Université de Toulouse

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :
Institut National des Sciences Appliquées de Toulouse (INSA Toulouse)

Discipline ou spécialité :

Informatique et Systèmes Embarqués

Présentée et soutenue par :
Linqing GUI

le : mercredi 13 février 2013

Titre :

Improvement of Range-free Localization Systems in Wireless Sensor Networks

---

Ecole doctorale :
Systèmes (EDSYS)

Unité de recherche :
IRIT

Directeur(s) de Thèse :

Thierry VAL et Anne WEI

Rapporteurs :

Benoît GELLER et Guillaume URVOY-KELLER

Membre(s) du jury :
Benoît GELLER, Professeur à l'ENSTA-Paristech
Guillaume URVOY-KELLER, Professeur à l'I3S-Nice
Eric GRESSIER-SOUDAN, Professeur au CNAM-Paris
Adrien VAN DEN BOSSCHE, Maître de Conférences à l'Université de Toulouse
Thierry VAL, Professeur à l'Université de Toulouse
Anne WEI, Professeur au CNAM-Paris

Remerciements

Le travail que nous présentons dans ce rapport thèse a été effectué à Toulouse au sien du groupe Ingénierie Réseaux et Télécoms (IRT) au laboratoire d'Institut de Recherche en Informatique de Toulouse (IRIT).

Tout d'abord je tiens à remercier chaleureusement les membres du jury. Merci au Professeur Benoît Geller (ENSTA-Paristech), et au Professeur Guillaume Urvoy-Keller (I3S-Nice) pour avoir accepter de rapporter mon travail. Mes remerciements s'adressent aussi au Professeur Eric Gressier-Soudan (CNAM-Paris), et au MCF Adrien Van den Bossche (Université de Toulouse), qui ont fait l'honneur de participer au jury de ma soutenance de thèse.

Je remercie vivement mes directeurs de thèse Monsieur Thierry Val professeur à l'Université de Toulouse, et Madame Anne Wei professeur au CNAM-Paris. Leurs conseils, orientations, et remarques ont été d'une importance crucial dans l'aboutissement de ce travail.

Mes vifs remerciements s'adressent aussi à tous ceux qui ont contribué à ces travaux, collègues d'IRIT que j'ai eu le plaisir de travailler avec eux (Réjane, Juan, Asma, Chiraz, Sabri, ...).

Je remercie le CSC (Chine Scholarship Council) qui m'a soutenu financièrement pendant cette thèse.

Finalement, j'exprime tous mes remerciements et mon affection à ma famille pour leur précieux soutien tout au long de ma thèse.

Résumé

Ces dernières années, les réseaux de capteurs sans fil attirent les intérêts de la recherche mondiale, en raison de leurs vastes applications comme les soins médicaux, les maisons intelligentes, et la surveillance de l'environnement. Pour ces applications, la localisation des équipements mobiles communicants est une problématique importante.

Les algorithmes de localisation existants peuvent être classés en deux catégories "range-based" et "range-free". Le principe "range-based" est de mesurer précisément la distance ou l'angle entre deux nœuds d'un même réseau. Par la suite, la position peut alors être obtenue simplement par trilatération ou triangulation. Le principe "range-free" utilise uniquement des informations de connectivité entre les nœuds du réseaux qui sont ou pas à portée les uns des autres. Généralement, les nœuds (fixes ou mobiles) dont on connait la position sont appelés "ancres" ou *anchors*. Les autres nœuds avec une position à déterminer sont appelés "nœuds normaux" ou *normal nodes*. Pour estimer leurs positions, les nœuds normaux recueillent tout d'abord des informations de connectivité réseaux ainsi que la position des ancres, puis calculent leurs positions. Par rapport au principe "range-based", la technique "range-free" est plus rentable, parce qu'il n'y a pas besoin de matériels supplémentaires pour la mesure et l'évaluation de la distance. Par conséquence, nous avons focalisé nos travaux de la thèse sur la technique "range-free". Ces dernières années, de nombreux algorithmes "range-free" ont été proposés. Parmi eux, Centroïde et CPE (*Convex Position Estimation*) nécessitent des nœuds normaux ayant au moins trois ancres voisines à un saut, tandis que DV-hop (*Distance Vector-Hop*) n'impose pas cette restriction. Toutefois, les algorithmes "range-free" ne sont pas assez précis. De plus, les algorithmes de la littérature sont généralement étudiés hors contexte réseau. Notre objectif est de proposer des algorithmes et des protocoles permettant d'améliorer la précision de localisation de ce type de méthode range-free.

Afin de permettre à chaque nœud normal de choisir son propre algorithme de localisation suivant la topologie environnante, nous avons proposé un mécanisme adapté en séparant les nœuds normaux en deux classes : les nœuds de la première classe ont au moins 3 ancres voisines (à 1 saut ou à portée radio), alors que les nœuds de la deuxième classe ont moins de trois ancres voisines.

Pour les nœuds normaux de la classe 1, nous avons proposé un nouvel algorithme "*Mid-perpendicular*", qui cherche à trouver un centre de la zone de recouvrement des cellules radio des ancres voisines. Les résultats des simulations par MATLAB montrent que, en moyenne, "*Mid-perpendicular*" offre une meilleure précision que Centroïde et CPE.

Pour les nœuds normaux de la classe 2, nous avons proposé deux algorithmes "*Checkout DV-hop*" et "*Selective 3-Anchor DV-hop*". En utilisant la distance estimée entre le nœud normal et sa plus proche ancre, "*Checkout DV-hop*" ajuste le résultat de la localisation de DV-hop. Bien que "*Checkout DV-hop*" n'ajoute qu'une étape simple à DV-hop, son amélioration sur la précision n'est pas très remarquable. Ainsi, nous avons proposé un autre nouvel algorithme "*Selective 3-Anchor DV-hop*", qui peut obtenir une meilleure précision au prix d'une augmentation plus importante de la complexité de calcul. Le principe de cet algorithme est le suivant: le nœud normal sélectionne toutes les trois ancres possibles afin de former des « 3-ancres groupes », puis il calcule les positions estimées grâce à ces « 3-ancres groupes ». Enfin, en fonction de la relation entre les positions estimées et les connectivités, le nœud normal choisit la position la plus précise.

Lors de la vérification de nos trois nouveaux algorithmes, nous avons trouvé que la plupart des algorithmes existants sont étudiés en utilisant uniquement des simulateurs algorithmiques tels que MATLAB, les problèmes liés aux réseaux et les influences des protocoles ont été généralement négligés comme la collision des trames et la synchronisation des nœuds. Ainsi, nous avons proposé deux protocoles : ”*DV-hop protocol*” et ”*Classe-1 protocol*”. Ensuite, nous avons combiné ces deux protocoles pour obtenir notre ”*adaptive range-free localization protocol*”. Dans ”*DV-hop protocol*”, nous avons défini des formats de trames adaptés, et une nouvelle méthode d'accès ”E-CSMA/CA” pour améliorer les performances de la couche MAC classique ”*non-slotted CSMA/CA*”. D'un côté, notre ”*DV-hop protocol*” peut être utilisé pour mettre en œuvre les algorithmes basés sur DV-hop, notamment ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”. De l'autre, notre ”*Classe-1 protocol*” peut être utilisé pour mettre en œuvre les algorithmes tels que *Centroïde*, CPE et ”*Mid-perpendicular*”.

Basé sur nos protocoles, en utilisant le simulateur WSNet, nous avons simulé différents algorithmes ”*range-free*” dans le contexte de réseaux conformes au standard IEEE 802.15.4. Les résultats sont présentés et analysés en termes de la précision de la localisation, charge du réseau, mobilité des nœuds, et synchronisation de ces derniers. Les résultats montrent que globalement nos nouveaux algorithmes sont plus précis que les algorithmes classiques. Par rapport à la charge du réseau, les algorithmes basés sur DV-hop sont beaucoup plus complexes que les algorithmes de la classe 1, parce que DV-hop nécessite des diffusions globales dans un réseau. Eu égard de la mobilité des nœuds, l'influence sur la précision des algorithmes basés sur DV-hop est plus importante que celle des algorithmes de la classe 1, parce que DV-hop nécessite une durée plus longue pour les diffusions globales. Finalement, nous avons aussi montré que la synchronisation des nœuds n'est pas nécessaire pour nos algorithmes et nos protocoles.

En perspectives, nous voulons étudier la performance des algorithmes en utilisant un modèle de couche radio réel. Nous sommes également intéressés par la combinaison des algorithmes ”range-based” et ”range-free”, et la mise en œuvre de nos méthodes sur prototypes.

Abstract

Wireless sensor networks have attracted worldwide research and industrial interest, because they can be applied in various areas such as hospital surveillance, smart home, and environmental monitoring. For most of these applications, localization is a fundamental issue.

The existing localization techniques can be generally categorized into two types: range-based and range-free. Range-based schemes need to first precisely measure the range information (the distance or the angle) between concerned equipments, and then calculate the desired position based on trilateration or triangulation approaches. Instead of the range information, the range-free scheme uses connectivity information between nodes. In this scheme, the nodes that are aware of their positions are called anchors, while others are called normal nodes. Anchors are fixed, while normal nodes are usually mobile. To estimate their positions, normal nodes first gather the connectivity information as well as the positions of anchors, and then calculate their own positions. Compared with range-based schemes, the range-free schemes are more cost-effective, because no additional ranging devices are needed. As a result, we focus our research on the range-free schemes in this thesis. During these years, many range-free localization algorithms have been proposed. Among them, Centroid, CPE (Convex Position Estimation), and DV-hop (Distance Vector-Hop) are well known algorithms. Centroid and CPE algorithms require a normal node has at least three neighbor anchors, while DV-hop algorithm doesn't have this requirement. However, these localization algorithms are not accurate enough, and they are usually studied without network context. Thus, we are interested in the investigation with wireless network context by implementing and improving new localization algorithms.

In order to permit each normal node to choose its suitable localization algorithm, we propose an adaptive mechanism to categorize normal nodes into two classes: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes.

For class-1 normal nodes, we propose a new algorithm named as *Mid-perpendicular*, which tries to find a centre point of the overlap communication area of neighbor anchors. The simulation results by MATLAB show that, on average, the accuracy of *Mid-perpendicular* algorithm is better than Centroid and CPE.

For class-2 normal nodes, we propose two algorithms *Checkout DV-hop* and *Selective 3-Anchor DV-hop*. Based on estimated distance between the normal node and its nearest anchor, Checkout DV-hop adjusts the result position of DV-hop algorithm. In order to further improve the accuracy of *Checkout DV-hop* algorithm, we provide another new algorithm *Selective 3-Anchor DV-hop*, which can obtain much better accuracy at the cost of higher computation complexity. The basic principle of the latter is as follows. The normal node first selects any three anchors to form a 3-anchor group, then it calculates the candidate positions based on each 3-anchor group, and finally according to the relation between candidate positions and their connectivities, the normal node chooses the best candidate position.

During the verification process of our three new algorithms, we noted that most of the existing algorithms were only studied using tools like MATLAB which neglects the possible problems of a real wireless network context such as frame collision and node synchronization. Therefore, in this thesis, we propose two protocols: *DV-hop protocol* and *Class-1 protocol*. Then we combine these two protocols into our adaptive range-free localization protocol. In our *DV-hop protocol*, we design new

data payload formats, and a new access method E-CSMA/CA to improve the performance of non-slotted CSMA/CA in the IEEE 802.15.4 wireless network. Note that in one part, our *DV-hop protocol* can be used to implement the DV-hop based algorithms, including *Checkout DV-hop* and *Selective 3-Anchor DV-hop*. In another part, our *Class-1 protocol* can be used to implement the class-1 algorithms including *Mid-perpendicular*.

Based on our protocols, using the network simulator WSNet, we simulate the concerned range-free localization algorithms in the IEEE 802.15.4 wireless network. The comparative network simulation results are presented and analyzed in terms of localization accuracy, overhead, node mobility, and node synchronization. Results show that, globally, our new algorithms have better accuracy than the existing typical range-free algorithms. We can also note that, in term of overhead, the DV-hop based algorithms have much higher network overhead than the class-1 algorithms like Centroid and CPE, because DV-hop based algorithms require the broadcasts throughout the network. In terms of mobility, node mobility can have a bigger influence on the accuracy of DV-hop based algorithms than that of the class-1 algorithms, because DV-hop based algorithms need longer localization period to support the broadcasts through the network. Finally, it should be noted that, node synchronization is not necessary for our algorithms and protocols.

In the future, we will investigate the performance of algorithms in real radio propagation scenarios. We will also be interested in the combination of range-based and rang-free algorithms, and in the implementation of our methods into prototypes.

Publications

International Journal

- Linqing GUI, Thierry VAL, Anne WEI, “An Adaptive Range-free Localization Protocol in Wireless Sensor Networks”, International Journal of Ad Hoc and Ubiquitous Computing, in review, submitted on 20 December 2012.
- Linqing GUI, Thierry VAL, Anne WEI, Réjane DALCE, “Improvement of Range-free Localization Systems Realized by a DV-hop Protocol in Wireless Sensor Networks”, Ad Hoc & Sensor Wireless Networks, in review, submitted on 5 July 2012.
- Linqing GUI, Thierry VAL, Anne WEI, “A Novel Two-Class Localization Algorithm in Wireless Sensor Networks”, Network Protocols and Algorithms, vol 3, no 3 (2011).

International Conference

- Linqing GUI, Anne WEI, Thierry VAL, “A Range-Free Localization Protocol for Wireless Sensor Networks”, International Symposium on Wireless Communications Systems, Paris, August 2012.
- Linqing GUI, Thierry VAL, Anne WEI, “Improving Localization Accuracy Using Selective 3-Anchor DV-hop Algorithm”, IEEE Vehicular Technology Conference (VTC 2011-fall), San Francisco, September 2011.
- Linqing GUI, Anne WEI, Thierry VAL, “A Two-level Range-free Localization Algorithm for Wireless Sensor Networks”, IEEE conference on wireless communications, networking, and mobile computing (WICOM 2010), Chengdu, September 2010.
- Rejane DALCE, Linqing GUI, Thierry VAL, Adrien VAN DEN BOSSCHE, Anne WEI, “Localisation par Méthodes Range-based et Range-free de Stations Mobiles Communicantes dans un Réseau sans Fil”, Colloque Francophone sur l'Ingénierie des Protocoles (CFIP 2011), Sainte-Maxime, mai 2011.

National Conference

- Linqing GUI, Thierry VAL, Anne WEI, “Un Nouvel Algorithme de Localisation pour les Réseaux de Capteurs sans Fil”, Congrès des doctorants EDSYS, Toulouse, mai 2011.

Table of Content

List of Figures

List of Tables

Glossary

Mathmatical Notations

A_i one anchor; the position of A_i is (x_i, y_i)

A_{near} the nearest anchor to N_x

BE backoff exponent in CSMA/CA access method

$d_{DV\text{-}hop}$ distance between $N_{DV\text{-}hop}$ and the nearest anchor A_{near}

$d_{i,k}$ the distance between A_i and A_k

$d_{i,Nx}$ estimated distance between the normal node N_x and each anchor A_i

$d_{<i,j,k>,t}$ the distances between $N_{<i,j,k>}$ and anchor A_t

dph_i the approximate average distance per hop of A_i

dph_{near} the distance per hop of A_{near}

$dph_{<i,j,k>,t}$ the distance per hop between $N_{<i,j,k>}$ and A_t

$frame\_dph_i$ distance-per-hop frame of anchor A_i

$frame\_pos_i$ position frame of anchor A_i

$frame\_pos_{i,Nx}$ position frame sent by the anchor A_i to N_x

$frame\_req$ localization request frame from each normal node

$hop_{i,k}$ the minimal hop count between A_i and A_k

$hop_{i,Nx}$ the minimal hop count between N_x and A_i

$hop_{<i,j,k>,t}$ the hop count between $N_{<i,j,k>}$ and A_t

m number of neighbor anchors for one normal node

m_d total number of anchors in the network

N_x one normal node

$N_{<i,j,k>}$ a 3-anchor estimated position of N_x based on the three anchors A_i A_j and A_k

$N_{DV\text{-}hop}$ estimated position of N_x by DV-hop algorithm; the position of $N_{DV\text{-}hop}$ is (x', y')

NB number of backoff in CSMA/CA access method

num the total number of nodes (including anchors and normal nodes)

num_wait_pos maximum number of anchors whose positions received by each normal node

num_wait_dph maximum number of anchors whose distance-per-hop received by each normal node

RA_{thresh} threshold for the ratio of anchors

t_{bo} one back-off period in CSMA/CA access method

t_p localization period of our range-free localization protocol

t_{recv} duration of Step #1 and Step #2 in our Class-1 protocol

t_{s1} maximum duration of Step #1 in our DV-hop protocol

t_{s2} maximum duration of Step #2 in our DV-hop protocol

t_{wdi} A_i's waiting time before sending $frame\_dph_i$

t_{wlr} normal node's waiting time before sending localization request

t_{wpi} A_i's waiting time before sending $frame\_pos_i$

TRd_{nod} number of random times for generating different geographical distributions of nodes

TRd_{anc} number of random times for selecting nodes as anchors in the simulation

Abbreviations

ACK	Acknowledgement
AOA	Angle of Arrival
AP	Access Point
APIT	Approximate Point-In-Triangulation
BSS	Basic Service Set
CCA	Clear Channel Assessment
CPE	Convex Position Estimation
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
CTS	Clear to Send
DCF	Distributed Coordination Function
DIFS	Distributed Inter Frame Space
DSSS	Direct Sequence Spread Spectrum
DV-hop	Distance Vector-hop algorithm
DDV-hop	Differential DV-hop algorithm
E-CSMA/CA	Enhanced-Carrier Sense Multiple Access with Collision Avoidance
ER	Estimated Rectangle
ESS	Extended Service Set
FCS	Frame Check Sequence
FFD	Full-Function Device
FHSS	Frequency Hopping Spread Spectrum
GPS	Global Positioning System
GSM	Global System for Mobile communications
GTS	Guaranteed Time Slots
IEEE	Institute of Electrical and Electronics Engineers
IRIT	Institut de Recherche en Informatique de Toulouse
LQI	Link Quality Indication
LOS	Line of Sight
MAC	Medium Access Control
MEMS	Micro-Electronic-Mechanical Systems
MFR	MAC Footer
MHR	MAC Header
MIMO	Multiple Input and Multiple Output
MLE	Maximum Likelihood Estimation
MPDU	MAC Protocol Data Units
OCARI	Optimization of Communication for Ad hoc Reliable Industrial network
OFDM	Orthogonal Frequency Division Multiplexing
PC	Personal Computer
PCF	Point Coordination Function
PHY	Physical Layer

PN	Pseudo-Noise
PSD	Power Spectral Density
RF	Radio Frequence
RFD	Reduced-Function Device
RSSI	Received Signal Strength Indicator
RTS	Request to Send
TDD	Time Division Duplex
TDOA	Time Difference of Arrival
TOF	Time of Flight
UWB	Ultra Wide Band
WCL	Weighted Centroid Localization algorithm
WiFi	Wireless Fidelity
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Networks

1. General Introduction

1.1 Wireless Sensor Networks

With the development of wireless communication technologies and MEMS (Micro-Electronic-Mechanical Systems) [XKR 10], wireless sensor networks have become an important research area nowadays. The first research in this area was motivated by the military application "Distributed Sensor Networks (DSN) program" [GKS 88] and "Smart Dust" [KKP 99]. Then, the researchers in Berkeley University proposed "PicoRadio project" [SSA 01] to develop a low-power and low-cost ubiquitous sensor network, which was supposed to be applied in civilian areas. Later, many sensor network systems have been proposed, for example, "OCARI" by researchers in France [ACG 09]. Recently, civilian applications of wireless sensor networks have been considered including hospital surveillance [LCM 10], smart home [SK 08] [SUR 12], environmental monitoring [PPG 09] [ERO 12], and object tracking [GLN 09] [POL 12].

A wireless sensor network is a collection of sensor nodes organized into a cooperative network. Each sensor node has typically several parts, as shown in Figure 1-1. A sensor node is usually a tiny electronic device equipped with a battery for an energy source. It has sensors for detecting environment conditions such as temperature, sound, vibration, pressure, humidity or motion. A wireless transceiver is fitted for two way communications with other sensors.

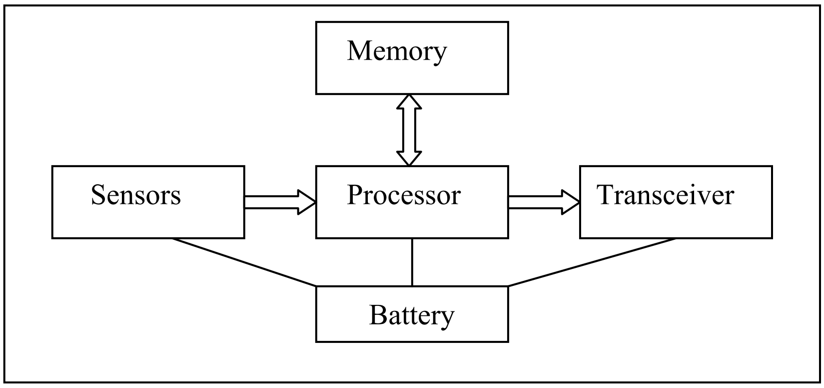


Figure 1-1. Structure of a Sensor Node

A sensor node has the following characteristics: (1) a small physical size, (2) low power consumption, (3) limited processing power, (4) short-range communications and (5) a small amount of memory storage.

By advanced networking protocols, sensor nodes can form various types of wireless networks that facilitate the life of human beings. For example, in a hospital, patients can be equipped with vital sign sensor nodes. These sensor nodes are able to measure and transmit the heart rate and blood oxygenation of patients [LC 09]. As shown in Figure 1-2, through the wireless network organized by these nodes, doctors can easily monitor the status of patients with a computer or a smart phone.

Unlike traditional wireless devices such as cell phones, wireless sensor nodes do not need to communicate directly with the nearest high-power control tower or base station, but only with their local peers. Instead of relying on a pre-deployed infrastructure, each sensor node becomes part of the overall infrastructure. This ad-hoc networking topology provides a mesh-like connection in a multi-hop fashion. The flexible mesh architecture dynamically adapts to support the adding of new nodes and the compensation for node failures.

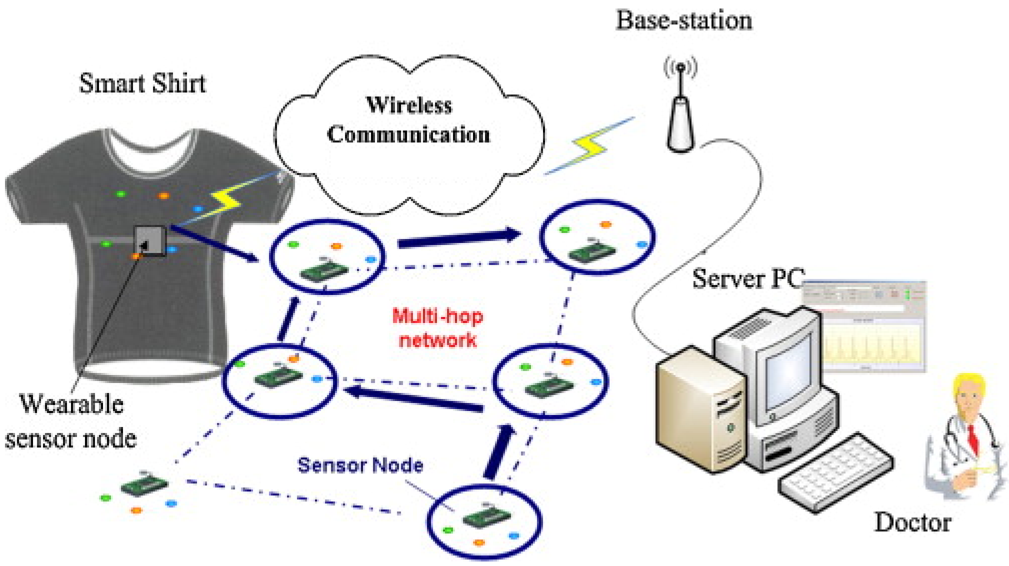


Figure 1-2. a Wireless Sensor Network for Hospital Monitoring [LC 09]

1.2 Localization in Wireless Sensor Networks

The location information of each sensor node in the network is critical for many applications. This is because users normally need to know not only what happens, but also where interested events happen or where the target is. For example, in hospital surveillance, the knowledge of where the patient is can help the doctors arrive at the right place as quickly as possible in urgent case [LC 09] [LCM 10]; in a disaster relief operation using WSN to locate survivors in a collapsed building, it is critical that sensors report monitoring information along with their location [WAL 06] [LY 07] [LP 05] [TAM 06]. On the other hand, the position parameters of sensor nodes are assumed to be available in many operations for network management, such as routing where a number of geographical algorithms have been proposed [BCS 98] [KK 00] [KGK 05], topology control that uses location information to adjust network connectivity for energy saving [ALW 03] [LH 05] [XHE 01], and security maintenance where location information can be used to prevent malicious attacks [HPJ 03] [LP 05].

Many ideas have been proposed for node localization in wireless sensor networks [CHA 06] [MFA 07] [AK 09] [LYW 10]. Based on whether accurate ranging is required, there are generally two types of methods: range-based and range-free.

Range-based schemes [OWW 10] [KRV 09] [VH 04] [KH 05] [RS 06] need to first precisely measure the range information (the distance or the angle) between concerned equipments, and then calculate the desired position based on trilateration or triangulation approaches. The ranging methods typically use Received Signal Strength Indicator (RSSI) [KRV 09], Time of Flight (TOF) [VH 04], and Angle of Arrival (AOA) [RS 06]. Different localization systems have rapidly evolued. For example, GPS (Global Positioning System) [OWW 10] is the most well-known range-based technique using TOF or TDOA (Time Difference of Arrival). However, the GPS devices not only consume lots of energy, but also fail to work indoors. An alternative system is GSM (Global System for Mobile communications), using RSSI and AOA methods. Now, the most precise system is based on UWB (Ultra Wide Band) which can be used to measure time of flight with high precision [LDG 09]. In general, the range-based techniques have two major drawbacks. First, the range information is very easily affected by multipath fading, noise and environment variations. Second, usually, additional ranging devices are needed, which consume more energy and increase the overall cost.

While the range-based scheme uses the distance or angle between nodes, the range-free scheme uses connectivity information between nodes. In this scheme, the nodes that are aware of their positions are called anchors, while others are called normal nodes. Anchors are fixed, while normal nodes are usually mobile. Normal nodes first gather the connectivity information as well as the positions of anchors, and then calculate their own positions. Here, the connectivity information of a node *N* can be its hop counts to other nodes. The connectivity is used as an indication of how close this node *N* to other nodes. For example, the nodes within *N*'s transmission range is said to be one hop away, and can be called as the neighbor nodes of *N*. Since no ranging information is needed, the range-free scheme can be implemented on low-cost wireless sensor networks. Another advantage of range-free scheme is its robustness; the connectivity information between nodes is not easily affected by the environment. As a result, we focus our research on the range-free scheme.

Many range-free algorithms have been proposed for several years, such as Centroid [BHE 00] [PAT 04], CPE (Convex Position Estimation) [DPG 01], Approximate Point-In-Triangulation (APIT) [HHB 03], DV-hop (Distance Vector-hop) [NN 03] [LCK 10] [HZL 10]. However, these existing schemes are not accurate enough. So, the localization accuracy must still be studied and improved.

Considering the limitations of existing work, we tried to investigate practical solutions to bridge the gap between low cost and high accuracy for range-free localization. In the following, we give an overview about objectives and contributions of this thesis.

1.3 Research Objectives and Contributions

In general, our research work has the following objectives and contributions.

(1) For each normal node to adaptively choose its proper localization algorithm, we categorize normal nodes into two classes according to the number of neighbor anchors. Here, the neighbor anchors of a normal node are those anchors within the transmission range of the normal node.

Based on our study of the existing range-free algorithms, we found that Centroid and CPE only work for the normal nodes which have at least 3 neighbor anchors, while DV-hop can work for all normal nodes. However, Centroid and CPE have much less network overhead and calculation complexity than DV-hop.

In order to let each normal node choose its suitable algorithm, we categorize normal nodes into two classes according to the number of neighbor anchors: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes. Then, the class-1 normal nodes can use the low-complexity localization algorithms like Centroid and CPE. As for class-2 nodes, only the DV-hop based algorithms are available. Therefore, using this classification, the total overhead of the network can be reduced.

(2) For class-1 normal nodes, we will propose a new algorithm, which can obtain better accuracy than Centroid and CPE.

Both in Centroid and CPE algorithms, anchors are assumed to have the same communication range in a circle shape. A normal node is located inside the intersection area formed by the ranges of neighbor anchors. Centroid algorithm can get relatively good accuracy when the distribution of anchors is regular. However, when the distribution of anchors is not regular, the estimated position derived from the Centroid algorithm can go out of the intersection area, resulting in low localization accuracy [SCH 08]. The CPE algorithm defines an estimated rectangular to bound the intersection area. But we find that sometimes the estimated position of CPE can still go out of the intersection area.

However, using our proposed “Mid-perpendicular” method, we can find the closest centre of the intersection area. Therefore, statistically, our method has better accuracy than CPE and Centroid.

(3) For class-2 normal nodes, at first, we aim to propose a simple algorithm, which can have better accuracy than DV-hop algorithm.

Although the DV-hop algorithm is much more complicated than Centroid and CPE, it is a suitable solution for normal nodes whose neighbor anchors are fewer than three. When DV-hop algorithm is used to localize a normal node, at first the normal node estimates its distance to anchors based on its hop counts to the anchors. Then based on the estimated distance values and the positions of anchors, the normal node can calculate its position.

However, the accuracy of the estimated distance varies with the hop count. Let’s consider two anchors. One has a fewer hop count to the normal node than the other anchor. That means, one anchor is nearer to the normal node than the other. *We prove that, in general, the normal node has a more accurate estimated distance to the nearer anchor than to the other anchor. This conclusion has been proved both by analysis and simulation.* As a result, statically, the estimated distance to the nearest anchor will have the best accuracy.

Based on estimated distance between the normal node and its nearest anchor, we propose Checkout DV-hop algorithm to adjust the result position of DV-hop algorithm. This additional calculation is designed as simple as possible, so that the Checkout DV-hop algorithm has the same complexity level with the DV-hop algorithm.

(4) Still for class-2 normal nodes, in order to significantly improve the accuracy, we want to propose a new algorithm.

Even if we develop the Checkout DV-hop to improve the accuracy of the DV-hop algorithm, it is still necessary to find a better solution in order to get a satisfactory result. The solution is our Selective 3-Anchor DV-hop algorithm. This algorithm brings many more candidates, and then selects the best candidate as the final position. Each candidate is generated based on 3 anchors, so called as 3-anchor estimated position.

The connectivity parameter is used as the criterion to select the best 3-anchor estimated position. We note that, in this context, the connectivity can identify the position of the normal node. If two normal nodes have similar connectivity, then they must have similar positions, that is to say, they are very near to each other. Based on this conclusion, the best 3-anchor estimated position is the one that has the most similar connectivity with the normal node.

(5) In order to study and compare the DV-hop based algorithms in practical network scenarios, we propose a new DV-hop localization protocol.

Most of DV-hop based algorithms are implemented using MATLAB. They all neglect the issues in a real network, such as frame collisions, node mobility and node synchronization. Therefore a new DV-hop protocol is designed. The new protocol covers the format of data payload, the improved collision reduction method E-CSMA/CA, several parameters that can decide the end of each DV-hop step, and the complete frame exchange procedure. Note that our protocol can be used in both synchronized and unsynchronized networks.

(6) Aiming to implement all the concerned range-free localization algorithms, we propose a new range-free localization protocol. This protocol, as an extended version of DV-hop protocol, solves the implementation problems for the range-free algorithms like Centroid, CPE, Mid-perpendicular, and the DV-hop based algorithms.

(7) Based on the above protocol, using the network simulator WSNet [HCG 08] [WSNET], *we simulate the typical range-free localization algorithms. The comparative simulation results are analyzed in terms of accuracy, overhead, mobility, and synchronization.*

1.4 Organization of the Manuscript

The rest of the thesis is organized as follows. Chapter 2 first provides a survey about physical layer and MAC layer specified in the standards about wireless sensor networks, then introduces the typical localization algorithms and techniques. Chapter 3 concentrates on the topic of improvement on range-free localization algorithms, and presents our algorithms (i) Mid-perpendicular, (ii) Checkout DV-hop, (iii) Selective 3-Anchor DV-hop. Their superior accuracy and flexibility over traditional solutions are demonstrated through simulations of the concerned algorithms. Chapter 4 presents our protocols (i) a new DV-hop localization protocol, and (ii) an extended version, a range-free localization protocol. Results from simulation and system evaluation validate the performance gain of our proposals comparing with previous works. Finally, Chapter 5 provides the conclusions and an outlook on future researches.

2. Background and Related Work

In this section, we first introduce the standards and technologies which have been frequently utilized in wireless sensor networks. The physical layer and MAC layer specifications of these standards will be compared in order to explain our choice that is based on the IEEE 802.15.4 standard. Then, we will present the existing localization algorithms and techniques for wireless sensor networks. The advantages and disadvantages of both range-free and rang-based algorithms will be listed.

2.1 Standards and Technologies in Wireless Networks

2.1.1 IEEE 802.11 Standard and WiFi

Wireless Fidelity (WiFi) is the technology based on IEEE 802.11 standard [802_11]. It is mostly deployed for Wireless Local Area Network (WLAN) applications.

A WLAN may either consist of stations running in ad-hoc mode (for example the new 802.11s amendment), or it may consist of stations and access points (AP) in infrastructure mode. These two modes are distinguished by the use of an access point (AP). The AP can not only provide access to a wired LAN, but also organize the communications between stations in the same service area.

The basic cell of a WLAN is called a Basic Service Set (BSS), which is a set of mobile or fixed stations. If a station moves out of its BSS, it can no longer directly communicate with other members of the BSS. Based on the BSS, IEEE 802.11 standard employs the Independent Basic Service Set (IBSS) and Extended Service Set (ESS) network configurations. As shown in Figure 2-1, the IBSS operation is possible when IEEE 802.11 stations are able to communicate directly without any AP. Because this type of WLAN is often formed without pre-planning, for only as long as the WLAN is needed, this type of operation is often referred to as an ad hoc network. Instead of existing independently, a BSS may also form an extended form of network that is built with multiple BSSs. The architectural component used to interconnect BSSs is the distribution system (DS). The DS with APs allow IEEE 802.11 to create an ESS network of arbitrary size and complexity. This type of operation is often referred to as an infrastructure network.

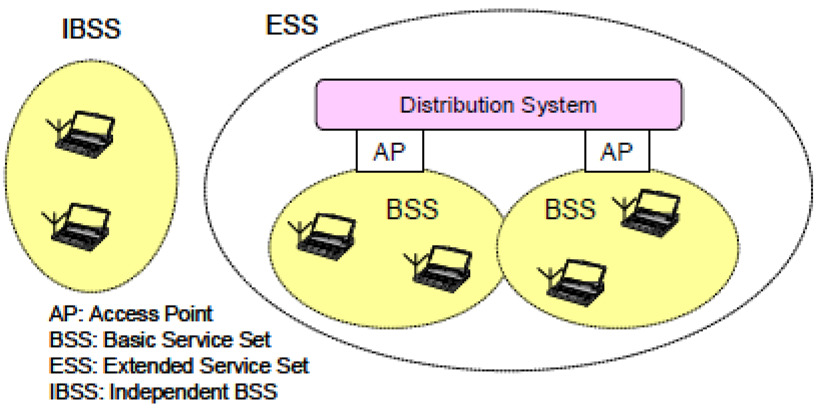


Figure 2-1. IBSS and ESS configurations of WiFi networks [LSS 07]

2.1.1.1 IEEE802.11 Physical Layer

The Physical layer (PHY) of 802.11 acts as an interface between the wireless media and the MAC layer. It is responsible for the actual transmitting of frames and for sensing whether the channel is idle or not and reporting this back to the MAC layer.

Since the 802.11 has been standardized by IEEE, a number of task groups have been formed to add functionalities and improve performance of WLAN. IEEE 802.11b, 802.11a, 802.11g and 802.11n are currently used for WLAN applications, while IEEE 802.11ac is under development [WIK 802.11ac]. Their key characteristics are summarized in Table 2-1.

Table 2-1. Summarized PHY Features of IEEE 802.11 standards

	802.11b	802.11a	802.11g	802.11n	802.11ac
Spectrum (GHz)	2.4	5	2.4	2.4/5	5
Max Data Rate (Mbps)	11	54	54	54--600	6900
Transmission Protocol	DSSS	OFDM	OFDM, DSSS	MIMO-OFDM	Multi-user MIMO (OFDM)
Typical Power (mW)	15 - 20 dBm				
Typical Range	50-100m				
Current Status	Widely Used	Limited Use	Widely Used	Emerging	Under Development

The 802.11 standard supports three different PHYs. For example, the 802.11b uses the Direct Sequence Spread Spectrum (DSSS). DSSS can transform and spread the energy of the transmitted signal in a wider frequency range. This makes it easier for the receiver to pick up the signal and recover the frame sent. The higher bit rates of 802.11g are achieved by using more advanced frequency modulation schemes, Orthogonal Frequency Division Multiplexing (OFDM). This scheme utilized multi-carrier modulation methods. A number of orthogonal sub-carriers are used to carry data to cope with severe channel conditions. The IEEE 802.11n is an amendment to IEEE 802.11-2007 to improve network throughput over the two previous standards - 802.11a and g. It offers significant increase in the maximum raw data rate from 54Mbps to 600 Mbps by using Multiple Input and Multiple Output (MIMO). In addition, IEEE 802.11n can operate at 5GHz frequency band, which may benefit its usage in present of other wireless system using 2.4GHz, such as Bluetooth and ZigBee. The IEEE 802.11ac is currently under development, providing high-throughput wireless local area networks on the 5 GHz band.

2.1.1.2 IEEE802.11 MAC Layer

The MAC layer of IEEE 802.11 is responsible for providing equal access to shared wireless media and association of the wireless devices. Although the media is shared, two transmissions cannot occur at the same time, since both transmissions would probably fail because of collision.

Two operating modes are offered, the Distributed Coordination Function (DCF) which is mandatory, and the Point Coordination Function (PCF) which is optional and centralized. DCF is sometimes referred to as contention mode, since each sender has to contend for access to the media.

As an approved amendment to the IEEE 802.11 standard, 802.11e enhances the DCF and the PCF, through a new coordination function: the hybrid coordination function (HCF). In this method, high-priority traffic has a higher chance of being sent than low-priority traffic. This is very importance for delay-sensitive applications, such as Voice over Wireless LAN and streaming multimedia.

In the DCF mode, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) is used to control media access, which is also utilized by ZigBee MAC layer (see the section 2.1.3.2).

CSMA mechanism is well known in the industry, where the most popular is the Ethernet using a CSMA/CD protocol (CD standing for Collision Detection). The CSMA works as follows: A station desiring to transmit senses the medium, if the medium is busy (i.e. some other station is transmitting) then the station will defer its transmission to a later time, if the medium is sensed free then the station is allowed to transmit. This mechanism is very effective when the medium is not heavily loaded, since it allows stations to transmit with minimum delay. But there is always a chance of stations transmitting at the same time (collision), caused by the fact that the stations sensed the medium free and decided to transmit simultaneously.

In the Ethernet case, this collision is recognized by the transmitting stations based on Collision Detection method. However, this method cannot be used on a Wireless LAN, because of two main reasons: [TAY 10]

(i). Implementing a Collision Detection mechanism would require the implementation of a Full Duplex radio, capable of transmitting and receiving simultaneously. However, a wireless station normally cannot transmit its own signal and receive another signal at the same time.

(ii). In a wireless environment we cannot assume that all stations hear each other (which is the basic assumption of the Collision Detection scheme). When a station is willing to transmit and senses the medium free, this only means the medium around the transmitting station is free. However, the medium around the receiving station can still be busy.

In order to overcome these problems, the 802.11 uses a Collision Avoidance (CA) mechanism together with the signals such as positive Acknowledge (ACK) and RTS/CTS (Request to Send/Clear to Send). The principle of CSMA/CA is as follows:

A station willing to transmit first senses the medium. If the medium is free for a specified time (called DIFS, Distributed Inter Frame Space) then the station is allowed to transmit. The receiving station will check the CRC (Cyclic Redundancy Check) of the received frame and send an acknowledgement (ACK) frame. Receipt of the ACK will indicate the transmitter that no collision occurred. If the sender does not receive the ACK then it will retransmit the frame until it gets ACK or thrown away after a given number of retransmissions.

The above case has a condition: when the station is ready to transmit, it senses that the medium has been free for the specified time DIFS. However, if the station senses that the medium is busy, then an Exponential Backoff method is necessary.

Backoff is a well known method to resolve contention between different stations willing to access the medium. This method requires each station to choose a random number between 0 and a given number, and wait for this random number of slots before accessing the medium. During this waiting time, the station always checks whether a different station has accessed the medium. Exponential Backoff means that each time if the station chooses a slot and happens to collide, it will increase the maximum number for the random selection exponentially.

It can be noted, this CSMA/CA access method is very adaptive for the distributed wireless sensor networks.

2.1.2 IEEE 802.15.1 Standard and Bluetooth

Bluetooth is an industry standard developed by Ericsson, which later was adopted by the IEEE 802.15 work group as a WPAN (Wireless Personal Area Network) standard, IEEE 802.15.1. It can enable several devices to communicate with each other, overcoming problems of synchronization.

Bluetooth is mostly applied for short-range and cheap devices to replace cables for digital peripherals, such as keyboards, printers, and hands-free earphones [802_15_1].

2.1.2.1 IEEE802.15.1 Physical Layer

A summary of some key features of Bluetooth physical layer is provided in Table 2-2, which is extracted from IEEE802.15.1-2005 and Bluetooth v3.0 specifications.

Table 2-2. Key Features of Bluetooth Physical Layer

Frequency Band	2.4GHz
Transmission Protocol	Spread Spectrum(Frequency hopping)
Transmission Power	1-100mW
Data Rate	0.723-24Mbps
Transmission Range	10-100m

Bluetooth uses the Frequency Hopping Spread Spectrum (FHSS) technique to transmit signals. This technique provides processing gain, which improves the chance of successful packet delivery in the presence of interference. Figure 2-2 shows the Bluetooth channel frequency hopping mechanism. 79 channels are used. Each channel has 1MHz bandwidth. During communication, a Bluetooth system can make 1,600 hops per second evenly spread over the 79 channels according to a pseudorandom pattern. Therefore if the system transmits on a bad channel, the next hop (which will occur 625 µs later), will hopefully be on a good channel [VAL 02].

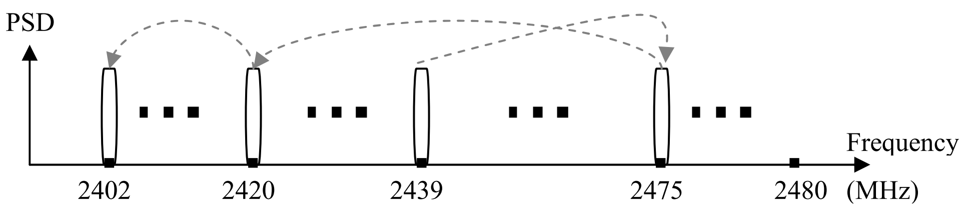


Figure 2-2. Power Spectral Density (PSD) of Frequency Hopping in Bluetooth

2.1.2.2 IEEE802.15.1 MAC Layer and Topology

Two connectivity topologies are defined in Bluetooth: the Piconet and Scatternet. A maximum of eight simultaneous devices can participate in a Piconet, which can comprise of one master device and up to seven active slave devices. All devices participating in communications in a given Piconet are synchronized using the clock of the master. Each Bluetooth device is capable of assuming the master or slave role, depending on its configuration. Usually a device that sends request for connection is determined as the master (i.e. the device that initializes the formation of the Piconet). Bluetooth provides both point-to-point and point-to-multipoint connections. Piconet permits master-to-slave and slave-to-master communications. Slave-to-slave traffic must to go through a master.

Within a piconet, the master device and slave devices communicate in Time Division Duplex (TDD) manner. The data transmissions use up the whole frequency slot (1 MHz) with downlink (from the master to slaves) and uplink (from slaves to master) transmission divided into different time slots. The master determines which device can have access to the communication channel by addressing a slave. This slave will then have the right to send its data in the next time slot.

A member of one Piconet could also be a member of another Piconet. A device participating multiple Piconets does so on based on time division. Before the device leaves one Piconet, it tells the master it will not be available for a predetermined interval and puts itself in non-active mode (sniff, hold or park mode), and then it adjusts its clock to another piconet and joins the conversation there. Such a device may act as the bridge between two Piconets. Several Piconets can be connected together to form a Scatternet. However, the Scatternet is rarely used and always stays on the theory level.

In fact, because of strict requirement on synchronization, Bluetooth is not so suitable for wireless sensor networks. Besides, the new standards Bluetooth v3.0 and v4.0 specify high date rate and power consumption, which is different from the features of wireless sensor networks.

2.1.3 IEEE 802.15.4 Standard and ZigBee

ZigBee technology is a low data rate, low power consumption, low cost, wireless networking protocol targeted towards automation and remote control applications. IEEE 802.15.4 committee started working on a low data rate standard a short while later. Then the ZigBee Alliance and the IEEE 802.15.4 group decided to join forces and use ZigBee as the commercial name for this technology. However, the two groups still work on different parts of the technology. The IEEE 802.15.4 group has standardized the physical (PHY) and the medium access control (MAC) layers, whereas the ZigBee alliance concentrates on the development of the upper layers and the overall development. Figure 2-3 shows the ZigBee protocol stack, as well as the relations between IEEE 802.15.4 and ZigBee in terms of the protocol [VVC 11] [ZAR 04].

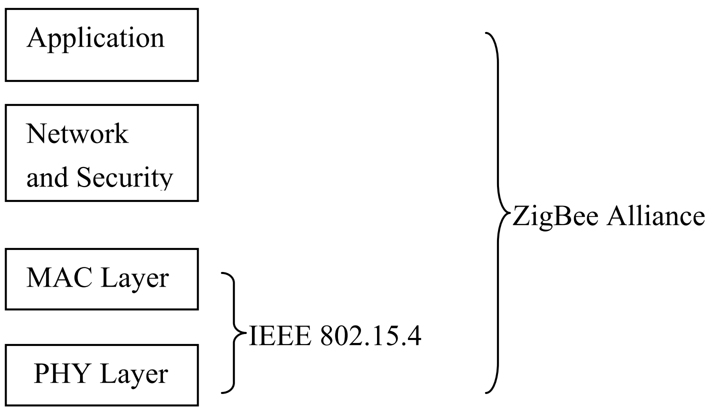


Figure 2-3. ZigBee and IEEE 802.15.4 Protocol Stack

Using the ZigBee technonology, a Wireless Personal Area Network (WPAN) consists of several components. The most basic is the device. A device can be a Full-Function Device (FFD) or Reduced-Function Device (RFD). A network shall include at least one FFD, operating as the PAN coordinator.

The FFD can operate in three modes: a PAN coordinator, a coordinator or a device. An RFD is intended for applications that are extremely simple and do not need to send large amounts of data. An FFD can talk to RFDs or FFDs, while an RFD can only talk to an FFD.

Figure 2-4 shows 3 types of topologies that ZigBee supports: star, peer-to-peer (or mesh) and cluster tree [802_15_4].

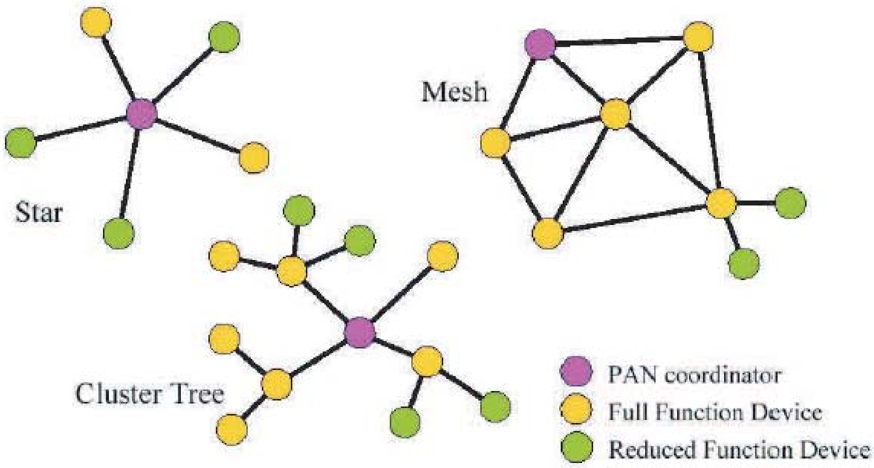


Figure 2-4. ZigBee Topology Models [802_15_4]

In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator. The PAN coordinator may be fixed and main powered, while the devices will most likely be battery powered. Applications that benefit from this topology include home automation, personal computer (PC) peripherals, toys and games.

In the mesh topology, there is also one PAN coordinator. In contrast to star topology, any device can communicate with any other device as long as they are in range of one another. A peer-to-peer network can be ad hoc, self-organizing and self-healing. Applications such as control and monitoring, asset and inventory tracking would benefit from such a topology. It also allows multiple hops to route messages from any device to any other device in the network. It can provide reliability by multipath routing.

Cluster-tree network is a special case of a peer-to-peer network in which most devices are FFDs and an RFD may connect to a cluster-tree network as a leave node at the end of a branch. Any of the FFD can act as a coordinator and provide synchronization services to other devices and coordinators. Only one of these coordinators however is the PAN coordinator. This clustered structure can increase coverage area, however, at the cost of increased message latency.

2.1.3.1 IEEE 802.15.4 Physical Layer

The features of the IEEE 802.15.4 PHY are activation and deactivation of the radio transceiver, energy detection (ED), link quality indication (LQI), channel selection, clear channel assessment (CCA) and transmitting as well as receiving frames across the physical medium.

IEEE 802.15.4 offers three choices for the PHY for low-power operations. The differences in the choices lie in the frequency band used. They differ with respect to the data rate as shown in Table 2-3.

Although IEEE 802.15.4 introduced in 2006 two optional specifications which support high data rate up to 250 kbps for the 868 and 915 MHz bands, they are rarely used because of their complexity in implementation and channel limitation. Therefore, 2.4 GHz is popularly used for higher data rate.

It should be noted that theoretically the highest data rate supported by a ZigBee channel is 250 kbps. However the calculation of this value does not take into account header bytes, CSMA waiting times, etc. As a result, the actual channel capacity can be less than 250kbps.

Table 2-3. IEEE 802.15.4 PHY Specifications

Frequency Band	868 MHz	915 MHz	2.4 GHz
Applied Area	Europe	America	Worldwide
Maximum Data Rate	20 kbps	40 kbps	250kbps
Typical Range	10-100m		
Transmit Power	(-25) – 0 dBm		
Receiver Sensitivity	-92dBm	-92dBm	-85dBm
Number of Channels	1	10	16
Channel Spacing	2MHz	2MHz	5MHz

2.1.3.2 IEEE 802.15.4 MAC Layer

The MAC layer provides services to the upper layers, and enables the transmission and reception of MAC Protocol Data Units (MPDU) across the PHY data service. According to the IEEE 802.15.4 standard, features of the IEEE 805.15.4 MAC layer include beacon management, channel access, guaranteed time slots (GTS) management, frame validation, acknowledged frame delivery, association and disassociation.

Two different modes of operation are allowed in PAN (Personal Area Network): the beacon mode and the non-beacon mode. In the beacon-enabled PAN, in order to synchronize all the devices, the coordinators emit regular beacons. However, in the non-beacon PAN, there is no emission of beacons, thus the devices in the network are not synchronized.

Mainly available for networks of star topology, the beacon mode can provide node synchronization as well as QoS (Quality of Service) for applications. In the beacon mode, the coordinators periodically diffuse beacon frames. When a node receives a beacon, this node uses the beacon to synchronize itself with the coordinator. This mechanism permits the best performance on energy saving, because the node can turn into sleep state as soon as it finishes the synchronization [VCV 08]. Besides, if a node wants to receive data from the coordinator, the node can also choose to wake up and put its data request in the reply frame. This kind of data transmission is known as indirect transmission in the star topology, where all the exchanges pass by the coordinator.

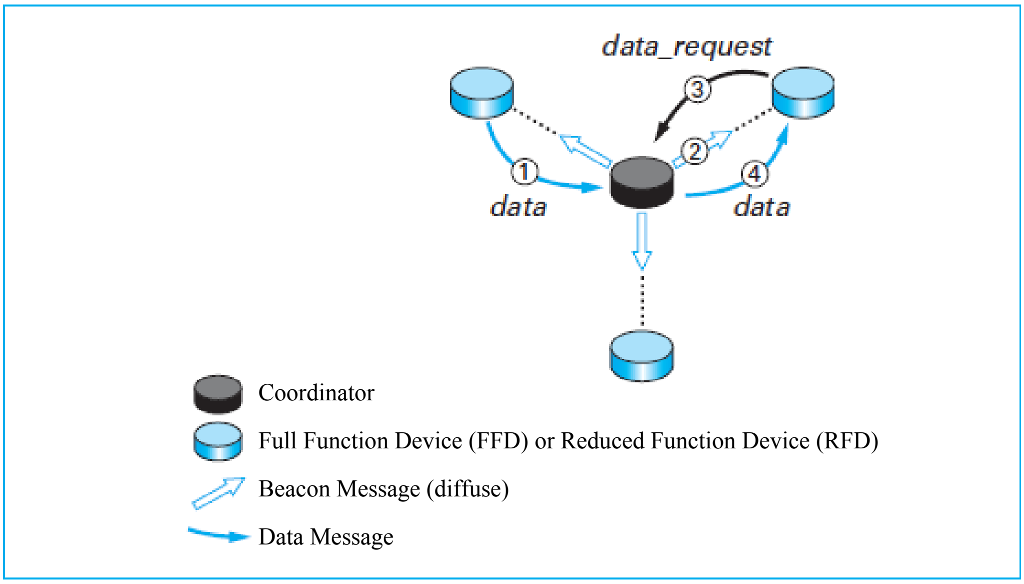


Figure 2-5. Principal of Data Transmission in Beacon Mode with Star Topology [VCV 08]

Figure 2-5 illustrates the principal of data transimission organized by the coordinator in the network with the star topology.

- Direct data transmission is shown by the delivery of the data message (1). In this case, the node sends directly its data to the coordinator; after the transmission, the node turns to sleep state.

- Indirect data transmission is dedicated for the nodes who want to retrieve the data. For example, the beacon message (2) announces all the nodes: the data message is pending. The node demanding the data periodically listens to the network beacon. When this destination node hears the beacon, it reclaims the data by sending the data request message (3) to the coordinator. Then, the coordinator transmits the pending data message (4).

In the beacon mode, the time space between two beacons is called superframe. A superframe is always equally divided into 16 time slots, and the beacon always occupies the first slot. The beacon is used to synchronize the attached nodes, to identify the PAN, and to describe the structure of the superframes. The structure of superframe is presented in Figure 2-6.

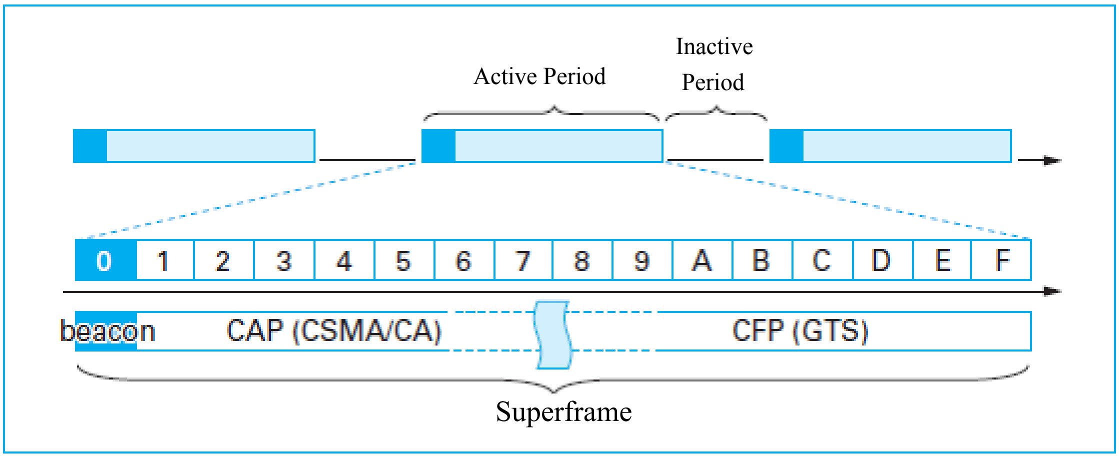


Figure 2-6. Structure of Superframe [VCV 08]

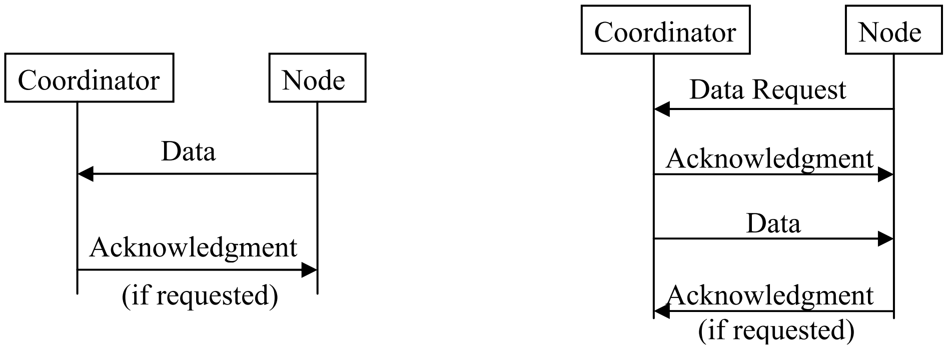
Shown in Figure 2-6, the superframe can have an active portion and an optional inactive portion. During the inactive portion, the coordinator may enter a low-power state. Beside the beacon, the active portion has two parts: CAP (Contention Access Period) and CFP (Contention Free Period). During CAP, any node wishing to communicate has to compete with other nodes using a slotted CSMA/CA medium access mechanism. The detailed of this access method will be introduced later. As an optinal part, the CFP appears at the end of the active portion of superframe, dedicated for low-latency applications or applications requiring specific data bandwidth. The CFP is composed of Guaranteed Time Slots (GTSs). The PAN coordinator may allocate up to seven of these GTSs, and a GTS may occupy more than one slot period. More information on the beacon mode can be referred to the work [VAN 07] about the star topology, or the work [FRA 08] about the tree topology.

Compared with the beacon mode, the non-beacon mode is simpler, where the coordinator does not send out a beacon. It should be noted that transmission of beacons will put extra payload on the network as well as consume more power. With intention of saving power and bandwidth, non-beacon mode is suggested in this thesis.

In the non-beacon PAN of mesh topology, every node can directly communicate with other nodes in its radio sphere. However, in the star topology, the communication should always pass by the coordinator.

In the non-beacon PAN of star topology, when a node wishes to transfer data, it directly transmits its data frame to the coordinator using non-slotted CSMA/CA medium access method. (The detail of this acces method will be introduced later.) Then, the coordinator acknowledges the successful reception of the data by transmitting an optional acknowledgment frame. This direct data transmission is shown in Figure 2-7(a).

When a coordinator wishes to transfer data to a node in a non-beacon PAN of star topology, the coordinator waits for the request of the node. The node may contact with the coordinator by transmitting a data request, using non-slotted CSMA/CA access method. The coordinator acknowledges the successful reception of the request by transmitting an acknowledgment frame. If the data is ready, the coordinator transmits the data frame to the node, using unslotted CSMA-CA. Finally, the node acknowledges the successful reception of the data frame by transmitting an optional acknowledgment frame. The procedure of transmission is summarized in Figure 2-7 (b).



(a) From Node to Coordinator (b) From Coordinator to Node

Figure 2-7. Principal of Data Transmission in NonBeacon Mode with Star Topology

Here, the acknowledgment (ACK) frame is used for confirming successful frame reception. When the acknowledgment is not required, the originating node shall assume that the transmission of the frame was successful. If the originator who had requested an ACK does not receive an acknowledgment after some period (denoted as *macAckWaitDuration* in IEEE 802.15.4), the originator assumes that the transmission was unsuccessful and retries the frame transmission. If an acknowledgment is still not received after several retries, the originator can choose either to terminate the transaction or to try again. The maximum number of retransmissions allowed is denoted as *acMaxFrameRetries* in IEEE 802.15.4.

The channel access mechanism supported by the IEEE 802.15.4 MAC layer is Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Depending on the network configuration, the IEEE 802.15.4 LR-WPAN (Low Rate-Wireless Personal Area Network) uses two types of CSMA/CA: slotted and non-slotted. (Acknowledgment and beacon frames are sent without using a CSMA/CA mechanism.) Each device shall maintain two variables for each transmission attempt: Number of Backoff (*NB*), and Backoff Exponent (*BE)*. *NB* is the number of times the CSMA/CA algorithm was required to backoff while attempting the current transmission. It is initialized to 0 before every new transmission. *BE* is the backoff exponent, which is related to how many backoff periods a device shall wait before attempting to assess the channel. *BE* can be used to reduce the frame collisions.

Beacon-enabled PANs use a slotted CSMA/CA channel access mechanism, where the backoff slots are aligned with the start of the beacon transmission. The backoff slots of all devices within one PAN are aligned to the PAN coordinator. Each time a device wishes to transmit data frames, it locates the boundary of the next backoff slot and then waits for a random number of backoff slots. If the channel is busy, following this random backoff, the device waits for another random number of backoff slots before trying to access the channel again. If the channel is idle, the device begins transmitting on the next available backoff slot boundary. IEEE standard 802.15.4 also supports a "Battery Life Extension" (BLE) mode, in which the CSMA/CA backoff exponent is limited to the range 0-2. This BLE mode can help to reduce receiver power consumption.

Non-beacon-enabled PANs use a non-slotted CSMA/CA channel access mechanism. In non-slotted CSMA/CA, the backoff periods of one device do not need to be synchronized to the backoff periods of another device. Before transmission, a node first waits for a random number (between 0 and 2^{BE}-1) of unit periods. Then, it performs Clear Channel Assessment (CCA) to sense the allocated channel to ascertain its availability. If the channel is found to be idle, the node transmits its data. If the sensor node detects the allocated channel is occupied, it delays the transmission and *NB* controls the times of planed CCA operation. If the node still cannot access the channel when the value of *NB* get to its upper threshold (which is 4 in default), it will declare a transmission failure and discard the waiting frame, resulting in data loss. Figure 2-8 illustrates the algorithm of non-slotted CSMA/CA.

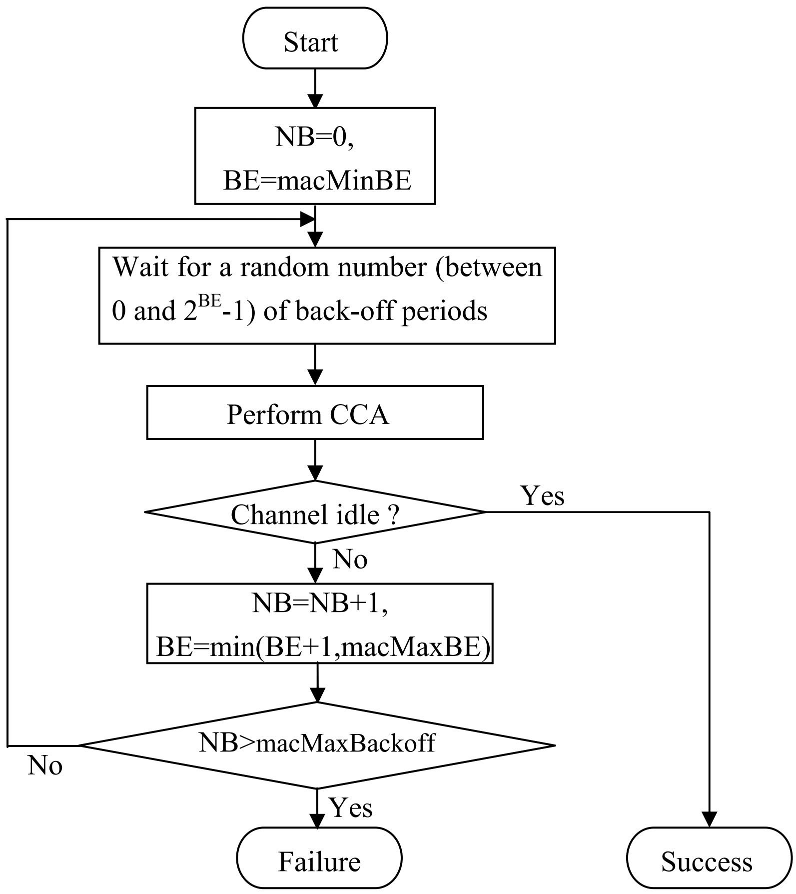


Figure 2-8. Non-slotted CSMA/CA Algorithm

2.1.4 Brief Summary of the Standards and Technologies

Table 2-4 summarizes the main differences among the three types of standards and the corresponding technologies.

From the comparison, we can note that:

(1) The IEEE standard 802.15.4 is expected to provide low cost and low power connectivity for equipments. This has drawn great interests by lots of wireless sensor network applications. Because the sensor nodes need to operate in low power, so that their batteries can last as long as several months even to several years.

(2) Meanwhile, many applications, such as smart home and health monitoring, do not require data rates as high as those enabled by Bluetooth or WiFi.

(3) In addition, IEEE standard 802.15.4 with its technology ZigBee can be implemented in mesh networks. This kind of ad-hoc topology has its advantages on reliability, flexibility, and scalability.

As a result, the IEEE standard 802.15.4 is the choice in this thesis. In this standard, non-beacon mode is always simpler and more cost-efficient than beacon-mode. Therefore, non-beacon mode, with its corresponding medium access method non-slotted CSMA/CA, is chosen to support our proposals.

Table 2-4. Comparison of IEEE 802.11, 802.15.1, and 802.15.4

IEEE Standard	802.11 a/b/g/n/ac	802.15.1	802.15.4
Technology	WiFi	Bluetooth	ZigBee
Frequency Band	2.4GHz; 5GHz	2.4GHz	868/915MHz; 2.4GHz
Maximum Data Rate	11; 54; 6900 Mbps	0.72; 24 Mbps	250kbps
Typical Range	50-100m	10-100m	10-100m
Transmission Power	15 - 20 dBm	0 - 20 dBm	(-25) - 0 dBm
Channel Bandwidth	22MHz	1MHz	0.3/0.6MHz; 2MHz
Spreading	DSSS, OFDM	FHSS	DSSS
Topology	IBSS, ESS	Piconet, Scatternet	Star, Tree, Mesh
MAC Layer Protocol	CSMA/CA with Exponential Backoff	TDD	slotted CSMA/CA, slotted CSMA/CA with BLE non-slotted CSMA/CA

2.2 Localization Algorithms and Techniques

Localization in wireless sensor networks has attracted a lot of research efforts in recent years [BT 05] [CHA 06] [MFA 07] [AK 09] [LYW 10]. It is commonly agreed that GPS [HWL 97] is not an excellent solution for sensor network applications, because of its expensive cost, high energy consumption, and rigid deployment constraints [BFA 05] [BT 05] [AK 09] [LYW 10] [BHE 00]. As a result, researchers have continued investigating innovative ideas to realize practical, inexpensive, flexible and robust localization in wireless sensor networks.

Most of the proposed localization solutions for WSN can be generally categorized into two categories: (a) range-based and (b) range-free. Their major difference lies in whether ranging information are required at sensor nodes in the network. In the following, we give a survey about the range-based and range-free localization techniques in Section 2.2.1 and Section 2.2.2 respectively.

2.2.1 Range-Based Localization

The methodology of range-based localization depends on accurate ranging results among sensor nodes. These ranging results include point-to-point distance, angle, or velocity relative measurements. After obtaining ranging results, the positions of sensor nodes can be estimated through geographical calculations such as trilateration (shown in Figure 2-9) [SPS 02] [MLR 04] [BT 05] [YL 10] or triangulation [EGH 99] [SRL 02] [BP 00] [XRS 10].

In Figure 2-9, three nodes (A_1, A_2, and A_3) already know their positions, called anchors (or beacon nodes). The Mobile node M is the node we want to localize. It is assumed that we know the position (x_i, y_i) of each anchor A_i as well as the distance d_i between M and A_i. The relationship between M and each anchor A_i can be written as Equation (2.1). The position of M is unknown, labeled as (x, y). The principle of trilateration is to find the position of M by solving this equation.

$$\begin{cases}(x-x_1)^2+(y-y_1)^2=d_1^{\ 2}\\(x-x_2)^2+(y-y_2)^2=d_2^{\ 2}\\(x-x_3)^2+(y-y_3)^2=d_3^{\ 2}\end{cases} \qquad \textbf{(2.1)}$$

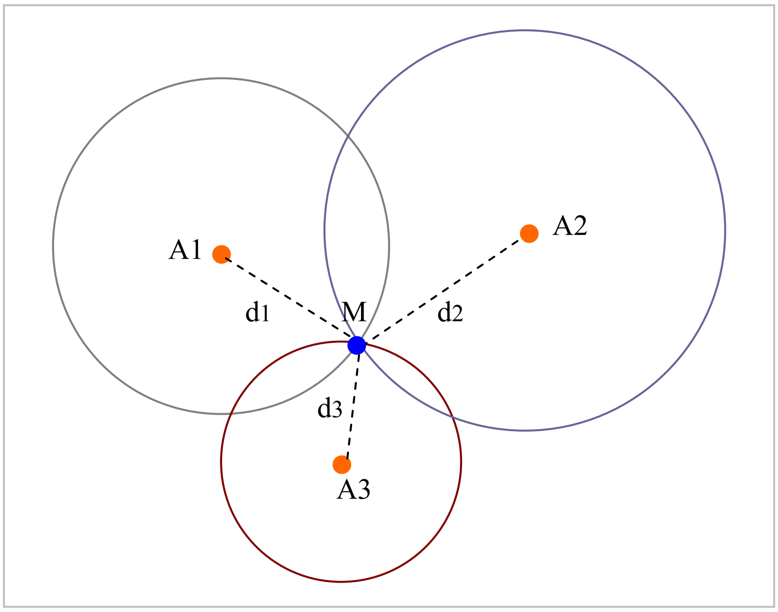


Figure 2-9. Trilateration

The principle of triangulation will be presented in the subsection 2.2.1.3.

In the following subsections, we explain range-based methods from the perspective of three types of elementary ranging information, including (i) received signal strength indicator, (ii) time of flight, and (iii) angle of arrival.

2.2.1.1 Received Signal Strength Indicator

Radio Signal Strength Indicator (RSSI) is considered as the most popular modality for range estimation in wireless sensor networks. That is because, almost every node in the market has the ability to analyze the strength of a received message [CHR 05], then RSSI information can be obtained at almost no additional cost [ELM 04] [WKC 07].

In order to effectively utilize RSSI for localization, two types of methods have been studied: (1) directly calculation of distance; (2) RSSI fingerprinting. In the following, we introduce basic ideas for the above two types of methods.

2.2.1.1.1 Direct Calculation of Distance

The intensity of an emitted signal decreases as the distance from the emission source increases. This decrease relative to the original intensity is the attenuation [GAR 07]. The signal strength decays with respect to distance in a polynomial manner. In the most ideal circumstances, signal power attenuation is proportional to d^2, where d denotes the distance between the transmitter and the receiver. This effect is sometimes referred to as free space loss [ZN 05].

Given a function correlating attenuation and distance, it is possible to estimate the distance between two nodes by measuring the strength of the signal. The widely used radio propagation model is the log-distance path loss model (without multipath effects):

$$RSSI(d)\ [\text{dBm}] = RSSI_{\text{ref}} - 10n\log_{10}\left(\frac{d}{d_{\text{ref}}}\right) \quad \textbf{(2.2)}$$

In Equation (2.2), *RSSI* is measured in dBm, which is a logarithmic measurement of signal strength. d is the distance between emitter and receiver. $RSSI_{\text{ref}}$ the signal strength value at reference distance d_{ref}. n is the attenuation constant (rate at which the signal decays). Usually, n is obtained through empirical data. n is around 2 in a free-space environment, but its value increases if the environment is more complex (walls, large metallic objects, etc.). In environments with many obstructions such as an indoor office space, an approximation of n is between 3 to 6 [SSS 01].

Based on Equation (2.2), a commonly used model for calculating the distance d is given in Equation (2.3), in which $RSSI_{\text{ref}}$ is measured at d_{ref} = 1 m.

$$d = 10^{\frac{RSSI_{\text{ref}} - RSSI}{10n}} \quad \textbf{(2.3)}$$

After obtaining this distance, the positions of sensor nodes can be estimated through trilateration.

Note that the RSSI value does not only depend on the distance, but also on the environment, antenna orientation, the movement of the emitter and the receiver, and the power supply [AWP 06]. This means, the RSSI information can be unpredictable [ZHK 04] [GKW 02] [SDT 06] [MZF 08] [FAL 09], because the reflecting and attenuating caused by objects in the environment can have much larger effects on RSSI than distance. Therefore, it is difficult to obtain accurate distance from RSSI without a detailed model of the physical environment [BP 00] [ELM 04] [WKW 05] [WKC 07]. For example, in the article [PSG 12], the authors do some experiments on localization using RSSI in open space. Their results show that, with the communication range about 25m, the location error is mainly between 3.7m and 4.5m.

2.2.1.1.2 RSSI Fingerprinting

Motivated by the fact that direct distance estimation from RSSI is found to be inaccurate in the indoor scenario, an alternative solution using RSSI for positioning is called RSSI fingerprinting [WKP 07] [DVB 11].

This method comprises two steps.

In the first step or learning step, beacon nodes (or anchors, they already know their positions) record the power level of frames sent periodically by the mobile nodes. These frames contain the current position and orientation of the mobile nodes. During this offline procedure, the beacon nodes map each frame with the measured RSSI and the time of reception. Since all the nodes have been synchronized, the time values are valid throughout the network. The first step is also qualified as off-line phase since it is usually performed before the activation of the localization service provided by the network.

When the offline phase ends, a database is built, containing each mobile node's position and orientation (north, south, east, and west), as well as the RSSI measurements taken by each beacon.

In the second step or online step, the beacon nodes use the empirical values contained in the database to determine the matched position for mobile nodes.

The disadvantages of the fingerprinting method are mainly its cost in terms of setup time and its demand of high data volume. Furthermore, any change in the configuration such as the coming of a new beacon node, will imply creating a new database, meaning this method is not so flexible.

2.2.1.2 Time of Flight

The distance measurement using RSSI is usually less accurate than using Time-of-Flight (TOF), especially in the environments with many obstructions such as an indoor office space [HB 01].

The principle of direct distance calculation based on Time-of-Flight (TOF) is shown in Figure 2-10. In the figure, t_a is the signal transmission instant from the beacon node, while t_b is the signal reception instant at the mobile node. Considering the fact that the nodes share a common clock (synchronized), the TOF is calculated as $|t_a - t_b|$. Since the speed of signal propagation know as c, then the distance between the nodes can be derived as $c \times |t_a - t_b|$.

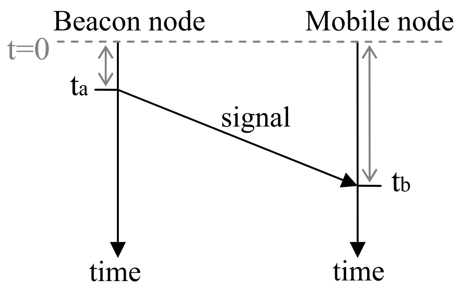


Figure 2-10. Distance Calculation of TOF in Synchronized Network

However, the above measurement of TOF demands that the two nodes should be perfect-synchronized, which is not practical in the real networks. In order to reduce the impact of synchronization, the IEEE standard 802.15.4a support a two-way ranging method, as shown in Figure 2-11.

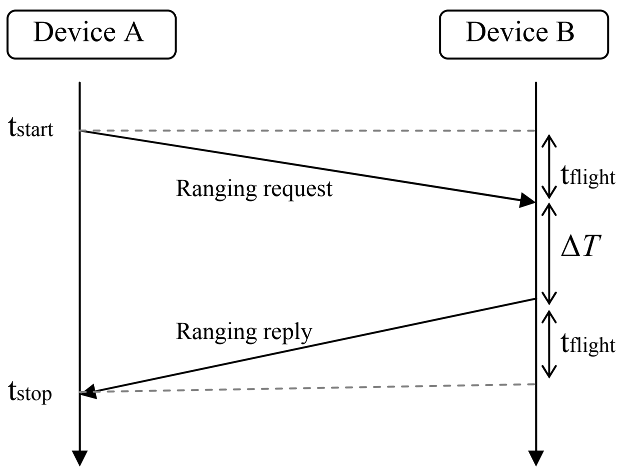


Figure 2-11. Example of Two-Way Ranging

The two-way ranging method requires the two devices to exchange at least two packets. In Figure 2-11, the device A starts a ranging measurement by sending a ranging packet to device B at time t_{start}.

Device B receives the packet from A, and replies with a second ranging packet, transmitted after a delay ΔT. The packet is received by the device A at time t_{stop}. Then the propagation time from A to B, denoted as t_{flight}, can be calculated as Equation (2.4).

$$t_{flight} = \frac{t_{stop} - t_{start} - \Delta T}{2} \quad \textbf{(2.4)}$$

Even if the two-way ranging method is used to reduce the influence of node synchronization, the localization techniques based on TOF still have the problem of node timing resolution. In order to obtain the precise measurement of TOF, sensor nodes need to have ultra-resolution timing ability. This requires the nodes to be equipped with specially designed radio chips [MDF 00] [LLP 06] or high speed clocks and processors [FG 02] [LS 02] [KR 06] [YYR 06].

Two mostly used technologies for TOF measurements are Ultra Wide Band (UWB) [WS 98] and Direct Sequence Spread Spectrum (DSSS) [KAP 96], both of which are wide-band or ultra-wideband signals with enhanced time domain resolution.

The DSSS signal has been used in ranging systems for many years (e.g., the GPS). In such a system, to be more robust to the noise and interference, a signal coded by a pseudo-noise (PN) sequence is transmitted by an emitter. Then at the receiver, the received signal needs also to be decoded with a local PN sequence [RAP 01]. The arrival time of the received signal is used to calculate the distance between the emitter and the receiver. The resolution of TOF estimation in DSSS ranging systems is mainly determined by the signal bandwidth. For example, if a bandwidth of 100 MHz is used, distance estimation errors are about or less than 3 meters under ideal LOS (line of sight) conditions [PLM 02]. However, this requires that the PN generator at the sender has a time resolution of at least 1/100MHz=10 ns. In other words, high speed clocks are required in the system.

Note that signal bandwidth is one of the key factors that affect TOF estimation, the wider the bandwidth, the higher the ranging accuracy. Ultra Wide Band (UWB) systems, that employ bandwidths more than 1 GHz, have attracted considerable attention, especially for indoor localizations [LS 02]. It has been demonstrated that the UWB signal is not seriously affected by multi-path fading [LS 02]. However, integrating the UWB systems on sensor nodes is quite challenging, because they demand sophisticated hardware to provide swift sampling and precise timing. Thus, we cannot find many UWB based products for wireless sensor networks.

2.2.1.3 Angle of Arrival

Except the above two localization techniques using distance estimates, there is another technique that utilizes the angle information called Angle of Arrival (AOA).

To perform localization with AOA, two angle measurements are required, as shown in Figure 2-12. The signal sending from the mobile node M is received by anchor A_1 and anchor A_2. The antenna array of A_1 can detect the signal's AOA denoted as α, while A_2 can measure the AOA as β. Then the two anchors send to M the angle information α and β as well as their positions (x_1, y_1) and (x_2, y_2). From the positions of anchors, M can calculate the distance between anchors, denoted as d.

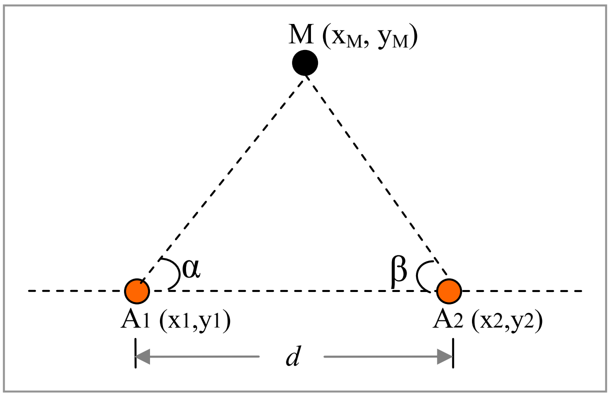


Figure 2-12. Example of Localization Using Angle of Arrival

Finally, M estimates its position (x_M, y_M) through the triangulation approach by solving the following equation. (For simplicity, assume that A_1 and A_2 are on x-axis, that means, $y_1 = y_2$.)

$$\begin{cases} x_M = x_1 + \dfrac{d \times \sin\alpha \times \sin\beta}{\sin(\alpha+\beta)} \\ y_M = \dfrac{d \times \cos\alpha \times \sin\beta}{\sin(\alpha+\beta)} \end{cases} \quad \textbf{(2.5)}$$

Solving the equation (2.5) is simple. However, the AOA based localization has two practical problems.

First, it is the use of antenna array for detecting the angles. The AOA data should be obtained by using antenna arrays [SHS 01] [WIN 06], which allow a receiver to determine the direction of a transmitter. Antenna array consists of multiple antennas separated with certain distance. Therefore, it is not practical to implement antenna array on tiny sensor nodes, considering the constraint on sensor nodes in terms of size, cost and energy.

Second, the accuracy of AOA measurements is affected by a combination of factors, including multi-path reflections and background noise. A multi-path reflection of signal, or some noise, may appear as a signal arriving from a totally different direction. When a mobile node receives this faked signal, a wrong AOA will be detected. This shall lead to large errors in angle estimation.

2.2.1.4 Brief Summary of Range-Based Localization

Summarizing the typical range-based localization techniques, Table 2-5 lists their advantages and disadvantages.

Table 2-5. Comparison of Range-Based Localization

<table>
<tr><th colspan="2">Range-based Methods</th><th colspan="2">Advantages</th><th colspan="2">Disadvantages</th></tr>
<tr><td rowspan="2">RSSI</td><td>Direct Calculation</td><td rowspan="2">No additional hardware</td><td>Scalable, Low overhead</td><td colspan="2">Low accuracy</td></tr>
<tr><td>Fingerprinting</td><td>Better accuracy</td><td colspan="2">Non-flexible, memory cost</td></tr>
<tr><td rowspan="2">TOF</td><td>Direct Calculation</td><td rowspan="2">Better accuracy than RSSI</td><td>Low overhead</td><td>Strict synchronization</td><td rowspan="2">Ultra-high timing requirement, expensive hardware</td></tr>
<tr><td>Two-way Ranging</td><td>No rigid synchronization</td><td></td></tr>
<tr><td colspan="2">AOA</td><td colspan="2">Low timing / synchronization requirement,
Better accuracy than RSSI</td><td colspan="2">Hardware constraints,
affected by multipath fading and noise</td></tr>
</table>

RSSI based methods have the lowest cost, because the RSSI information can be obtained without any additional hardware. However, since RSSI information is very easily affected by environment, RSSI based methods have the lowest accuracy, compared with TOF and AOA. Among the RSSI methods, although the fingerprinting has better accuracy than the direct distance calculation, the fingerprinting requires large memory for the database storing offline RSSI measurements.

TOF and AOA based methods can achieve much better accuracy than RSSI. However, they all require additional hardware. The TOF based methods need high-speed clocks to support ultra-high resolution timing, while the AOA based methods demand antenna array to effectively detect the angles. Furthermore, rigid synchronization is necessary for simple TOF methods such as the direct distance calculation method by TOF. And the AOA information is sensitive to multipath fading and noise.

Therefore, normally, the range-based techniques require additional ranging hardware, except the RSSI based methods. However, localization using RSSI has a low accuracy, because the RSSI measurement is unstable and varies greatly with the environment.

2.2.2 Range-Free Localization

While the range-based localization techniques precisely measure the distance or angle between nodes, the range-free schemes use connectivity information. The connectivity information can be the hop count between two sensor nodes, indicating how close the two nodes are. For example, if one node is within the communication range of the other node, the distance between two nodes can be estimated as one hop, and these two nodes can be called as neighbors.

In range-free localization schemes, the nodes that are aware of their positions are called anchors, while others are called normal nodes. In general, anchors are fixed, while normal nodes are mobile. Normal nodes first gather the connectivity information as well as the positions of anchors, and then calculate their own positions. Since no ranging information is needed, the range-free schemes can be implemented on low-cost wireless sensor networks. Another advantage of range-free schemes is their robustness; the connectivity information between nodes is not easily affected by the environment. As a result, we focus our research on the range-free localization.

Many range-free localization algorithms have been proposed for these years, such as Centroid [BHE 00] [RT 06], Convex Position Estimation (CPE) [DPG 01], Approximate Point-In-Triangulation (APIT) [HHB 03], and DV-hop (Distance Vector-hop) based algorithms [NN 03] [LCK 10] [HZL 10]. In this section, these typical range-free algorithms will be introduced and compared.

2.2.2.1 Centroid Algorithm

Centroid algorithm is first proposed by Bulusu [BHE 00]. The basic principle is to regard the centroid point of neighbor anchors as the estimated position of the normal node. The author chooses a simple radio propagation model, which fits quite well for outdoor environment. In this model, there are two assumptions: the first is perfect spherical radio propagation, and the second is identical transmission range for all radios.

The scenario is shown in Figure 2-13. In the network, there are totally m anchors situated at known positions, $A_1 (x_1, y_1)$, $A_2 (x_2, y_2)$ … $A_m (x_m, y_m)$. All these anchors have the same communication range denoted as R. Their transmission areas have an overlap, as shown by the shaded part in the figure. Inside the overlap locates the normal node N_x. That means, all these m anchors are the neighbor anchors of N_x.

The network topology is mesh. Each anchor periodically (period=T) transmit one beacon signal containing its position. It is assumed that all anchors are well synchronized and no collisions occur during the transmissions. For the facility of explanation, before the introduction of the algorithm, the author defines a few terms listed below:

R: Node transmission range

T: Time interval between two beacon signals transmitted by an anchor

t: Normal node N_x uses this amount of time to collect beacon signals, $t > T$

$N_{sent}(i, t)$: Number of beacons sent by anchor A_i in time t

$N_{recv}(i, t)$: Number of beacons received by normal node in time t (beacons are sent by anchor A_i)

CM_i: Connectivity metric for anchor A_i

CM_{thresh}: Threshold for CM

(x_{cen}, y_{cen}): Estimated position of the normal node by the Centroid algorithm

(x_a, y_a): Actual (or real) position of the normal node

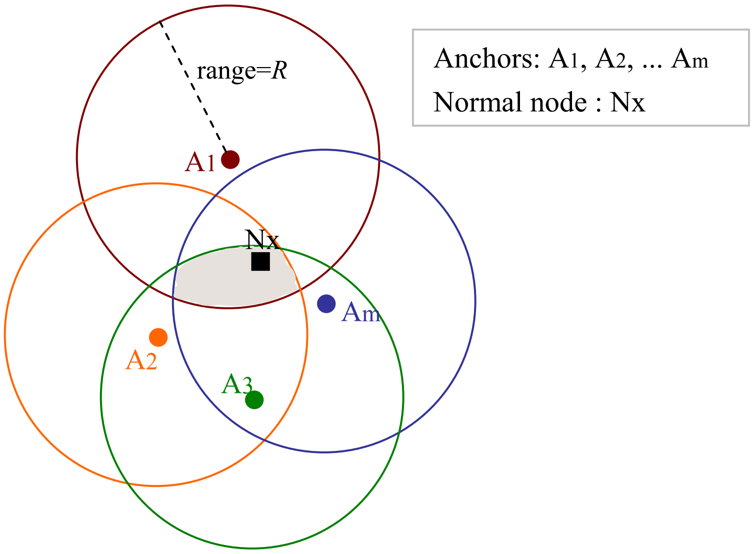


Figure 2-13. Example for Centroid Localization

During the fixed time period t, the normal node N_x listens to the channel and collects all the beacon signals from various anchors. Although each anchor A_i has sent $N_{sent}(i, t)$ signals, because of radio propagation interference, the normal node can actually receive $N_{recv}(i, t)$ signals from A_i (Note that $N_{recv}(i, t) <= N_{sent}(i, t)$).

In order to know whether an anchor is really within the radio range of the normal node, the author defines connectivity metric for each anchor A_i, denoted as CM_i:

$$CM_i = \frac{N_{recv}(i,t)}{N_{sent}(i,t)} \tag{2.6}$$

The author also defines a threshold for CM_i, denoted as CM_{thresh}. If CM_i is larger than CM_{thresh}, the normal node N_x will regard the corresponding anchor A_i is neighbor to N_x. Therefore, when calculating the position, N_x will take A_i into account. However, if CM_j is smaller than CM_{thresh}, N_x will see that the anchor A_j is not in its proximity, then N_x will not consider A_j when estimating its position.

Assume that finally N_x can have k anchors whose connectivity metrics are larger than CM_{thresh}. These k anchors are $A_1, A_2 \ldots, A_k$. Then N_x localizes itself at the centroid of these k anchors:

$$\begin{cases} x_{cen} = (x_1 + x_2 + ... + x_k)/k \\ y_{cen} = (y_1 + y_2 + ... + y_k)/k \end{cases} \quad \textbf{(2.7)}$$

The program procedure of Centroid algorithm is summarized as follows.

```
Algorithm "Centroid":
During a period t, normal node N obtains the positions of k anchors (A1, A2 …, Ak).
xcen ← 0 ;  ycen ← 0
 for i ← 1 to k
     do  xcen ← ( xcen + xi)  ;   ycen ← ( ycen + yi)      where (xi, yi) is the position of Ai
 xcen ← xcen / k;    ycen ← ycen / k
 return xcen and ycen
```

Figure 2-14. Procedure of Centroid Algorithm

In [BHE 00], the author estimates the accuracy of the Centroid algorithm by the metric *location error*, which is defined as:

$$location\ error = \sqrt{(x_{cen} - x_a)^2 + (y_{cen} - y_a)^2} \quad \textbf{(2.8)}$$

The author evaluates the algorithm performance by an experiment in a 10 × 10 m^2 outdoor parking lot. 4 anchors are placed at the different corners. Their radio range is about 8.94 m. They transmit beacon signals containing their positions every 2 seconds (that means, T=2s). CM_{thresh} is set to be 90%. Whenever the normal node moves to a new place inside the parking lot, it keeps static for 41.9s (that is t=41.9s), receiving the beacon signals and finally calculating its position. As a result, the average location error is about 1.83m.

Some improvements have been proposed in these years. However, most of them need to add the RSSI information. For example, a Weighted Centroid Localization (WCL) algorithm is proposed in an article [RT 06]. WCL adds the RSSI information to the original Centroid algorithm, by associating weights to the links between the mobile node and the anchors. The estimated position of the mobile node, (x_{wcl}, y_{wcl}), is calculated as:

$$x_{wcl} = \frac{\sum_{i=1}^{m}(w_i \times x_i)}{\sum_{i=1}^{m} w_i} \quad , \quad y_{wcl} = \frac{\sum_{i=1}^{m}(w_i \times y_i)}{\sum_{i=1}^{m} w_i} \quad \textbf{(2.9)}$$

In Equation (2.9), (x_{i}, y_{i}) is the coordinates of anchor A_{i}, and w_{i} is the weight associated with the link between the mobile node and A_{i}. The value of w_{i} is determined by $RSSI_{\text{i}}$, which is the RSSI value of the received signal (sent from A_{i}, arrive at the mobile node). The authors execute some experiments, and find that the RSSI values detected by the mobile node are always in the range of [–110 dB, –50 dB]. Then, in order to make w_{i} to be positive, w_{i} is calculated as the following equation [RT 06].

$$w_i = \frac{1}{-(RSSI_i + 49)} \quad \textbf{(2.10)}$$

The results have shown that, compared with the original Centroid algorithm, the estimated position of WCL algorithm moved closer to anchors with higher weight.

2.2.2.2 CPE Algorithm

In order to improve the accuracy of Centroid algorithm, the Convex Position Estimation (CPE) algorithm was first proposed [DPG 01] by Doherty et al.

The authors of CPE algorithm first provide an optimization concept, and then the locations of normal nodes in a WSN are found as a result of a joint optimization problem.

Except this abstract mathematic modeling, the authors also give an example. Consider the case shown in Figure 2-15, where the three anchors A_1, A_2, A_3 have the same communication range. The normal node N_x locates inside the overlap of anchors radio transmission.

The principle of CPE algorithm is to find the smallest rectangle (in Figure 2-15) that bounds the overlap, and then to take the center of this rectangle as the estimated position of N_x. Now the problem is how to find the smallest rectangle.

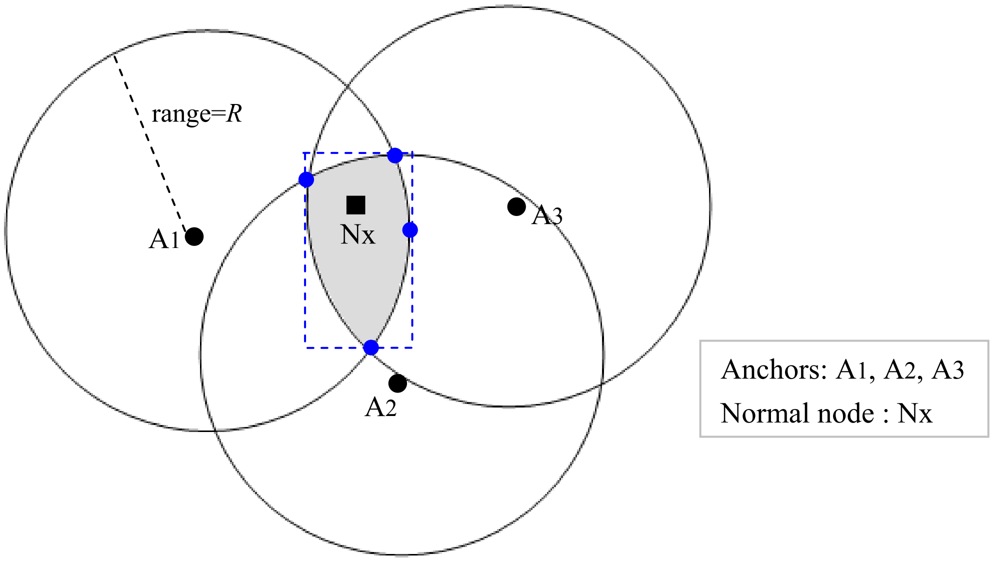
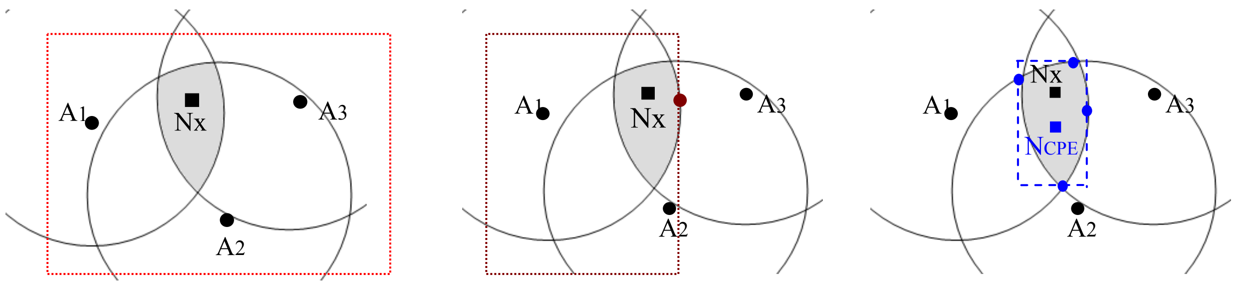


Figure 2-15. Example of CPE Algorithm

The authors propose an abstract optimization model to explain how they find the smallest rectangle. Nevertheless, here, for better understanding, we give an example to explain part of optimization process. As shown in Figure 2-16, the normal node N_x starts with a big rectangle (Figure 2-16 (a)). Then N_x begins to optimize one side, for example, the right size of the rectangle. After large amount of tests and calculations, in Figure 2-16 (b), the exact right side is found, and then N_x continues to look for other sizes. And finally, the smallest rectangle is found, shown in Figure 2-16 (c), and the centre is the estimated position N_{CPE}.



(a) start (b) one side finished (c) final case

Figure 2-16. Example of Process to Find the Smallest Rectangle

The CPE algorithm is a centralized localization scheme, because the resource-limited normal node is unable to do numerous and complex calculations required by optimization process. Therefore, all normal nodes need to first send the collected connectivity information to a centralized controller. The centralized controller then calculates the position of every normal node, and transmits the estimated positions back to the corresponding normal nodes. Thus, the traffic load is heavy. The CPE algorithm scales poorly when the network is large.

However, a simplified and distributed version of CPE algorithm has been proposed by some researchers [SLH 06] [SCH 08]. Unlike the original CPE finding the smallest rectangle, the simplified CPE algorithm defines an Estimated Rectangle (*ER*) which bounds the communication range of anchors, as shown in Figure 2-17. This *ER* is bigger than the smallest rectangle. Its centre point, denoted as N_{ER}, is the estimated position of the simplified CPE algorithm.

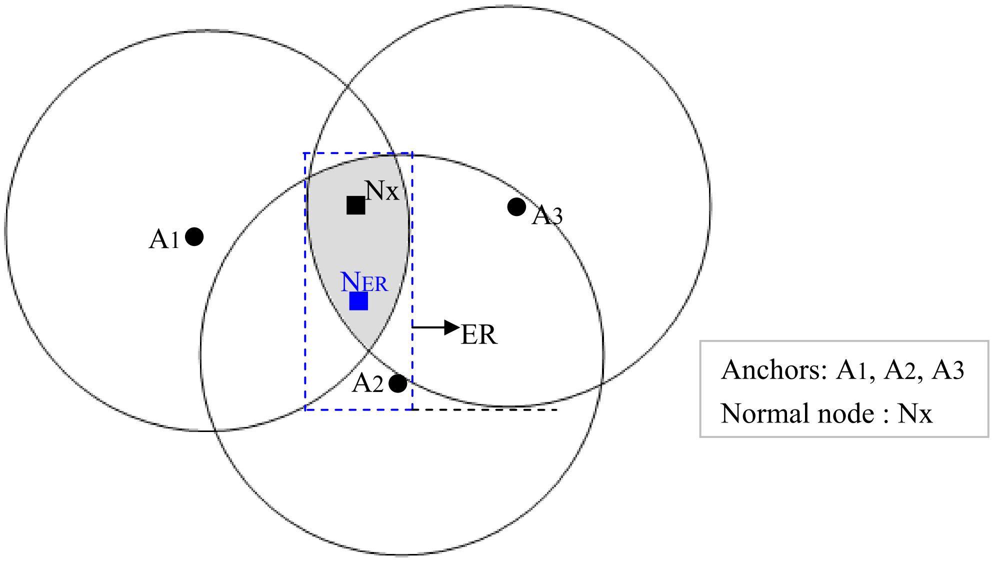


Figure 2-17. Example of a Simplified CPE Algorithm

The principle of this simplified algorithm is as follows. First, the normal node N_x sends location request signals to its neighbour anchor nodes. Second, after receiving the request, the neighbour anchors A_1, A_2, A_3 immediately send the agree response as well as their coordinates (x_1, y_1), (x_2, y_2), (x_3, y_3) to N_x. Third, N_x calculates its position N_{ER} (x_{ER}, y_{ER}) as the centre of *ER*:

$$x_{ER} = \frac{\min_i x_i + \max_i x_i}{2}, \quad y_{ER} = \frac{\min_i y_i + \max_i y_i}{2} \tag{2.11}$$

Then, the program procedure of simplified CPE algorithm can be described as:

Algorithm "simplified CPE":
suppose the normal node N_x has m neighbor anchors $A_1, A_2 \ldots, A_m$. The position of A_i is (x_i, y_i).
$x_{max} \leftarrow x_1$; $x_{min} \leftarrow x_1$; $y_{max} \leftarrow y_1$; $y_{min} \leftarrow y_1$;
for $i \leftarrow 2$ **to** m
    if $x_i < x_{min}$ **then** $x_{min} \leftarrow x_i$; **elseif** $x_i > x_{max}$ **then** $x_{max} \leftarrow x_i$
    if $y_i < y_{min}$ **then** $y_{min} \leftarrow y_i$; **elseif** $y_i > y_{max}$ **then** $y_{max} \leftarrow y_i$
$x_{ER} \leftarrow (x_{min} + x_{max})/2$; $y_{ER} \leftarrow (y_{min} + y_{max})/2$
return x_{ER} and y_{ER}

Figure 2-18. Procedure of simplified CPE algorithm

2.2.2.3 APIT Algorithm

Unlike the Centroid and CPE algorithms that assume the node communication range as circle, Approximate Point-in-Triangulation (APIT) algorithm doesn't have this ideal assumption.

Assume that in a network there are 5 anchors in total, A_1, A_2, A_3, A_4, and A_5. As shown in Figure 2-19, the solid circles are anchors, while the hollow circles are normal nodes. The concerned normal node is marked as N_x. The basic principle of APIT algorithm is: let us assume that the normal node N_x is aware of the positions of anchors, and then N_x can form triangles using any three anchors, as shown in Figure 2-20. If N_x can determine whether it is inside or outside of these triangles, the overlap of the triangles (N_x inside) is where N_x resides. The detailed principle is presented in the following.

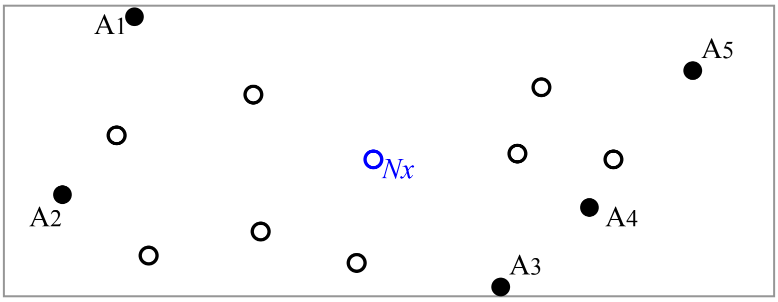


Figure 2-19. a Network Example (APIT Algorithm)

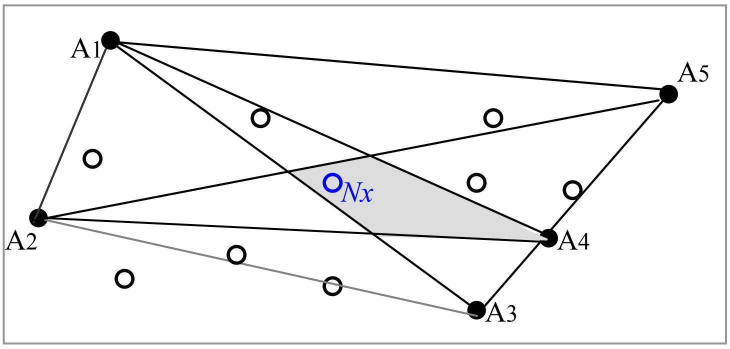


Figure 2-20. Triangles Formed by Any Three Anchors

APIT algorithm consists of four steps: (1) beacon exchange, (2) Point-in-Triangulation (PIT) test, (3) position calculation. Next, we describe each step. Since the second step is the most important, the emphasis will be placed on the second step.

(1) Beacon exchange: the anchors periodically broadcast beacon signals (containing their positions) to its neighbor nodes. In this algorithm, it is necessary for each anchor to equip with a powerful transceiver, so that its signal can be received by all normal nodes in the network. Receiving the signal from an anchor A_i, each normal node detects and notes down the received signal's RSSI value, as well as the position of A_i. The RSSI information is used to estimate whether a node is inside a triangle formed by three anchors in the PIT testing step.

(2) The Point-in-Triangulation (PIT) test is performed to determine whether a normal node N_x is inside a triangle formed by three anchors.

The Perfect PIT test can be performed by moving N_x along any direction, as shown in Figure 2-21. In Figure 2-21 (a), N_x moves in every possible direction, and compares its distance to anchors with the distance before moving. The distance is measured based on RSSI. After moving a tiny step toward every direction, N_x finds that its distance to the three anchors never increase or decrease simultaneously. For example, when N_x moves a little to A_1, its distance to A_1 decreases, but its distances to A_2 and A_4 both increase. Thus, N_x is judged to be inside the triangle $\Delta A_1A_2A_4$. On the contrary, N_x will be judged outside a triangle if there exists a direction such that when N_x is moved a little, its distances to the three vertexes of the triangle increase or decrease simultaneously. For example, in Figure 2-21 (b), when N_x moves a little, its distances to three anchors decrease simultaneously. Therefore, N_x is outside the triangle $\Delta A_1A_2A_3$.

(a) Inside the triangle $\Delta A_1A_2A_4$ (b) Outside the triangle $\Delta A_1A_2A_4$

Figure 2-21. Perfect PIT Test

In terms of implementation, the Perfect PIT test has two problems. First, it is impossible to test all directions, because there are infinite directions around the normal node *N*. Second, the Perfect PIT test requires that normal nodes can move, however, normal nodes may be fixed in some applications.

Therefore, instead of Perfect PIT, an Approximate PIT (APIT) test is performed. The APIT test assumes that normal nodes are static. Although normal nodes cannot move, the APIT method imagines that they could move, and regards their neighbor nodes as their positions after moving. For example, as shown in Figure 2-22, N_x has three neighbor normal nodes *T*, *U* and *V*. Like N_x, These three nodes have also received signals sent from anchors, and noted down the corresponding RSSI values. N_x can communicate with its neighbors, and obtain their RSSI values. Although here the RSSI values are not used to calculate the exact distance, the difference between the RSSI values of two nodes is used to determine whether a node is farther to an anchor than the other node.

Let us consider the triangle $\Delta A_1A_3A_4$. In order to determine whether N_x is inside the triangle, the Perfect PIT test controls *N* to move a very tiny step and then observes the change of its distances to anchors. However, here, in APIT test, the static N_x virtually moves to its three neighbors *T*, *U*, and *V*. Instead of tiny moves in Perfect PIT test, here, the big moves to neighbors sometimes can cause test errors, which will be discussed later on. Among the three nodes (*T*, *U*, and *V*), N_x checks whether there is one node that is farther from A_1 A_3 and A_4 simultaneously. N_x compares its RSSI value to A_1 with *T*'s RSSI value to A_1. Normally (that means, if RSSI values are relatively stable, not much influenced by the environment), *T* is closer to A_1 than N_x. In the same manner, it can be tested that *T* is farther to A_3 and A_4 than N_x. So, compared with N_x, *T* is not farther from A_1 A_3 and A_4 simultaneously. As for *U*, the same phenomenon can be observed. If N_x had only two neighbor nodes *T* and *U*, then through this APIT test, N_x could have determined that it was inside $\Delta A_1A_3A_4$. However, in reality, N_x has the third neighbor *V*. Unfortunately, compared with N_x, *V* is farther from A_1 A_3 and A_4 simultaneously. Thus, by the APIT test, finally, N_x will judge itself to be outside of the triangle, although it is actually inside the triangle. This test error is caused by the big virtual moves in APIT test. As shown in Figure 2-22, if *V* was *V'*, then N_x could have determined to be inside the triangle.

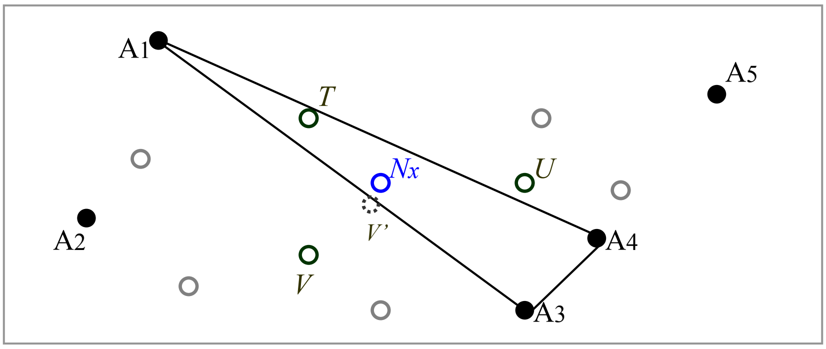


Figure 2-22. Example for APIT test

(3) Position calculation: An overlap is formed by the triangles inside which the normal node N_x locates. Then, the centre of this overlap is calculated as the position of N_x.

The APIT algorithm may achieve good accuracy. However, it requires anchors have high power transmitters. And the APIT test can sometimes cause serious errors. Further more, the RSSI is necessary in this algorithm, although the RSSI is usually not stable. Considering these disadvantages, the APIT algorithm is rarely practiced for localization.

2.2.2.4 DV-hop Based Algorithms

The above three algorithms (Centroid, CPE and APIT) all require a normal node have at least 3 neighbor anchors. Nevertheless, in practical scenarios, the number of normal nodes is always more than that of anchors. Normal nodes may have less than 3 neighbor anchors, or even no neighbor anchors. For example, in a small network topology shown by Figure 2-23, there are 3 anchors and 4 normal nodes. The connectivity between nodes is displayed by lines of dashes. We can see that the concerned normal node N_x has no neighbor anchor, while the other normal nodes all have only one neighbor anchors. None of the algorithms already introduced in previous subsections (including the range-based schemes) can localize these normal nodes.

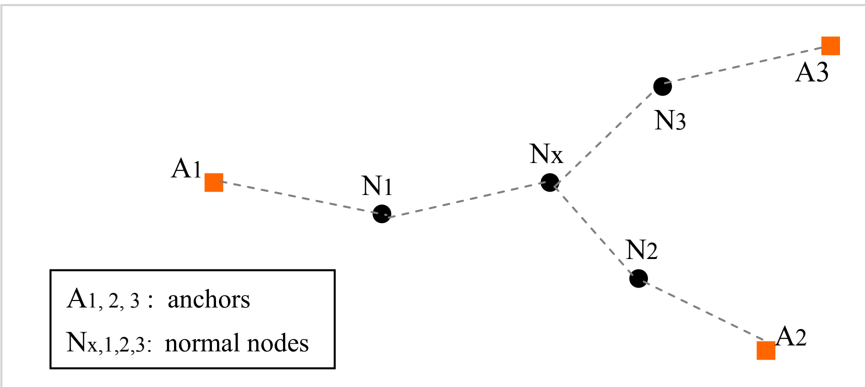


Figure 2-23. Example of Network Topology

However, this tough task can be fulfilled by Distance Vector - hop (DV-hop) algorithm. In the following, this algorithm together with its typical improved solutions will be introduced.

2.2.2.4.1 DV-hop Algorithm

The DV-hop localization algorithm was proposed by Niculescu [NN 03]. It is a suitable solution for normal nodes having few neighbour anchors. As shown in Figure 2-23, although the normal node

N_x has no neighbour anchors, N_x can use the DV-hop algorithm for localization. The algorithm consists of the following three steps.

Step #1: First, each anchor A_i broadcasts through the network a message containing the position of A_i and a hop count field initialized as 0. This hop count value will increase with augment of hop during the broadcast of the message. That means, as soon as this message is received by a node, the hop count value in the message will be incremented. On the first reception of the message, every node K (here, K can be either anchor or normal node) records the position of A_i, and initializes $hop_{i,K}$ as the hop count value in the message. Here, $hop_{i,K}$ is the minimum hop count between K and A_i. If the same message is received again, K maintains $hop_{i,K}$. If the received message contains a lower hop count value than $hop_{i,K}$, K will update hop_i with that lower hop count value, and relay the message. Otherwise, K will ignore the message. Through this mechanism, all the nodes in the network can get the minimum hop count to each anchor.

For example, shown in Figure 2-23, at Step #1, the anchor A_1 broadcasts a message carrying its position (x_1, y_1) and a hop count field initialized as 0. Upon receiving this message, the normal node N_1 first increases the hop count field by 1. Thus, right now, the message is renewed by N_1, and in the message, the value of the hop count field is 1. Then, N_1 records the information in the message in its database, noting down A_1's position (x_1, y_1) and setting $hop_{1,N1}$ (here, $hop_{1,N1}$ is N_1's hop count to A_1) to be 1. After this, N_1 broadcast the renewed message.

Then, N_1's message is received by N_x. N_x does the same process as N_1, including renewing the message, recording the message information into N_x's database, and relaying the message. As a result, N_x can know its hop count to A_1 (denoted as $hop_{1,Nx}$) as 2. In the same manner, N_x can be aware of it hop counts to other anchors, as well as other anchors' positions.

Step #2: Since each anchor A_i has received at Step #1 the positions of other anchors as well as its minimum hop counts to other anchors, now A_i can calculate its average distance per hop, denoted as dph_i. Once dph_i is calculated, it will be broadcasted through the network by A_i.

Here, we give an example on the calculation of dph_1, which is the average distance per hop of A_1 (in Figure 2-23). After Step #1, the anchor A_1 can obtain the positions of A_2 and A_3 denoted as (x_2, y_2) and (x_3, y_3), as well as its minimum hop count to A_2 and A_3 denoted as $hop_{1,2}$ and $hop_{1,3}$. Then, at Step #2, A_1 first calculates its distances to A_2 and A_3, denoted as $d_{1,2}$ and $d_{1,3}$ respectively:

$$\begin{aligned} d_{1,2} &= \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \ , \\ d_{1,3} &= \sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2} \end{aligned} \tag{2.12}$$

Then, the average distance per hop of A_1, that is dph_1, can be calculated as Equation (2.13). The average distance per hop of other anchors can be obtained in the same manner.

$$dph_1 = \frac{d_{1,2} + d_{1,3}}{hop_{1,2} + hop_{1,3}} \tag{2.13}$$

Step #3: when receiving dph_i, the normal node N_x multiplies $hop_{i,Nx}$ (its hop count to A_i) by dph_i, so that N_x obtains its approximate distance to each anchor A_i, denoted as $d_{i,Nx}$. Here, $i \in \{1, 2\ldots m_d\}$, if we assume that there are totally m_d anchors. Thus, the following equation can be derived, where (x', y') is the estimated position of N_x:

$$\begin{cases} (x'-x_1)^2+(y'-y_1)^2=d_{1,Nx}^2 \\ (x'-x_2)^2+(y'-y_2)^2=d_{2,Nx}^2 \\ \vdots \\ (x'-x_{m_d})^2+(y'-y_{m_d})^2=d_{m_d,Nx}^2 \end{cases} \quad \textbf{(2.14)}$$

Equation (2.14) has m_d quadratic equations. For the simplicity of computation, we can transform the quadratic equations to linear equations. Making each quadratic equation subtracted from the last quadratic equation, the equation (2.15) is turned into:

$$\begin{cases} 2(x_{m_d}-x_1)x' + 2(y_{m_d}-y_1)y' = d_{1,Nx}^2-d_{m_d,Nx}^2-x_1^2+x_{m_d}^2-y_1^2+y_{m_d}^2 \\ 2(x_{m_d}-x_2)x' + 2(y_{m_d}-y_2)y' = d_{2,Nx}^2-d_{m_d,Nx}^2-x_2^2+x_{m_d}^2-y_2^2+y_{m_d}^2 \\ \vdots \\ 2(x_{m_d-1}-x_2)x'+2(y_{m_d-1}-y_2)y' = d_{m_d-1,Nx}^2-d_{m_d,Nx}^2-x_{m_d-1}^2+x_{m_d}^2-y_{m_d-1}^2+y_{m_d}^2 \end{cases} \quad \textbf{(2.15)}$$

Solving the above equation based on least square approximations, the normal node N_x can obtain its estimated position $N_{\text{DV-hop}}$:

$$N_{DV-hop}: \begin{bmatrix} x' \\ y' \end{bmatrix} = (A^T A)^{-1} A^T B \quad \textbf{(2.16)}$$

Where $A = -2\times\begin{bmatrix} x_1-x_{m_d} & y_1-y_{m_d} \\ x_2-x_{m_d} & y_2-y_{m_d} \\ \vdots & \vdots \\ x_{m_d-1}-x_{m_d} & y_{m_d-1}-y_{m_d} \end{bmatrix}$, $B = \begin{bmatrix} d_{1,Nx}^2-d_{m_d,Nx}^2-x_1^2+x_{m_d}^2-y_1^2+y_{m_d}^2 \\ d_{2,Nx}^2-d_{m_d,Nx}^2-x_2^2+x_{m_d}^2-y_2^2+y_{m_d}^2 \\ \vdots \\ d_{m_d-1,Nx}^2-d_{m_d,Nx}^2-x_{m_d-1}^2+x_{m_d}^2-y_{m_d-1}^2+y_{m_d}^2 \end{bmatrix}$,

A^T is the transpose of the matrix A, and A^{-1} is the inverse of the matrix A.

We should note a condition for DV-hop algorithm: the anchors cannot be on the same line. Otherwise, in the equation, $A^T A$ will be singular, thus $(A^T A)^{-1}$ doesn't exist.

Although the DV-hop algorithm can localize the normal nodes which have less than three neighbor anchors, its localization accuracy needs to be improved. Thus, many DV-hop based algorithms have been proposed in recent years. In the following, several typical algorithms will be introduced.

2.2.2.4.2 Typical DV-hop Based Algorithms

In this section we describe a few DV-hop based localization algorithms such as DDV-hop (Differential DV-hop), Self-adaptive DV-hop, and Robust DV-hop.

(i) DDV-hop: In [HZL 10], the authors propose a DDV-hop (Differential DV-hop) algorithm. This algorithm changes Step #2 and Step #3 of the original DV-hop algorithm.

In Step #2 of DDV-hop, each anchor A_i not only broadcasts its distance-per-hop dph_i through the network, but also broadcasts the differential error of dph_i to the entire network. This differential error, denoted as $diff\_err_i$, is calculated as:

$$diff\_err_i = \frac{\sum_{j \neq i} \left| dph_i - \frac{d_{i,j}}{hop_{i,j}} \right|}{m_d - 1} \quad \textbf{(2.17)}$$

Where m_d is the number of anchors, $hop_{i,j}$ is the hop count between A_i and every other anchor A_j, and $d_{i,j}$ is the distance between A_i and A_j. Here, $hop_{i,j}$ is obtained through Step #1, while $d_{i,j}$ is calculated as $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

In Step 3, DDV-hop and DV-hop differ on the calculation of the estimated distance between a normal node Nx and each anchor A_i. In the original DV-hop algorithm, when a normal node Nx receives the distance-per-hop value of A_i, Nx immediately calculates its estimated distance to A_i as $dph_i \times hop_{i,Nx}$. But in DDV-hop algorithm, Nx uses its own distance-per-hop value denoted as dph_{ddv} to replace dph_i. Here, dph_{ddv} is obtained as the weighted sum of all anchors' distance-per-hop. The weighting coefficient of dph_i, denoted as λ_i, is decided by the differential errors of anchors' distance-per-hop:

$$\lambda_i = \frac{diff\_err_i}{\sum_{k=1}^{m^*} \left| diff\_err_k \right|} \quad \textbf{(2.18)}$$

Where m^* is the number of differential errors that have been received by Nx. Then, dph_{ddv} is calculated as:

$$dph_{ddv} = \sum_{i=1}^{m^*} \left(\lambda_i \times dph_i \right) \quad \textbf{(2.19)}$$

We can find that: in order to obtain a more accurate average distance per hop for normal nodes, the authors use a weighing method based on the differential error of each anchor's distance-per-hop. The authors want to apply this algorithm to improve the accuracy in asymmetry distributed wireless sensor networks. Their simulation results show that, when sensor nodes are distributed asymmetrically (not regularly), DDV-hop algorithm is about 18%-20% more accurate than DV-hop algorithm. However, compared with the original DV-hop, DDV-hop increases the network traffic, because the new data "differential error" ($diff\_err_i$) should be broadcast by every anchor.

(ii) Self-Adaptive DV-hop: In [ZXL 09], a DV-hop based Self-Adaptive Positioning algorithm is proposed. This algorithm composed of two methods. Because the second method needs RSSI information, we only consider the first method of this self-adaptive algorithm. This algorithm has the same network overhead as the original DV-hop but slightly changes Step #3. At Step #3, when a normal node Nx calculates its estimated distance to A_i, Nx uses its own distance-per-hop value denoted as dph_{adp} to replace the anchors' distance-per-hop. dph_{adp} is also obtained as the weighted sum of anchors' distance-per-hop. But compared with DDV-hop algorithm, this self-adaptive algorithm has a different way to decide the weighing coefficients for dph_{adp}. In this algorithm, λ_i^a, that is the weighting coefficient of dph_i (each anchor A_i's distance-per-hop), is decided based on Nx's hop counts to A_i. The more hops between Nx and A_i, the smaller value assigned to λ_i^a. The calculation of λ_i^a is:

$$\lambda_i^{\mathrm{a}} = \frac{(\sum_{k=1}^{m_d} hop_{k,Nx}) - hop_{i,Nx}}{(m_d - 1)\sum_{k=1}^{m_d} hop_{k,Nx}} \quad \textbf{(2.20)}$$

Then, dph_{adp} is calculated as:

$$dph_{ddv} = \sum_{i=1}^{m_d}\left(\lambda_i^a \times dph_i\right) \quad \textbf{(2.21)}$$

As a conclusion, the authors in [ZXL 09] believe that the nearest anchor to a normal node always has the most accurate average distance-per-hop. Based on this concept, the authors obtain a new distance-per-hop. Simulation results show that, the accuracy of this self-adaptive algorithm is 30% better than DV-hop algorithm. However, the simulation scenario is very special: the anchors are distributed at the corners of the simulation area, and the normal nodes are regularly distributed inside the area.

(iii) Robust DV-hop: a Robust DV-hop (RDV) algorithm is proposed in [LCK 10]. Different from the above two algorithms, in order to replace dph_{i} (the average distance per hop of A_{i}), RDV-hop algorithm defines a distance-per-hop value between Nx and A_{i}, denoted as $dph_{\mathrm{Nx,i}}$. And $dph_{\mathrm{Nx,i}}$ is calculated as the weighted sum of the distance-per-hop values between A_{i} and every other anchor A_{k}. Here, the distance-per-hop between A_{i} and A_{k} is denoted as $dph_{\mathrm{i,k}}$, which is calculated as:

$$dph_{i,k} = \frac{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}{hop_{i,k}} \quad \textbf{(2.22)}$$

Where the positions of A_{i} and A_{k} are (x_{i}, y_{i}) and (x_{k}, y_{k}), $hop_{\mathrm{i,k}}$ is the hop count between A_{i} and A_{k}. Thus, Nx needs to know the hop count between any two anchors. This requires each anchor to broadcast at Step #2 its hop counts (to other anchors) throughout the network.

Then, the weighting coefficient $\lambda_{\mathrm{i,k}}$, is calculated as:

$$\lambda_{i,k} = \frac{1}{(hop_{i,Nx} + hop_{Nx,k}) - hop_{i,k} + 1} \quad \textbf{(2.23)}$$

So, $dph_{\mathrm{Nx,i}}$ can be calculated:

$$dph_{Nx,i} = \frac{\sum_{\mathrm{k \neq i}} (\lambda_{i,k} \times dph_{i,k})}{\sum_{\mathrm{k \neq i}} \lambda_{i,k}} \quad \textbf{(2.24)}$$

This weighting coefficient will have the maximum value, if Nx is on the shortest path between A_{i} and A_{k}. As a conclusion, the authors find that if a normal node stands on the shortest path between two anchors, then the distance-per-hop between the two anchors will be the most accurate for this normal node. Based on this concept, this algorithm obtains a more accurate average distance-per-hop for each normal node. The simulation results indicate that, compared with DV-hop algorithm, Robust DV-hop algorithm has an accuracy improvement from 10% to 40%, depending on the distribution of sensor nodes. However, the three distributions of nodes in the simulation are very special: the first is an isotropic (regular) distribution, the second is C-shaped distribution, and the third is X-shaped distribution. We can also note that, this algorithm increases the network traffic compared with the original DV-hop algorithm. That is because: at Step #2, besides the distance per hop, each anchor needs also to broadcast its hop counts (to other anchors) throughout the network.

All these typical DV-hop based algorithms use a weighing method to determine a weighted distance-per-hop value for each normal node. However, in order to get a more accurate weighted distance-per-hop value, sometimes additional information is demanded, such as differential error in [HZL 10], or hop counts in [LCK 10]. Broadcasting this additional information always increases the network traffic. We should also note that, the simulation results of the above algorithms are not so convincing, because the distributions of sensor nodes are particularly designed rather than randomly obtained. In order to obtain a better accuracy without increasing the network overhead, we are motivated to provide improved methods.

2.2.2.5 Brief Summary of Range-Free Localization

Compared with range-based schemes, the range-free localization schemes don't need any additional ranging hardware, and can be pursuit as cost-effective solutions for wireless sensor networks. Thus, we focus on range-free schemes. The following table compares the main advantages and disadvantages of range-based and rang-free schemes.

Table 2-6. Comparison between Rang-based and Rang-free Schemes

"Range-based"	"Rang-free"
higher precision	lower precision
need additional ranging devices	don't need additional devices
easily affected by multi-path fading and noise	more robust

In the previous subsections, we have introduced several popular range-free localization algorithms. In the following, these algorithms are listed and compared.

Table 2-7. Comparison of Range-Free Localization

<table>
<tr><th colspan="2">Range-free Algorithms</th><th>Advantages</th><th colspan="2">Disadvantages</th></tr>
<tr><td colspan="2">Centroid</td><td>Low overhead</td><td>Low accuracy</td><td rowspan="4">A normal node needs at least 3 neighbor anchors.</td></tr>
<tr><td rowspan="2">CPE</td><td>original CPE</td><td>Good accuracy</td><td>Centralized, high overhead</td></tr>
<tr><td>simplified CPE</td><td>Low overhead</td><td>Low accuracy</td></tr>
<tr><td colspan="2">APIT</td><td>No ideal radio assumption</td><td>High power, RSSI needed</td></tr>
<tr><td colspan="2">DV-hop</td><td rowspan="4">No restrict on the number of neighbor anchors</td><td colspan="2" rowspan="4">Low accuracy, big overhead (network)
(the overheads of DDV-hop and Robust DV-hop are even higher than DV-hop and Self-adaptive DV-hop)</td></tr>
<tr><td colspan="2">DDV-hop</td></tr>
<tr><td colspan="2">Self-adaptive DV-hop</td></tr>
<tr><td colspan="2">Robust DV-hop</td></tr>
</table>

Centroid and the simplified CPE algorithms both have low network overhead and low calculation complexity, but with relatively low accuracy. On the contrary, the original CPE algorithm has much better accuracy, but it is centralized, resulting in large network traffic and high computation complexity. The APIT algorithm is not frequently used, because the anchors need to have high power transmitters, and the unstable RSSI information is required.

Unlike the above methods requiring at least 3 neighbor anchors for a normal node, DV-hop based algorithms provide the solutions when there are not many anchors in the network. However, their network overhead cannot be neglected, and the accuracy needs to be improved.

In the following chapters, we will introduce our improved range-free algorithms. Considering few works on how to implement the algorithms into network, we will also introduce our corresponding localization protocols.

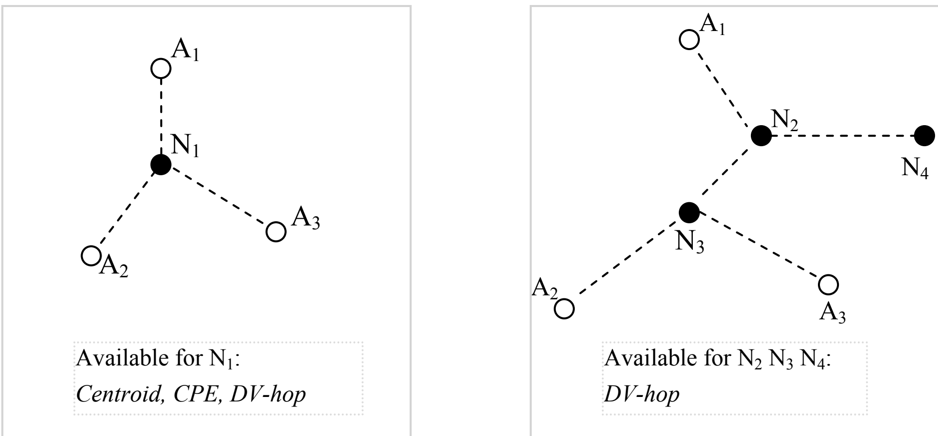
3. Improvement on Range-free Algorithms

3.1 Context

According to the previous comparison on the typical range-free algorithms, when a normal node has at least 3 neighbor anchors, it can localize itself using algorithms such as Centroid, CPE (in this chapter, it refers to simplified version of CPE), and DV-hop based algorithms. On the contrary, when a normal node has less than 3 neighbor anchors, the available localization algorithms are only the DV-hop based algorithms. Here, the neighbor anchors of a normal node are those within the transmission range of the normal node.

Examples of the above two cases are shown in Figure 3-1. In Figure 3-1(a), the normal node N_1 has three neighbor anchors A_1, A_2, and A_3. However, in Figure 3-1(b), every normal node (such as N_2, N_3, and N_4) has less than 3 neighbor anchors.

Among the three available algorithms for N_1, Centroid and CPE are better than DV-hop based algorithms, in terms of overhead (including network overhead and calculation cost). As discussed in the section 2.2.2.5, compared with DV-hop based algorithms, Centroid and CPE algorithm have lower network overhead and lower calculation cost. Thus, using Centroid and CPE, N_1 can calculate its position as soon as possible. However, DV-hop algorithm requires broadcast from each anchor throughout the network at Step #1 and Step #2. The broadcast takes time, especially when the network is large. Thus, using DV-hop, N_1 needs to wait longer time before computing its position than using Centroid and CPE. As a result, for normal nodes with at least 3 neighbor anchors, Centroid and CPE are recommended instead of DV-hop.



(a) at least 3 neighbor anchors (b) less than 3 neighbor anchors

Figure 3-1. Normal Nodes with Different Number of Neighbor Anchors

However, in general, anchors are always scattering in a network, making a normal node has less than 3 neighbor anchors. As shown in Figure 3-1 (b), N_2 has only 1 neighbor anchor, N_3 has 2, and N_4 has none. In this case, these normal nodes cannot use Centroid and CPE. Thus, DV-hop is recommended for localization.

Therefore, different localization methods are suggested for normal nodes when they have different number of neighbor anchors. This encourages us to categorize normal nodes into two classes according to the number of neighbor anchors: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes.

The advantage of this classification is for each normal node to choose its suitable localization algorithm [GWV 10]. Class-1 nodes can choose those low-overhead algorithms such as Centroid and CPE, while class-2 nodes need to use DV-hop.

For normal nodes in each class, we will present our improved localization methods in the following subsections. As shown in Figure 3-2, Mid-perpendicular algorithm will be introduced for class-1 nodes. Since there are generally only a few anchors in practical scenarios, most of normal nodes must be in class 2. Thus, we take more attention on DV-hop algorithm. We have proposed Checkout DV-hop and Selective 3-Anchor DV-hop algorithms for class-2 nodes.

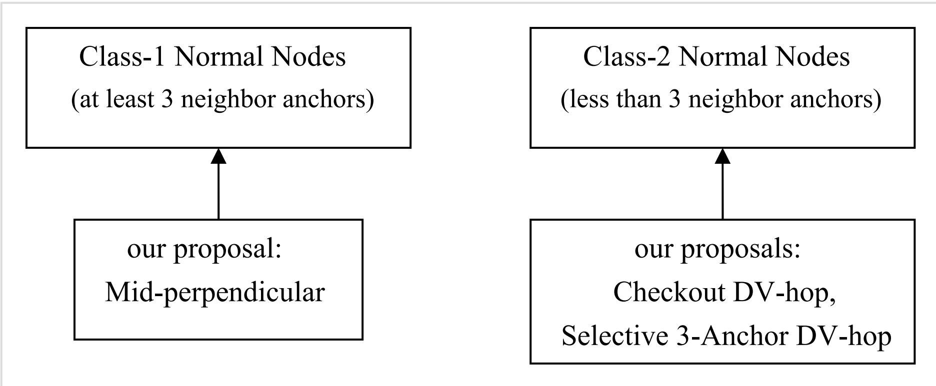


Figure 3-2. Proposals for Corresponding Class of Normal Nodes

3.2 Mid-perpendicular Algorithm

For class-1 normal nodes, Centroid and CPE are popular algorithms because of their low communication and computation cost, regardless of their inaccuracy. Our aim is to propose a new algorithm which can achieve a higher accuracy, at the cost of higher calculation complexity.

In the following, we will first analyze where can be improved in Centroid and CPE algorithms, then illustrate the principle of our new algorithm called Mid-perpendicular [DGV 11] [GVW 11].

3.2.1 Preliminary Analysis

The class-1 algorithms, such as Centroid and CPE, assume that the communication range of nodes is identical in the shape of sphere. These algorithms can localize the class-1 normal nodes which have at least 3 neighbor anchors. Centroid and CPE algorithms try to find a centre point of the overlap communication region of neighbour anchors. Centroid algorithm regards the centroid point of anchors as the estimated position, while CPE algorithm uses the centre point of the rectangle which bounds the communication range of anchors.

The Centroid algorithm can get relatively good accuracy when the distribution of anchors is regular (evenly or equally distributed). In this case, the communication areas of anchors form a relatively large overlap, which is shown as the shaded region in Figure 3-3. In this scenario, there are 3 anchors in total, and the communication range of anchors is set to be 10 meters. The real position of the normal node N_x is (19, 1.5). It has all the three anchors as its neighbours. Using Equation (2.7) in Centroid algorithm, N_x can calculate its estimated position N_{cen} as (12+26+19, 0+0+7) / 3 = (19, 2.33). N_{cen} locates in the overlap, and it is very close to the real position of N_x. The location error is $\sqrt{(19-19)^2+(2.33-1.5)^2}=0.83$m.

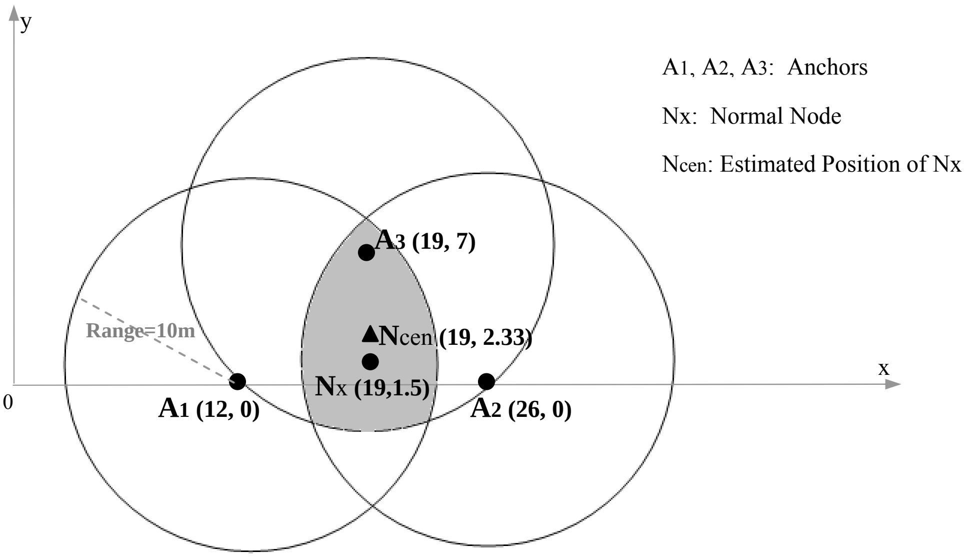


Figure 3-3. Scenario 1: Centroid with Good Accuracy

However, when the distribution of anchors is not regular, the estimated position derived from the Centroid algorithm will be inaccurate. For example, in scenario 2 (see from Figure 3-4), the normal node N_x with coordinates (19.5, 6.5) locates in the overlap communication region of its neighbour anchors A_1 (12, 10), A_2 (27, 0), and A_3 (27, 12). This overlap (shaded part in Figure 3-4) is much smaller than that in Figure 3-3. The communication range of nodes is set as 10 meters. When using the Centroid algorithm, the estimated position is N_{cen} (22, 4), which goes obviously out of the overlap communication region. In this scenario, the location error is $\sqrt{(19.5-22)^2+(6.5-4)^2}=3.54$m. This shows a much lower accuracy, compared with scenario 1 (its location error is 0.83m).

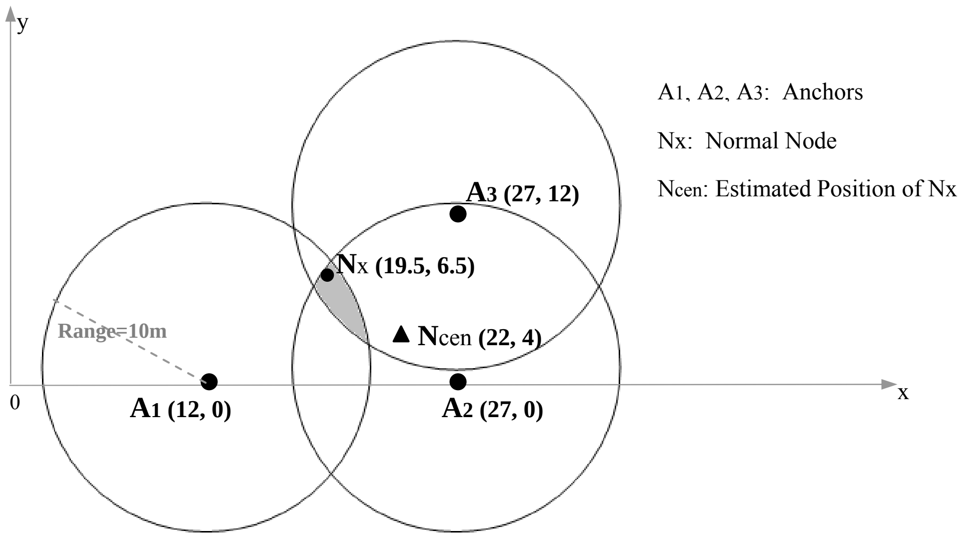


Figure 3-4. Scenario 2: Centroid with Lower Accuracy

Generally, the simplified version of CPE algorithm has better accuracy than Centroid algorithm, when anchors are distributed not evenly. An example is shown in Figure 3-5. Like Figure 3-4, scenario 2 is configured in Figure 3-5, so that the performance difference between Centroid and CPE can be observed. Using CPE algorithm, the estimated position of N_x, denoted as N_{CPE}, is the centre of the estimated rectangle. This rectangle bounds the communication range of anchors. Using Equation (2.11), we can obtain the position of N_{CPE} as (19.5, 6). Thus, the location error of CPE algorithm is $\sqrt{(19.5-19.5)^2+(6.5-6)^2}=0.5$m, which is much smaller than that of Centroid algorithm.

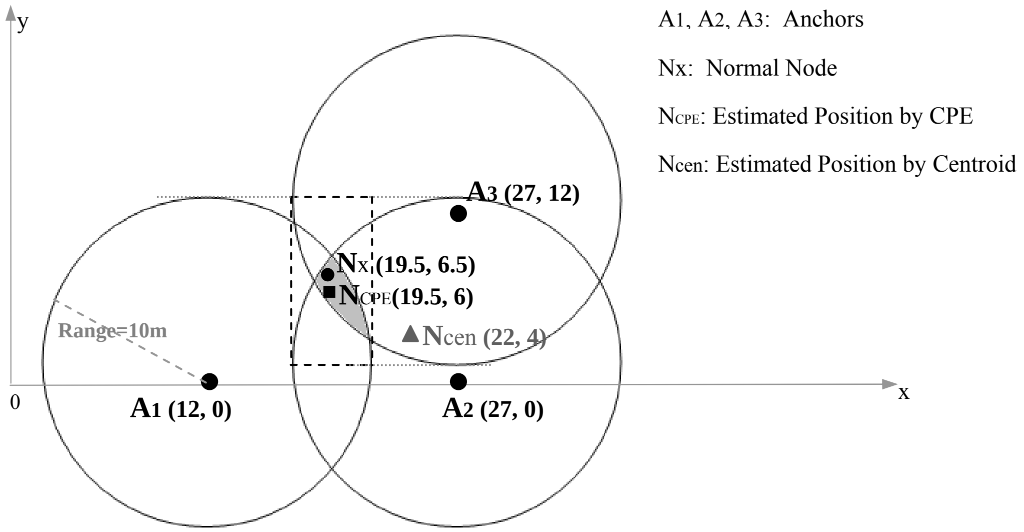


Figure 3-5. Scenario 2: CPE Algorithm with Good Accuracy

However, the estimated position from CPE algorithm is not always inside the overlap. Sometimes, it may go out of the overlap communication region of anchors. As shown in Figure 3-6, in Scenario 3, the overlap is very small. We can see that, the estimated positions by Centroid and CPE algorithms are both out of the overlap. In this case, the location error of CPE algorithm is 1.50m, while that of Centroid algorithm is 1.21m.

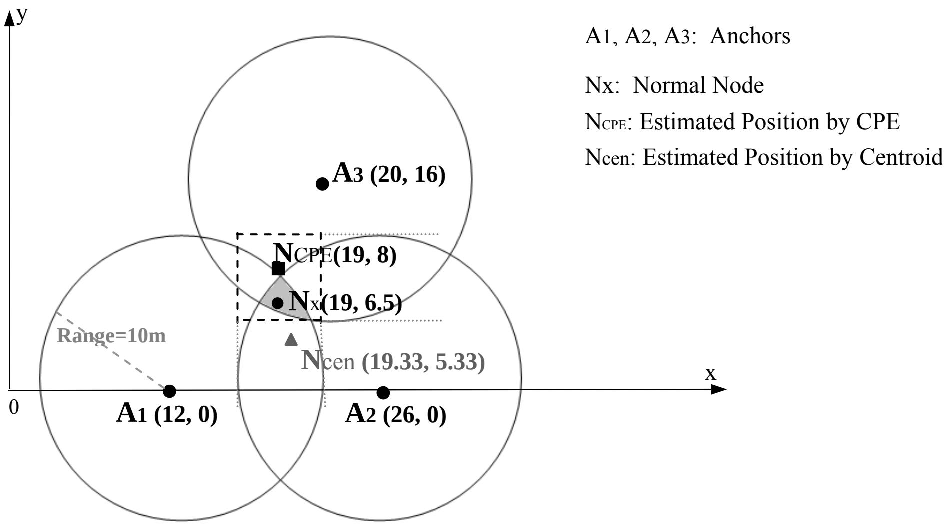


Figure 3-6. Scenario 3: CPE Algorithm with Lower Accuracy

From the above examples, it can be observed that, the performances of Centroid and CPE algorithms vary with the distribution of anchors. We can also note that, although N_x is inside the overlap of anchors communication range, Centroid and CPE algorithms sometimes localize N_x outside the overlap. This encourages us to propose a new algorithm.

3.2.2 Principle of New Mid-perpendicular Algorithm

In order to improve the accuracy of Centroid and CPE algorithm, we propose Mid-perpendicular algorithm [GWV 10]. The basic principle of this algorithm is to find the centre of anchors communication overlap, and regards this centre as the estimated position.

First, we present our algorithm when a normal node has only 3 neighbor anchors. Then, we extend the algorithm to support more neighbor anchors.

3.2.2.1 Mid-perpendicular in Case of 3 Neighbor Anchors

Like Centroid and CPE algorithms, here, it is also assumed that the communication ranges of anchors are all the same. As shown in Figure 3-7, the normal node N_x has three neighbor anchors (A_1 A_2 and A_3). It means that N_x locates in the overlap communication region of A_1 A_2 and A_3. This overlap is marked as the shaded part in Figure 3-7.

Now we present how to derive the centre of overlap region. As shown in Figure 3-7, "*Line1*" is the mid-perpendicular of the line connecting the anchors A_2 and A_3. That means, *Line1* passes the middle point between A_2 and A_3, and it crosses the line (which connects A_2 and A_3) at a right angle. According to the symmetry, *Line1* goes through the center of the overlap region.

In the same manner, *Line2* is the mid-perpendicular of the line connecting A_1 and A_3, while *Line3* is the mid-perpendicular of the line connecting A_1 and A_2. Both *Line2* and *Line3* go through the center of the overlap region. Thus, the cross point of the three mid-perpendiculars (*Line1*, *Line2* and *Line3*) can be regarded as the center of overlap. This cross point is denoted as N_{mid}, which is also the estimated position of Mid-perpendicular algorithm.

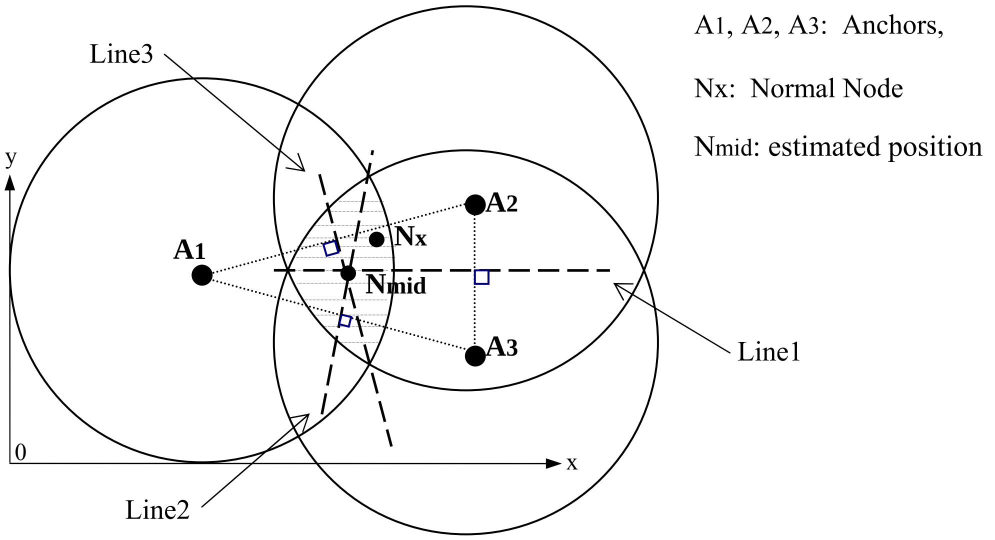


Figure 3-7. Mid-Perpendicular (An Acute Triangle by 3 Neighbor Anchors)

In fact, in order to calculate the cross point N_{mid}, only two mid-perpendiculars are needed, for example, *Line1* and *Line2*. If the coordinates of the three anchors (A_1, A_2 and A_3) are respectively (x_1, y_1), (x_2, y_2), and (x_3, y_3), then *Line1*, which is the mid-perpendicular of line A_2A_3, can be expressed as:

$$y - \frac{y_2 + y_3}{2} = (x - \frac{x_2 + x_3}{2})\frac{x_2 - x_3}{y_3 - y_2} \qquad \textbf{(3.1)}$$

Line2, which is the mid-perpendicular of line A_1A_3, can be expressed as:

$$y - \frac{y_1 + y_3}{2} = (x - \frac{x_1 + x_3}{2})\frac{x_1 - x_3}{y_3 - y_1} \qquad \textbf{(3.2)}$$

The cross point of the above two mid-perpendiculars, that is N_{mid} with its coordinates (x_{mid}, y_{mid}), can then be calculated as:

$$
\begin{cases}
x_{mid} = \dfrac{(x_1^2 - x_2^2)(y_3 - y_1) + (x_1^2 - x_3^2)(y_1 - y_2) + (y_1 - y_2)(y_2 - y_3)(y_3 - y_1)}{2[y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_3(x_1 - x_2)]} \\
y_{mid} = \dfrac{(y_1^2 - y_2^2)(x_3 - x_1) + (y_1^2 - y_3^2)(x_1 - x_2) + (x_1 - x_2)(x_2 - x_3)(x_3 - x_1)}{2[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)]}
\end{cases}
\quad \textbf{(3.3)}
$$

It should be noted that, there is one condition for the above derivation: N_x's 3 neighbor anchors (A_1, A_2 and A_3) form an acute triangle, where all the angles are less than 90 degrees. However, if the 3 neighbor anchors form a right triangle or an obtuse triangle, then the calculation of N_{mid} will be much simpler than the equation (3.3). This will be illustrated as following.

Figure 3-8 shows the scenario when A_1, A_2 and A_3 form a right triangle. $\angle A_1A_3A_2$ is 90 degrees, and the side A_1A_2 is the longest side of the triangle. From Figure 3-8, we can see that, the cross point of the three mid-perpendiculars is just the middle point of side A_1A_2. That means, when the three neighbour anchors form a right triangle, N_{mid} will be the middle point of the longest side in the triangle. This conclusion is expressed in the following equation, where side A_iA_k is the longest side of the triangle $\Delta A_1A_2A_3$, with the coordinates for A_i and A_k respectively (x_i, y_i) and (x_k, y_k):

$$
(x_{mid}, y_{mid}) = (x_i + x_k, y_i + y_k)/2, \quad \textbf{(3.4)}
$$
$$
\text{where } A_i \text{ and } A_k \text{ form the longest side of triangle}
$$

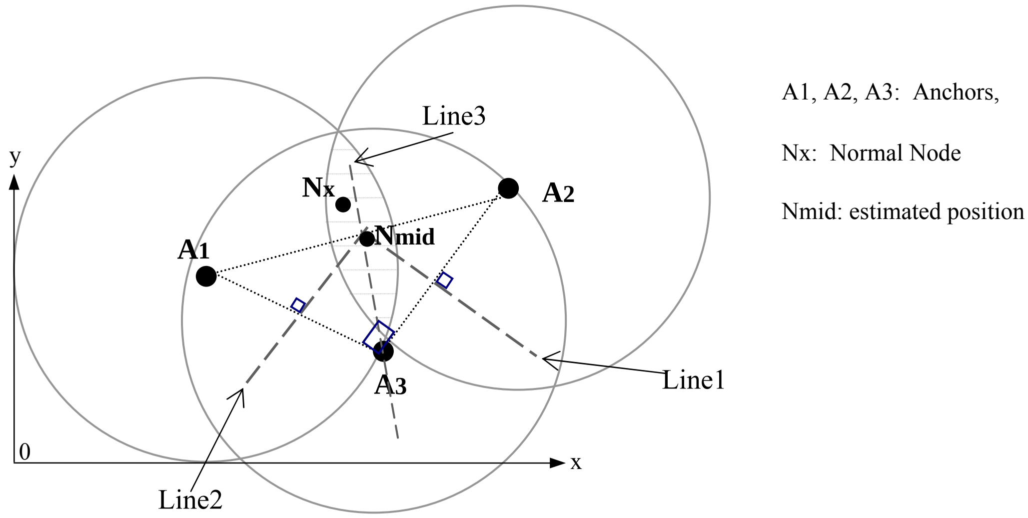


Figure 3-8. Mid-Perpendicular (Right Triangle by 3 Neighbor Anchors)

Not only a right triangle, A_1, A_2 and A_3 can also form an obtuse triangle, as shown in Figure 3-9. $\angle A_1A_3A_2$ is larger than 90 degrees, and the side A_1A_2 is still the longest side of the triangle. From Figure 3-9, we can see that, the cross point of the three mid-perpendiculars, denoted as "*P*", is going out of the overlap region. Instead, the middle point of side A_1A_2, denoted as N_{mid}, becomes the centre of overlap. Thus, like the above scenario (right triangle), when the three neighbour anchors form an obtuse triangle, N_{mid} is still the middle point of the longest side in the triangle. In this case, the equation (3.4) is still applicable.

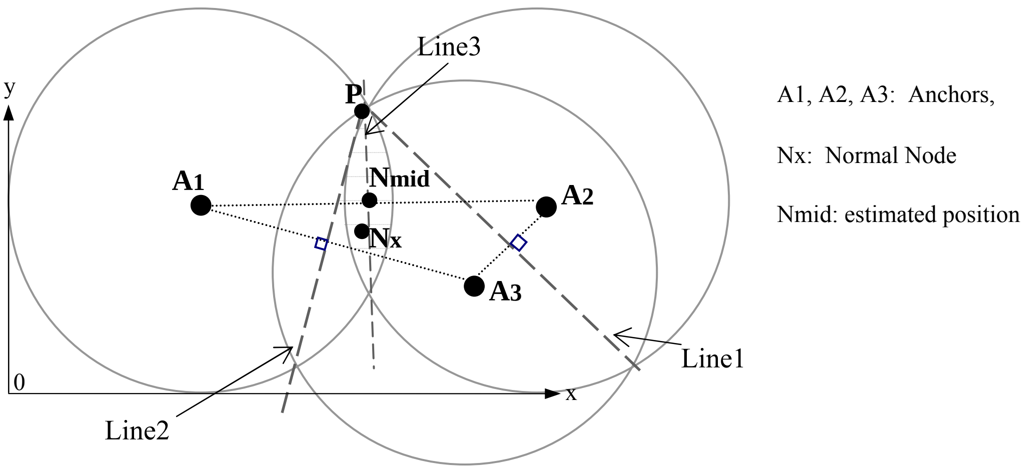


Figure 3-9. Mid-Perpendicular (Obtuse Triangle by 3 Neighbor Anchors)

To summarize the above derivation with different triangles, we can give the calculation of N_{mid} when N_{x} has only three neighbor anchors:

$$N_{mid}\ (x_{mid}, y_{mid}) = \begin{cases} \text{Equation (3.3), if three anchors form an acute triangle} \\ \text{Equation (3.4), others} \end{cases} \qquad \textbf{(3.5)}$$

3.2.2.2 Mid-perpendicular in Case of More than 3 Neighbor Anchors

The equation (3.5) gives a solution to calculate the position when the normal node N_{x} has 3 neighbor anchors. However, a more complex scenario should also be considered, when N_{x} has in total *m* neighbor anchors and m>3. In this case, an extended version of Equation (3.5) must be developed.

The direct manner to apply the equation (3.5) is as follows. Among the *m* neighbor anchors A_1, A_2, … A_{m}, any three anchors can generate one estimated position "N_{mid}" using the equation (3.5). Thus, as many as C_{m}^{3} positions can be generated. The average of all these positions can be regarded as final estimated position. However, this direct manner is complicated. (Note: the complexity of algorithms will be discussed in Section 3.5.)

The program procedure of this direct version can be described as:

Algorithm "direct version of Mid-perpendicular":
suppose the normal node N has m neighbor anchors $A_1, A_2 \ldots, A_{\mathrm{m}}$.
$\overline{x_{mid}} \leftarrow 0$; $\overline{y_{mid}} \leftarrow 0$;
for $i \leftarrow 1$ **to** $(m$-2)
  A_{i} is chosen. $(x_{\mathrm{i}}, y_{\mathrm{i}})$ is the position of A_{i}.
  for $j \leftarrow (i+1)$ **to** $(m$-1)
    A_{j} is chosen. $(x_{\mathrm{j}}, y_{\mathrm{j}})$ is the position of A_{j}.
    for $k \leftarrow (j+1)$ **to** m
      A_{k} is chosen. $(x_{\mathrm{k}}, y_{\mathrm{k}})$ is the position of A_{k}.
      $(x_{\mathrm{mid}}, y_{\mathrm{mid}}) \leftarrow$ calculated as Equation (3.5) based on the anchors $A_{\mathrm{i}}, A_{\mathrm{j}}, A_{\mathrm{k}}$
      $\overline{x_{mid}} \leftarrow \overline{x_{mid}} + x_{\mathrm{mid}}$; $\overline{y_{mid}} \leftarrow \overline{y_{mid}} + y_{\mathrm{mid}}$
$\overline{x_{mid}} \leftarrow \overline{x_{mid}} / C_m^3$; $\overline{y_{mid}} \leftarrow \overline{y_{mid}} / C_m^3$
return $\overline{x_{mid}}$ and $\overline{y_{mid}}$

Figure 3-10. Procedure of Direct Version of Mid-perpendicular Algorithm

Considering this direct version is complicated, in the following, we introduce a simplified method.

First, we use an example to show a phenomenon: the overlap communication region of all the *m* anchors is contributed mainly by three anchors. This example has 4 anchors in total, as shown in Figure 3-11. The normal node N_x has 4 neighbor anchors A_1, A_2, A_3, and A_4. From the figure, we can see that, the overlap region formed by all the 4 anchors is actually the overlap of the three anchors A_1, A_2, and A_4.

These three anchors have the following characteristics: (1) Two of them have the longest distance, compared with distances between any two of the entire anchors. That is because the longest two anchors have the smallest overlap. In this example, it can be observed that, the distance between A_1 and A_4 is longest. Thus, the two longest anchors here are A_1 and A_4. (2) The third anchor is farthest to the line connecting the two longest anchors. In this example, since the two longest anchors are A_1 and A_4, the anchors except them are A_2 and A_3. From the figure, obviously, compared with A_3, A_2 has a longer distance to the line connecting A_1 and A_4. Thus, the third anchor is A_2.

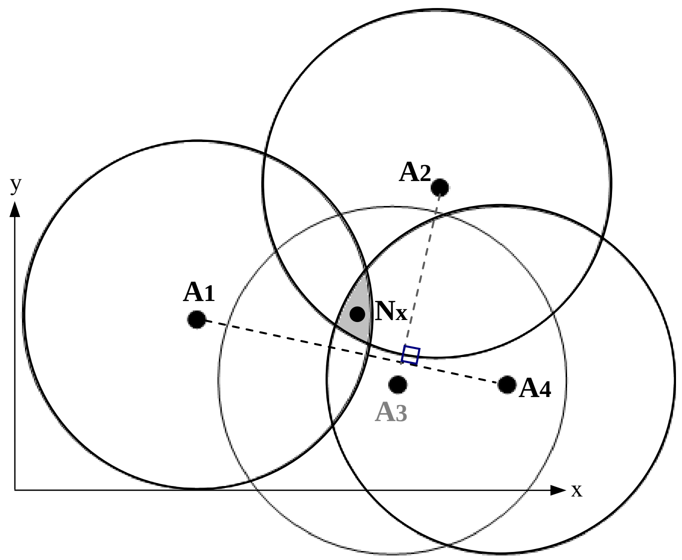


Figure 3-11. Example with 4 Neighbor Anchors

Based on the above analysis, we know how to find the three anchors which contribute the overlap communication region of the entire *m* neighbor anchors (like the Centroid Algorithm, it is assumed that the anchors periodically broadcast their positions.) First, N_x calculates the distance between any two anchors. Since there are *m* neighbor anchors, there will be C_m^2 distances in total. Comparing these distances, N_x can find out the two farthest anchors, denoted as A_i and A_k. Then, among all other anchors except A_i and A_k, N_x finds out the anchor which has the longest distance to the line connecting A_i and A_k. This anchor is denoted as A_j. Thus, A_i, A_k and A_j are the three anchors which contributes the overlap of all the *m* anchors. Finally, using the equation (3.5), N_x calculates the centre of overlap formed by the three anchors A_i, A_k and A_j. The result is final estimated position of this simplified Mid-perpendicular algorithm.

The program procedure of this simplified version can be described as:

```
Algorithm "simplified version of Mid-perpendicular":
suppose the normal node N has m neighbor anchors A1, A2 ..., Am.
for i ← 1 to (m-1)
    Ai is chosen. (xi, yi) is the position of Ai.
    for k ← (i +1) to m
        Ak is chosen. (xk, yk) is the position of Ak.
        calculate the distance between Ai and Ak; look for the two farthest anchors
Find the two farthest anchors, suppose they are denoted as Ai and Ak
for j ← 1 to m
    calculate the distance between Aj and the line AiAk
Find the anchor farthest to the line AiAk, suppose the anchor is Aj
( x̄mid , ȳmid ) ←calculated as Equation (3.5) based on the anchors Ai, Aj, Ak
return x̄mid and ȳmid
```

Figure 3-12. Procedure of Simplified Version of Mid-perpendicular Algorithm

3.3 Checkout DV-hop Algorithm

In the previous section, we introduce our method to localize the class-1 normal nodes. Now, we would like to focus on class-2 normal nodes. Generally, in a network there are always a few anchors and much more normal nodes. As a result, most normal nodes will be in class 2, having less than 3 neighbor anchors.

DV-hop algorithm is frequently used to localize class-2 normal nodes. However, its accuracy is not satisfactory. Thus, we aim to propose better solutions. This section presents our Checkout DV-hop algorithm [GWV 10]. In the following, we first prove one conclusion which inspires us to propose the algorithm. This conclusion shows that, the nearest anchor to a normal node always has the most accurate estimated distance to that node. Then, we introduce the procedure of Checkout DV-hop algorithm in detail.

3.3.1 Accuracy of Estimated Distance

In this subsection, by both theoretical analysis and simulation results, we prove that the nearest anchor always has the most accurate estimated distance to a normal node.

3.3.1.1 Theoretical Analysis

The key issue of DV-hop algorithm is calculating the approximate distance between the normal node N_x and each anchor A_i. This estimated distance, denoted as $d_{i,Nx}$, is obtained by multiplying the hop count by the average distance per hop:

$$d_{i,\mathrm{N}x} = hop_{i,\mathrm{N}x} \times dph_i, i = 1, 2 \ldots m_d \quad \textbf{(3.6)}$$

In the equation (3.6), $hop_{i,Nx}$ is the minimal hop number between N_x and A_i, and dph_i is the approximate average distance per hop of A_i. These two values can be obtained through the first two steps of DV-hop algorithm. It is assumed that in the network there are m_d anchors in total, and N_x has less than 3 neighbor anchors (one example of N_x is shown in Figure 2-20 in the section 2.2.2.4). The calculation of dph_i is:

$$dph_i = (\sum_{k(k \neq i)} d_{i,k}) / (\sum_{k(k \neq i)} hop_{i,k}) \quad \textbf{(3.7)}$$

Where $d_{i,k}$ is the distance between A_i and A_k, $hop_{i,k}$ is the minimal hop count between A_i and A_k.

From the equation (2.16) in section 2.2.2.4.1, we can see that, $d_{i,Nx}$ is an important element for calculating the position of N_x. Thus, $d_{i,Nx}$ has a considerable influence on the accuracy of DV-hop. We denote the true distance from N_x to A_i as $d_{i,Nx\text{True}}$, and the difference between $d_{i,Nx\text{True}}$ and $d_{i,Nx}$ as $\Delta d_{i,Nx}$, where obviously $\Delta d_{i,Nx}$ is one reason for the inaccuracy of DV-hop. If we denote Δdph_i as the difference between its value given by (3.7) and the actual distance along the path from A_i to N_x, we have:

$$\Delta d_{i,Nx} = hop_{i,Nx} \times \Delta dph_i \quad \textbf{(3.8)}$$

Equation (3.8) indicates that when $hop_{i,Nx}$ increases, $\Delta d_{i,Nx}$ also increases, and the accuracy of DV-hop decreases. If A_{near} is the nearest anchor to N_x among all anchors A_1 A_2 ... A_{md}, then correspondingly $hop_{near,Nx}$ is the smallest, so that $\Delta d_{near,Nx}$ is the smallest position error. So we can conclude that, compared to other anchors, the distance from N_x to its nearest anchor A_{near}, denoted as $d_{near,Nx}$, has the highest reliability in terms of precision.

3.3.1.2 Simulation Results

In this section, we aim to obtain the simulation results to verify that the nearest anchor has the most accurate estimated distance to the normal node.

Our simulation tool is MATLAB. The ideal radio propagation is assumed, with no signal loss, no interference, and no collisions. The main parameters of simulation scenario are listed in Table 3-1.

Table 3-1. Parameters of Simulation Scenario for Checkout DV-hop

Simulation Area	100 × 100 m^2
Node Communication Range	20 m
Total Number of Nodes	100
Ratios of Anchors	[10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%]
Random Times	20 × 100 = 2000 times

As displayed in Table 3-1, there are 100 nodes in total. They include anchors and normal nodes. The parameter "ratio of anchors" is defined as the ratio between the number of anchors and the total number of nodes. For example, if ratio of anchors is 10%, then there are 100 × 10%=10 anchors, while the rest 90 nodes are normal nodes. In our simulation, the ratios of anchors vary from 10% to 90%.

For each ratio of anchors, the simulations run randomly for 2000 times. This value of random times "2000" is composed of two multipliable parts "TRd_{nod}" and "TRd_{anc}", thus $TRd_{nod} \times TRd_{anc}$=2000. Here, TRd_{nod} is the number of random times for generating different geographical distributions of nodes. Every time, for each ratio of anchors, all the 100 nodes are uniform-randomly distributed inside the simulation area. So, through TRd_{nod} times, we can have as many as TRd_{nod} geographical distributions of nodes. The second random times, TRd_{anc}, is the number of random times for selecting nodes as anchors. Every time, for each distribution of nodes, anchors are uniform-randomly selected from the entire 100 nodes. In the simulation, as an example, we set TRd_{nod} to be 20, and TRd_{anc} to be 100. That means, $TRd_{nod} \times TRd_{anc}$=20 × 100=2000.

The metric used for measuring the accuracy of estimated distance is the deviation between the estimated distance and its real value. Assume that $d_{i,Nx}$ is the estimated distance between N_x and A_i,

$d_{i,Nx\text{True}}$ is the true distance from N_x to A_i. The deviation between $d_{i,Nx\text{True}}$ and $d_{i,\text{Nx}}$ is denoted as $\Delta d_{\text{i,Nx}}$. Thus, $\Delta d_{\text{i,Nx}}$ is used to judge the accuracy of $d_{i,\text{Nx}}$. Lower $\Delta d_{\text{i,Nx}}$ indicates better accuracy of $d_{i,\text{Nx}}$.

In general, if an anchor has smaller hop count to N_x, then this anchor is nearer to N_x. Thus, in order to know whether the nearest anchor has the most accurate estimated distance, we sort all the $\Delta d_{\text{i,Nx}}$ according to the hop count between N_x and A_i.

When the hop count between N_x and A_i is 1, we note down the corresponding 1-hop estimated distance, that is $d_{1\text{hop}}$, as well as the real distance which is $d'_{1\text{hop}}$. Then we calculate the deviation between $d_{1\text{hop}}$ and $d'_{1\text{hop}}$, that is $|d_{1\text{hop}} - d'_{1\text{hop}}|$. During the 2000 simulations, there exist many 1-hop estimated distances $d_{1\text{hop}}$, thus we can get many distance deviations $|d_{1\text{hop}} - d'_{1\text{hop}}|$. Then, the average value of all these 1-hop deviations, denoted as $\overline{\left|d_{1hop} - d'_{1hop}\right|}$, can be used to quantify the accuracy of 1-hop estimated distances. This average deviation of 1-hop estimated distances is listed in the first column of Table 3-2. In the same manner, we can obtain the average deviation of *i*-hop estimated distances. Theoretically, *i* can be any positive integer, but in reality, *i* can't be too large. Here, considering the area size 100×100 m^2 and the communication range of nodes 20m, we set the maximum of *i* to be 10. All the average deviations are listed in Table 3-2.

Table 3-2. Average Deviation of *i*-hop Estimated Distances

i-hop	1-hop	2-hop	3-hop	4-hop	5-hop	6-hop	7-hop	8-hop	9-hop	10-hop
Deviation (m)	3.80	4.23	4.91	5.54	6.11	6.57	7.20	9.14	14.64	28.92

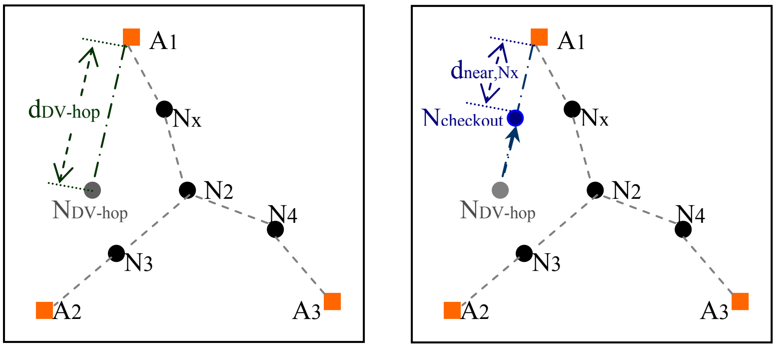
From Table 3-2, we can find that, generally, the deviation of estimated distance augments with the hop count. On average, the 1-hop estimated distance is the most accurate. That means, the nearest anchors to normal nodes has the most accurate estimated distance.

3.3.2 Principle of Checkout DV-hop Algorithm

The above conclusion is exploited for our proposed Checkout DV-hop algorithm. The basic principle of this algorithm is to correct the position of DV-hop algorithm based on the estimated distance to the nearest anchor.

Our algorithm adds a checkout step to DV-hop algorithm, as shown in Figure 3-13. For the purpose of comparison, Figure 3-13 (a) shows the result of DV-hop without "checkout", while Figure 3-13 (b) shows the impact of our checkout step. As shown in Figure 3-13 (a), the normal node N_x uses DV-hop to obtain its estimated position at $N_{\text{DV-hop}}$ with its coordinates denoted as (x', y'). It then calculates the distance between $N_{\text{DV-hop}}$ and the nearest anchor A_{near} (here, A_{near} is A_1), denoted as $d_{\text{DV-hop}}$. Note that alternatively, N_x has used equation (3.6) to get its estimated distance to A_{near}, denoted as $d_{\text{near,Nx}}$.

The purpose of the checkout step is to change the estimated position from $N_{\text{DV-hop}}$ (see Figure 3-13 (b)) to a new one called N_{checkout}, whose distance to A_{near} is $d_{\text{near,Nx}}$. To achieve this, the easiest and quickest way is to adjust the position along the line connecting $N_{\text{DV-hop}}$ and A_{near}. N_{checkout} is on the line from $N_{\text{DV-hop}}$ to A_{near}, and the distance between N_{checkout} and A_{near} is $d_{\text{near,Nx}}$. The position of A_{near} is (x_{Anear}, y_{Anear}) and $N_{\text{DV-hop}}$ is located at (x', y'), therefore the position of N_{checkout}, denoted as (x_{checkout}, y_{checkout}) can be derived as follows. N_{checkout} is finally the estimated position of N_x.



(a) DV-hop (b) Checkout DV-hop

Figure 3-13. Principle of Checkout DV-hop

$$\begin{cases} x_{checkout} = x' - \left(\dfrac{d_{\text{DV-hop}} - d_{\text{near},Nx}}{d_{\text{DV-hop}}}\right) \times (x' - x_{Anear}) \\ y_{checkout} = y' - \left(\dfrac{d_{\text{DV-hop}} - d_{\text{near},Nx}}{d_{\text{DV-hop}}}\right) \times (y' - y_{Anear}) \end{cases} \quad \textbf{(3.9)}$$

Our Checkout DV-hop algorithm comprises four steps. The fourth step, which is the checkout step, is proposed by us, while the first three steps are the same with DV-hop algorithm. The procedure of our algorithm is presented as follows.

Step #1: Initially, the system installer makes each anchor A_i be aware of its own position (x_i, y_i). Every node K (including anchors and normal nodes) holds a variable $hop_{i,K}$, which represents the minimal hop count from K to A_i. K initializes $hop_{i,K}$ as -1(if K is not A_i) or 0(if K is A_i). Here, $hop_{i,K}$ is the minimum hop count between K and A_i. Then, A_i broadcasts a message containing the position of A_i and a hop count field initialized as 0. This hop count value will increase with augment of hop during the broadcast of the message. That means, as soon as this message is received by a node, the hop count value in the message will be incremented. On the first reception of the message, every node K records the position of A_i, and initializes $hop_{i,K}$ as the hop count value (already incremented) in the message. If the same message is received again, K maintains $hop_{i,K}$. If the received message contains a lower hop count value (already incremented) than $hop_{i,K}$, K will update $hop_{i,K}$ with that lower hop count value, and relay the message. Otherwise, K will ignore the message. Through this mechanism, all the nodes in the network can get the minimum hop count to each anchor. Step #1 ends on condition that all nodes have received the position information from each anchor. Note: this ending condition is difficult to be implemented in real network scenarios, which will be discussed in the chapter 4.1.3.

Step#2: at this point, each normal node N_x knows its $hop_{i,Nx}$ (minimal hop count from A_i to N_x). Each anchor A_i has also obtained its minimal hop count to other anchors. So A_i can calculate its average distance per hop dph_i, and then broadcasts dph_i throughout the network. After receiving dph_i, the normal node N_x can use equation (3.6) to get $d_{i,Nx}$, which is the approximate distance between N_x and each anchor A_i. If A_{near} is the nearest anchor to N_x, their estimated distance $d_{near,Nx}$ will be used in the fourth step (our checkout step).

Step #3: The normal node N_x can use the estimated distances to anchors to calculate its estimate position $N_{DV\text{-}hop}$ (x', y'). The details of the calculation can be found in section 2.2.2.4.1.

Step #4: Finally, with our proposed checkout step, the normal node N_x calculates the distance between $N_{DV\text{-}hop}$ and A_{near}, denoted as $d_{DV\text{-}hop}$. Because N_x already knows $d_{i,Nx}$, A_i's position (x_i, y_i), $N_{DV\text{-}}$

hop's position (x', y'), and $d_{\text{DV-hop}}$, N_x uses the equation (3.9) to calculate N_{checkout}, which is the final estimated position of N_x.

The performance evaluation of Checkout DV-hop algorithm, in terms of localization accuracy and computation complexity, will be presented in the section 3.5.4 and 3.6.2.

3.4 Selective 3-Anchor DV-hop Algorithm

Since the accuracy improvement by Checkout DV-hop is not so considerable [GWV 10], we have proposed Selective 3-Anchor DV-hop algorithm [GWV 11]. First, this algorithm generates a group of candidates. Then, from this pool, it chooses one based on its connectivity vector.

In order to facilitate our presentation of this algorithm, we first give a typical network example. Then, we introduce two basic elements: 3-anchor group and 3-anchor estimated position. And then, the principle of our algorithm is presented.

3.4.1 Network Example

In order to present the principle of our algorithm, we use a typical example of network topology as shown in Figure 3-14. The network operates in a $50 \times 50\text{m}^2$ area, with a total of 10 nodes randomly distributed inside. The maximum communication range of all the nodes is set to 20m. Among the 10 nodes, 4 are anchors A_1 A_2 A_3 and A_4, who already know their positions. The remaining units are normal nodes N_1 N_2… N_6. These normal nodes do not know their positions.

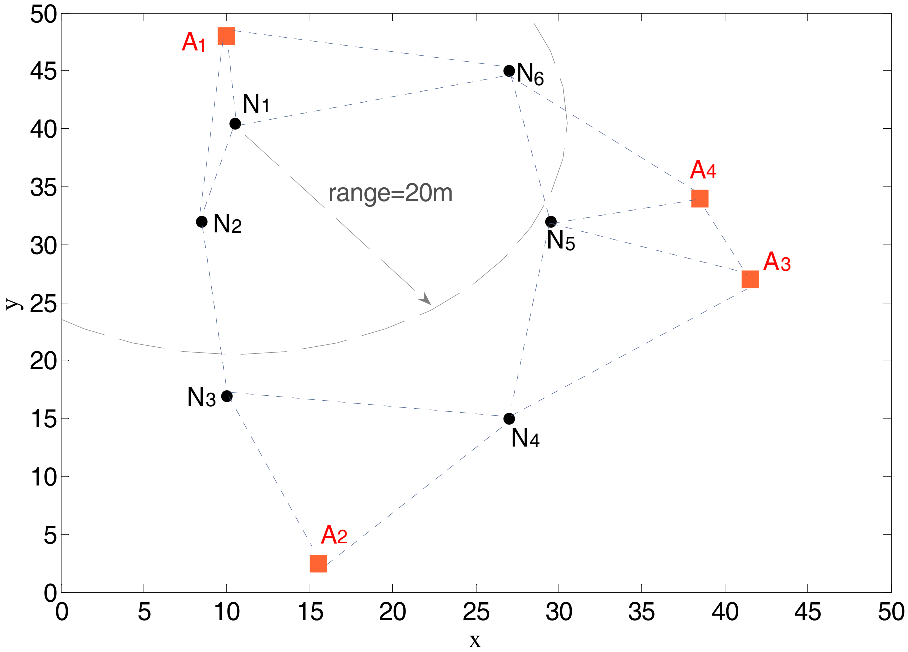


Figure 3-14. Example of Nodes Distribution

The range-free localization scheme is generally based on the exchange of connectivity information between nodes. For example, in Figure 3-12, because of the constraint of communication range, N_1 can only find A_1, N_2 and N_6 in its neighborhood. That means, N_1 can only directly connect to A_1 N_2 and N_6. Based on exchanges of connectivity information, we can turn Figure 3-14 into a connectivity graph, represented by the lines of dashes in the same figure.

3.4.2 3-Anchor Groups and 3-Anchor Estimated Positions

Let's consider a network with m_d anchors A_1 A_2 ... A_{md}. Through the first two steps of DV-hop, a normal node N_x can obtain $hop_{i,Nx}$, which is its minimum hop count to each anchor A_i, as well as $d_{i,Nx}$, which is the estimated distance between N_x and A_i. Then, N_x can calculate its estimated position $N_{DV\text{-}hop}$ by trilateration based on the m estimated distance values $d_{1,Nx}$ $d_{2,Nx}$ ... $d_{md,Nx}$. So, the quality of these estimated values has a great influence on the accuracy of DV-hop.

In fact, instead of using all m_d estimated values, three estimated distance values to three different anchors are sufficient for N_x to calculate its position. For example, we use $d_{i,Nx}$, $d_{j,Nx}$, $d_{k,Nx}$, which are the three estimated distance values from N_x to the three corresponding anchors A_i, A_j, A_k. If we denote the true position of N_x as (x, y), and the positions of A_i A_j A_k respectively as (x_i, y_i), (x_j, y_j), (x_k, y_k), then we can have the following equation:

$$\begin{cases}(x-x_i)^2+(y-y_i)^2=d_{i,\,Nx}{}^2\\(x-x_j)^2+(y-y_j)^2=d_{j,\,Nx}{}^2\\(x-x_k)^2+(y-y_k)^2=d_{k,\,Nx}{}^2\end{cases} \tag{3.10}$$

Solving (3-10) by trilateration, we can get a 3-anchor estimated position of N_x, denoted as $N_{<i,j,k>}$ $(x_{<i,j,k>}, y_{<i,j,k>})$. It is calculated as:

$$N_{<i,j,k>}:\ \begin{bmatrix}x_{<i,j,k>}\\y_{<i,j,k>}\end{bmatrix} = C^{-1}B,$$

$$\text{and } C=-2\times\begin{bmatrix}x_i-x_k & y_i-y_k\\x_j-x_k & y_j-y_k\end{bmatrix},\ B=\begin{bmatrix}d_{i,\,Nx}^2-d_{k,\,Nx}^2-x_i^2-y_i^2+x_k^2+y_k^2\\d_{j,\,Nx}^2-d_{k,\,Nx}^2-x_j^2-y_j^2+x_k^2+y_k^2\end{bmatrix} \tag{3.11}$$

Where the dimension of matrix C is 2 by 2, and that of matrix B is 2 by 1. Here, it should be mentioned that the three anchors A_i A_j A_k cannot be collinear. Otherwise, matrix C will be singular.

Among the m_d available anchors, if we select any three anchors to form a 3-anchor group, then there are totally $C_{m_d}^3$ groups. Using the equation (3.11), based on each group, N_x can generate a 3-anchor estimated position. So, totally N_x can have $C_{m_d}^3$ 3-anchor estimated positions. They are all candidate positions for N_x.

Some 3-anchor estimated positions of N_x have much higher accuracy than $N_{DV\text{-}hop}$, and some others are not so accurate. For example, based on Group<A_1, A_2, A_3>, Group<A_1, A_2, A_4>, Group<A_2, A_3, A_4>, and Group<A_1, A_3, A_4>, N_x, which corresponds to N_1 in Figure 3-14, can get its 3-anchor estimated positions respectively $N_{1<1,2,3>}$, $N_{1<1,2,4>}$, $N_{1<2,3,4>}$, and $N_{1<1,3,4>}$. Table 3-3 lists these estimated positions and their corresponding location errors. We can note that $N_{1<1,2,3>}$ is much more accurate than other estimated positions.

Table 3-3. Examples of 3-Anchor Estimated Positions for N_1

3-anchor estimated positions (m)	Location Error (m)
$N_{1<1,2,3>}$ (7.77, 44.82)	5.11
$N_{1<1,2,4>}$ (18.44, 46.11)	9.72
$N_{1<2,3,4>}$ (0, 73.92)	35.03
$N_{1<1,3,4>}$ (45.90, 102.02)	70.98
DV-hop estimated position (m)	10.23
$N_{1,\,DV\text{-}hop}$ (17.30, 48.14)	

Here, the location error is defined as the Euclidean distance between a normal node's estimated position and its real position. For example, N_1 (*10.50, 40.50*) (in Figure 3-14) can use DV-hop method to obtain its estimated position $N_{1,DV\text{-}hop}$ (*17.30, 48.14*). Then, the location error of $N_{1,DV\text{-}hop}$ is calculated as $\sqrt{(10.50-17.30)^2+(40.50-48.14)^2}$ =*10.23*.

Note that from Table 3.3, A_4 has no contribution to the best estimation of N_1 though it is at a smaller hop distance than A_3. Based on our observations above, our selective 3-Anchor DV-hop algorithm will select the most accurate 3-anchor estimated position and regard it as the final estimated position.

3.4.3 Principle of Selective 3-Anchor DV-hop Algorithms

3.4.3.1 Position vs. Connectivity

Range-free localization schemes are based on two kinds of information: anchors' positions, and the connectivity between nodes. In DV-hop, the connectivity of N_x is specified as the minimum hop counts between N_x and each anchor. Since this chapter focuses on the algorithms based on DV-hop, the connectivity mentioned will be considered as an array which contains the minimum hop counts to anchors. For example, if there are totally m_d anchors, and the minimum hop count from N_x to each anchor A_i is $hop_{i,Nx}$, then the connectivity of N_x is the array [$hop_{1,Nx}$, $hop_{2,Nx}$ ... $hop_{md,Nx}$].

In fact, the connectivity of a normal node can identify its position. For example, from Figure 3-14, the connectivity of each normal node can be observed. The results are summarized in Table 3-4. From this table, we can find that each normal node has a unique connectivity, which allows us to identify its position.

Since the connectivity of a normal node can represent its position, if two normal nodes have similar connectivities, then they must have similar positions. It means, they are very near to each other.

So, we can deduce the relationship between connectivity difference and the distance: smaller connectivity difference between two normal nodes will result in smaller distance between them.

Table 3-4. Connectivities of Normal Nodes

Normal Node	Connectivity
N_1	[1, 3, 3, 2]
N_2	[1, 2, 3, 3]
N_3	[2, 1, 2, 3]
N_4	[3, 1, 1, 2]
N_5	[2, 2, 1, 1]
N_6	[1, 3, 2, 1]

Then, we use the sum of absolute difference to quantify the connectivity difference. For example, in Table 3-4, the connectivity difference between N_1 and N_2 is |1-1|+|3-2|+|3-3|+|2-3| = 2. According to our conclusion, this small connectivity difference indicates a small distance between N_1 and N_2, which then can be observed from Figure 3-14.

To give a better understanding of this conclusion, we here investigate the relationship between N_x (N_1 in Figure 3-14) and any other normal node. From Table 3-4, we can calculate the connectivity difference between N_1 and any other normal node. The results are listed in Table 3-5. In this table, we also give the distance value between N_1 and any other normal node. Comparing the last two lines, we can find that larger connectivity difference always reflects the longer distance between two normal nodes. For example, the connectivity difference between N_3 and N_1 is bigger than that between N_2 and N_1. Correspondingly, N_3 is further to N_1 than N_2.

Table 3-5. Connectivity Difference and Distance to N_1

Normal Node	N_2	N_6	N_5	N_3	N_4
Connectivity Difference to N_1	2	2	5	5	6
Distance to N_1 (m)	8.73	17.10	20.36	23.51	30.37

This conclusion on the relation between the distance and connectivity difference can be used for finding the most accurate 3-anchor estimated position. In a network with m_d anchors, a normal node N_x has totally $C_{m_d}^{3}$ 3-anchor estimated positions. Each 3-anchor estimated position denoted as $N_{<i,j,k>}$, can be a candidate position for N_x. According to the conclusion, the 3-anchor estimated position which has the smallest connectivity difference to N_x must be nearest to N_x, and as a result, will be the most accurate estimated position. So, in order to estimate the position of N_x, the basic principle of our selective 3-anchor DV-hop algorithm is to choose the 3-anchor estimated position which has the smallest connectivity difference to N_x.

However, the connectivity of each 3-anchor estimated position is still unknown. That is, the hop count from $N_{<i,j,k>}$ to each anchor is unknown. We therefore need to find a method for N_x to calculate the hop count between $N_{<i,j,k>}$ and each anchor.

3.4.3.2 Hop Count for 3-Anchor Estimated Position

Through the first two steps of DV-hop, N_x can obtain the anchors' positions as well as its minimum hop counts to all anchors. Section 3.4.2 allows N_x to obtain for various anchors triplets A_i, A_j, A_k, different estimated positions $N_{<i,j,k>}$. Then to select the best candidate among those various estimates, we now elaborate a virtual hop estimate in order to select the minimum connectivity difference estimate; the problem of calculating the estimate hop count between any estimate $N_{<i,j,k>}$ and any anchor A_t can be linked to a classical calculation of distance per hop as follows; as N_x knows the distance between any 3-anchor estimate $N_{<i,j,k>}$ and any anchor A_t, the hop count between them can be estimated as:

$$hop_{<i,j,k>,t} = \frac{d_{<i,j,k>,t}}{dph_{<i,j,k>,t}} \tag{3.12}$$

where $hop_{<i,j,k>,t}$ is the hop count between $N_{<i,j,k>}$ and A_t, and $dph_{<i,j,k>,t}$, is their distance per hop.

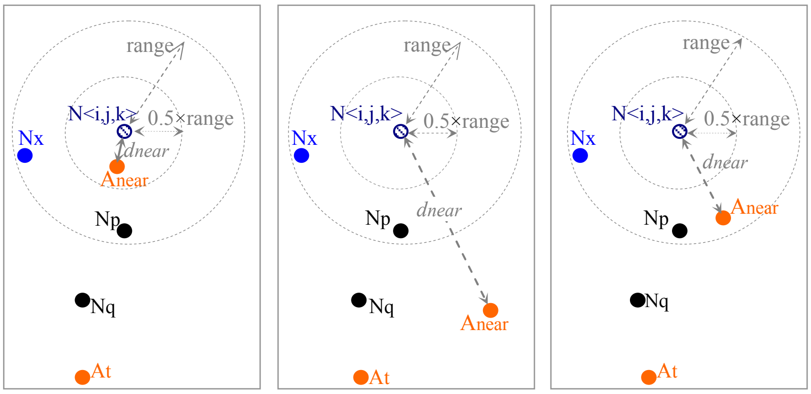
We must then find a method to estimate the value of $dph_{<i,j,k>,t}$. In fact, all the distance-per-hop information that N_x has obtained are anchors' distance-per-hop values: dph_1, dph_2, ..., dph_{md}, including the distance per hop of A_t denoted as dph_t.Thus, we need to estimate $dph_{<i,j,k>,t}$ based on the anchors' distance-per-hop values.

In order to get an approximate value of $dph_{<i,j,k>,t}$, three kinds of relative positions between $N_{<i,j,k>}$ and its nearest anchor A_{near} are considered, based on the euclidean distance between $N_{<i,j,k>}$ and A_{near}. In the first case, the euclidean distance between $N_{<i,j,k>}$ and A_{near} is so small that we can use the distance-per-hop value of A_{near} (denoted as dph_{near}) as an approximate value of $dph_{<i,j,k>,t}$. Here, as an example, we can set the distance threshold as half of the radio range of nodes. The second case is the opposite: the Euclidean distance between $N_{<i,j,k>}$ and A_{near} is so large that we can only use dph_t as an approximate value of $dph_{<i,j,k>,t}$. Here, also as example, the threshold of distance is set as the radio range of nodes. Since the third case is between the above two cases, the value of $dph_{<i,j,k>,t}$, in the third case can be set as the average of dph_{near} and dph_t. These three cases are shown in Figure 3-15.

In Figure 3-15, $N_{<i,j,k>}$ is a 3-anchor estimated position of the normal node N_x, while N_p and N_q are two other normal nodes which connect N_x and A_t. Summarizing the three cases, we can estimate the value of $dph_{<i,j,k>,t}$ as follow:

$$dph_{<i,j,k>,\,t} \approx \begin{cases} dph_{near} & \text{, when } d_{near} < range/2 \\ dph_t & \text{, when } d_{near} > range \\ (dph_{near} + dph_t)/2 & \text{, others} \end{cases} \quad \textbf{(3.13)}$$

where d_{near} is the distance between $N_{<i,j,k>}$ and A_{near}, dph_{near} is the distance per hop of A_{near}.



4(a) d_{near}<range/2 4(b) d_{near} >range 4(c) range/2< d_{near} <range

Figure 3-15. Three Kinds of Relative Positions

Using the equations (3.12) and (3.13), N_x can obtain $hop_{<i,j,k>,t}$, which is the estimated hop count between $N_{<i,j,k>}$ and each anchor A_t. Then, the connectivity difference between $N_{<i,j,k>}$ and N_x can be calculated as $\sum_{t=1}^{m_d} \left| hop_{\{i,j,k\},t} - hop_t \right|$. Then, from the $C_{m_d}^3$ 3-anchor estimated positions, N_x selects the position having the smallest connectivity difference as the final estimated position.

3.4.3.3 Procedure of the Algorithm

The procedure of our Selective 3-Anchor DV-hop algorithm is summarized as follows. The first and second steps are the same as DV-hop algorithm. In the third step, a normal node N_x selects any three non-collinear anchors to form a 3-anchor group, and correspondingly generates a 3-anchor estimated position. Then, based on equations (3.12) and (3.13), N_x calculates the connectivity of each 3-anchor estimated position. Finally, N_x chooses the best 3-anchor estimated position which has the smallest connectivity difference to N_x.

For better understanding the procedure, we give an example. The example scenario is the same as Figure 3-14. The coordinates of the 4 anchors (randomly distributed) are: A_1 (10, 48), A_2 (15.5, 2.5), A_3 (41.5, 27), and A_4 (38.5, 34). We assume that the concerned normal node N_x is the node N_1.

At Step #1 of our algorithm (this step is the same as DV-hop), each anchor broadcasts its position throughout the network. Thus, at the end of Step #1, every node (including anchors) knows its hop counts to each anchor as well as the positions of anchors. For example, A_2 can know its hop counts to A_1, A_3, A_4 respectively 3, 2, 3, while N_1 can know its hop counts to A_1, A_2, A_3, A_4 respectively 1, 3, 3, 2. This means, N_1's connectivity is [1, 3, 3, 2].

At Step #2 (this step is also the same as DV-hop), each anchor first calculates its distance-per-hop value, then broadcasts this value to the entire network. For example, the four anchors (A_1, A_2, A_3, and A_4) can use the equation (3.7) to obtain their distance-per-hop values respectively as: 14.43, 15.07, 13.53, and 13.06. These values are then broadcast by their corresponding anchors.

The Step #3 is contributed by our Selective 3-Anchor DV-hop algorithm. In this example, N_1 first selects any three anchors to form 3-anchor groups. Thus, four group are generated: Group<A_1, A_2, A_3>, Group<A_1, A_2, A_4>, Group<A_2, A_3, A_4>, and Group<A_1, A_3, A_4>. Based on these groups, N_1 can use the equation (3.11) to get its 3-anchor estimated positions: $N_{1<1,2,3>}$, $N_{1<1,2,4>}$, $N_{1<2,3,4>}$, and $N_{1<1,3,4>}$. Their coordinates are listed in the first column of the following Table 3-6. Then, using the equations (3.12) and (3.13), N_1 calculates the connectivity of each 3-anchor estimated position. These connectivity results are listed in the second column of Table 3-6. Thus, the absolute connectivity difference between N_1 and its 3-anchor estimated position can be obtained, as shown in the third column of Table 3-6. Finally, comparing the connectivities in Table 3-6, N_1 chooses $N_{1<1,2,3>}$ (7.77, 44,82) to be its estimated position, because $N_{1<1,2,3>}$ has the smallest connectivity difference to N_1.

In fact, comparing Table 3-3 with Table 3-6, we can verify that, the final choice by Selective 3-Anchor algorithm is correct. Table 3-6 tells us that the final choice $N_{1<1,2,3>}$ has the most similar connectivity to N_1, while Table 3-3 shows that $N_{1<1,2,3>}$ is closest to N_1. This also proves our previous conclusion: "similar connectivity" indicates "shorter distance".

Table 3-6. Connectivity Differences with N_1

<table>
<tr><th>3-anchor Estimated Positions (m)</th><th>Connectivity</th><th>Connectivity Difference with N_1 [1,3,3,2]</th></tr>
<tr><td>$N_{1<1,2,3>}$ (7.77, 44.82)</td><td>[1, 2.98=43.02/14.43, 2.64=38.15/14.43, 2.26=32.58/14.43]</td><td>0.64=|1-1|+|2.98-3|+|2.64-3|+|2.26-2|</td></tr>
<tr><td>$N_{1<1,2,4>}$ (18.44, 46.11)</td><td>[1, 3.03=43.71/14.43, 2.08=29.95/14.43, 1.62=23.43/14.43]</td><td>1.33=|1-1|+|3.03-3|+|2.08-3|+|1.62-2|</td></tr>
<tr><td>$N_{1<2,3,4>}$ (0, 73.92)</td><td>[1.93=27.78/14.43,4.85=73.08/15.06,4.63=62.64/13.53,4.25=55.46/13.05]</td><td>5.73=|1.93-1|+|4.85-3|+|4.63-3|+|4.25-2|</td></tr>
<tr><td>$N_{1<1,3,4>}$ (45.90, 102.02)</td><td>[4.49=64.86/14.43,6.91=104/15.06, 5.55=75.15/13.53, 5.24=68.42/13.05]</td><td>13.19=|4.49-1|+|6.91-3|+|5.55-3|+|5.24-2|</td></tr>
</table>

The program procedure of Selective 3-Anchor DV-hop algorithm can be described as:

```
Algorithm "Selective 3-Anchor DV-hop":
suppose in the network there are m_d anchors, A_1, A_2 ..., A_md.
 for i ← 1 to (m-2)
     A_i is chosen. (x_i, y_i) is the position of A_i.
     for j ← (i+1) to (m-1)
         A_j is chosen. (x_j, y_j) is the position of A_j.
         for k ← (j+1) to m
             A_k is chosen. (x_k, y_k) is the position of A_k.
             N_x calculates an estimated position N_<i,j,k> based on Equation (3.11).
             The connectivity of N_<i,j,k> is calculated based on Equation (3.12) and (3.13).
             The connectivity difference between N_<i,j,k> and N_x can be calculated.
 return the N_<i,j,k> which has the smallest connectivity difference
```

Figure 3-16. Procedure of Selective 3-Anchor DV-hop Algorithm

We should mention an exceptional case concerning the very low ratio of anchors. For example, let's consider a network with 100 nodes, with only 5 of them being anchors. With such a few anchors, the connectivity information collected by a normal node is very limited. Thus, several 3-anchor estimated positions of a normal node may have the same connectivity. That means, normal nodes don't have enough connectivity information to select their best estimate positions. In this case, since our Selective 3-Anchor DV-hop algorithm doesn't perform well, we suggest that Checkout DV-hop algorithm be used.

The performance evaluation of Selective 3-Anchor DV-hop algorithm, in terms of localization accuracy and computation complexity, will be presented in the section 3.5.5 and 3.6.2.

3.5 Computation Complexity of Range-free Algorithms

Because of limitation on the size and the cost, sensor nodes always have limited capacity of computation, which makes them sensitive to complicated algorithms. Thus, in this section, we analyze the computation complexity of the range-free localization algorithms. The algorithms considered in this section include Centroid, CPE, DV-hop, our proposed Mid-Perpendicular, Checkout DV-hop and Selective 3-Anchor DV-hop.

The study of an algorithm's complexity involves determining the amount of resources (such as time and storage) necessary to execute it. Theoretically, it is commonly expressed using "O" notation, which suppresses multiplicative constants and lower order terms [SB 09]. For example, if the number of elementary operations required by an algorithm on all inputs of size m is at most $5m^3 + 3m$, then its calculation complexity is $O(m^3)$. The following is the detailed analysis of calculation complexity for the related algorithms.

3.5.1 Complexity of Centroid Algorithm

The computation in Centroid Algorithm is referred to the equation (2.7) in section 2.2.2.1. In this equation, the calculation of x_{cen} (N_x's x-axis coordinate by Centroid Algorithm) involves two elementary operations: "+" and "/". The number of "+" operation is $m-1$, and the number of "/" operation is 1, if N_x has m neighbor anchors in total.

Then the amount of elementary operations for calculating x_{cen} is ($m-1$) **"+"** and one **"/"**. For y_{cen}, the same result can be obtained. So, the total amount of elementary operations for Centroid algorithm is $2(m-1)$ **"+"** and 2 **"/"**. Finally, we can conclude that the calculation complexity of Centroid is $O(m)$.

3.5.2 Complexity of CPE Algorithm

We can find all the calculation of CPE algorithm from the equation (2.11) in section 2.2.2.2. In this equation, the computation of x_{CPE} (N_x's x-axis coordinate by CPE algorithm) demands three elementary operations: "comparison", "+" and "/".

To calculate x_{CPE}, it is necessary to obtain $\max_{i=1}^{m}(x_i)$ and $\min_{i=1}^{m}(x_i)$, assumed that N_x has m neighbor anchors in total. $\max_{i=1}^{m}(x_i)$ is the maximum value among x_1 x_2 … x_m, while $\min_{i=1}^{m}(x_i)$ is their minimum value. In order to get these two values, we first compare x_1 and x_2. Here, without loss of generality, we assume that $x_1 > x_2$. Thus, after this first comparison operation, the temporary maximum value is set to be x_1, and the temporary minimum value to be x_2. Then, for each x_i among x_3 x_4 … x_m, we compare x_i with the temporary maximum value x_1. If x_i is greater, then x_i is assigned to be the temporary maximum value. Otherwise, x_i will be compared with the temporary minimum value x_2. If x_i is smaller than x_2, then x_i is assigned to be the temporary minimum value. Therefore, for each x_i, 1 or 2 comparison operations are needed, thus the average number of operations is 3/2. As a result, in order to obtain $\max_{i=1}^{m}(x_i)$ and $\min_{i=1}^{m}(x_i)$, the number of comparison operations should be *1+3/2×(m-2)=3/2×m-2*. In addition, referred from the equation (2.11), to calculate x_{CPE}, we need another one "+" operation and one "/" operation. As a result, the amount of elementary operations for x_{CPE} is (*3/2×m-2*) **"**compare**",** one **"+"**, and one **"/"**.

For the calculation of y_{CPE}, the same result can be obtained. So, the total amount of elementary operations for CPE algorithm is (*3×m-6)* **"**compare**",** 2 **"+"**, and 2 **"/"**. Finally, we can conclude that the calculation complexity for CPE algorithm is O(*m*), which is the same as Centroid algorithm.

3.5.3 Complexity of Mid-perpendicular Algorithm

Assume that the normal node N_x has *m* neighbor anchors. If *m* is 3, then the position of N_x is calculated by Mid-perpendicular algorithm as the equation (3.5) in section 3.2.2.1. This equation gives two cases. As the first case, the three neighbor anchors form an acute triangle, thus the equation (3.3) is utilized. From the equation (3.3), we can find that, the computation of x_{mid} (N_x's x-axis coordinate by Mid-perpendicular algorithm) demands four elementary operations: "+", "—", "×", and "/". Their amounts are respectively 4, 8, 10, and 1. As for y_{mid}, the same result can be obtained. So, the amount of elementary operations for the equation (3.3) is 8 **"+"**, 16 **"—"**, 20 **"×"** and 2 **"/"**. As the second case of the equation (3.5), the three neighbor anchors form a right triangle or an obtuse triangle. In this case, we should use the equation (3.4), which has just one "+" operation and one "/" operation to compute x_{mid} or y_{mid}. Thus, in the second case, the amount of elementary operations is 2 **"+"** and 2 **"/"**. The average number of elementary operations for the two cases is 5 **"+"**, 8 **"—"**, 10 **"×"** and 2 **"/"**. This is the result for Mid-perpendicular algorithm in case of *m*=3.

When *m* is larger than 3, two extended versions of Mid-perpendicular algorithm are discussed in the section 3.2.2.2.

One is the direct version: from the *m* neighbor anchors, we first select any three of them, thus there are in total C_m^3 groups; then based on the three anchors in each group, we calculate a position based on the equation (3.5); finally, the average of all these C_m^3 positions is regarded as the final estimated postion of N_x. So, the total number of elementary operations is $(6C_m^3-1)$ **"+"**, $8C_m^3$ **"—"**, $10C_m^3$ **"×"** and $(2C_m^3+1)$ **"/"**. As a result, the complexity for this direct version of Mid-perpendicular algorithm can be denoted as O(m^3).

The other is the simplified version: N_x first finds out the two farthest anchors, which needs to compare the distance between any two anchors, requiring C_m^2 **"+"**, $2C_m^2$ **"—"**, $2C_m^2$ **"×"** and (C_m^2-1) **"compare"**; then N_x finds the third anchor which has the longest distance to the line connecting the two farthest anchors, requiring *3*(*m-2*) **"+"**, *6*(*m-2*) **"×"**, (*m-2*) **"/"** and (*m-3*) **"compare"**; finally N_x calculates the position by the equation (3.5) based on the three anchors founded. So, in total, the amount of elementary operations is (C_m^2+3m-1) **"+"**, $(2C_m^2+8)$ **"—"**, $(2C_m^2+6m-2)$ **"×"**, *m* **"/"** and (C_m^2+m-4) **"compare".** Thus, the complexity for this simplified version of Mid-perpendicular algorithm can be denoted as O(m^2).

We recommend the simplified version is utilized, thus the complexity of Mid-perpendicular algorithm can be regarded as O(m^2).

3.5.4 Complexity of DV-hop Algorithm

The computation for a node to be localized with DV-hop algorithm is included in third step, as shown by the equation (2.16) in the section 2.2.2.4.1. In this equation, the matrix computation should be analyzed. The matrix A is a $(m_d\,-1)$ by 2 matrix, A^T is a 2 by (m_d-1) matrix, and B is a $(m_d\,-1)$ by 1 matrix.

Since each element in matrix A is an expression which demands one "—" operation and one "×" operation, the amount of elementary operations ("—" and "×") in matrix A is $2(m_d\,-1)$ **"—"** *and* $2(m_d\,-1)$ **"×"**. The amount of elementary operations ("+", "—" and "×") in matrix B is $2(m_d\,-1)$ **"+"**, $3(m_d\,-1)$ **"—"**, and $3(m_d+1)$ **"×"**. $A^T A$, that is the multiplication of two matrixes A and A^T, demands $4(m_d\,-1)$ **"×"** and $4(m_d\,-2)$ **"+"**. Since $A^T A$ is a 2 by 2 matrix, its inverse $(A^T A)^{-1}$ only needs one **"—"**, 4 **"/"** and 4 **"×"**. Then the multiplication of $(A^T A)^{-1}$ and A^T needs $2(m_d\,-1)$ **"+"** and $4(m_d\,-1)$ **"×"**. The multiplication of $(A^T A)^{-1} A^T$ and B needs $2(m_d\,-1)$ **"×"** and $2(m_d\,-2)$ **"+"**.

As a result, the equation (2.16) totally demands $(10m_d\,-16)$ **"+"**, $(5m_d\,-4)$ **"—"**, $(15m_d\,-5)$ **"×"**, and 4 **"/"**. So, the calculation complexity for DV-hop algorithm is $O(m_d)$.

3.5.4 Complexity of Checkout DV-hop Algorithm

While DV-hop algorithm has three steps, our proposed Checkout DV-hop method adds the fourth step. The equation (3.9) indicates that the fourth step has only 6 **"—"**, 2 **"×"**, and 2 **"/"**. Thus, the total amount of elementary operations for Checkout DV-hop is $(10m_d\,-16)$ **"+"**, $(5m_d\,+2)$ **"—"**, $(15m_d\,-3)$ **"×"**, and 6 **"/"**. So, its calculation complexity is still $O(m_d)$, the same as DV-hop algorithm.

3.5.5 Complexity of Selective 3-Anchor DV-hop Algorithm

Our Selective 3-Anchor DV-hop algorithm has three steps. The first two steps are the same as DV-hop algorithm. The difference lies on the third step, which includes most computations of our algorithm. In the third step, the normal node N_x first selects any three anchors to generate a 3-anchor estimated position. Based on the equation (3.11), the calculation of a 3-anchor estimated position requires 14 **"+"**, 11 **"—"**, 40 **"×"**, and 4 **"/"**. At most, as many as $C^3_{m_d}$ 3-anchor estimated positions can be generated. Then, N_x calculates the connectivity difference of each estimated position, requiring (m_d-1) **"+"**, m_d **"—"**, and m_d **"/"**. Finally, N_x chooses the best 3-anchor estimated position which has the smallest connectivity difference to N_x, requiring ($C^3_{m_d}\,-1)$ **"compare".**

Thus, in total, the amount of elementary operations is $C^3_{m_d}\,(m_d\,-13)$ **"+"**, $C^3_{m_d}\,(m_d+11)$ **"—"**, $40\,C^3_{m_d}$ **"×"**, $C^3_{m_d}\,(m_d+4)$ **"/"** and ($C^3_{m_d}\,-1)$ **"compare"**. We can conclude that, the complexity of Selective 3-Anchor DV-hop algorithm is $O(m_d^4)$.

3.5.6 Comparison of the Complexity

In this section, we compare the computation complexity of the above algorithms. All the theoretical analysis results are listed in the following Table 3-7.

Shown in Table 3-7, for class-1 normal nodes, Centroid and CPE algorithms have the lowest computation complexity, while our Mid-perpendicular algorithm is more complicated.

Table 3-7. Computation Complexity of Range-free Algorithms

Range-free Algorithms		Number of Elementary Operations	Complexity
Algorithms for Class-1 Nodes	Centroid	$2(m\text{-}1)$ "+", 2 "/"	O(m)
	CPE	2 "+", 2 "/", $(3m\text{-}6)$ "compare"	O(m)
	Mid-perpendicular	$(C_m^2+3m\text{-}1)$ "+", $(2C_m^2+8)$ "—", $(2C_m^2+6m\text{-}2)$ "×", m "/", $(C_m^2+m\text{-}4)$ "compare"	O(m^2)
Algorithms for Class-2 Nodes	DV-hop	$(10m_d\text{-}16)$ "+", $(5m_d\text{-}4)$ "—", $(15m_d\text{-}5)$ "×", 4 "/"	O(m_d)
	Checkout DV-hop	$(10m_d\text{-}16)$ "+", $(5m_d+2)$ "—", $(15m_d\text{-}3)$ "×", 6 "/"	O(m_d)
	Selective 3-Anchor DV-hop	$C_{m_d}^3(m_d\text{-}13)$ "+", $C_{m_d}^3(m_d+11)$ "—", $40C_{m_d}^3$ "×", $C_{m_d}^3(m_d+4)$ "/", $(C_{m_d}^3\text{-}1)$ "compare"	O(m_d^4)

As a popular algorithm for class-2 normal nodes, DV-hop algorithm is more complicated than Centroid and CPE, having more elementary operations. However, the complexity level of DV-hop algorithm is still O(m_d). Compared with DV-hop algorithm, our Checkout DV-hop algorithm just slightly increases the computation, thus its complexity remains in the level of O(m_d). However, our Selective 3-Anchor DV-hop algorithm puts much more effort on the accuracy improvement, thus having the complexity as high as O(m_d^4).

3.6 Evaluation on Accuracy of Range-free algorithms by MATLAB

In this section, we evaluate and compare the performance of the concerned range-free algorithms. This is fulfilled through simulations using a mathematic simulation tool MATLAB. Thus, the simulations in this section have ideal scenarios: ideal radio propagation without path loss or interference, no mobility for nodes, and no frame collisions. Because of the simplicity, research works on range-free algorithms usually prefer to use MATLAB and apply these ideal scenarios. To adapt in practical scenarios, the range-free algorithms discussed before should be designed and modified into range-free protocols. These protocols will be introduced in Chapter 4 as well as more practical evaluations using network simulator.

In the following, we first evaluate the algorithms for class-1 nodes, including Centroid, CPE and Mid-perpendicular. Then, the algorithms for class-2 nodes, such as DV-hop, Checkout DV-hop and Selective 3-Anchor DV-hop, will be investigated in terms of accuracy and computation complexity. Finally, we propose to combine these two classes of algorithms, so that more adaptive performance will be expected.

3.6.1 Performance of Algorithms for Class-1 Nodes

A class-1 normal node has at least 3 neighbor anchors. To make sure the anchors locate inside the range of the normal node, a special simulation area is configured in this section. We denote the communication range (radius, not diameter) of nodes "*range*". Assume that the normal node locates at the centre of the simulation area. Then, we set the side length of the square simulation area to be "*2×range*", so that the anchors in this simulation area probably resides within the radio range of the normal node.

In this section, several scenarios will be applied from different point of view to investigate the performance of algorithms for class-1 nodes. The main parameters of all these scenarios are listed in the following Table 3-8. Most of the parameters are shared by these scenarios, while other parameters marked with "*" vary with each scenario. From the table, we can see that, *range* is set to be 20m, and the simulation area is 40×40m^2 Square Area. The real position of the normal node is (20m, 20m), which is also the centre of this simulation area. The "random simulation number" means the number of simulations in a scenario. During each simulation, the anchors are uniform-randomly distributed inside the area. That means, in a specific scenario, the positions of anchors in one simulation will be different from those in another simulation. So, "random simulation number" is also the number of geographic distribution of anchors.

Table 3-8. Scenario Parameters for Class-1 Algorithms

Scenario Parameters	Values
Node Radio Range	20 meters
Simalation Area	40m×40m Square Area
Radio Propagation	Ideal, no pathloss, no interference
Real position of Nx	(20m, 20m)
* Number of Anchors "*m*"	to be decided in specific scenario
* Random Simulation Number	to be decided in specific scenario

In the following, we will present each scenario with the corresponding simulation results for the range-free algorithms (Centroid, CPE, and Mid-perpendicular).

3.6.1.1 Scenario 1 for Class-1 Localization

The parameters of the first scenario have already been listed in Table 3-8. Here, we give the values of those particular parameters (marked with "*"): the number of neighbor anchors "*m*" is 3, and the random simulation number is 1, which means that only one random distribution of anchors will be obtained. Here, we configure the random simulation number to be only 1, because we want to investigate the algorithms performance in a particular case. The geographic distribution of anchors as well as the normal node is shown in Figure 3-17.

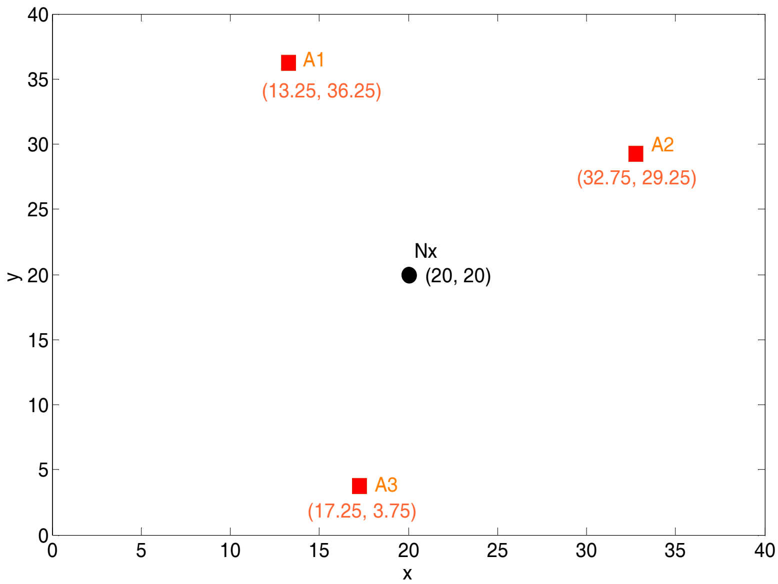


Figure 3-17. Nodes Distribution for Scenario 1 with 3 Neighbor Anchors

Assume that the normal node N_x has communicated with the anchors and known their positions respectively as A_1 (13.25, 35.25), A_2 (32.75, 29.25), and A_3 (17.25, 3.75). The unit is meter. Then, based on the equations (2.7), (2.11), and (3.5), N_x can calculate its estimated positions respectively by the algorithms Centroid, CPE, and Mid-perpendicular. The accuracy of these algorithms is quantized by the metric "location error" and "location error % radio range". The location error is the distance between N_x's estimated position and the real position. Then, "location error % radio range" is obtained as the percentage of location error by the node radio range. Lower location error always indicates better accuracy. The simulation results are listed in Table 3-9.

Table 3-9. Simulation Results for Scenario 1 with 3 Neighbor Anchors

Algorithms	Estimated Positions	Location Error (m)	Location Error (% radio range)
Centroid	(21.08, 23.08)	3.27	3.27 / 20 = 16%
CPE	(23.00, 20.00)	3.00	3.00 / 20 = 15%
Mid-perpendicular	(18.57, 20.41)	1.49	1.49 / 20 = 7.5%

From the simulation results, we can see that, our Mid-perpendicular algorithm has better accuracy than Centroid and CPE. Next, we will increase the number of neighbor anchors.

3.6.1.2 Scenario 2 for Class-1 Localization

For Scenario 2, the values of particular parameters (marked with "*" in Table 3-8) are: the number of neighbor anchors "*m*" is 4, and the random simulation number is still 1. The geographic random distribution of anchors is shown in Figure 3-18.

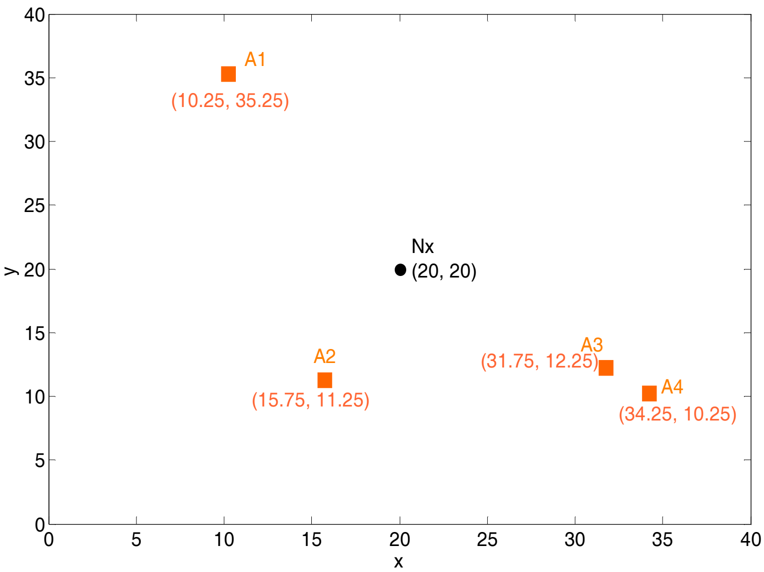


Figure 3-18. Nodes Distribution for Scenario 2 with 4 Neighbor Anchors

The simulation results for Scenario 2 are listed in Table 3-10.

Comparing Scenario 1 and Scenario 2, we have the following analysis and conclusions:

(1) The distribution of neighbor anchors has influence on the performance of these range-free algorithms. Regular distribution of anchors can lead to better localization accuracy, even with fewer anchors.

Normally, when a node has more neighbor anchors, its estimated position can be more accurate, thus the localization algorithms can have better performance. However, when compare Table 3-9 and Table 3-10, we can find that: although Scenario 2 has more neighbor anchors than Scenario 1, the performance of algorithms in Scenario 2 is not as good as that in Scenario 1. The reason lies on the distribution of anchors. From Figure 3-17 and Figure 3-18, we can see that, anchors in these two scenarios have different geographic distributions. Anchors in Scenario 1 are distributed regularly (shown in Figure 3-14), while those in Scenario 2 are not so regularly distributed (shown in Figure 3-18).

(2) Our Mid-perpendicular algorithm still has the best accuracy. However, we should also notice that, the direct version and the simplified version of our algorithm have different performance. In this scenario, the direct version has better accuracy. But sometimes, the simplified version may perform better, which will be shown in the next.

Table 3-10. Simulation Results for Scenario 2 with 4 Neighbor Anchors

Algorithms		**Estimated Positions**	**Location Error (m)**	**Location Error (% radio range)**
Centroid		(23.00, 17.25)	4.07	4.07 / 20 = 20%
CPE		(22.25, 22.75)	3.55	3.55 / 20 = 18%
Mid-perpendicular	Direct	(22.63, 20.00)	2.63	2.63 / 20 = 13%
	Simplified	(22.25, 22.75)	3.55	3.55 / 20 = 18%

3.6.1.3 Scenario 3 for Class-1 Localization

The Scenario 3 has the same parameters as Scenario 2. But, as shown in Figure 3-19, the four anchors have a random distribution, which is different from that in Scenario 2.

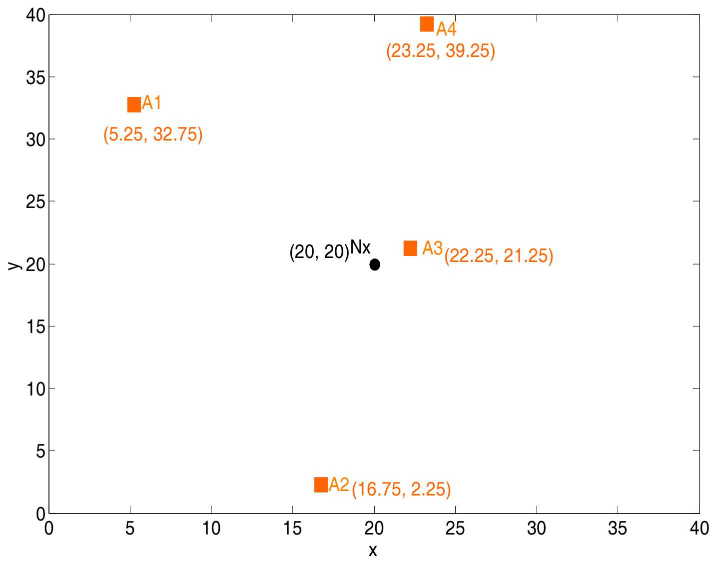


Figure 3-19. Nodes Distribution for Scenario 2 with 4 Neighbor Anchors

The simulation results for Scenario 3 are listed in Table 3-11.

Comparing Table 3-10 and Table 3-11, we can find that: when the distribution of anchors changes, the algorithm with the best accuracy may also change. In Scenario 2, it is the direct version of Mid-perpendicular algorithm that has the best accuracy, while Centroid algorithm has the lowest accuracy.

However, in Scenario 3, the best algorithm is the simplified version of Mid-perpendicular, while CPE algorithm is the least accurate.

Therefore, only by one particular case such as the above three scenarios, we cannot tell which the best algorithm is. Thus, instead of only 1 simulation, we will have to produce a large number of random simulations, so that plenty of results can be obtained. From these massive results, their average value, their maximum value and their minimum value should be investigated, so that we can have a comprehensive view on the performance of algorithms. All these will be discussed in the next.

Table 3-11. Simulation Results for Scenario 3 with 4 Neighbor Anchors

Algorithms		Estimated Positions	Location Error (m)	Location Error (% radio range)
Centroid		(16.88, 23.88)	4.98	4.98 / 20 = 25%
CPE		(14.25, 20.75)	5.80	5.80 / 20 = 29%
Mid-perpendicular	Direct	(16.73, 22.41)	4.06	4.06 / 20 = 20%
	Simplified	(19.74, 20.80)	0.84	0.84 / 20 = 4%

3.6.1.4 Scenario 4 for Class-1 Localization: a General Scenario

In order to get more general results, in this scenario, we set the parameters as: the number of neighbor anchors "*m*" ranges from 3 to 8 (here, in purpose of comparison, we assume the maximum number is 8, but in reality, the number of neighbor anchors may never reach 8); then, for each value of "*m*", the random simulation number is 5000. That means, for a given number of neighbor anchors, there are as many as 5000 different geographic distribution of anchors.

Table 3-12. Location Error (% Radio Range): Maximum, Average, Minimum

Number of Anchors \ Algorithms		Centroid	CPE	Mid-perpendicular	
				Direct	Simplified
3	Maximum	63%	62%	62%	
	Average	28%	26%	19%	
	Minimum	0.6%	0	0	
4	Maximum	54%	55%	53%	53%
	Average	25%	24%	18%	15%
	Minimum	0.5%	0	0.5%	0
5	Maximum	46%	48%	42%	42%
	Average	17%	17%	14%	13%
	Minimum	0.4%	0	0.5%	0
6	Maximum	44%	42%	42%	39%
	Average	18%	17%	14%	10%
	Minimum	0.6%	0	0.4%	0
7	Maximum	39%	36%	36%	39%
	Average	15%	16%	13%	11%
	Minimum	0.6%	0	0.6%	0
8	Maximum	38%	36%	35%	35%
	Average	13%	13%	11%	9%
	Minimum	0.3%	0	0.05%	0

Under each distribution of anchors, the location error of the algorithms can be obtained like the previous three scenarios. Thus, a total of 5000 distributions can generate massive location errors for every number of neighbor anchors. The average value of these location errors, as well as their maximum value and minimum value, are listed in Table 3-12.

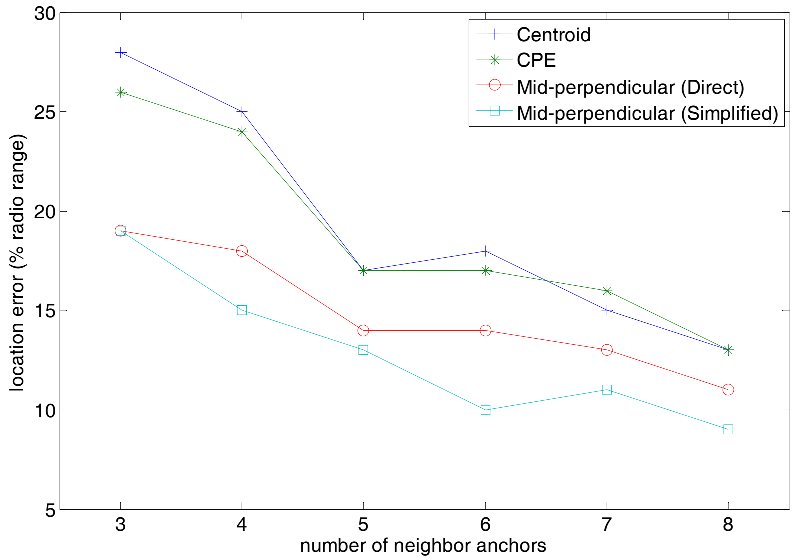


Figure 3-20. Average Location Error for Scenario 4

Based on the average values in Table 3-12, the corresponding figure is presented as Figure 3-20, so that we can get a clear view on the average performance of algorithms. From this figure, we can see:

(1) Our Mid-perpendicular algorithm (both the direct and simplified versions) is more accurate than Centroid and CPE algorithms. On average, the improvement is about 15%.

(2) When the number of neighbor anchors is small (like 3 or 4), the advantage of our algorithm is obvious. However, when there are more neighbor anchors (for example 7 or 8), the improvement by our algorithm is not so significant. The reason is that, in case of more anchors, more information can be available for the normal node, thus the algorithms like Centroid and CPE can have relatively good accuracy. Then, the gap between our algorithm and other algorithms is reduced.

(3) The simplified version of our Mid-perpendicular algorithm has a little better accuracy than the direct version. The reason is: the direct version calculates the average of all estimated positions without any particular selection or filtering. (These positions are obtained based on any three anchors.) However, as we discussed in the section 3.2.2.2, the overlap by all the anchors is mainly contributed by just three anchors. Thus, the key point is to find only these three anchors to calculate one estimated position, as the simplified version of our algorithm does. The average calculation of all estimated positions, which describes the direct version of our algorithm, will import additional location error.

Based on the data in Table 3-12, the maximum and minimum location error can be added to Figure 3-20. This generates the following Figure 3-21.

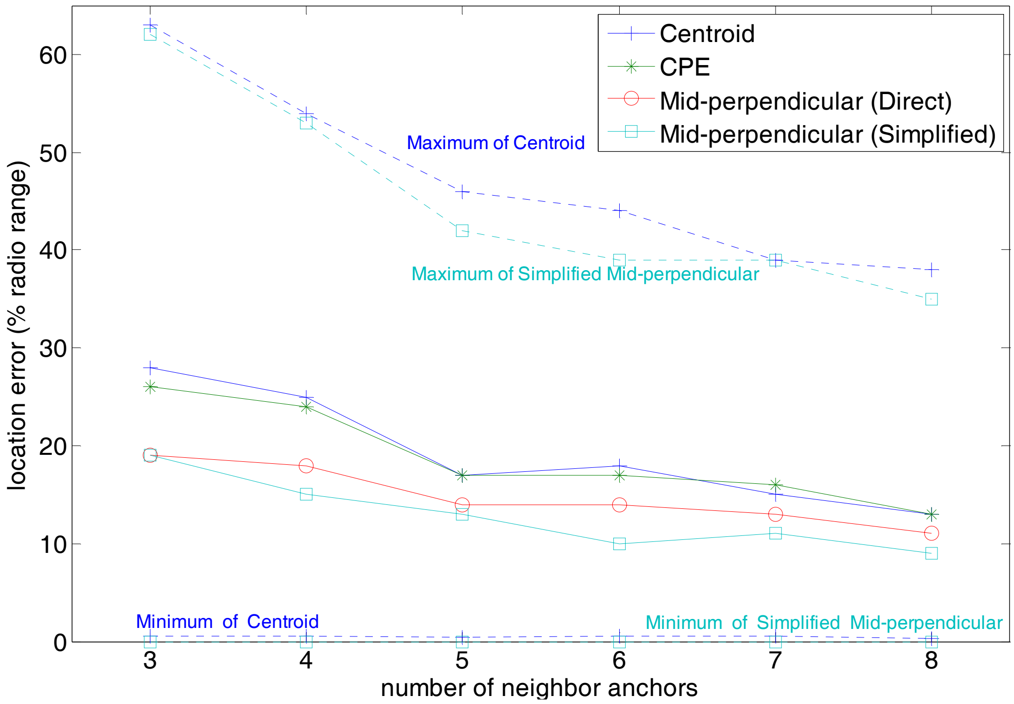


Figure 3-21. Maximum and Minimum Location Errors of Centroid and Simplified Mid-perpendicular

If the maximum and minimum location errors of all the 4 algorithms are displayed in the figure, it will be too complex to recognize them. So, in Figure 3-21, we only show the maximum and minimum values of Centroid and Simplified Mid-perpendicular algorithms.

From the figure, for both algorithms, we can observe the big vertical interval between the maximum and minimum location error. The minimum values are very low, nearly 0, showing that sometimes the algorithms can have very good accuracy. But the maximum location errors are relatively much higher, about twice of the average location error. At this point, a question can enter our minds: in most cases, is the performance of the algorithms close to the average location error? If the answer is yes, then that means the maximum and minimum location errors are just from a few cases, not frequently occur. But if the answer is not, then it indicates that the performance of algorithms change a lot.

To respond the question, we need to know the probabilities of location error values. This probability parameter can evaluate how frequent a location error exists in our random simulations. As an example, Figure 3-22 shows the probabilities of location errors in the case of 3 neighbor anchors. Based on simulation results, Figure 3-22 is generated by the normal fit function in MATLAB stastics toolbox. From the figure, we can note that, for the three algorithms, the maximum location error occurs very rarely, with a probability nearly 0. For each algorithm, the average location error occurs the most frequently. Mid-perpendicular algorithm has the smallest average location error 19% whose probability is 0.038, while Centroid and CPE have bigger average errors that share the same probability 0.032.

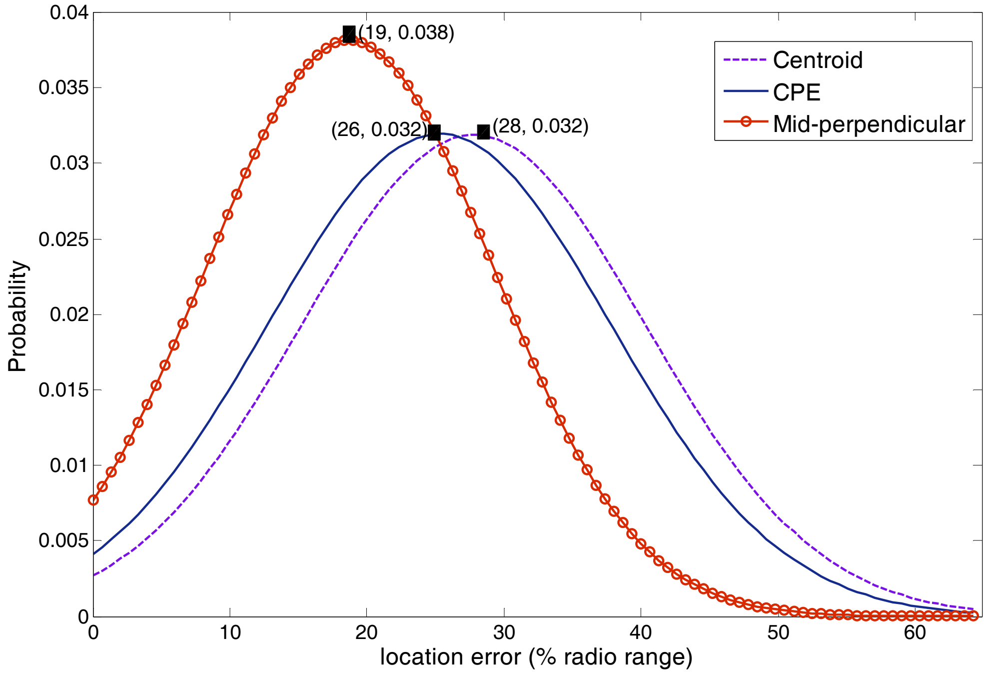


Figure 3-22. Probability of Location Error (3 Neighbor Anchors)

From the simulation results, using MATLAB stastics function, we can also obtain the interval where most location errors reside. In the above case, for Centroid algorithm, 90% of location errors values are between 8% and 48%, so the interval is [8%, 48%]. For CPE algorithm, 90% of location error values are in the interval [6%, 46%]. For Mid-perpendicular algorithm, 90% of location error values are in the interval [3%, 35%].

This indicates that the performance of the algorithms varies a lot in the simulation. The reason is the distribution of anchors (the influence by the distribution has been discussed in previous scenarios). Every time we run the simulation, we use a different random distribution of anchors. Under a particular distribution of anchors, all the algorithms might have very good accuracy, with the location error as low as 0. But under another distribution, the location error might be as high as 50%.

3.6.2 Performance of Algorithms for Class-2 Nodes

In the previous section, the simulation area is relatively small, so that anchors have more possibility to be within the communication range of one normal node. However, now, this restriction is not necessary. The simulation area can be larger, and there can be many more normal nodes. For example, when the node radio range remains 20 meters, the simulation area can be as large as $100\times100m^2$ with 100 nodes inside. Most of them are normal nodes, while only a few are anchors. Some normal nodes may have less than 3 neighbor anchors. So, DV-hop based algorithms will be used in this section, including DV-hop, DDV-hop (Differential DV-hop), Self-adaptive DV-hop, Robust DV-hop, our Checkout DV-hop, and our Selective 3-Anchor DV-hop. These DV-hop based algorithms have been discussed in the sections 2.2.2.4, 3.3, and 3.4.

In this section, several scenarios will be applied from different point of view to investigate the performance of algorithms for class-2 nodes. The main parameters of all these scenarios are listed in Table 3-13. Most of the parameters are shared by these scenarios, while other parameters marked with “*” vary with each scenario.

Table 3-13. Scenario Parameters for Class-2 Algorithms

Scenario Parameters	Values
Node Radio Range	20 meters
Simalation Area	100m×100m Square Area
Radio Propagation	Ideal, no pathloss, no interference
Number of Nodes	100 (all static)
* Ratio of Anchors	to be decided in specific scenario
* Random Simulation Number (= $TRd_{\text{nod}} \times TRd_{\text{anc}}$)	to be decided in specific scenario

In the following, we will present each scenario with the corresponding simulation results for the DV-hop based algorithms.

3.6.2.1 Scenario 1 for Class-2 Localization

The parameters of the first scenario have already been listed in Table 3-8. Here, we give the values of those particular parameters (marked with "*"): the ratio of anchors is 5%; and the random simulation number is 1, which means TRd_{nod} is 1 and TRd_{anc} is also 1. So, there is only one random distribution of nodes. Among these nodes, 5 anchors are randomly selected, and then never change in this scenario. Thus, a particular case will be presented. The geographic distribution of nodes is shown in Figure 3-23. The 5 anchors are shown as the little squares in the figure.

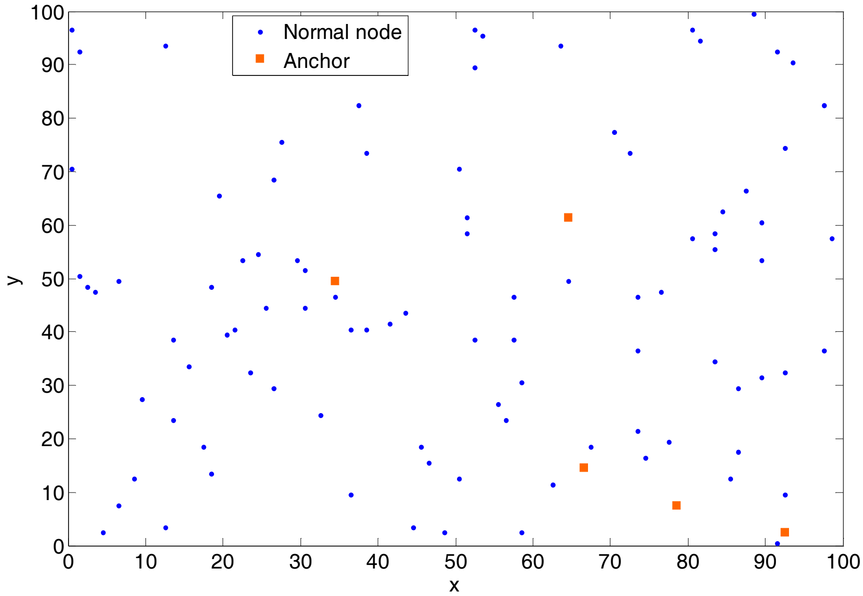


Figure 3-23. Nodes Distribution for Scenario 1 with 5% Ratio of Anchors

In the section 3.6.1.1, we have introduced the metric "location error (% radio range)" to measure the accuracy of algorithms, which estimate the position of one normal node N_x. However, in this section, many more normal nodes appear. We need to use "average location error (% radio range)" to quantize the accuracy. This metric is calculated as the average of location errors from all normal nodes. The simulation results for Scenario 1 are listed in Table 3-14.

It should be noted that, as discussed in the section 3.4.3.3, our Selective 3-Anchor DV-hop algorithm doesn't work in the case of very few anchors. In this scenario, since the ratio of anchors is only 5%, the performance of the algorithm is not studied, resulting in "----" in Table 3-14. However, in reality, it is recommended to temporarily use Checkout DV-hop to replace Selective 3-Anchor DV-hop, when the ratio of anchors is small.

Table 3-14. Simulation Results of Scenario 1 for Class-2 Nodes

Algorithms	Average Location Error (% radio range)
DV-hop	73%
DDV-hop	74%
Self-adaptive DV-hop	75%
Robust DV-hop	71%
Checkout DV-hop	62%
Selective 3-Anchor DV-hop	------

From the table, we can see: comparing with DV-hop algorithm, on average, our Checkout DV-hop algorithm has an improvement about |73%-62%| / 62% = 20%. We can also note that, DDV-hop and Self-adaptive DV-hop algorithms have no good performances. This is different from the simulation results in the literature of DDV-hop and Self-adaptive DV-hop algorithms, which are discussed in the section 2.2.2.4.2. The reason is the distribution of nodes. For example, in the literature about Self-adaptive DV-hop algorithm [ZXL 09], four anchors are deployed at the corners of a square area, and normal nodes are distributed regularly inside the area. However, in our simulation, nodes are randomly distributed.

In the next, we will increase the number of anchors, and then investigate algorithms performance.

3.6.2.2 Scenario 2 for Class-2 Localization

In this scenario, those particular parameters are: the ratio of anchors is 30%; and the random simulation number is still 1. So, a particular case will be presented. The geographic distribution of nodes is shown in Figure 3-24. The 30 anchors are shown as the little squares in the figure. Other points in the figure indicate the positions of the 70 normal nodes.

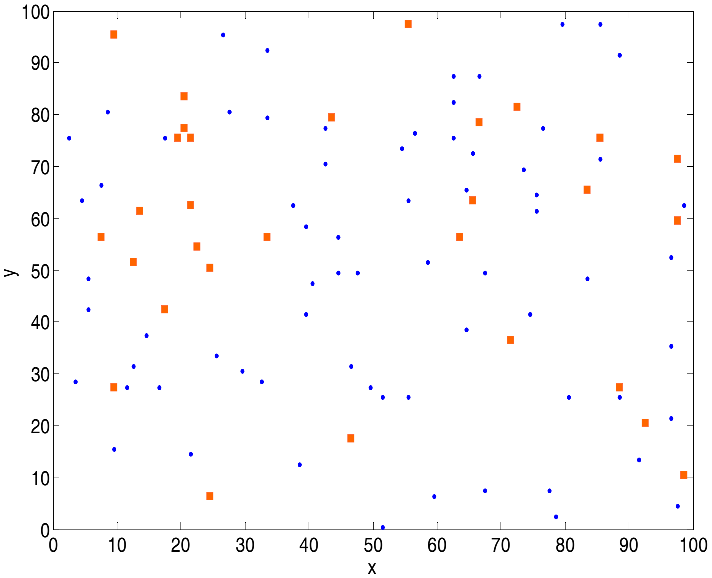


Figure 3-24. Nodes Distribution for Scenario 2 with 30% Ratio of Anchors

The simulation results for Scenario 2 are listed in Table 3-15.

Table 3-15. Simulation Results of Scenario 2 for Class-2 Nodes

Algorithms	Average Location Error (% radio range)
DV-hop	45%
DDV-hop	48%
Self-adaptive DV-hop	48%
Robust DV-hop	42%
Checkout DV-hop	39%
Selective 3-Anchor DV-hop	29%

Comparing Table 3-14 and Table 3-15, we can find that:

(1) When the ratio of anchors increase from 5% to 30%, since more information from anchors can be used for localization, the accuracy of each DV-hop based algorithm has been much better. For example, the average location error of DV-hop algorithm has been improved from 73% in Scenario 1 to 45% in Scenario 2.

(2) While our Checkout DV-hop algorithm has some improvement that is yet not so significant, our Selective 3-Anchor DV-hop algorithm can achieve much better accuracy than the other DV-hop based algorithms. For example, its average location error in Scenario 2 is 29%. Compared with DV-hop algorithm, our Selective 3-Anchor DV-hop algorithm has an improvement about |45%-29%| / 45% = 35%.

3.6.2.3 Scenario 3: a General Scenario

In order to get more general results, in this scenario, we set the parameters as: the ratios of anchors range from 5% to 90% (here, in purpose of comparison, we assume the maximum ratio of anchors is as high as 90%, but in reality, there may not be so many anchors); then, for each ratio of anchors, the random simulation number is 2000, which composed of two multipliable parts "TRd_{nod}" and "TRd_{anc}", thus $TRd_{\mathrm{nod}} \times TRd_{\mathrm{anc}}$=2000. Every time, for each ratio of anchors, all the 100 nodes are uniform-randomly distributed inside the simulation area. So, through TRd_{nod} times, we can have as many as TRd_{nod} geographical distributions of nodes. The second random times, TRd_{anc}, is the random times for selecting nodes as anchors. Every time, for each distribution of nodes, anchors are uniform-randomly selected from the entire 100 nodes. In the simulation, as an example, we set TRd_{nod} to be 20, and TRd_{anc} to be 100. That means, $TRd_{\mathrm{nod}} \times TRd_{\mathrm{anc}} = 20 \times 100 = 2000$.

The average value of these location errors, as well as their maximum value and minimum value, are listed in Table 3-16. In this table, each algorithm has three columns which from left to right correspondingly contain the maximum, average and minimum location errors (% radio of range).

Observed from Table 3-16, for all the algorithms, the average location error is not close to the maximum and minimum values. A big gap between the maximum and minimum location errors can be noted. Thus, we need to know the probabilities of location errors.

As an example, Figure 3-25 shows the probabilities of location errors when the ratio of anchors is 10%. Based on simulation results, Figure 3-25 is generated by the normal fit function in MATLAB stastics toolbox. From the figure, we can note that, for all the three algorithms, the maximum location error occurs very rarely with a probability nearly 0. On the contrary, the average location errors appear the most frequently. Compared with DV-hop and Checkout DV-hop, Selective 3-Anchor DV-hop algorithm has the smallest average location error 43% whose probability 0.0179 is highest.

Table 3-16. Location Error (% Radio Range): Maximum, Average, Minimum for Scenario 3

Ratio of Anchors	DV-hop			DDV-hop			Self-adaptive DV-hop			Robust DV-hop			Checkout DV-hop			Selective 3-Anchor DV-hop		
5%	327	71	5	319	73	6	319	73	6	305	67	5	220	61	4	-------		
10%	284	65	5	301	67	6	295	67	6	277	58	5	209	55	3	163	43	2
20%	195	62	4	203	64	5	202	63	6	186	55	4	157	53	3	114	40	2
30%	125	56	3	138	59	3	138	57	3	113	51	2	130	48	3	83	36	1
40%	162	58	3	177	62	3	177	60	3	150	52	2	143	50	3	108	37	1
50%	151	57	3	159	61	4	154	59	3	133	50	2	136	49	2	89	35	1
60%	176	58	3	184	62	3	187	60	3	143	52	3	125	50	2	83	35	1
70%	164	57	2	163	60	3	163	58	2	138	52	2	112	50	2	81	36	1
80%	138	56	1	157	62	3	156	58	3	124	52	2	109	49	2	81	34	1
90%	137	56	2	136	58	2	133	57	3	111	50	2	105	49	2	79	33	1

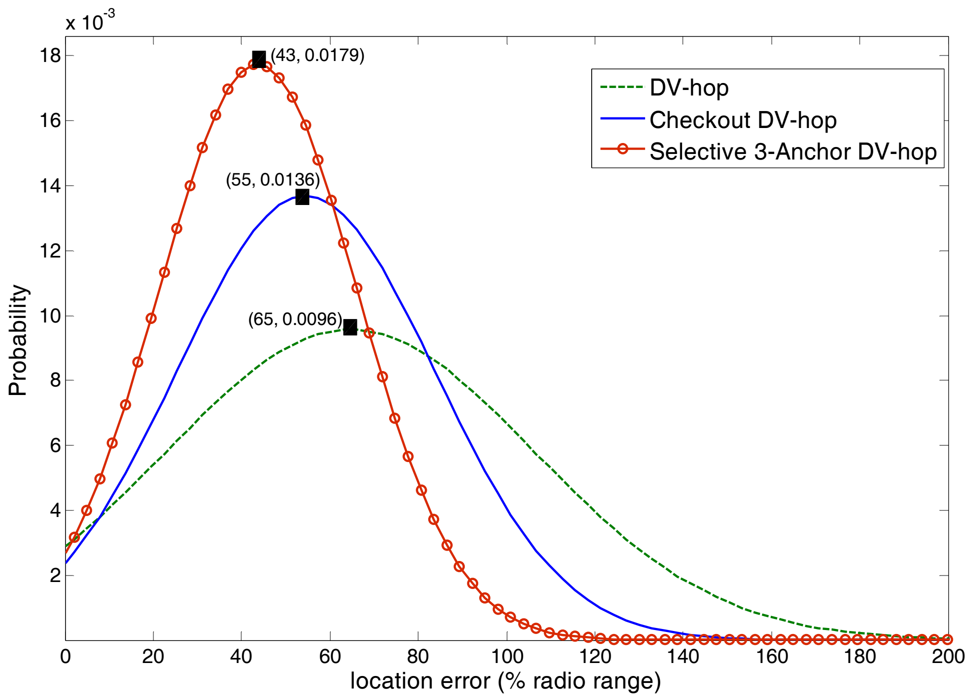


Figure 3-25. Probability of Location Error in case of Ratio of Anchors 10%

From the simulation results, we can also obtain the interval where most location errors reside. For DV-hop algorithm, 90% of location errors values are in the interval [26%, 104%]. For Checkout DV-hop algorithm, 90% of location error values are in the interval [27%, 83%]. For Selective 3-Anchor DV-hop algorithm, 90% of location error values are in the interval [23%, 63%].

This shows that the performances of the three algorithms vary a lot. This performance variation is mainly caused by two factors. First is the random distribution of nodes. Nodes distribution can influence the accuracy of DV-hop based algorithms, as discussed in the section 3.6.2.1. The second factor is the estimated distance between a normal node and each anchor. This estimated distance is an important element for DV-hop based algorithms. However, the accuracy of estimated distance is not steady, which has been investigated in the section 3.3.1.2. Thus, in the future, when we improve the stability of the performance of DV-hop based algorithms, we should take these two factors into consideration.

Based on the average values in Table 3-16, the corresponding figure is presented as Figure 3-26, so that we can get a clear view on the average performance of algorithms.

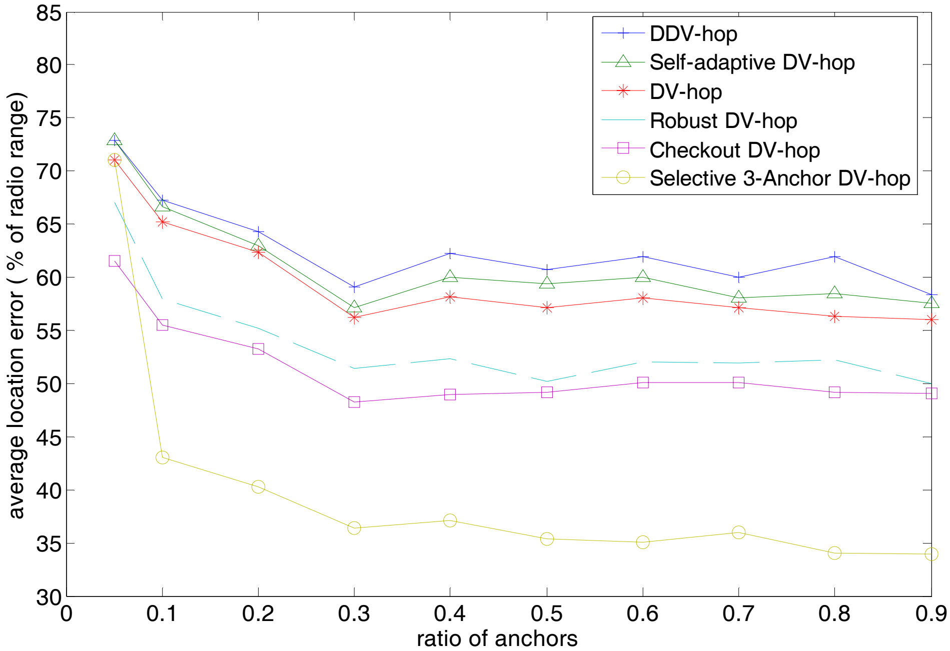


Figure 3-26. Average Location Error of Algorithms for Class-2 Nodes

From these simulation results, we can see:

(1) Compared with the existing algorithms (like DV-hop, DDV-hop, Self-adaptive DV-hop, and Robust DV-hop), on average, our Checkout DV-hop algorithm has better accuracy, although the improvement is not so significant, which is at least 5% and at most 20% depending on which algorithm to be compared with. However, the improvement by our Selective 3-Anchor DV-hop algorithm is considerable. On average, its localization accuracy is about 30% better than Checkout DV-hop algorithm, and about 45% better than DV-hop algorithm.

(2) In general, for each algorithm, the location error decreases when the ratio of anchors increases. However, this doesn't happen when the ratio of anchors is large. For example, as shown in Figure 3-26, considering the ratio of anchors larger than 40%, when the ratio increases, the accuracy of the DV-hop based algorithms doesn't get better. The reason can be: the DV-hop based algorithms use the estimated distance between each anchor and the normal node. Each distance has an estimation error, which has been shown in the section 3.3.1. In the case of high ratio of anchors, since there are more anchors, more estimated distances bring in more estimation errors.

In fact, when the ratio of anchors is high, around each normal node, there can be more than 3 neighbor anchors. These neighbor anchors are much closer to the normal node than other anchors. The information from these neighbor anchors is more helpful than from other anchors. Thus, in this case, we should make best use of these neighbor anchors, just like the algorithms for class-1 nodes do.

Comparing Figure 3-20 with Figure 3-26, we can see that, the algorithms for class-1 nodes have better accuracy than the DV-hop based algorithms, in the case of high ratio of anchors. Therefore, we can have the following conclusion: Although the DV-hop based algorithms (mainly the six algorithms discussed in this section) can localize the class-2 normal nodes, they are not recommended to localize the class-1 nodes due to their unsatisfactory accuracy.

In fact, this conclusion can also be supported from the view of network overhead which will be discussed in Chapter 4.

3.6.3 Combined Evaluation of the Algorithms for Class-1 and Class-2 Nodes

From the previous section, although the DV-hop based algorithms can serve for both class-1 and class-2 nodes, we suggest that it is better for them to only localize class-2 nodes. Thus, we have the idea to combine these range-free algorithms in an adaptive mode: when a normal node has at least 3 neighbor anchors, it uses the class-1 algorithms among which our Mid-perpendicular is recommended; while the normal node has less than 3 neighbor anchors, it changes to the DV-hop based algorithms, among which we recommend our Checkout DV-hop and Selective 3-Anchor DV-hop algorithms.

In this section, the simulation scenario is the same as Scenario 3 in the section 3.6.2.3. In this scenario, except the anchors, some are class-1 normal nodes, while others are class-2 normal nodes. Since the algorithms for class-1 nodes cannot work for class-2 nodes, we need to combine them with DV-hop algorithm. For example, instead of using only “Centroid”, a combination “Centroid+DVhop” will be used, so that it can serve all nodes in the scenario. Here, “Centroid+DVhop” means Centroid for class-1 nodes and DV-hop for class-2 nodes.

In the following, the combined evaluation will be presented in terms of accuracy and complexity.

3.6.3.1 Evaluation on Accuracy of the Range-free Localization Algorithms

The concerned algorithms are: DV-hop, Centroid+DVhop, CPE+DVhop, Mid-perpendicular+CheckoutDVhop, Mid-perpendicular+Selective3AnchorDVhop. Here, the Mid-perpendicular algorithm refers to its simplified version.

The average location errors of these algorithms are shown in Figure 3-27. From the figure, we can have the following conclusions:

(1) The combination of two-class algorithms has good advantages compared with separate use of algorithms.

The first advantage is the complete coverage on all normal nodes. This is the advantage for class-1 algorithms like Centroid, CPE and Mid-perpendicular. For example, if not combined with DV-hop algorithm, Centroid algorithm cannot localize class-2 normal nodes. However, the combination “Centroid+DVhop” covers both class-1 and class-2 nodes.

The second advantage is the better accuracy, which is aimed at class-2 algorithms (also known as DV-hop based algorithms). If not combined with Centroid or CPE, the DV-hop algorithm doesn’t have good accuracy when the ratio of anchors is high. However, seen from Figure 3-27, when the ratio of anchors increases, more class-1 nodes exist, thus the class-1 algorithms like Centroid and CPE begins to have good effect. When the ratio of anchor is only 5%, all normal nodes are at class-2. Thus, only DV-hop algorithm works, and “Centroid+DVhop” has the same accuracy as DV-hop. But when there are more anchors, some normal nodes begins to own at least 3 neighbor anchors. Then we can see "Centroid+DVhop" has better accuracy than DV-hop algorithm. And when the ratio of anchors gets bigger, the gap between "Centroid+DVhop" and DV-hop also gets larger, indicating the greater advantage of "Centroid+DVhop".

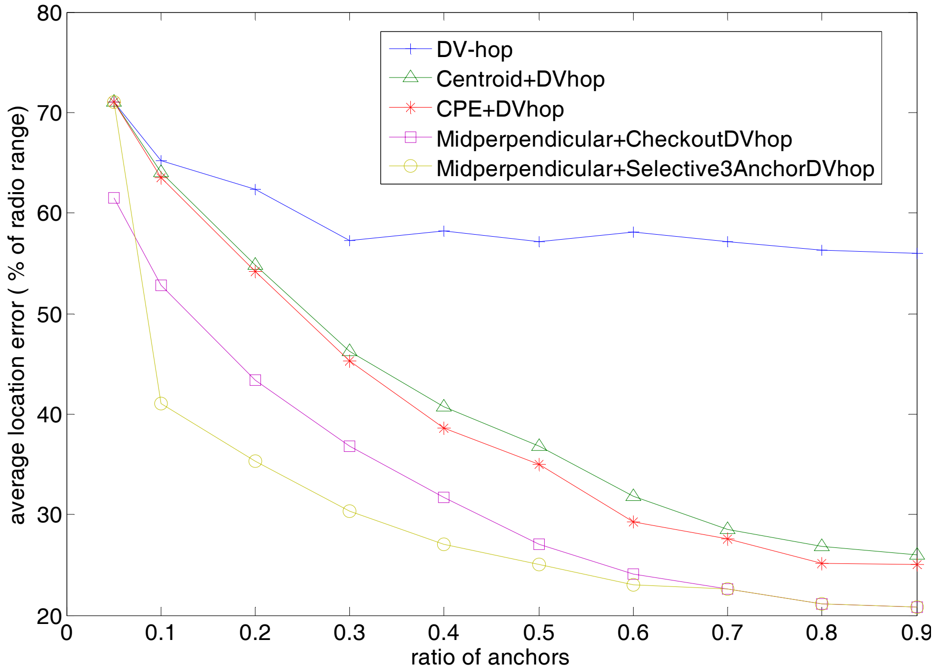


Figure 3-27. Combined Evaluation on Location Error

(2) Among the different combinations of two-class algorithms, the combination of our algorithms has the best accuracy. Our class-1 algorithm is Mid-perpendicular (in its simplified version), while our class-2 algorithms include Checkout DV-hop and Selective 3-Anchor DV-hop. Thus, the combinations of our algorithms include "Mid-perpendicular+CheckoutDVhop" and "Mid-perpendicular+Selective3AnchorDVhop". From Figure 3-27, we can observe the better accuracy of our combinations than other combinations.

3.7 Evaluation on Computation Complexity of Range-free Algorithms

The theoretical analysis on computation complexity of the algorithms has been presented in the section 3.5. Now, we give the evaluation through simulations.

The computation of an algorithm takes certain amount of runtime when it is simulated with Matlab on computers. We use this runtime as the metric for evaluating the computation complexity. Hence, the longer the runtime of an algorithm, the higher its complexity.

Generally, sensors have limited computation capability, while the computers that we used for simulation possess high-speed powerful CPUs. The computation capacity of device has influence on the runtime of algorithm. In order to present this influence, the same simulation is done by two computers, which have different computation strength. Computer A has a 3.07GHz CPU and 24GB RAM, while computer B has a 1.6 GHz CPU and 0.99 GB RAM. In Matlab, the default data type is double-precision floating point, which requires 64 bits for storage. The simulation results are shown in the following figures.

Figure 3-28 presents the runtimes of class-1 algorithms such as Centroid, CPE and our proposed Mid-perpendicular (simplified version). We can see that, when the ratio of anchors increases, the complexity of Mid-perpendicular algorithm increase much more quickly than that of Centroid and CPE algorithms. The reason is: the complexity level of Centroid and CPE algorithms is O(m), while that of Mid-perpendicular algorithm is $O(m^2)$. Thus, Mid-perpendicular algorithm can increase more sharply than Centroid and CPE algorithm.

The influence of device computation capacity can also be observed from Figure 3-28. Since Computer A has a more powerful CPU and a bigger RAM than Computer B, the calculation by Computer A is much faster. From Computer B to Computer A, we can see from the figure a considerable increase of calculation time for each algorithm.

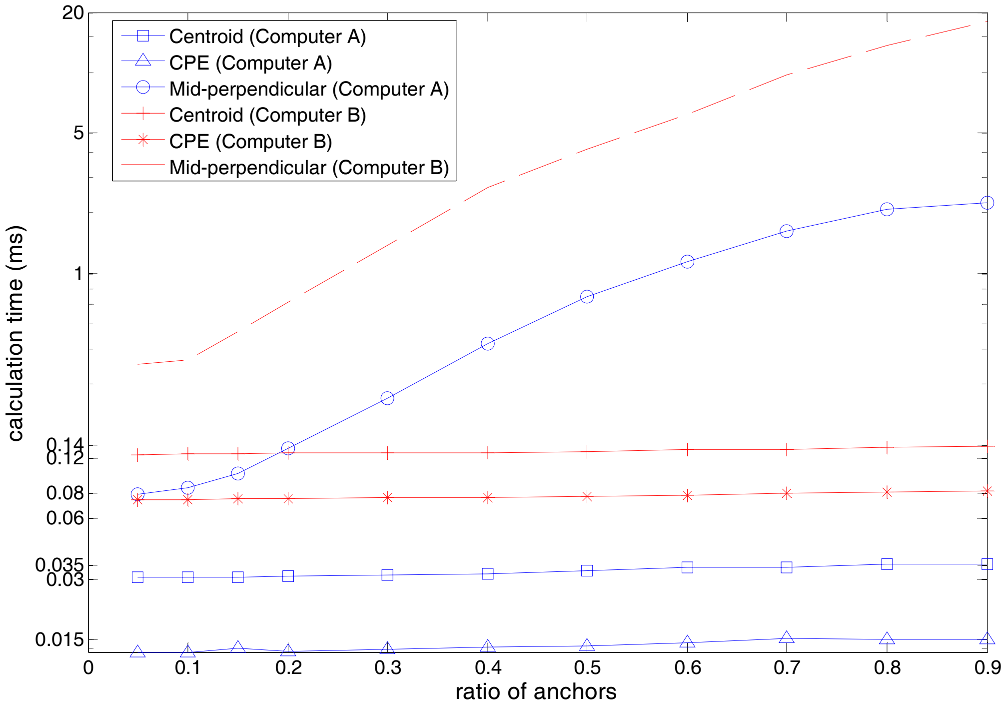


Figure 3-28. Calculation Time of Class-1 Algorithms

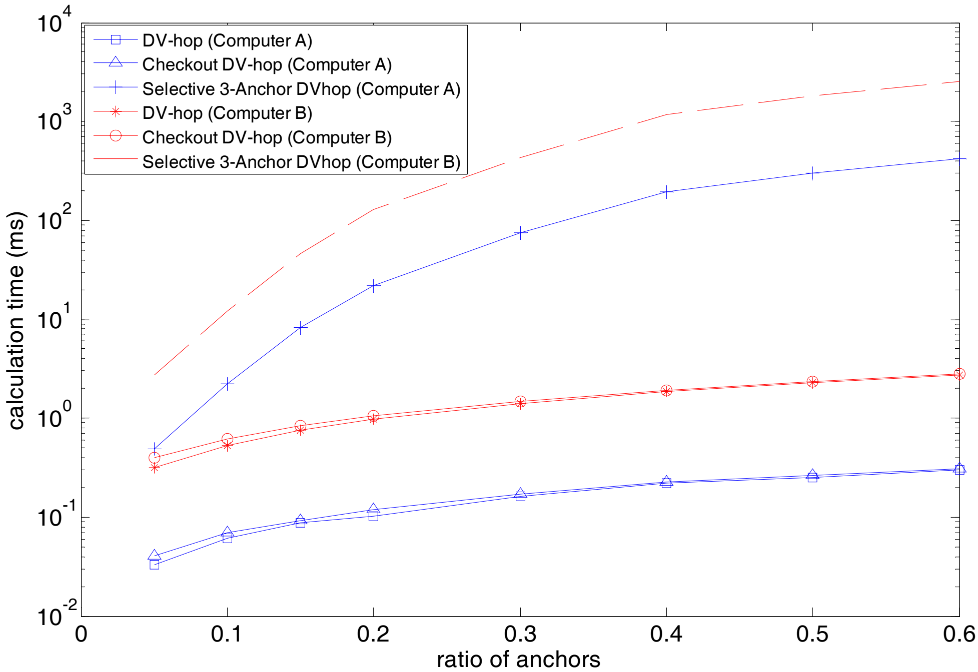


Figure 3-29. Calculation Time of Class-2 Algorithms

Figure 3-29 presents the runtimes of class-2 algorithms such as DV-hop, Checkout DV-hop and Selective 3-Anchor DV-hop. In Figure 3-29, the curves for DV-hop and Checkout DV-hop algorithms are almost overlapped. This indicates that Checkout DV-hop algorithm has very similar computation complexity with DV-hop algorithm. On the contrary, compared with DV-hop algorithm, Selective 3-Anchor DV-hop algorithm needs much more calculation time.

In the following table, the theoretical analysis in the section 3.5 is compared with simulation results in this section. The mathematical analysis includes big O notation, while simulation results are presented as runtime in millisecond. From the table, we can conclude that the mathematical analysis fits well with simulation results.

Table 3-17. Comparison of Mathematical Analysis and Simulation results

	Theoretical Results	Simulation Results* (calculation time in millisecond)
Centroid	O(m)	0.12
CPE	O(m)	0.08
Mid-perpendicular	$O(m^2)$	1.37
DV-hop	$O(m_d)$	1.41
Checkout DV-hop	$O(m_d)$	1.49
Selective 3-Anchor DV-hop	$O(m_d^{\ 4})$	435.16

(* : The simulation examples are conducted by Computer B. During the simulation, the anchor ratio is 30%.)

3.8 Brief Summary of Chapter 3

The normal nodes are categorized into two classes according to the number of neighbor anchors: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes.

For class-1 normal nodes, Mid-perpendicular algorithm is proposed to give a better accuracy than Centroid and CPE algorithms. The proposed algorithm finds a centre of anchors communication overlap, and regards this centre as the estimated position of the normal node. When the normal node has only 3 neighbor anchors, the centre of the overlap is calculated as the cross point of mid-perpendiculars between any two anchors. When there are more than 3 neighbor anchors, the proposed algorithm first finds the 3 anchors which contribute the communication overlap of all anchors, and then calculates the centre in the same way for 3 neighbor anchors.

Class-2 normal nodes need to use DV-hop algorithm for localization. Two algorithms have been proposed to improve the accuracy of DV-hop.

One is Checkout DV-hop, which simply adjusts the estimated position of DV-hop algorithm based on the nearest anchor. The principle is: from the statistical average view, the nearest anchor has the most accurate distance to the normal node. This has been proved both by analysis and simulation.

Since Checkout DV-hop algorithm just does a little modification to DV-hop algorithm, its improvement is not so significant. DV-hop and Checkout DV-hop algorithms both create only one candidate position for the normal node. However, another proposal, our Selective 3-Anchor DV-hop algorithm, creates many more candidates. Each candidate is obtained based on any three anchors. The metric for judging a candidate is its connectivity which is its hop counts to all anchors. Similar connectivity means closer in distance. Thus, the candidate which has the most similar connectivity to the normal node is selected as the final estimated position.

Simulations have been executed for each class of normal nodes. From the simulation results, several conclusions can be obtained:

(1) The distribution of nodes has influence on the algorithms. Under different distributions, the performance might change a lot. This performance variance of the algorithms has been observed from the view of confidence level. In addition, for class-1 algorithms, regular distribution of anchors can lead to better localization accuracy, even with fewer anchors. For DV-hop based algorithms, the anchors cannot be positioned on a line.

(2) Our Mid-perpendicular algorithm has better accuracy than Centroid and CPE algorithms. On average, the improvement can be about 15%.

(3) Our Selective 3-Anchor DV-hop algorithm has an average localization accuracy that is about 30% better than Checkout DV-hop algorithm and about 45% better than DV-hop algorithm.

(4) Although the DV-hop based algorithms can serve for both class-1 and class-2 nodes, we suggest that it is better for them to only localize class-2 nodes. That is because, when localizing class-1 normal nodes, the DV-hop based algorithms have lower accuracy than the algorithms like Centroid and CPE. Thus, a combination of our algorithms is recommended. That is, Mid-perpendicular for class-1 nodes, while Checkout DV-hop or Selective 3-Anchor DV-hop for class-2 nodes.

(5) As class-1 algorithms, Centroid and CPE both have low computation complexity at the level $O(m)$, while Mid-perpendicular increases the complexity to $O(m^2)$. As class-2 algorithms, DV-hop and Checkout DV-hop both remain at the level $O(m_d)$, but they need more calculation time than Centroid and CPE. Selective 3-Anchor DV-hop algorithm has a complexity as high as $O(m_d^4)$.

4. Protocols for Range-free Localization

During the verification process of our three new algorithms, we noted that most of the existing algorithms were only studied using tools like MATLAB which neglects the possible problems of a real network. For example, DV-hop based algorithms need the broadcasts of position related information throughout the network, then some problems such as collisions and link congestion must be solved by a localization protocol. Having found no such protocol, we propose a DV-hop protocol and a Class-1 protocol, whose combination is an adaptive range-free protocol. Our protocols are based on the IEEE 802.15.4 standard, with the chosen medium access method being non-slotted CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance). The network topology is assumed as ad-hoc.

4.1 Our DV-hop Localization Protocol

To the best of our knowledge, most of DV-hop based algorithms are implemented using MATLAB. They all neglect the issues inherent to a real network, such as collisions, mobility and synchronization. We noted that these problems can significantly influence the localization accuracy. As a result, it is important to estimate the performance of a localization algorithm from a networking point of view. However, the initial version of IEEE 802.15.4 standard doesn't define a localization protocol suitable for the range-free algorithms like DV-hop. Hence, we decided to implement a DV-hop localization protocol [GWV 12] in order to evaluate the original DV-hop, Checkout DV-hop and 3-Anchor DV-hop algorithms.

Our DV-hop localization protocol is implemented in the WSNet network simulator using C language. In the following subsections, we will introduce our DV-hop localization protocol, including the format of *data payload*, the improved collision reduction methods and the procedure of the protocol.

4.1.1 Proposed Formats of Data Payload in Each Step of DV-hop Algorithm

Like DV-hop algorithm, our protocol consists of 3 steps. At Step #1, anchors need to broadcast their positions throughout the network. At Step #2, anchors also need to diffuse their distance-per-hop values. So we must define the frame formats for the message exchange at the first two steps.

Conforming to the MAC general frame format specified in IEEE standard 802.15.4-2009, here, the frames in DV-hop protocol consist of three basic fields: MHR (MAC Header), MAC payload and MFR (MAC Footer). As shown in Table 4-1, MHR is composed of frame control, sequence number, destination address and source address. The Frame Control field contains information defining the frame type, security enabled or not, and other control flags. We should mention that the destination and source addresses use 16-bit short format. Since the frames in DV-hop protocol are all to be broadcasted, the destination address should be *0xFFFF*.

Data payload carries the information from a certain anchor. The information could be the position of the anchor, or the distance-per-hop value of the anchor. The detailed formats of *data payload* will be given later on. MFR contains the FCS (Frame Check Sequence) which is a 16-bit ITU-T CRC.

Table 4-1. Format of Data Frame in DV-hop Protocol

MHR				Data Payload (variable length)	MFR
Frame Control (16 bits)	Sequence Number (8 bits)	Destination Address (16 bits)	Source Address (16 bits)		FCS (16 bits)

Two formats of *data payload* are proposed for the first two steps of DV-hop protocol.

At Step #1, each anchor A_i broadcasts through the network a position frame *"frame_pos$_i$"*, so that all nodes (including anchors and normal nodes) can know the position of A_i and the minimum hop count to A_i. The format of *frame_pos$_i$* is shown in Table 4-2.

"Sequence Number" can identify every frame transmitted from the sender (here, the sender is A_i). It has 8 bits in length, defined by IEEE standard 802.15.4-2009.

The MHR and MFR of *frame_pos$_i$* have already been introduced in Table 4-1. Here, in Table 4-2, the *data payload* is composed of four parts: "Data Type", "x_i", "y_i" and "HopCount".

Data Type identifies the type of information that the frame contains. In fact, in DV-hop algorithm, each anchor A_i only need to broadcasts two types of information. One is its position at Step #1 of DV-hop, and the other is the distance-per-hop at Step #2 of DV-hop. So, we define that Data Type (1 bit) is "*0*" if this is a position frame, or "*1*" if this is a distance-per-hop frame. Here, Data Type is "*0*" since *"frame_pos$_i$"* is a position frame.

"HopCount" is the hop count value initialized to "*0*" by the initial sender A_i. This hop count value will increase with augment of hop during the flooding of this frame. Here, HopCount is limited to 7 bits, with the maximum value 127 that is sufficient for the network.

"x_i" and "y_i" represents A_i's coordinates. "x_i", as well as "y_i", is a 32-bit single precision float-point value. According to the standard IEEE 754, the single precision float point has the 24-bit mantissa precision and 8-bit exponent width, which means the precision is about 10^{-7} and the range is about $\pm10^{38}$.

Table 4-2. Format of *frame_pos$_i$*

<table>
<tr><td rowspan="3">MHR</td><td colspan="4">Data Payload</td><td rowspan="3">MFR</td></tr>
<tr><td>Data Type
(1 bit)</td><td>HopCount
(7 bits)</td><td rowspan="2">xi
(32 bits)</td><td rowspan="2">yi
(32 bits)</td></tr>
<tr><td colspan="2">(in total 8 bits)</td></tr>
</table>

On the first reception of the frame *frame_pos$_i$*, a node *N* records the position of A_i, and initializes its *hop$_i$* as HopCount+1. The value *hop$_i$* is *N*'s minimum hop count to A_i. Then *N* increases HopCount by 1 and broadcasts *frame_pos$_i$*. When receiving again a frame with the same header as *frame_pos$_i$*, *N* compares its *hop$_i$* with HopCount in this received frame and then makes decision. If HopCount+1 is smaller than *hop$_i$*, *N* will renew its *hop$_i$* as HopCount+1, increase HopCount by 1, and then relay the broadcast of this frame. If not, *N* will ignore this frame. Through this mechanism, all the nodes in the network (including normal nodes and anchors) can get the minimum hop counts to all anchors.

Through Step #1, each anchor A_i can collect the positions of the other anchors as well as its minimum hop count to them. From this, A_i can calculate its average distance per hop, denoted as *dph$_i$*. Then, at Step #2, A_i provides the normal nodes with its *dph$_i$* by broadcasting a distance-per-hop frame *"frame_dph$_i$"*.

The format of *frame_dph$_i$* is shown in Table 4-3. The *data payload* of *frame_dph$_i$* consists of Data Type and *dph$_i$*. The value of Data Type is *1*. "*dph$_i$*" is a single precision float-point value. In our case, the length of a single precision float-point value should be 32 bits. However, considering the length of "Data Type" is just 1 bit, we assume that the first bit of the float-point value is used for "Data Type". The other 31 bits are used for *dph$_i$*. However, when a node retrieves the value of *dph$_i$*, it should automatically add one bit "0" to the end of dph_i, so that a 32-bits float-point format can be obtained. Since the "0" is the last bit, its influence to the value of dph_i is very little.

When receiving *frame_dph*$_i$, a normal node N multiplies its hop_i by this received dph_i, so that N obtains its estimated distance to each anchor A_i, denoted as d_i. Here, $i \in \{1,2,...m\}$, if we assume that there are in total m anchors.

Table 4-3. Format of *frame_dph*$_i$

MHR	*Data Payload*		MFR
	Data Type (1 bit)	*dphi* (31 bits)	
	in total 32 bits		

At Step #3, since N has obtained its estimated distances to anchors, N can calculate its estimated position $N_{\text{DV-hop}}$ using trilateration.

4.1.2 Our Enhanced CSMA/CA (E-CSMA/CA) Access Method

4.1.2.1 Principle of E-CSMA/CA

The IEEE standard 802.15.4-2009 defines several medium access methods that can help to reduce collisions, for example, slotted CSMA/CA and non-slotted CSMA/CA [802_15_4]. The slotted CSMA/CA method requires a network coordinator which at regular intervals sends beacon messages for synchronization and network association. On the other hand, the non-slotted CSMA/CA does not require the transmission of beacons. So it can not only serve for star or tree networks, but can also serve for ad-hoc networks. Due to this simplicity and flexibility, the non-slotted CSMA/CA is a popular method for low-cost sensor networks. Therefore, in this work, we mainly focus on the non-slotted CSMA/CA method.

The original DV-hop algorithm hasn't considered the problem of frame collisions, which however is easy to happen during the broadcasts of position frames and distance-per-hop frames at the first two steps of DV-hop algorithm. Even if the 802.15.4 non-slotted CSMA/CA is used as the MAC layer protocol, it can't completely solve the collision problem of DV-hop. That is because, normally, in point-to-point communication, the CSMA/CA scheme generates the ACK (acknowledgement) response to ensure a final successful transmission. However, as for DV-hop protocol, since all the communications are fulfilled as broadcast, no ACK frame is sent, so it here becomes non-slotted CSMA/CA without ACK, which cannot make sure transmissions succeed if collisions exist. So we must propose a solution to effectively reduce collisions. In the following, we first analyze how the collisions take place, and then introduce our solution E-CSMA/CA (non-slotted Enhanced CSMA/CA without ACK).

The collisions may happen when anchors simultaneously broadcast their position frames or distance-per-hop frames. At Step #1 of DV-hop, initially, it is assumed that some anchors are simultaneously ready to broadcast their position frames. According to the principle of CSMA/CA without ACK, each anchor need first wait for a short random period, and then if the channel is still free, the position frame is sent immediately. Here, the maximum value of the short random period is $(2^{BE}-1) \times t_{\text{bo}}$, where t_{bo} is the back-off period and BE is the backoff exponent (c.f. pages 164, 171, and 172 in the IEEE standard 802.15.4-2006). Considering the default value of BE is 3, the short random period is randomly chosen among 8 values which are 0, t_{bo}, $2 \times t_{\text{bo}}$, …, $7 \times t_{\text{bo}}$. According to the standard

IEEE 802.15.4-2009, if the data rate is 250kbps, then t_{bo} is 320 μs, and the maximum value of this random period is 7×320μs=2.24ms. With such a short random waiting period, when anchors simultaneously broadcast position frames throughout the network, collisions easily occur. The same phenomenon could also happen at Step #2 of DV-hop, when several anchors send their distance-per-hop frames simultaneously.

The solution that we use to reduce collisions is to make the senders (nodes ready for sending frames) wait for another longer random duration before they perform CSMA/CA. So the probability of collision is reduced. The details about this longer waiting period are described in the following.

At the beginning of Step #1 of DV-hop, each anchor A_i first wait for a random duration denoted as t_{wpi}. Then, A_i performs CSMA/CA and sends its position frame. Similarly, at the beginning of Step #2 of DV-hop, after each anchor A_i has calculated its distance per hop denoted as dph_i, A_i waits for a random duration denoted as t_{wdi}. Then, it performs CSMA/CA before sending its distance-per-hop frame $frame\_dph_i$.

The following two figures show how collisions happen and how our access method E-CSMA/CA works. In Figure 4-1, it is assumed that three anchors A_1 A_2 A_3 start their first step simultaneously at the time T_0 when they perform the non-slotted CSMA/CA without ACK. A_1 and A_2 happen to choose the same period $2 \times t_{bo}$, while A_3 wait for a longer period $5 \times t_{bo}$ before broadcasting its position frame. Since A_1 and A_2 send out their position frames at the same time, the two frames will arrive simultaneously at the common neighbor node of both A_1 and A_2, thus a collision occurs at Step #1. The same phenomenon could take place at Step #2, with A_2 and A_3 choosing the same waiting period $1 \times t_{bo}$.

Figure 4-2 shows an example of our collision reduction method, using the same scenario of Figure 4-1. Comparing these two figures, we can see that our method adds an extra random duration before the beginning of the CSMA/CA procedure at each anchor. At the cost of additional waiting time, our method reduces the probability of simultaneous emissions; therefore, fewer collisions can occur.

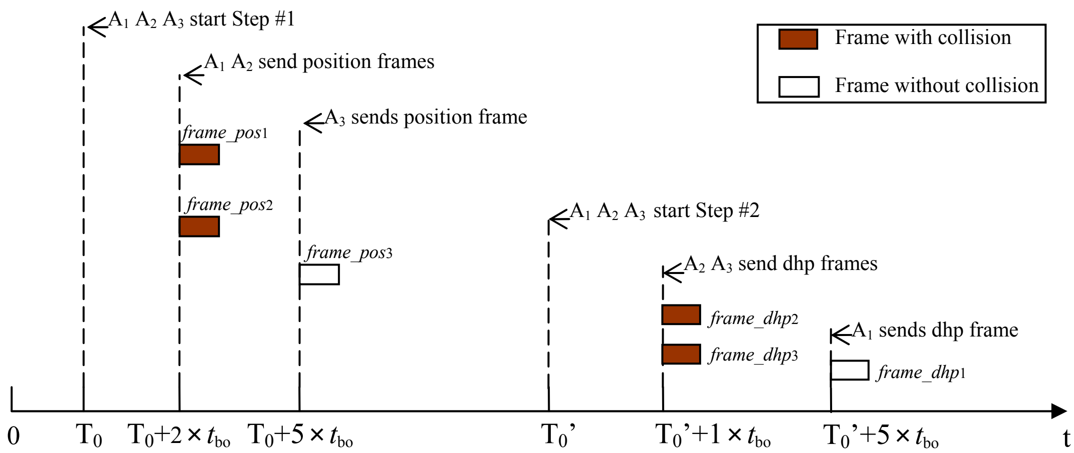


Figure 4-1. Collisions Occur at Step #1 and Step #2

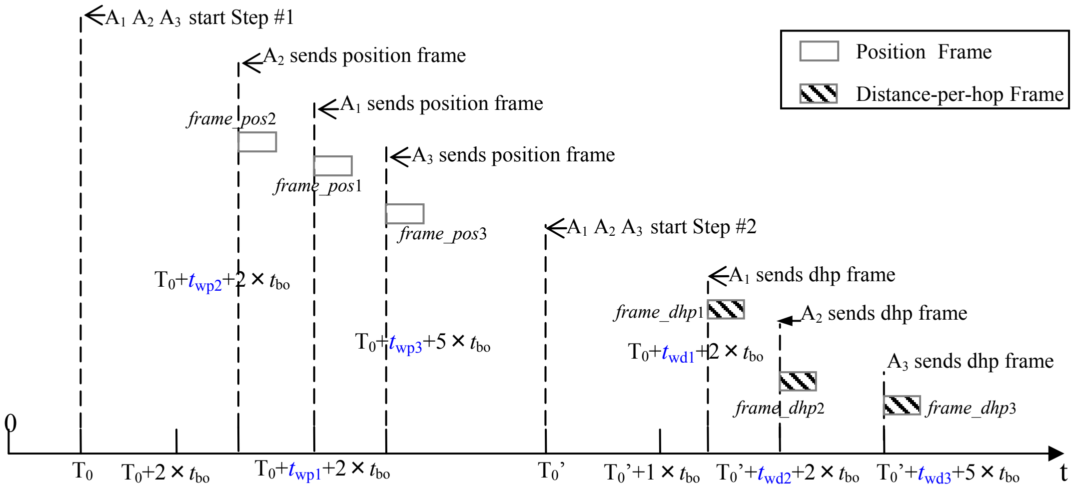


Figure 4-2. Example of Our Access Method E-CSMA/CA

In fact, our collision reduction method E-CSMA/CA should also be applied to the relay nodes. These relay nodes, either anchors or normal nodes, help relay the position frame or distance-per-hop frame by broadcast. According to our method, every time a relay node is ready to perform CSMA/CA, this node needs to wait for a supplementary random duration t_{wr}.

4.1.2.2 Effect of E-CSMA/CA Observed from Simulation

Through simulations, we want to prove the good effect of E-CSMA/CA method. The network simulator we use is WSNet, which is an event-driven simulator designed by three researchers from INRIA. Comparing with other simulators like NS-2 [NS_2] and OPNET [OPNET], WSNet has two main advantages [WSNet]. First, it supplies many modules for each layer based on IEEE standard 802.15.4, including different radio propagation modules for PHY layer and medium access modules for MAC layer. Second, WSNet facilitates the development of new algorithms. WSNet has integrated sensor nodes with its behaviors such as mobility, birth, death and packet transmission. Thus, it is easy to access and control the behaviors of nodes when we create our algorithms. Using WSNet, in C language, we have implemented some DV-hop based algorithms.

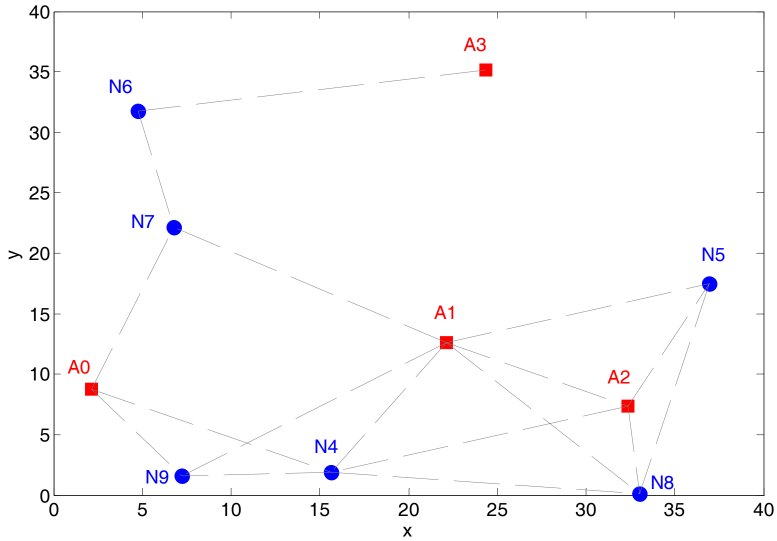


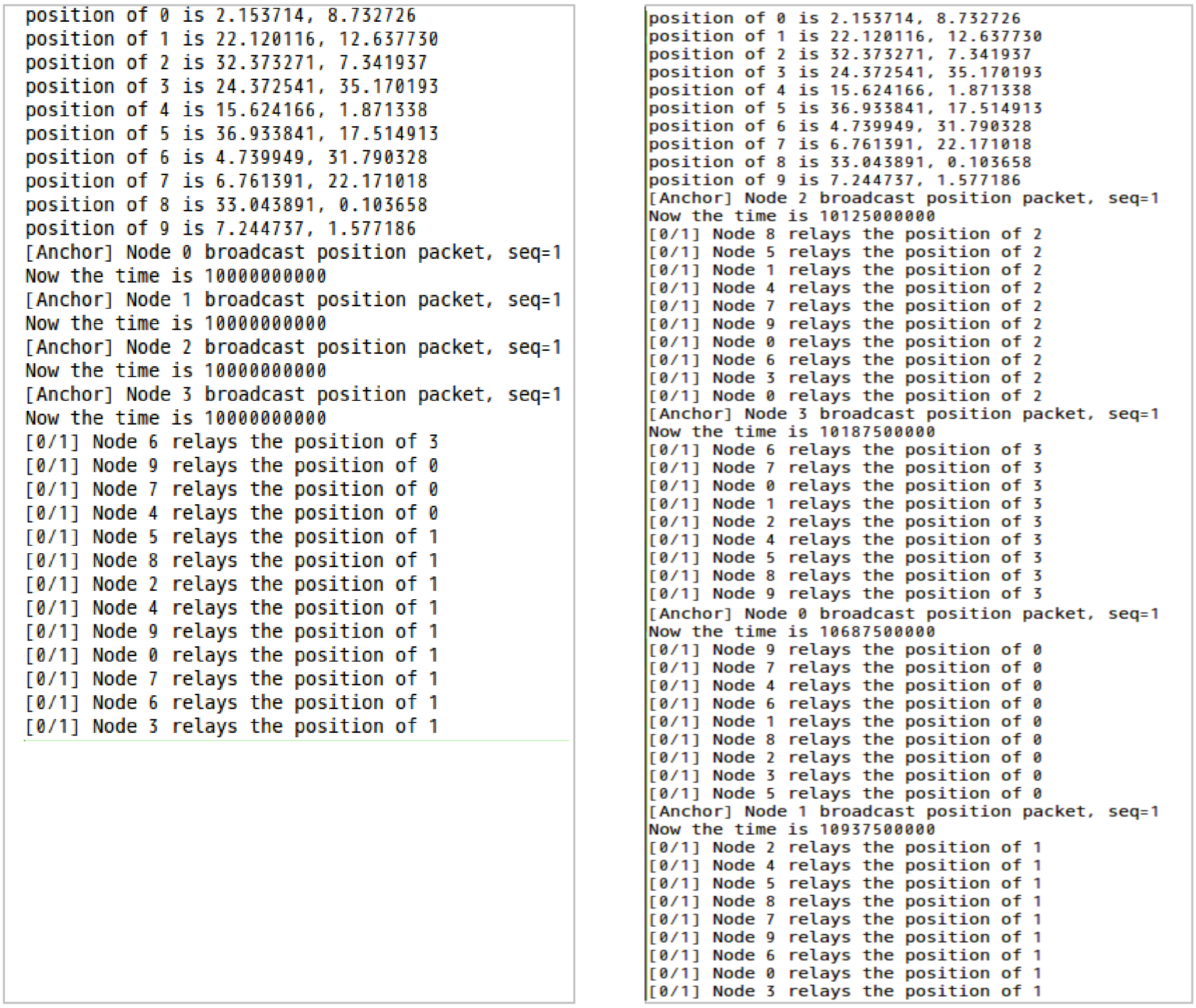
Figure 4-3. Network Topology in the Simulation

In the simulation, the topology of reference network is shown in Figure 4-3. This small network locates in a 40×40m^2 area. Inside the area, we uniform-randomly distribute 10 nodes. That means, the positions of the nodes are randomly assigned. All the nodes are static. Among the nodes, 4 are anchors, while other 6 are normal nodes. The communication range is set to be 20m.

The physical layer of the network conforms to the IEEE standard 802.15.4-2009. The radio frequency is 2.4GHz, and the signal modulation is OQPSK. In order to exclude the influence from radio propagation, in this subsection, we use an ideal radio environment with no interference and no signal loss. In MAC layer, we will investigate and compare two methods: non-slotted CSMA/CA, and our E-CSMA/CA.

The major difference between non-slotted CSMA/CA and our E-CSMA/CA is: E-CSMA/CA has additional random waiting time before CSMA/CA. As for our E-CSMA/CA method, as an example, the values for the key parameters are set as following. t_{wpi} or t_{wdi}, which is the waiting time for each anchor A_i sending its position frame or distance-per-hop frame, is randomly selected between 0 and 0.5s.

In the following, the simulation results will step by step display the process of DV-hop algorithm. The process of Step #1 is shown in Figure 4-4, which comprises two subfigures. In Figure 4-4(a), non-slotted CSMA/CA is used, while in Figure 4-4(b) it is our E-CSMA/CA method.



(a) non-slotted CSMA/CA (b) our E-CSMA/CA

Figure 4-4. Step #1 of DV-hop Algorithm Simulated by WSNet

It is assumed that, ideally, all nodes can receive and then relay the position frame from each anchor. However, in Figure 4-4(a), we can observe that, using non-slotted CSMA/CA, when the anchors broadcast their positions simultaneously, nobody receives the position of A_2, while the position frame of A_3 is only received by N_6. Why other nodes cannot receive the position of A_2 or A_3? The reason should be frame collisions. For example, from the network topology in Figure 4-3, since A_3 has only one neighbor node N_6, its position frame should be first received by N_6. Then, N_6 relays the position frame of A_3. The neighbor of N_6, that is N_7, is supposed to receive this relayed frame. But at the same time, N_7 is also relaying the position frame of A_0. Thus, collision happens on these two relayed frames.

On contrary, the good results are shown in Figure 4-4(b) for our E-CSMA/CA method. We can note that, the position frame from each anchor has been successfully received and relayed by all nodes. This is contributed by the random waiting time added to non-slotted CSMA/CA.

The similar phenomenon can also be observed for Step #2. Considering the similarity of simulation results for Step #1 and Step #2, here we don't give the result figure of Step #2.

From the simulation results, we can conclude that, our E-CSMA/CA method is an efficient solution to reduce the frame collisions in DV-hop localization algorithm.

4.1.3 Our Parameters for the End of Each Step

As for DV-hop algorithm, the first step ends as soon as every node in the network has received all anchors' position frames, while the second step ends on condition that all anchors' distance-per-hop frames have been received. These two ending conditions can be fulfilled in an ideal scenario by a mathematic simulator such as MATLAB. However, in practical network scenarios, the ending conditions cannot be reached because the algorithm will encounter two problems. Solving the problems, we propose several parameters to control the end of the first two steps of DV-hop.

As for the first problem, it is unnecessary for nodes to receive all anchors' positions, especially when the total number of anchors is very large. Because mobile normal nodes need to calculate their positions as quickly as possible, it could take too much time for them to collect all anchors' positions. Therefore, each node needs to set a maximum number of anchors whose information they take into account: the node will then wait until it has identified this number of distinct anchors. This maximum number of anchors can be denoted as '*num_wait_pos'*. Then, as long as a normal node has received *num_wait_pos* anchors' positions, it can stop relaying position frames and end Step #1 of DV-hop algorithm. As for anchors, when an anchor has received *num_wait_pos-1* anchors' positions, it can end Step #1, (here, it is '*num_wait_pos -1'* instead of '*num_wait_pos',* because the number 'num_*wait_pos'* includes A_i). Similarly, if a normal node has received *num_wait_dph* anchors' distance-per-hop, it can end Step #2. As for anchors, the number threshold will be *num_wait_dph-1*. Normally, *num_wait_pos* is no less than *num_wait_dph*.

The second problem occurs when some frames are lost or the total number of anchors is less than '*num_wait_pos' or* '*num_wait_dph'*. When transmitted frames are partly lost at the first two steps because of collisions or bad channel quality, a few nodes may miss some anchors' position frames as well as distance-per-hop frames. As a result, these nodes might never receive as many as '*num_wait_pos'* anchors positions, neither *num_wait_dph* anchors' distance-per-hop. Of course, this phenomenon could also happen if the total number of anchors is less than '*num_wait_pos' or* '*num_wait_dph'*. Timers will be used to solve this problem.

In order to end Step #1, we need to set a timer for each node N_i at the time instant $T^0_i+t_{s1}$. Here, since all nodes periodically execute DV-hop localization protocol, T^0_i is N_i's beginning time of its localization period. All nodes could have the same beginning time if the network is well synchronized. If this is not the case, each node might begin its period at a different instant. t_{s1} is the maximum duration of Step #1, which is configured and shared by all nodes. Before the expiration of $T^0_i+t_{s1}$, those anchors who have already received as many as '*num_wait_pos-1'* anchors' positions must immediately end Step #1 and enter Step #2. When $T^0_i+t_{s1}$ arrives, the anchors who haven't yet received the specified amount of data need to immediately end Step #1 and enter Step #2.

In order to end Step #2, we need to set a timer at the time instant $T^0_i+t_{s1}+t_{s2}$. Here, t_{s2} is the maximum duration of Step #2, which is shared by all normal nodes. In fact, Step #3 of DV-hop algorithm is designed for normal nodes to calculate their positions. Hence, the timer for starting Step #3 is specific to the normal nodes. Before $T^0_i+t_{s1}+t_{s2}$, those normal nodes, who have already received as many as '*num_wait_dph'* anchors' distance-per-hop frames and '*num_wait_pos'* anchors' position frames, could immediately start Step #3. When the time '$T^0_i+t_{s1}+t_{s2}$' arrives, other normal nodes, who haven't yet received the specified amount of data, need to nevertheless start Step #3.

As a summary for this subsection, these parameters are illustrated in the following state transit diagrams. Note that in these diagrams, t_p is the duration of a localization period.

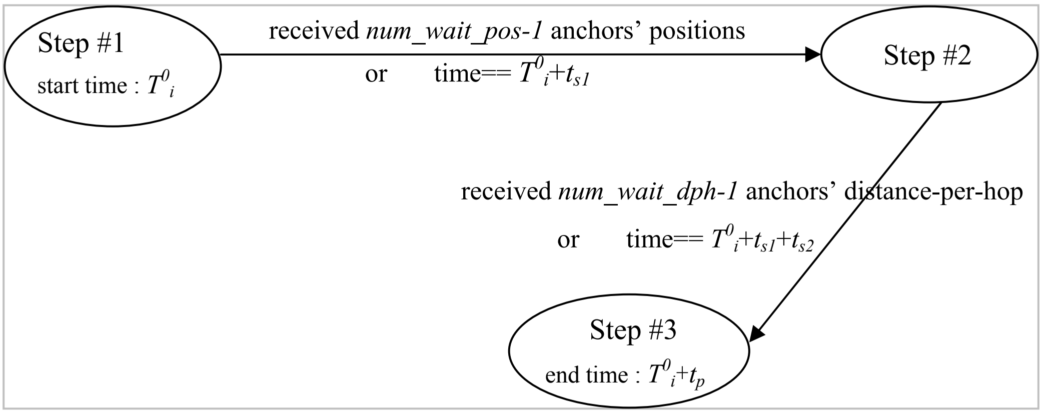


Figure 4-5. Transit Diagram in One Localization Period for an Anchor A_i

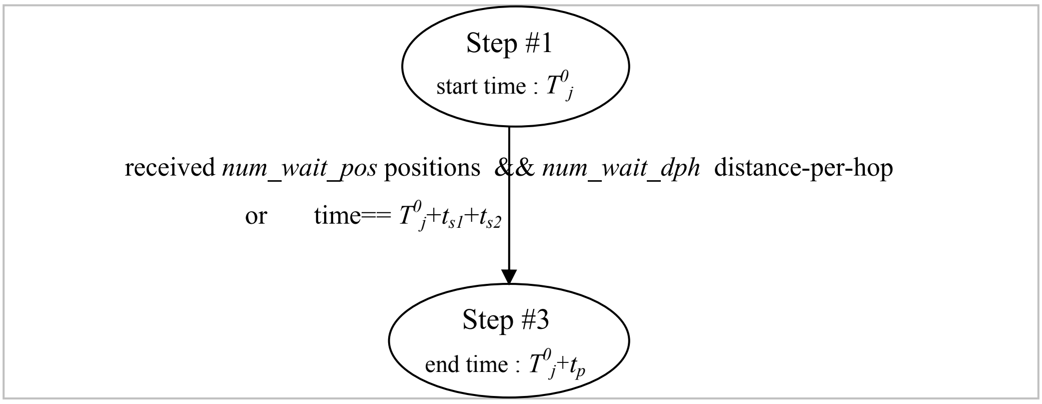


Figure 4-6. Transit Diagram in One Localization Period for a Mobile Node N_j

In DV-hop algorithm, all broadcasts of frames are included at Step #1 and Step #2, while Step #3 only includes the position calculation. Since broadcasts normally take much more time than calculation, the total duration of Step #1 and Step #2 is very close to the entire period of localization. That is, $t_{s1} + t_{s2} \approx t_p$. Besides, since Step #1 and Step #2 both broadcast frames, their duration should be similar. We can consider $t_{s1}\approx t_{s2}$. For example, t_{s1} could be set as t_p /2, while t_{s2} could be set as t_p*(3/8). Then, the time left is devoted to Step #3, that is: t_p- t_{s1} - t_{s2} =t_p/8.

4.1.4 Procedure of Our DV-hop Localization Protocol

The execution of our DV-hop localization protocol is shown in the following two figures. One figure shows the procedure for anchors and another illustrates the procedure for normal nodes.

Figure 4-7 shows the procedure followed by each anchor A_i. The duration of the localization period is t_p, and A_i begins its period at the time T^0_i. Then, according to our collision avoidance method, A_i first waits for a random duration t_{wpi}, and then broadcasts through the network its position frame which has been defined in the section 4.1.1. Meanwhile, A_i also receives and relays the positions frames of other anchors. When A_i has received '*num_wait_pos -1*' anchors' position frames, it will immediately end Step #1 and enter Step #2. This time instant is denoted as Tr_i. However, if A_i couldn't receive as many as '*num_wait_pos -1*' anchors' position frames until the time instant T^0_i+t_{s1}, it will still end Step #1 when it reaches T^0_i+t_{s1}. So A_i ends Step #1 at the time instant Tr_i or T^0_i+t_{s1}.

Immediately after Step #1, A_i begins Step #2 by calculating its distance-per-hop. Then, according to our collision reduction method, A_i waits for a random duration t_{wdi}, and then broadcasts through the network its distance-per-hop frame. Meanwhile, A_i also helps relay the distance-per-hop frames of other anchors. Here, the end of A_i's Step #2 is also the end of its participation in the localization period, since the third step is designed for normal nodes.

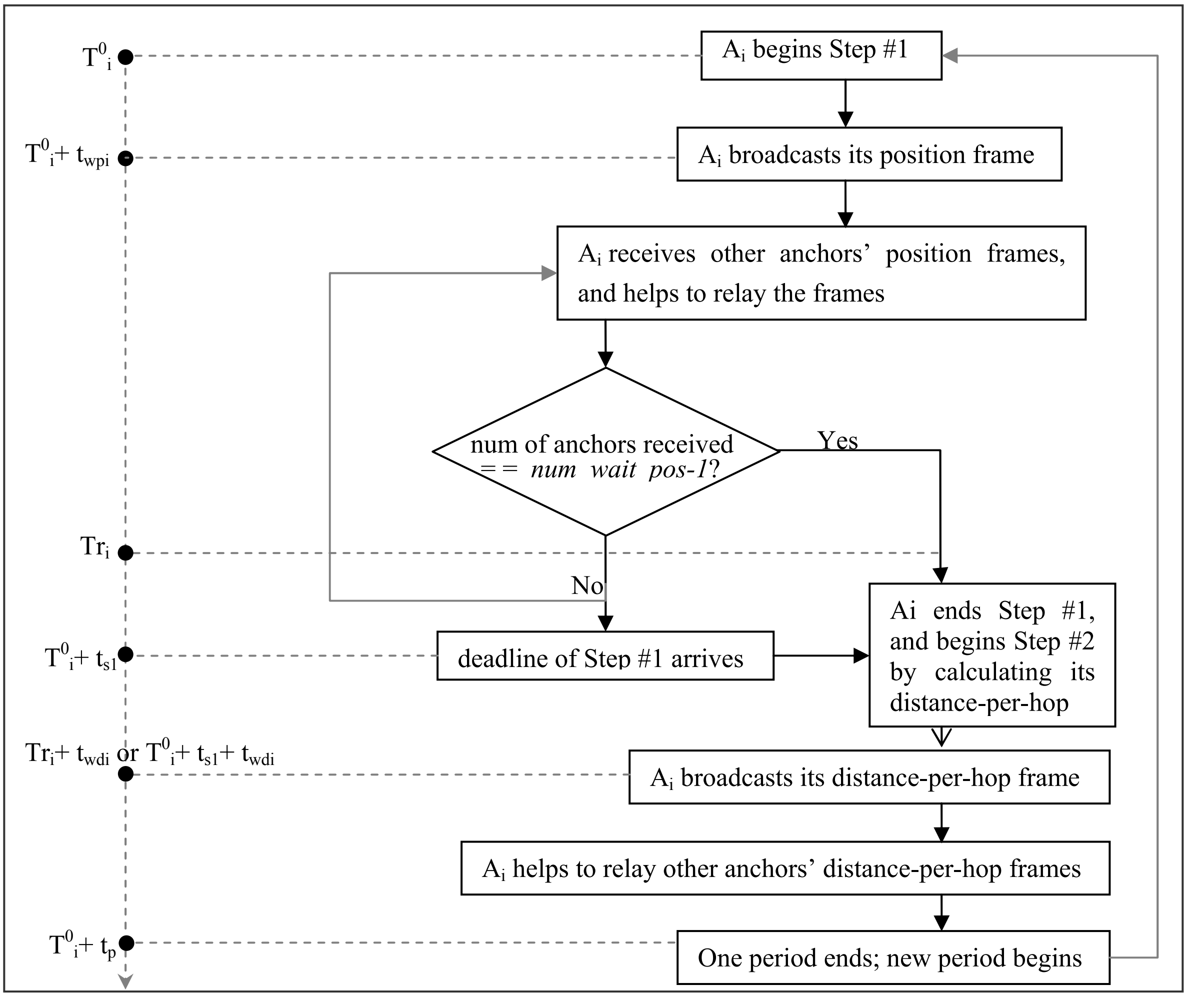


Figure 4-7. Procedure for Each Anchor A_i

Figure 4-8 shows the procedure for each normal node N_j. It begins its period at the time T^0_j. During the first two steps, N_j receives and relays anchors' frames. When N_j has received as many as *num_wait_pos* anchors' position and as many as *num_wait_dph* anchors' distance-per-hop frames, it will immediately end the first two steps and start the third step. This time instant is denoted as Tr_j. However, if N_j couldn't receive as many as *num_wait_dph* distance-per-hop frames until the time $T^0_j+t_{s1}+t_{s2}$, it will end Step #2 anyway. Since t_p is the duration of the period, at the time $T^0_j+t_p$, N_j will end the current period.

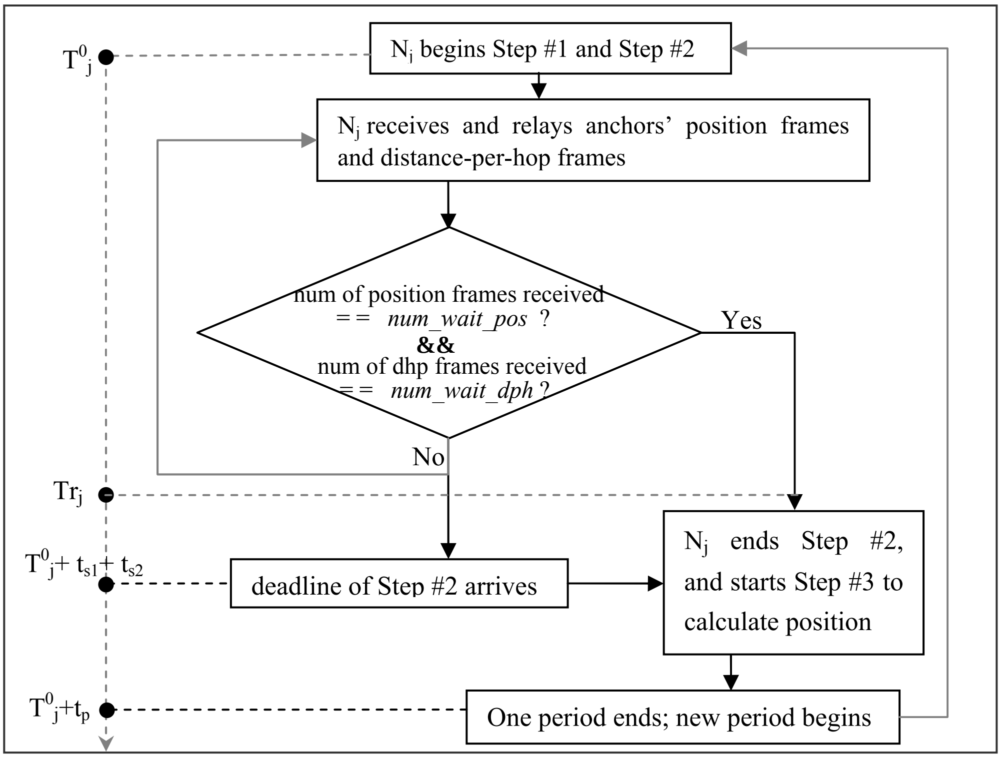


Figure 4-8. Procedure for Each Normal Node N_j

In this section about our DV-hop localization protocol, we have presented the frame structure, the improved collision reduction method, several parameters to end each step of DV-hop, and finally the procedure of protocol. Now, using this protocol, the original DV-hop algorithm can be implemented in network scenarios.

4.2 Evaluation of DV-hop Protocol by WSNet

In this section, based on the implementation of our DV-hop protocol, we evaluate the performance of the original DV-hop, Checkout DV-hop, and Selective 3-Anchor DV-hop algorithms. First, we assign values to the parameters of our DV-hop protocol. Second, we configure simulation scenarios and implement the protocol by using the network simulator WSNet. Third, through the WSNet simulations, we investigate the specific performance of the original DV-hop algorithm. Finally, in terms of mobility, synchronization and network overhead, we present comparative evaluation of the original DV-hop, Checkout DV-hop, and Selective 3-Anchor DV-hop algorithms.

4.2.1 Parameters Quantization

In the previous section, we have proposed the DV-hop localization protocol as well as several important parameters of the protocol. But when we implement the protocol, we need first quantize its parameters.

Proposed in subsection 4.1.2, t_{wpi} is A_i's random waiting time before performing CSMA to broadcast its position frame, while t_{wdi} is A_i's random waiting time before broadcasting its distance-per-hop frame. As for the range of t_{wpi} or t_{wdi}, as an example, we can set their minimum value as 0. Their maximum value cannot be too small; otherwise different anchors might easily send frames at the same time, making collisions happen. We consider that 0.5s could be big enough for this maximum value, comparing with an example (just 2.24ms) of maximum waiting time of CSMA/CA in subsection 4.1.2. So, in simulation, t_{wpi} and t_{wdi} are uniform-random values between 0 and 0.5s.

Also proposed in subsection 4.1.2, t_{wr} is any relay node's random waiting time before it resends the position frame or distance-per-hop frame. The maximum value of t_{wr} should not be too big, because mobile nodes cannot wait too long to receive the positions or distance-per-hop from the faraway anchors. In our simulation, as an example, the maximum value of t_{wr} is set as 10ms, and its minimum value is 0.

Since low-cost sensor nodes have limited memory, we assume that, each node can receive at most 30 anchors' positions at Step #1, and at most 20 anchors' distance-per-hop at Step #2. That is to say, *num_wait_pos* and *num_wait_dph* proposed in subsection *4.1.3* are respectively 30 and 20.

4.2.2 Scenario Configuration

Our simulation scenario takes place within a $100 \times 100m^2$ area. Inside the area, 100 nodes including anchors and normal nodes are randomly placed according to a uniform distribution. An example of distribution is shown in Figure 4-9. In this example, 5 of the 100 nodes are anchors which are represented as squares, while others are normal nodes. So, this figure gives an example of a 5% ratio of anchors, which is defined as the ratio of the number of anchors to the total number of nodes.

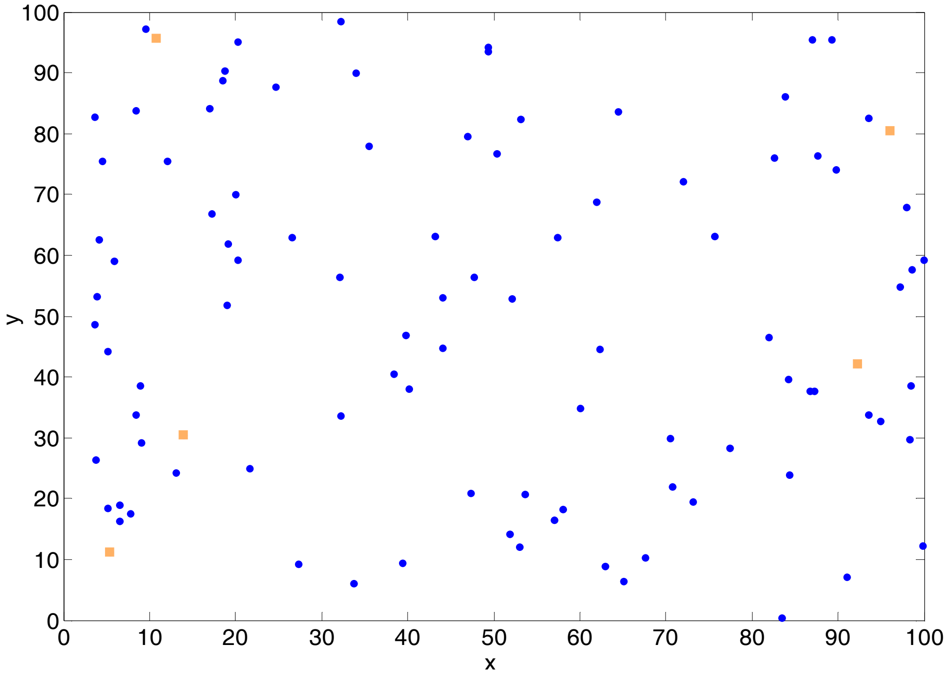


Figure 4-9. Example of Nodes Distribution

The scenario parameters and their values are listed in Table 4-4 where the last 5 parameters marked by '*' have different values in different scenarios, while other parameters are constant over the scenarios.

We use a log-distance pathloss radio propagation model, which is usually applied in indoor scenarios [DPS 08] [ARH 10] [MA 12]. Note that the problem of interference from other technologies using the same 2.4GHz frequency band is not studied in our scenarios.

The network can be synchronized so that all nodes can simultaneously begin their localization period), or unsynchrized (nodes start time will be different). As for mobility, the anchors are static, while normal nodes may be mobile. All these scenarios are considered, and their simulation results will be presented in the following subsections. We will first investigate the performance of DV-hop algorithm, and then compare it with that of Checkout DV-hop and Selective DV-hop algorithms.

Table 4-4. Senario Parameters

Radio range of nodes	20 meters
Physical Data rate	250kbps
Radio propagation	Log-distance pathloss propagation model
Interference	none
Physic layer protocol	IEEE 802.15.4, 2.4GHz, OQPSK
MAC layer protocol	IEEE 802.15.4 non-slotted CSMA/CA
Localization period t_p	6s
A_i's waiting time before sending: t_{wpi} and t_{wdi}	both randomly selected between 0 and 0.5s
Maximum duration of Step #1: t_{s1}	$1/2*t_p$=3s
Maximum duration of Step #2: t_{s2}	$3/8*t_p$=2.25s
Maximum waiting number: *num_wait_pos*	30
Maximum waiting number: *num_wait_dph*	20
Network synchronized or not *	to be decided in specific scenario
Ratio of anchors *	to be decided in specific scenario
Nodes mobility *	to be decided in specific scenario
Network simulation time *	to be decided in specific scenario

* : parameters having different values in different scenarios

4.2.3 Evaluation on DV-hop Algorithm Using Our DV-hop Protocol

Based on our DV-hop localization protocol, we will present in total 6 scenarios for the original DV-hop algorithm, including 4 static scenarios, 1 mobile synchronized scenario and 1 mobile unsynchronized scenario. From the first three static scenarios, we aim to obtain specific performance of DV-hop algorithm without the influence of node movement. But from the fourth static scenario and other 2 mobile scenarios, we aim to know the general performance. So, as for Static Scenario 1, 2 and

3, we set the network simulation time of each scenario as only 18s (equal as 3 localization periods), so that we can get 3 particular cases for each. As for the general static scenario and mobile scenarios, the simulation time is set as 3000s (equal as 500 periods), so that the average performance is presented.

4.2.3.1 Static Scenario 1

Most parameters having already been listed in Table 4-4, we will only assign values to the parameters marked with an asterisk. Table 4-5 lists these parameters.

Table 4-5. Particular Parameters of Static Scenario 1

Network synchronized or not	Synchronized and all nodes start at the same time
Ratio of anchors	5%
Nodes mobility	Static (distribution as Figure 4-9)
Network simulation time	18s (3 localization periods)

Since the simulation runs for 3 localization periods, we can obtain 3 particular results, as shown in Table 4-6. The results are examined using two criteria, location error and number of transmitted frames. As shown in Table 4-6, the location error (in meters) is the average of all distances between each normal node's estimated position and its real position. The location error can be used to evaluate the accuracy of the DV-hop protocol. A smaller location error indicates better accuracy performance. Another parameter is the number of transmitted frames, which is the total number of frames transmitted by all nodes during one localization period of the DV-hop protocol. The number of transmitted frames can be used for evaluating the network overhead. A higher value indicates higher network overhead.

Table 4-6. Performance Results of Static Scenario 1

Result 1		Result 2		Result 3	
Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames
17.60/20 = 88%	1071	12.03/20 = 60%	1223	10.78/20 = 54%	1063

From Table 4-6, we can reach the following conclusions:

(1) Even if the same scenario is applied, as for each period, we could obtain different results. This is caused by the random nature of some parameters, for example, in Table 4-4, t_{wpi} and t_{wdi} (A_i's random waiting time before sending its frames). Consequently, for each period, the collisions might happen between different nodes and at different times. As a result, the performance will be different for each result.

(2) The accuracy could be quite different from a run to the other. For example, the location error of Result 1 is much bigger than that of Result 3. However, the network overhead difference is similar, as shown by the number of transmitted frames of Result 1 and Result 2.

(3) The average location error of the three results is 13.47 meters (that is 67% in percentage of radio range), while the average number of transmitted frames is 1119. These average results can be finally regarded as the average performance of the Static Scenario 1.

4.2.3.2 Static Scenario 2

From Static Scenario 1 to Static Scenario 2, only the ratio of anchors changes from 5% to 40%.

We can also obtain 3 particular results, as shown in Table 4-7.

Table 4-7. Performance Results of Static Scenario 2

Result 1		Result 2		Result 3	
Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames
14.97/20 =75%	6783	10.01/20 = 50%	7001	16.89/20 = 84%	6780

From Table 4-7, we can deduce the following:

(1) As expected, when there are more anchors in the network, the network overhead of DV-hop protocol will increase. This can be observed by comparing the number of transmitted frames in Table 4-6 and Table 4-7. When the number of anchors is 40, the number of transmitted frames is very large, normally more than 6700, which brings heavy traffic to the network.

(2) An increase in the number of anchors doesn't necessarily improve the localization accuracy of DV-hop algorithm. This conclusion can be obtained by comparing Table 4-6 and Table 4-7. The location errors in Table 4-7 (with 40 anchors) are a little higher than those in Table 4-6 (with 5 anchors). One reason is that when the anchors number is large, the traffic in the network becomes heavy, which leads to more collisions. This in turn prevents normal nodes from receiving the right position frames.

4.2.3.3 Static Scenario 3

From Static Scenario 2 to Static Scenario 3, the ratio of anchors changes from 40% to 80%.

We can also obtain 3 particular results, as shown in Table 4-8.

Table 4-8. Performance Results of Static Scenario 3

Result 1		Result 2		Result 3	
Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames	Location error (% radio range)	Number of transmitted frames
15.75/20 =79%	12072	17.87/20 =89%	11895	20.02/20 =100%	11981

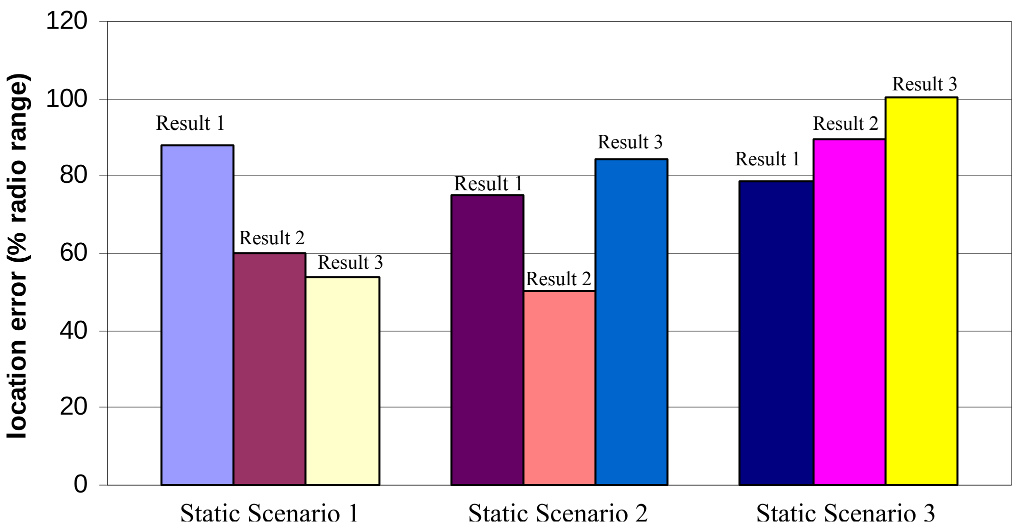


Figure 4-10. Location Error (% radio range) of Static Scenario 1, 2 and 3

Generated from Tables 4-6, 4-7 and 4-8, Figure 4-10 compares the location error (in percentage of radio range) of the above 3 scenarios, while Figure 4-11 compares the number of transmitted frames.

From these two figures, we can reach the following conclusion: if there are too many anchors, the network traffic will be too heavy, generating excessive collisions and causing the localization accuracy to decline. As a result, when the ratio of anchors is as greater than 40%, instead of using DV-hop algorithm, we need to use other low-traffic localization algorithms, such as Centroid and CPE.

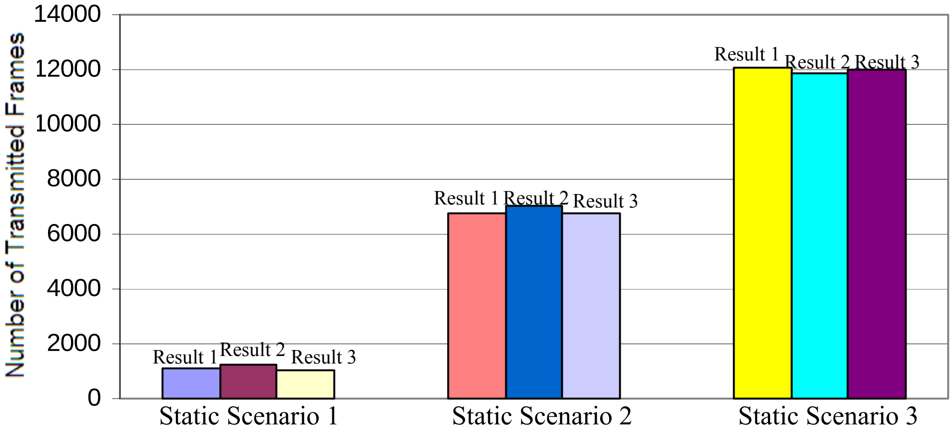


Figure 4-11. Number of Transmitted Frames of Static Scenario 1, 2 and 3

4.2.3.4 General Static Scenario and Mobile Scenarios

From the above three static scenarios, we found that DV-hop protocol is not suitable for scenarios with large number of anchors. From now on, we will configure the scenarios with no more than 30 anchors in total (the total number of nodes still being 100). In the following, we present three scenarios, including general static scenario, synchronized mobile scenario and unsynchronized mobile scenario. First, we list the particular parameters for each scenario (the common parameters are the same as Table 4-4). Then, their results are presented together.

4.2.3.4.1 Particular Parameters of General Static Scenario

The particular parameters of general static scenario are listed in Table 4-9. In order to obtain more general results than the previous three static scenarios, we increase the simulation duration to 3000 seconds which allows for 500 localization periods.

Table 4-9. Particular Parameters of General Static Scenario

Network synchronized or not	Synchronized and all nodes start at the same time
Ratio of anchors	5, 10, 15, 17, 19, 20, 25, 30 / 100
Nodes mobility	Static (distribution as Figure 4-9)
Network simulation time	3000s (500 localization periods)

4.2.3.4.2 Particular Parameters of Synchronized Mobile Scenario

The particular parameters of the synchronized mobile scenario are listed in Table 4-10. Anchors remain static, while normal nodes move in billiard [RMN 11] [ST 11] mode. That means, when a normal node reaches the edge of the $100 \times 100m^2$ simulation area, this node will rebound like a billiard ball. The speed is fixed as 0.5m/s, which corresponds to low-speed human movement.

Table 4-10. Particular Parameters of Synchronized Mobile Scenarios

Network synchronized or not	Synchronized and all nodes start at the same time
Ratio of anchors	5, 10, 15, 17, 19, 20, 25, 30 / 100
Nodes mobility	Anchors are static, normal nodes move at a speed of 0.5m/s in billiard mode.
Network simulation time	3000s (500 localization periods)

4.2.3.4.3 Particular Parameters of Unsynchronized Mobile Scenario

The particular parameters of the unsynchronized mobile scenario are the same as described in Table 4-10, except for the synchronization. Here, nodes will start at different time. Some nodes might start very late, while others start earlier. This means that when late nodes begin Step #1, some early nodes might have already finished their Step #2. For example, as shown in Figure 4-12, anchor A_i starts its Step #1 so late that anchor A_k has already ended its Step #2.

However, this kind of unsynchronized situations has been considered by our DV-hop protocol. In the protocol, when a normal node is working at Step #2, it can receive both distance-per-hop and position frames. Therefore, no matter how late an anchor begins Step #1, its position frame and distance-per-hop frame will sooner or later be received by all nodes.

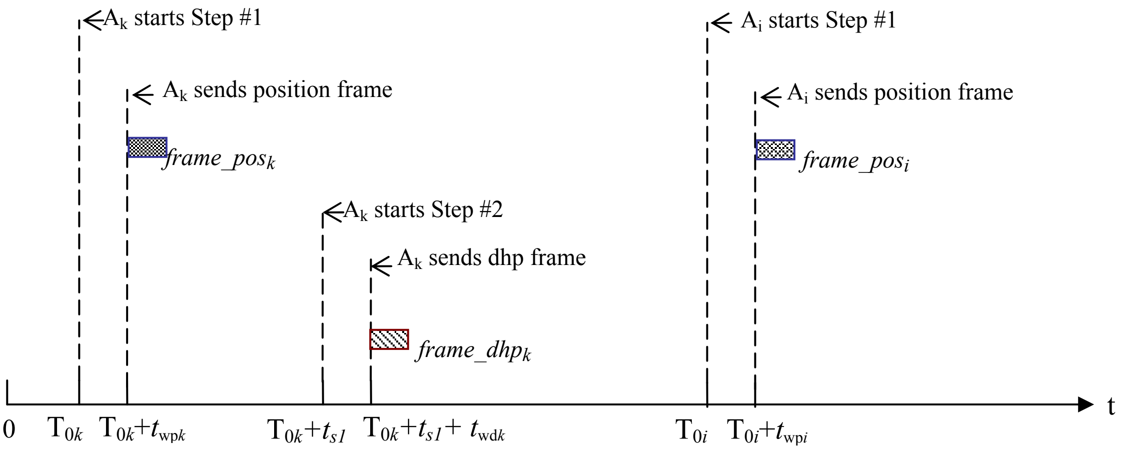


Figure 4-12. Example of Unsynchronized Scenario

4.2.3.4.4 Simulation Results of General Static Scenario and Mobile Scenario

The simulation results of our three scenarios (general static, synchronized mobile, and unsynchronized mobile) using the original DV-hop algorithm are presented in Figure 4-13 and 4-14. The data is collected on a per anchor ratio basis. Figure 4-13 shows the average location error per node per localization period, expressed as a percentage of the radio range. Figure 4-14 presents the average number of transmitted frames per localization period.

From Figure 4-13, we can see that, for all scenarios, as the number of anchors increases, the location error declines, which means the localization accuracy improves. As expected, the location error increases when the number of anchors goes over 20 or 25. This is caused by the increase in frame collisions. As there are many anchors, a large number of frames are broadcasted through the network, where the collisions can easily occur.

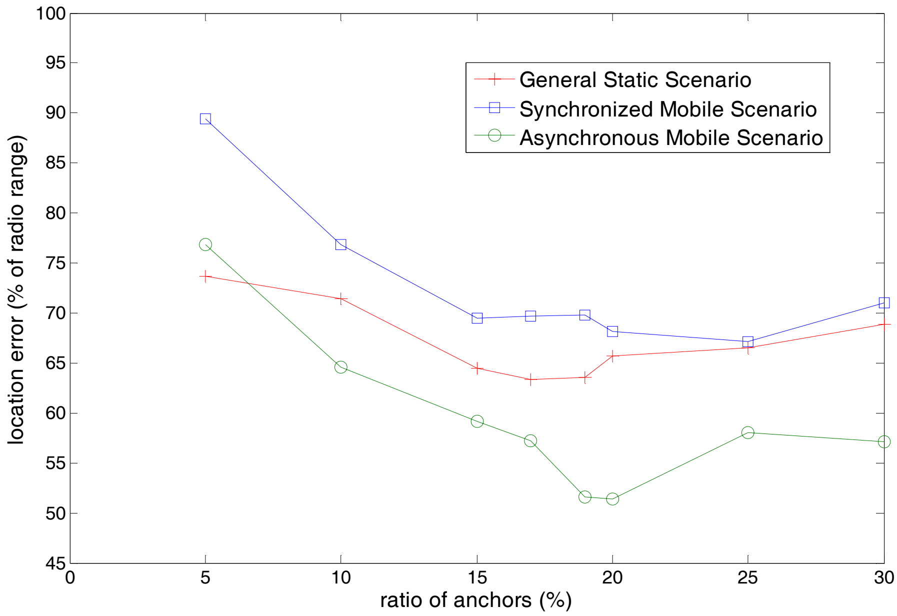


Figure 4-13. Location Error in General Static and Mobile Scenarios

Comparing the location error between general static and synchronized mobile scenarios in Figure 4-13, we can see the influence of node mobility. The location error of synchronized mobile scenario is normally a little bigger than that of general static scenario. The reason may be that we haven't used any position prediction method. Therefore, when nodes are mobile, their estimated positions do not match their latest positions.

From Figure 4-13, we can also notice that, although lacking a position prediction mechanism, the unsynchronized mobile scenario generally has the best accuracy. That is because, in the unsynchronized scenario, nodes generally start their localization period at different times. Hence, compared with the synchronous scenario, the anchors have less chance to broadcast their positions simultaneously, resulting in fewer collisions.

We notice that the accuracy performance of DV-hop is not so satisfying. Its minimum location error corresponds to half the radio range. These results will nevertheless serve as a benchmark in the evaluation of Checkout DV-hop and Selective 3-Anchor DV-hop algorithms. Their simulation results will be presented in the next section.

The three scenarios have almost the same simulation results regarding the number of transmitted frames, as shown in Figure 4-14. We can see that when the number of anchors is less than 20, the number of transmitted frames increases linearly with the number of anchors. But this linearity ends when the number of anchors exceeds 20. That is because, according to the settings, each node is supposed to keep at most 20 anchors' distance-per-hop values at Step #2. That means, when a node has obtained as many as 20 anchors' distance-per-hop, its memory for distance-per-hop is supposed to be completely occupied. If this node receives another distance-per-hop frame in the future, it has to discard this frame. However, in a scenario with less than 20 anchors, since the memory for distance-per-hop can never be completely occupied, new anchors' distance-per-hop frames are always recorded and then transmitted instead of being discarded.

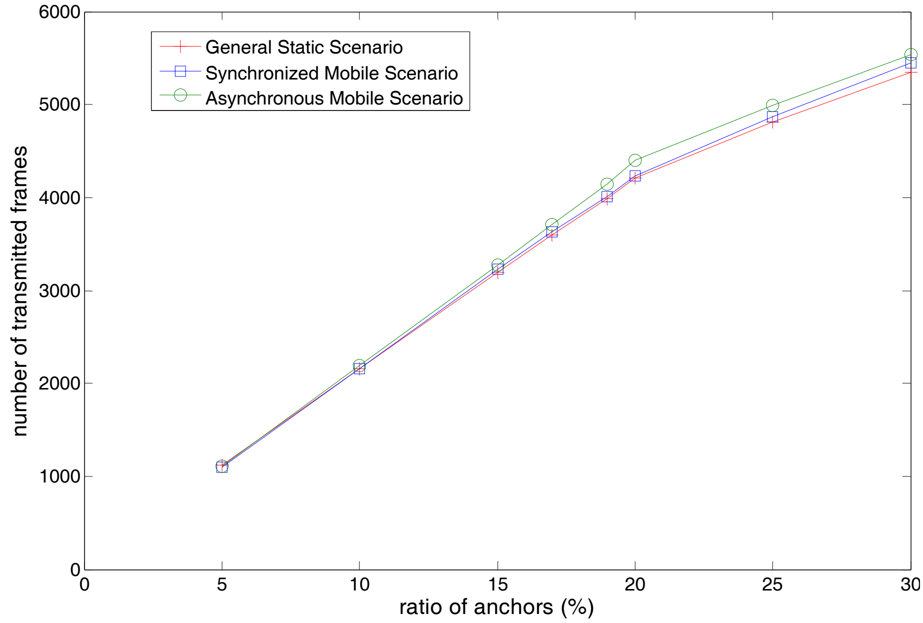


Figure 4-14. Number of Transmitted Frames in General Static and Mobile Scenarios

4.2.4 Comparative Evaluation of DV-hop Based Algorithms

Checkout DV-hop and Selective 3-Anchor DV-hop algorithms both share the same Step #1 and Step #2 with DV-hop algorithm. The difference between these 3 algorithms lies in the calculation part that is Step #3. Therefore, Checkout DV-hop and Selective 3-Anchor DV-hop can use the same protocol as the one used for DV-hop algorithm. The following sections will present the comparison of the simulation results of these 3 algorithms.

4.2.4.1 Comparison under Static Scenarios

The static scenarios we use here are the same as those in the section 4.2.3.4.1. The simulation results about the number of transmitted frames remain the same as Figure 4-14. The results on location error are shown in Figure 4-15.

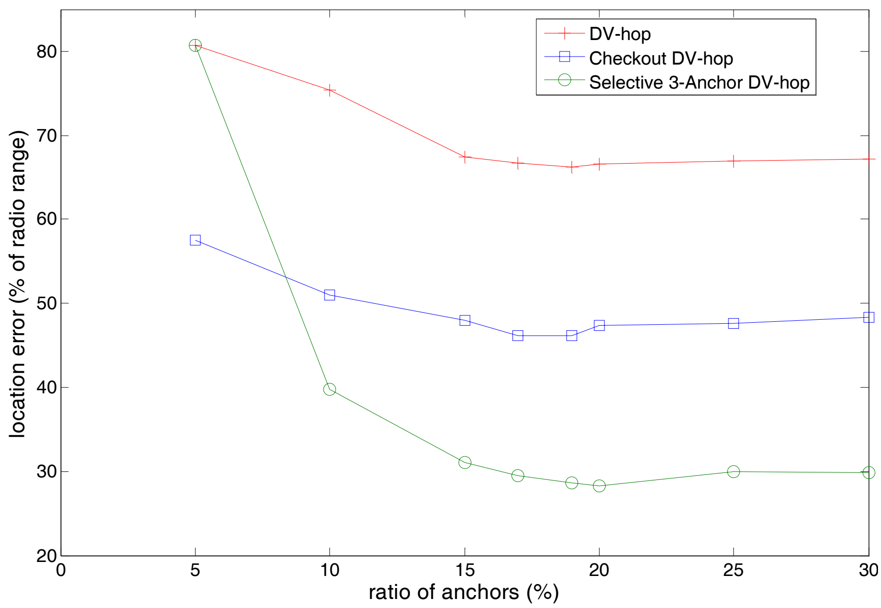


Figure 4-15. Comparison of Location Error in Static Scenarios (Range of 20m)

Figure 4-15 indicates that, in general, the localization accuracy of Checkout DV-hop is about 25% better than that of DV-hop. When the anchor ratio is larger than 5%, Selective 3-Anchor DV-hop has better accuracy. The improvement is about 30% when considering Checkout DV-hop and about 55% compared to DV-hop.

It should be mentioned that, when the ratio of anchors is as low as 5%, many normal nodes will have the same connectivity. Thus, Selective 3-Anchor DV-hop algorithm can't identify the unique solution. It then temporarily utilizes DV-hop algorithm. That is why in Figure 4-15 Selective 3-Anchor DV-hop and the original DV-hop both start from the same point.

In order to investigate the radio range's influence on accuracy, we change the node radio range from 20 meters to 15 meters. Meanwhile, all other scenario parameters remain the same. Then, as shown in Figure 4-16, we obtain the WSNet simulation results of the three algorithms under static scenarios with the radio range set to 15 meters.

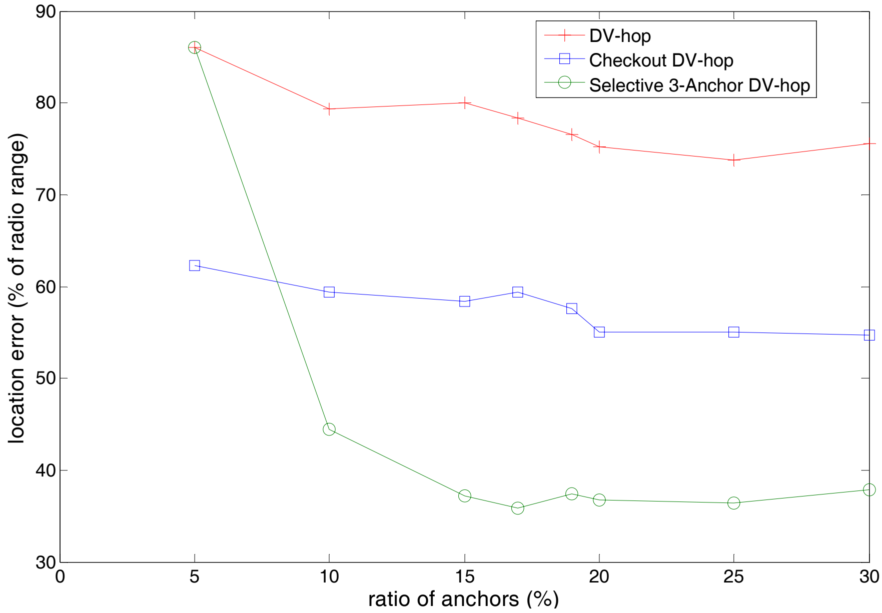


Figure 4-16. Comparison of Location Error in Static Scenarios (Range of 15m)

Figure 4-16 shows that, in general, the accuracy of Selective 3-Anchor DV-hop is 25% better than Checkout DV-hop's and about 50% better than the original DV-hop algorithm. Comparing Figure 4-15 and 4-16, the accuracy improvement is similar when the radio range passes from 20m to 15m. The reason can be: when the radio range decreases, there are fewer neighbor nodes around each normal node, thus less connectivity information can be obtained; but at the same time, there are fewer collisions in the network.

4.2.4.2 Comparison in Synchronized Mobile Scenarios

The scenarios here are the same as those in the section 4.2.3.4.2. The number of transmitted frames during the execution of the 3 algorithms remains the same as described in Figure 4-14. The simulation results in terms of location error are presented in Figure 4-17.

Figure 4-17 presents the relationship between accuracy and anchor ratio for DV-hop, Checkout DV-hop and Selective 3-Achor DV-hop in the synchronized mobile scenarios. The accuracy improvement of Checkout DV-hop over DV-hop is between 20% and 25%. When the number of anchors is larger than 5, the improvement of Selective 3-Anchor DV-hop over Checkout DV-hop ranges from 18% to 32%, and is between 37%, and 48% compared to DV-Hop.

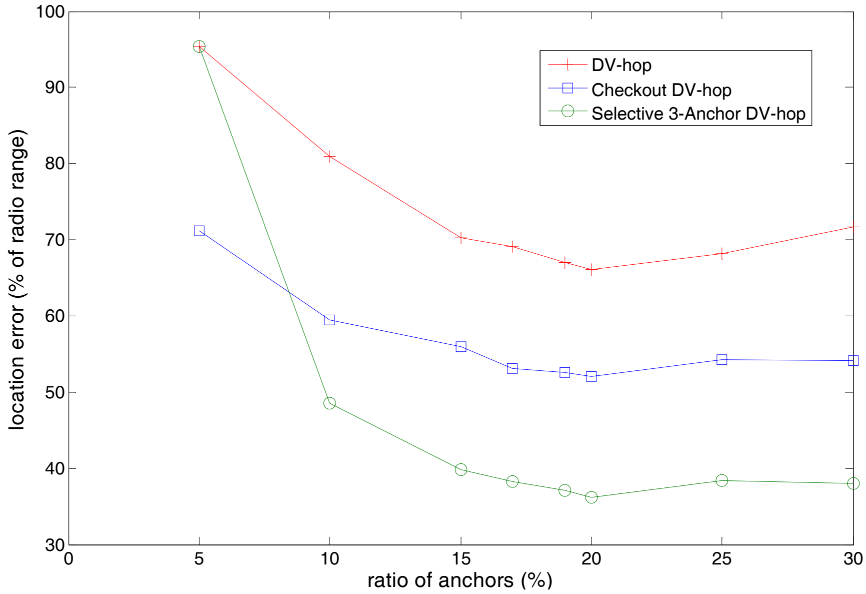


Figure 4-17. Comparison of Location Error in Synchronized Mobile Scenarios (Range of 20m)

In order to investigate the accuracy with a different radio range, we reduce the radio range to 15 meters. The other parameters remain the same. The results are shown in Figure 4-18. Comparing Figure 4-17 and Figure 4-18, we can find that the accuracy is not affected by the change in the radio range. Selective 3-Anchor DV-hop's accuracy is about 20% better than Checkout DV-hop algorithm and about 50% better than the original DV-hop algorithm.

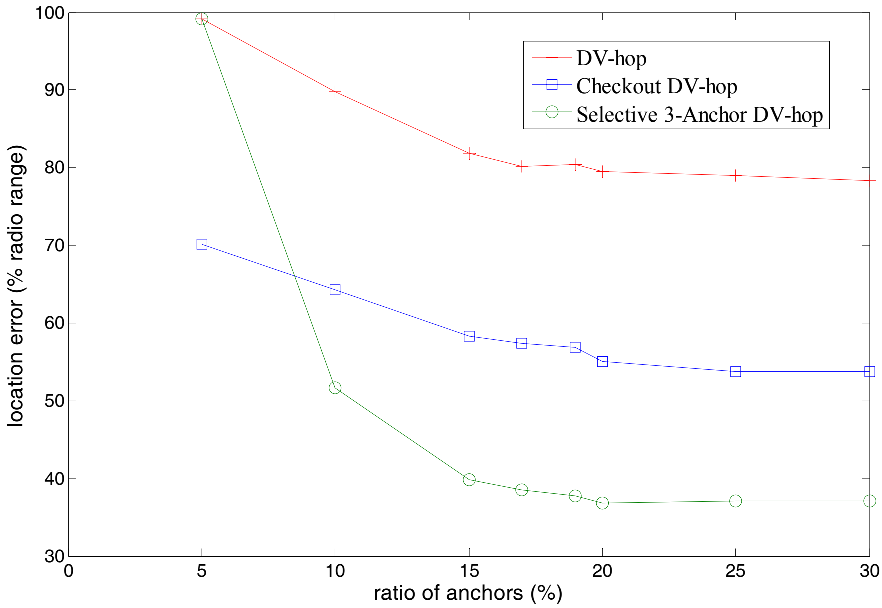


Figure 4-18. Comparison in Synchronized Mobile Scenarios (Range of 15m)

4.2.4.3 Comparison in Unsynchronized Mobile Scenarios

The scenarios of this section are the same as those in Section 4.2.3.4.3. The number of transmitted frames when executing the 3 algorithms remains the same as illustrated by Figure 4-14. The simulation results in terms of location error are presented in Figure 4-19.

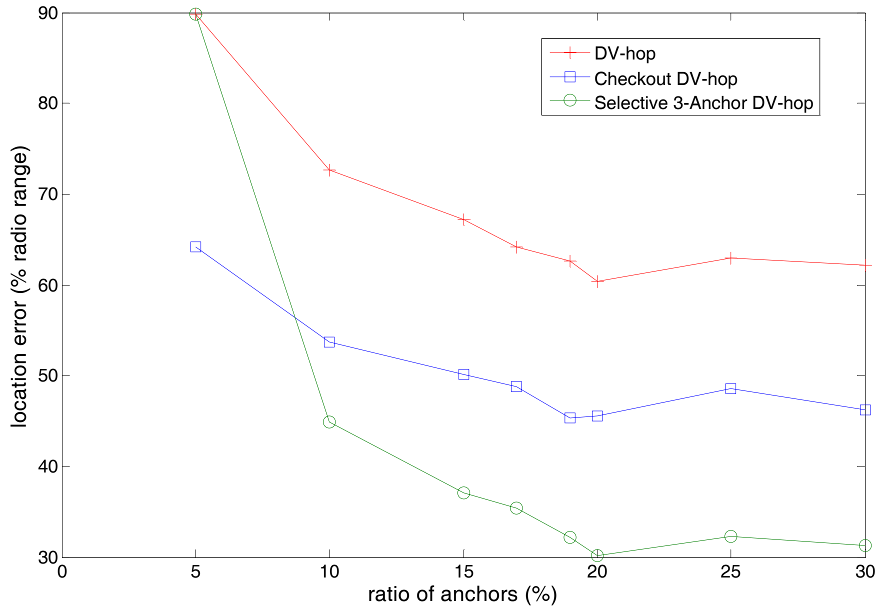


Figure 4-19. Comparison of Location Error in Unsynchronized Mobile Scenarios (Range of 20m)

Figure 4-19 shows that the accuracy improves by 10% to 20% when using Checkout DV-hop instead of DV-hop. When the number of anchors is larger than 5, the improvement of Selective 3-Anchor DV-hop over Checkout DV-hop is between 20% and 34%, and when compared to DV-hop, it is between 32% and 45%.

We also change the radio range from 20 meters to 15 meters, while all other scenario parameters remain the same. The simulation results for the three algorithms under unsynchronized mobile scenarios with the radio range set to 15 meters are shown in Figure 4-20.

Figure 4-20 indicates that, in general, the accuracy of Selective 3-Anchor DV-hop is 30% better than Checkout DV-hop and about 45% better than the original DV-hop algorithm. We can conclude that the change in radio range had almost no impact on the performance.

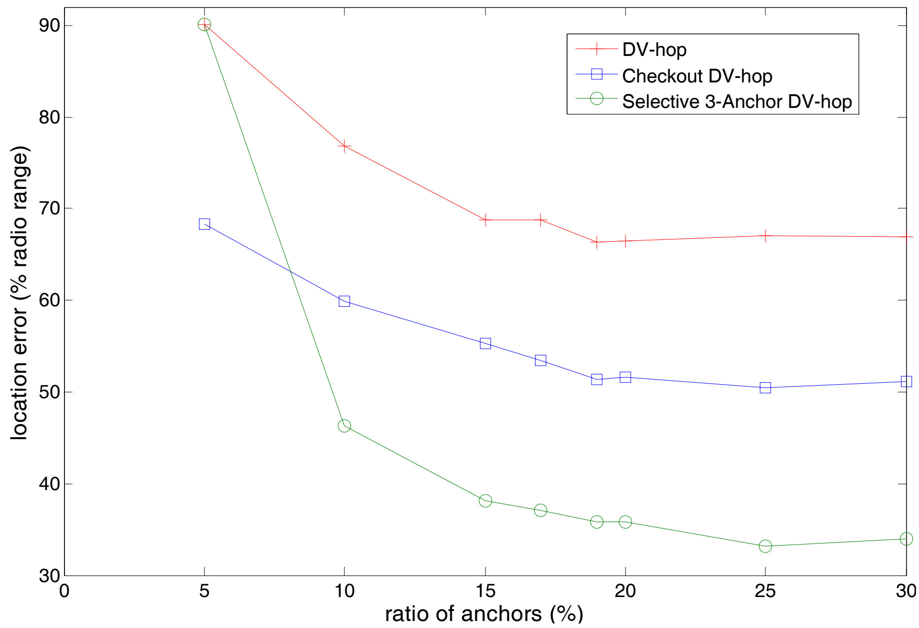


Figure 4-20. Comparison in Asynchronized Mobile Scenarios (Range of 15m)

4.2.4.4 Summary of Evaluation

As the end of this section 4.2, we give a resume of comparisons on the three algorithms, the original DV-hop, the Checkout DV-hop and the Selective 3-Anchor DV-hop.

The original DV-hop and the Checkout DV-hop have the same requirement on the minimum number of anchors. They need at least 3 anchors in any network. But the Selective 3-Anchor DV-hop generally requires more anchors. As shown in previous sections, Selective 3-Anchor DV-hop algorithm shows the advantage on accuracy when there are at least 10 anchors in a network with 100 nodes in total.

As for accuracy, on average, the original DV-hop's location error is about 70% of the radio range. When using Checkout DV-hop, this error is about 55% of the radio range. Selective 3-Anchor DV-hop is the best choice as the location error drops to 35% on average.

Finally, since these three algorithms share the same communication procedure (Step #1 and Step #2) and only differ on the position calculation (Step #3), they have the same network overhead.

As for calculation time, since the position evaluation is restricted to Step #3, the calculation time doesn't exceed the duration of Step #3. In our simulation, the duration of Step #3 is set as $1/8*t_p$=0.75s. Therefore, all the three algorithms spend no more than 0.75s calculating the position.

The three algorithms report more accurate results in unsynchronized scenarios. That is because in synchronous scenarios, all nodes are configured to start their localization period simultaneously, which leads to more collisions. But in unsynchronized scenarios, anchors generally have fewer chances to broadcast their positions simultaneously. Therefore, synchronization is not a necessary condition for the use of DV-hop based solutions.

The last parameter is mobility. With the speed as low as 0.5m/s, the accuracy of the three algorithms in synchronized mobile scenarios is about 10% lower than in static scenarios. However, the accuracy in unsynchronized mobile scenarios is about 10% better than that in static scenarios. This suggests that, in the context of low-speed mobility, the influence of synchronization becomes more noticeable.

The following table gives a brief comparison on these three algorithms.

Table 4-11. Brief Comparison on the Three Algorithms under All Scenarios

	DV-hop	Checkout DV-hop	Selective 3-Anchor DV-hop
Accuracy	Modest	Better: 20% > DV-hop	Best: 50% > DV-hop
	scenarios: unsynchronized mobile > static > synchronized mobile		
Calculation Complexity	O(m_d)	O(m_d)	O(m_d^3)
Network Overhead	share the same network overhead		

Note: in this table, ">" means "better accuracy than"

4.3 An Adaptive Range-free Protocol: Combination of Class-1 Protocol and DV-hop Protocol

In the previous section, we focused on DV-hop protocol which is very useful to localize class-2 normal nodes (with less than 3 neighbor anchors). However, we also note a disadvantage of DV-hop protocol: it has high network overhead when there are many anchors.

As discussed in the section 4.2.3.3, when the ratio of anchors is high (for example, more than 40%), instead of using DV-hop algorithm to localize normal nodes, we recommend class-1 algorithms such as Centroid, CPE, and Mid-perpendicuar. Thus, we need to design the corresponding low-overhead protocol for these algorithms. This protocol is called “Class-1 protocol”.

In the following, we first present the principle of Class-1 protocol, and then integrate it with DV-hop protocol to provide an adaptive range-free protocol.

4.3.1 Class-1 Localization Protocol

Class-1 localization protocol can implement those class-1 algorithms such as Centroid, CPE and Mid-perpendicular. This protocol is supposed to function in case of high ratio of anchors. In this case, we don’t suggest that all anchors periodically broadcast their positions, because this leads to much more network overhead. It is better for the anchors to broadcast positions only when normal nodes ask them to do it.

Thus, the basic principle of Class-1 protocol is as follows, including 3 steps. First, when a normal node N_x needs to calculate its position, it broadcasts a localization request to its neighborhood. Second, if a neighbor anchor of N_x detects this request, this anchor sends its position to N_x. Finally, if collecting at least 3 neighbor anchors’ positions during a certain period, N_x can calculate its position by Centroid, CPE or Mid-perpendicular.

In the following, the protocol will be explained in details. At Step #1, the normal node N_x broadcasts to its neighborhood a localization request frame, denoted as *frame_req*. When broadcasting *frame_req*, our E-CSMA/CA method should be used to reduce collisions, because several normal nodes may be simultaneously ready to send their request frames. Here, the additional random waiting time in E-CSMA/CA method is denoted as t_{wlr}. Considering this request frame is broadcasted, no ACK signal is required.

frame_req conforms to the command frame format in IEEE standard 802.15.4-2009. Shown in Table 4-12, *frame_req* has 3 parts: MHR, MAC payload, and MFR. Here, since *frame_req* is broadcasted by N_x, the source address in MHR must be the address of N_x, and the destination address is *0xFFFF*. The MAC payload only has an 8-bit field “Command Type”. Its value is set to be 04. According to the IEEE standard, this value means *frame_req* is used to request data (positions of anchors).

Table 4-12. Format of *frame_req*

MHR				MAC Payload	MFR
Frame Control (16 bits)	Sequence Number (8 bits)	Destination Address (*0x*ffff , 16 bits)	Source Address (16 bits)	Command Type (*0x*04, 8 bits)	FCS (16 bits)

At Step #2, the anchors who have received N_x‘s *frame_req* should send their positions to N_x. These anchors are N_x‘s neighbor anchors. The number of neighbor anchors is at least 3, maybe even bigger such as 7 or 8, depending on the specific scenario. If all these neighbor anchors demand each normal node N_x to send back ACK signals, then the network overhead will increase a lot. So, at this step, no ACK signal is demanded.

The position frame sent by the anchor A_i to N_x is denoted as $frame\_pos_{i,Nx}$. It conforms to the data frame format in IEEE standard 802.15.4-2009. Shown in Table 4-13, $frame\_pos_{i,Nx}$ has 3 parts: MHR, *data payload*, and MFR. The source address in MHR is the 16-bit short MAC address of A_i, while the destination address is that of N_x. The *data payload* comprises the coordinate of A_i, which is represented by "x_i" and "y_i".

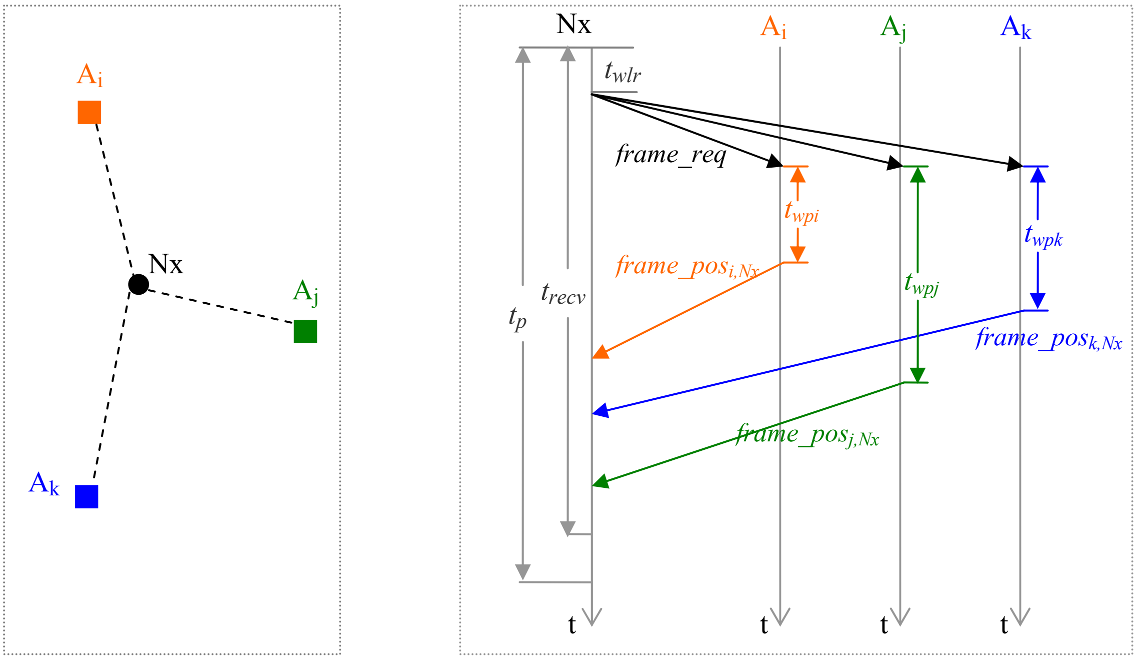
Table 4-13. Format of *frame_ pos$_{i,Nx}$*

MHR				*Data Payload*		MFR
Frame Control (16 bits)	Sequence Number (8 bits)	Destination Address (16 bits)	Source Address (16 bits)	*xi* (32bits)	*yi* (32 bits)	FCS (16 bits)

For transmitting this data frame *frame_pos$_{i,Nx}$*, instead of non-slotted CSMA/CA, our E-CSMA/CA method is recommended to reduce frame collisions. Because N_x may have quite a few neighbor anchors (for example, as many as 6), all these anchors receive N_x's localization request at the same time. If they perform non-slotted CSMA/CA before sending position frames, since the random waiting time is very small, two anchors may simultaneously send out frames, resulting in a collision. Here, we can still use our E-CSMA/CA method, which has an additional random waiting time, denoted as t_{wpi}.

Before Step #3, assume that N_x has received *m* anchors' position frames during a certain period t_{recv}. In fact, t_{recv} is also the duration of Step #1 and Step #2, because N_x is always collecting anchors' positions after it sending the request.

At Step #3, N_x calculates its estimated position by class-1 algorithm such as Mid-perpendicular. An example of the three steps for class-1 protocol is shown in Figure 4-21(b).



(a) Network Topology (b) Procedure of Class-1 Protocol

Figure 4-21. Example of Procedure of Class-1 Protocol

Figure 4-21(a) gives an example of network topology. The normal node N_x has there neighbor anchors, A_i, A_j, and A_k. Shown in Figure 4-21(b), N_x collectes the position frames from the anchors during the period t_{recv}. The entire duration of the three steps is denoted as t_{p}.

4.3.2 Combination of Class-1 Protocol and DV-hop Protocol

Class-1 protocol and DV-hop protocol both have advantages and disadvantages. Class-1 protocol is very simple, but it requires normal node has at least 3 neighbor anchors. DV-hop protocol can serve in the case of low ratio of anchors, but it has considerable network overhead. In order to take advantage of the two protocols, the combination of Class-1 protocol and DV-hop protocol is regarded as our adaptive range-free localization protocol.

In this adaptive protocol, the choice between Class-1 protocol and DV-hop protocol can be decided by each normal node or by the network administrator. If it is decided by each normal node, the corresponding protocol is chosen according to the number of neighbor anchors. That means, a normal node will choose Class-1 protocol when it has at least 3 neighbor anchors; otherwise, it will choose DV-hop protocol. However, this method has a practical problem, considering the different communication manners between Class-1 protcol and DV-hop protocol. In Class-1 protocol, anchors are in passive mode: They wait for requests from normal nodes, if receive the requests, then just broadcast position frames to neighbor nodes. However, in DV-hop protocol, anchors are in active mode: they don't need to listen to request, and they should broadcast their positions related information throughout the network. Suppose that in a network, anchors stay in passive mode by default, and most normal nodes use Class-1 protocol, while only one normal node needs to use DV-hop protocol. The problem is how can the particular one normal node informs all anchors to change from passive mode into active mode. The solution can be: this normal node broadcasts a special request frame throughout the network; when receiving this special frame, anchors need to begin the process of DV-hop protocol. We can note that, the broadcast of this special frame increases the network overhead.

Thus, we suggest that the choice between Class-1 protocol and DV-hop protocol is decided by the network administrator. So, when the choice is made, the network will use one protocol, Class-1 or DV-hop.

From the evaluation results in the section 4.2, we have noticed that, the network overhead of DV-hop protocol increases with the ratio of anchors. In fact, in case of high ratio of anchors, considering the overhead, Class-1 protocol is a better solution than DV-hop protocol. Thus, we need to set a threshold for the ratio of anchors, denoted as RA_{thresh}.

Suppose that in the network, the number of anchors is stable, and the administrator has known the ratio of anchors. Then, if the ratio of anchors is lower than RA_{thresh}, the administrator chooses DV-hop protocol because normal nodes are mostly class-2 nodes. But when the ratio of anchors is higher than RA_{thresh}, in order to avoid a large number of network traffic, Class-1 protocol should be used.

The value of RA_{thresh} is decided by the administrator according to the maximum traffic that the network can accept. A lower RA_{thresh} indicates the network can only accept lower network overhead. But the value of RA_{thresh} cannot be too low; otherwise, many class-2 normal nodes are unable to be localized. In the subsection 4.2.3.3, as an example, we proposed a value for RA_{thresh} which is 40%. After the administrator sets the value for RA_{thresh}, comparing the ratio of anchors with RA_{thresh}, the corresponding protocol can be chosen, which is either Class-1 protocol or DV-hop protocol. The idea is summarized in Figure 4-22.

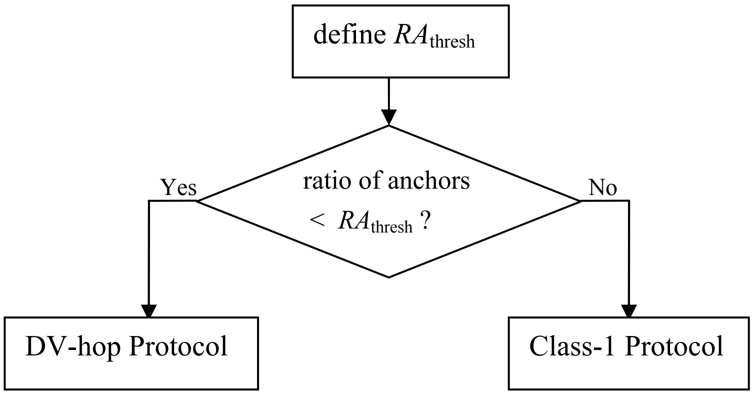


Figure 4-22. Basic Principle of Adaptive Range-free Protocol

4.4 Evaluation of our Range-free Protocol by WSNet

4.4.1 Scenarios

The scenarios for evaluating our range-free protocol are the same as those in the section 4.2 for our DV-hop protocol. Three scenarios are used in the simulation: static scenario (same as in the section 4.2.4.1), synchronized mobile scenario (same as in the section 4.2.4.2, all nodes begin their periods simultaneously), and unsynchronized mobile scenario (same as in the section 4.2.4.3, nodes start their periods at different time).

100 nodes are randomly distributed inside a $100\times100\text{m}^2$ area. The distribution of nodes is shown in Figure 4-9 in the section 4.2.2. Most simulation parameters are the same as those in Table 4-4 in the section 4.2.2.

The particular parameters of Class-1 protocol are listed in Table 4-14. As an example, RA_{thresh} is set to be 40%. That means, when the ratio of anchors is less than 40%, our DV-hop protocol is used. When the ratio of anchors is no less than 40%, Class-1 protocol turns to work.

Before each normal node N_x sends a localization request, it waits for a random duration t_{wpi} which has the maximum value 100ms. Before each anchor A_i sends its position, the additional waiting time t_{wpi} is also randomly selected between 0 and 100ms. The duration of Step #1 and Step #2 is set to be 2.5s, which is assumed to be enough for a normal node to communicate with its neighbor anchors. The duration of one localization period is 3s. That indicates the duration of Step #3 is 3-2.5=0.5s. For each ratio of anchors, the total simulation time is 3000s that equals to 1000 periods.

Table 4-14. Particular Parameters of Class-1 Protocol

RA_{thresh}	40%
Ratio of anchors	40%, 50%, 60%, 70%, 80%, 90%
t_{wlr} (waiting time before normal node sending request)	randomly selected between 0 and 100ms
t_{wpi} (waiting time before the anchor sending position)	randomly selected between 0 and 100ms
t_{recv} (duration of Step #1 and #2)	2.5s
t_{p} (duration of one period)	3s
Network simulation time	3000s (1000 localization periods)

4.4.2 Simulation Results on Accuracy

The simulation results of the three scenarios (static, synchronized mobile, and unsynchronized mobile) are presented from Figure 4-23 to 4-26. The data is collected at each ratio of anchors. Figures 4-23, 4-24, 4-25 show the average location error per node per localization period, expressed as a percentage of the radio range. Figure 4-26 presents the average number of transmitted frames per localization period. In total, six algorithms are compared. Three of them are class-2 algorithms, including DV-hop, Checkout DV-hop and Selective 3-Anchor DV-hop, which are evaluated based on our DV-hop protocol. The other three are class-1 algorithms such as Centroid, CPE and Mid-perpendicular, which function with our Class-1 protocol.

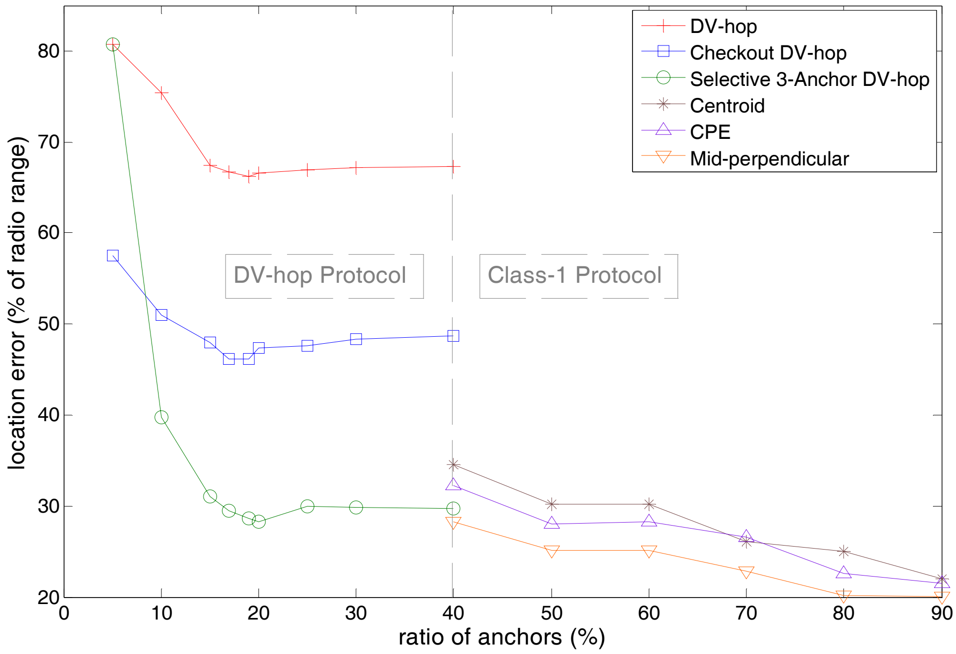


Figure 4-23. Location Error in Static Scenario with Range-free Protocol

The average location errors of the algorithms in static scenario are shown in Figure 4-23. The results of DV-hop based algorithms come from the results we have obtained in the section 4.2.4.1 (shown in Figure 4-15). We can note that, the class-1 algorithms all have much better accuracy than DV-hop and Checkout DV-hop. But Selective 3-Anchor DV-hop algorithm can achieve similar accuracy as those class-1 algorithms. We can also notice that, among the three class-1 algorithms, our Mid-perpendicular has the best accuracy, although the improvement is only about 15% on average.

Figure 4-24 shows the average location error of the algorithms in synchronized mobile scenario. The results of DV-hop based algorithms are obtained from the section 4.2.4.2 as shown in Figure 4-17. Comparing Figure 4-23 and Figure 4-24, we can note that, when nodes change from static to mobile, the accuracy of class-1 algorithms decreases a little. However, the accuracy of class-2 algorithms (DV-hop based algorithms) has a larger decrease.

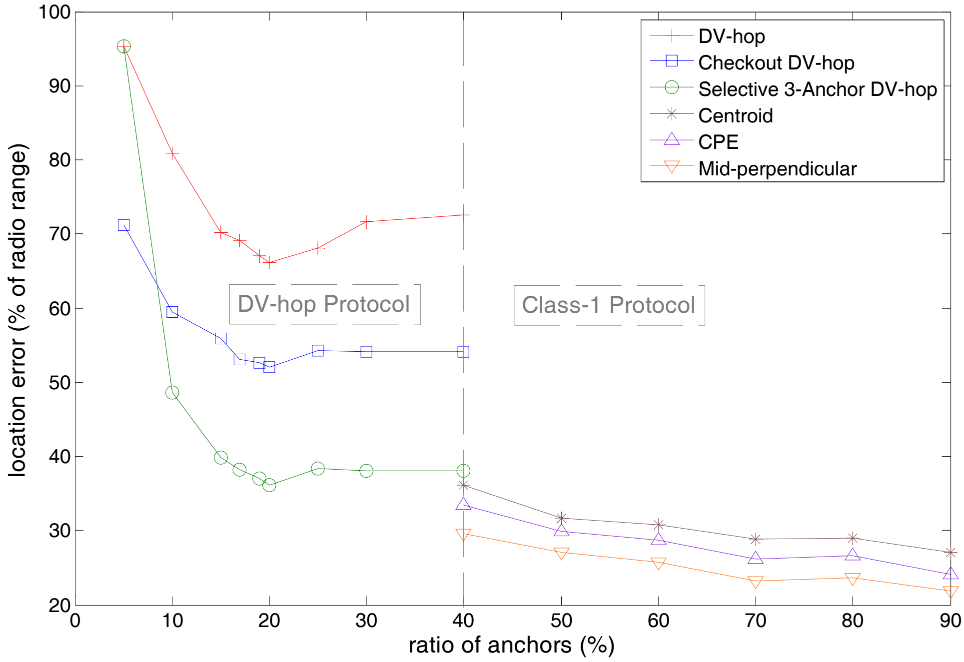


Figure 4-24. Location Error in Synchronized Mobile Scenario with Range-free Protocol

This small decrease of class-1 algorithms is contributed by the small localization period of Class-1 protocol, which is only 3s. Considering the speed of nodes movement is 0.5m/s, normal nodes move only 0.5×3=1.5m. However, in DV-hop protocol, frames need to have enough time to broadcast through the network, thus the localization period of DV-hop protocol is configured as big as 6s. During this 6s, normal nodes can move as far as 0.5×6=3m. Therefore, node movement has a bigger influence on the accuracy of DV-hop protocol than that of Class-1 protocol.

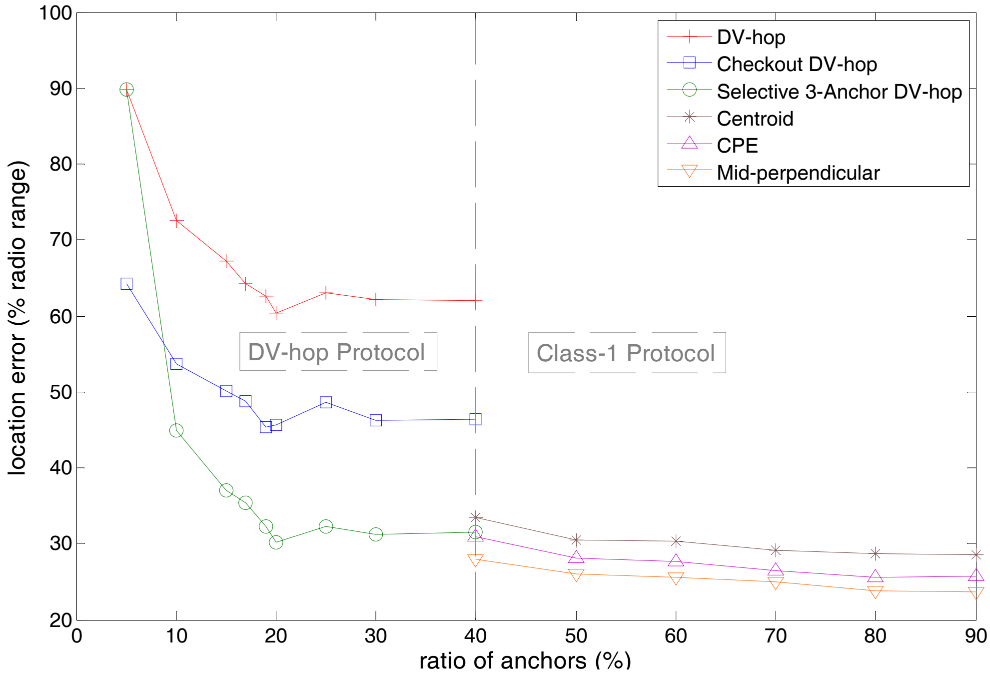


Figure 4-25. Location Error in Unsynchronized Mobile Scenario with Range-free Protocol

Figure 4-25 presents the average location error in unsynchronized mobile scenario. The results of DV-hop based algorithms come from the section 4.2.4.3 as shown in Figure 4-19. Comparing Figure 4-24 with Figure 4-25, we can see the different improvement of DV-hop protocol and Class-1 protocol.

Based on DV-hop protocol, the DV-hop based algorithms have an obvious accuracy improvement in unsynchronized mobile scenario, compared with the accuracy in synchronized mobile scenario. However, the class-1 algorithms using Class-1 protocol only have a slight improvement in unsynchronized mobile scenario. The reason is as follows. DV-hop protocol has a large number of broadcast traffic, while Class-1 protocol has much fewer traffic. So, the possibility of frame collisions in Class-1 protocol is much less than that in DV-hop protocol. Thus, as for Class-1 protocol, the few collisions can be effectively reduced by our E-CSMA/CA method, resulting in no big change between synchronized scenario and unsynchronized scenario. However, considering the massive traffic in DV-hop protocol, our E-CSMA/CA may be not enough to avoid collisions. In unsynchronized scenario, nodes (especially the anchors) begin their period at different time, which helps to further reduce frame collisions. Therefore, as for DV-hop protocol, the accuracy in unsynchronized scenario is obviously better than synchronized scenario.

4.4.3 Simulation Results on Network Overhead

The above analysis focuses on the localization accuracy of algorithms. In the following, the network overhead is discussed. The DV-hop based algorithms all use the same protocol that is our DV-hop protocol, thus they have the same network overhead. The class-1 algorithms using our Class-1 protocol also have the same network overhead. Therefore, we need to compare the network overhead of DV-hop protocol and that of Class-1 protocol.

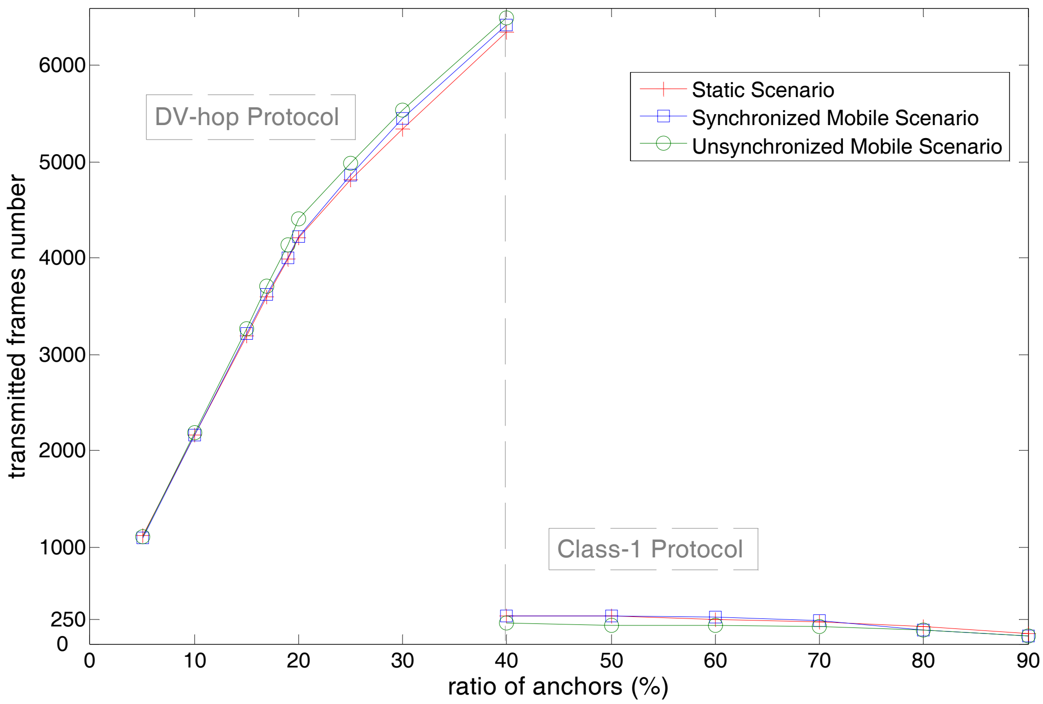


Figure 4-26. Number Transmitted Frames for Range-free Protocols

The network overhead of protocol is quantized by the average number of transmitted frames by all 100 nodes per localization period. Simulation results are shown in Figure 4-26. From this figure, we can note that, the network overhead of DV-hop protocol is much higher than that of Class-1 protocol.

Now, we estimate the approximate value of number of transmitted frames for each protocol. Here, we cannot give the exact value, because the exact value of number of transmitted frames varies with different network topologies.

In DV-hop protocol, the network traffics exist only at the first two steps. At Step #1, each anchor A_i broadcasts its position frame $frame\_pos_i$ throughout the network. In order to make all nodes be aware of $frame\_pos_i$, every node in the network needs to relay this frame once. Thus, if the total number of nodes is *num,* the number of anchors is $num \times$ "*ratio of anchors*", so the number of transmitted frames at Step #1 of DV-hop protocol is at least $num \times (num \times \text{"}ratio\ of\ anchors\text{"})$ = $num^2 \times \text{"}ratio\ of\ anchors\text{"}$. The same result can be obtained for Step #2. Thus, as for all the three scenarios, the approximate value of number of transmitted frames for DV-hop protocol is $2 \times num^2 \times \text{"}ratio\ of\ anchors\text{"}$. To verify this, for example, in Figure 4-26, when the ratio of anchors is 5%, the number of transmitted frames is about 1000. This value is just $2 \times 100^2 \times 5\%$, considering the total number of nodes is 100.

In Class-1 protocol, the network overhead also exists only at the first two steps. At Step #1, each normal node broadcasts its localization request just to its neighbor nodes. Thus, the number of transmitted frames at Step #1 is exactly $num \times (1-$"*ratio of anchors*"$)$, which is also the number of normal nodes. At Step #2, the neighbor anchors of each normal node respond the request by sending back their positions. Thus, if on average there are *m* neighbor anchors for each normal node, then the approximate value of number of transmitted frames for Class-1 protocol is $num \times (1 - \text{"}ratio\ of\ anchors\text{"}) \times m$. To verify this result, for example, in Figure 4-26, when the ratio of anchors is 40%, the number of transmitted frames is about 250, which is nearly $100 \times (1\text{-}40\%) \times 4 = 240$, where *m* is assumed to be 4. Considering *m* is usually a small value (at least 3) depending on network topology and the ratio of anchors, the number of transmitted frames of Class-1 protocol is much less than that of DV-hop protocol.

4.5 Brief Summary of Chapter 4

When we implement the typical range-free algorithms in network scenarios, some problems such as frame collisions, node mobility and synchronization, should be taken into consideration. Thus, in this chapter, based on the IEEE standard 802.15.4-2009, we propose two protocols: DV-hop protocol and Class-1 protocol. The combination of these two protocols is our adaptive range-free localization protocol.

In our DV-hop protocol, we design new *data payload* formats, and a new access method E-CSMA/CA to improve the performance of non-slotted CSMA/CA. In addition, several parameters such as timers and maximum number of received anchors are proposed to end each step of DV-hop based algorithms. Our DV-hop protocol can be used to implement the DV-hop based algorithms, including the original DV-hop algorithm, our Checkout DV-hop algorithm and our Selective 3-Anchor DV-hop algorithm.

In our Class-1 protocol, normal nodes broadcast their localization request to neighbor nodes, and then their neighbor anchors respond by sending back anchors' positions. In the protocol, our E-CSMA/CA method is also used to reduce frame collisions. Our Class-1 protocol can be used to implement the class-1 algorithms including Centroid, CPE, and Mid-perpendicular.

The DV-hop protocol has much higher overhead than Class-1 protocol. The overhead can be quantized by the metric “number of transmitted frames”. The approximate value of number of transmitted frames for DV-hop protocol is $2 \times num^2 \times "ratio\ of\ anchors"$, while that for Class-1 protocol is $num \times (1 - "ratio\ of\ anchors") \times m$, where m is the average number of neighbor anchors for each normal node, and *num* is the total number of nodes. So, given the ratio of anchors, the network administrator can estimate the network overhead for both protocols.

Thus, the maximum acceptable network overhead has its corresponding maximum ratio of anchors, which is defined as the threshold of ratio of anchors “RA_{thres}”. When the ratio of anchors is lower than RA_{thresh}, DV-hop protocol needs to be used; but when the ratio of anchors is higher than RA_{thresh}, in order to avoid a large number of network traffic, Class-1 protocol should be used. This is the basic principle of our range-free protocol.

Based on the corresponding protocols, the accuracy of the related algorithms has been evaluated in network scenarios. Although the improvement by our Mid-perpendicular algorithm and Checkout DV-hop algorithm is not so significant, our Selective 3-Anchor DV-hop algorithm has the accuracy about 35% better than our Checkout DV-hop algorithm and about 50% better than DV-hop algorithm.

Node mobility has a bigger influence on the accuracy of DV-hop based algorithms than that of the class-1 algorithms. The reason is: while Class-1 protocol has the broadcast only to neighbor nodes, DV-hop protocol need more time to broadcast information throughout the network. Thus, the localization period of DV-hop protocol is longer than that of Class-1 protocol. Therefore, moving at the same speed, in DV-hop protocol, normal nodes move away further during one period than in Class-1 protocol.

Synchronization also has an important influence on DV-hop protocol. Compared with synchronized mobile scenario, DV-hop protocol has an obvious accuracy improvement in unsynchronized mobile scenario. However, Class-1 protocol only has a slight improvement in unsynchronized scenario. This reveals that our E-CSMA/CA method is already qualified for Class-1 protocol, but not sufficient for DV-hop protocol. After all, synchronization is not a necessary condition for both protocols.

As for calculation time, for both protocols, since the position calculation is restricted to Step #3, the calculation time doesn’t exceed the duration of Step #3. In our simulation, the duration of Step #3 for DV-hop protocol is set to be 0.75s, while that for Class-1 protocol is 0.5s. Therefore, all the related algorithms spend a little time calculating the position.

The following table gives a brief comparison on accuracy and overhead of the protocols.

Table 4-15. Brief Comparison on the Protocols and Algorithms

	adaptive range-free localization protocol	
	Class-1 protocol	DV-hop protocol
Accuracy	Mid-perpendicular > CPE > Centroid >	Selective 3-Anchor DV-hop > Checkout DV-hop > DV-hop
	scenarios: unsynchronized mobile > static > synchronized mobile	
Network Overhead	$num \times (1 - "ratio\ of\ anchors") \times m$	$2 \times num^2 \times "ratio\ of\ anchors"$

Note: in this table, “>” means “better accuracy than”

5. Conclusions and Perspectives

5.1 Conclusions

Wireless sensor networks (WSN) have attracted worldwide research and industrial interest. They are typically composed of resource-constrained sensor nodes which can communicate with each other and cooperatively collect information from the environment. Among the available standards, the IEEE standard 802.15.4 is expected to provide low cost and low power connectivity for sensor network equipments. This standard has drawn great interests by lots of wireless sensor network applications, such as hospital surveillance, smart home, and object tracking. Because the sensor nodes need to operate in low power, so that their batteries can last as long as several months even to several years.

Localization has been a fundamental issue for many wireless sensor network applications. For example, in hospital surveillance, the knowledge of where the patient is can help the doctors arrive as quickly as possible in urgent case. In this kind of networks, compared with the topologies like star and tree, ad-hoc topology has its advantages on reliability, flexibility, and scalability.

The localization solutions for WSN can be generally categorized into two categories: range-based and range-free. Their major difference lies in whether ranging information are required. The range-based localization depends on accurate ranging results among sensor nodes. These ranging results include point-to-point distance, angle, or velocity relative measurements. Instead of the range information, the range-free localization uses connectivity information between nodes. In this scheme, the nodes that are aware of their positions are called anchors, while others are called normal nodes. Anchors are fixed, while normal nodes are usually mobile. To estimate their positions, normal nodes first gather the connectivity information as well as the positions of anchors, and then calculate their own positions. Compared with range-based schemes, the range-free schemes are more cost-effective, because no additional ranging devices are needed. As a result, we focus our research on the range-free schemes in this thesis.

During these years, many range-free localization algorithms have been proposed. Among them, Centroid, CPE (Convex Position Estimation), and DV-hop (Distance Vector-Hop) are well known algorithms. Centroid and CPE algorithms require a normal node has at least three neighbor anchors, while DV-hop algorithm doesn't have this requirement. However, these localization algorithms are not accurate enough, and they are usually studied without network context. Thus, we are interested in the investigation with wireless network context by implementing and improving new localization algorithms. The main contributions of this thesis are listed in the following.

- In order to permit each normal node to choose its suitable localization algorithm, we propose an adaptive mechanism to categorize normal nodes into two classes: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes.
- For class-1 normal nodes, *Mid-perpendicular* algorithm is proposed to give a better accuracy than Centroid and CPE algorithms. The proposed algorithm finds a centre point of the overlap communication area of neighbor anchors. When the normal node has only 3 neighbor anchors, the centre of the overlap is calculated as the cross point of mid-perpendiculars between any two anchors. When there are more than 3 neighbor anchors, the proposed algorithm first finds the 3 anchors which contribute the communication overlap of the anchors, and then calculates the centre in the same way as for 3 neighbor anchors.

- For class-2 normal nodes, we propose two algorithms *Checkout DV-hop* and *Selective 3-Anchor DV-hop*. Based on estimated distance between the normal node and its nearest anchor, *Checkout DV-hop* adjusts the result position of DV-hop algorithm. In order to further improve the accuracy of *Checkout DV-hop* algorithm, we provide another new algorithm *Selective 3-Anchor DV-hop*, which can obtain much better accuracy at the cost of higher computation complexity. The basic principle of the latter is as follows. The normal node first selects any three anchors to form a 3-anchor group, then it calculates the candidate positions based on each 3-anchor group, and finally according to the relation between candidate positions and their connectivities, the normal node chooses the best candidate position.
- As class-1 algorithms, Centroid and CPE both have low computation complexity at the level O(m), while *Mid-perpendicular* increases the complexity to O(m^2). As class-2 algorithms, DV-hop and *Checkout DV-hop* both remain at the level O(m_d), but *Selective 3-Anchor DV-hop* algorithm has a complexity as high as O(m_d^3). Here, m is the number of neighbor anchors for a normal node, while m_d is the number of all anchors in the entire network.
- In order to evaluate the accuracy of these algorithms, simulations have been done using MATLAB. On average, the accuracy of Our *Mid-perpendicular* algorithm is 15% better than Centroid and CPE algorithms. Our *Checkout DV-hop* algorithm has an average localization accuracy that is about 15% better than DV-hop algorithm. However, our *Selective 3-Anchor DV-hop* algorithm is 45% better than DV-hop algorithm.
- From the simulation results, we also note that: The distribution of nodes has influence on the accuracy of algorithms. Under different distributions, the performance might change a lot. This performance variance of the algorithms has been observed from the view of confidence level.
- During the verification process of our three new algorithms, we noted that most of the existing algorithms were only studied using tools like MATLAB which neglects the possible problems of a real wireless network context such as frame collision and node synchronization. Therefore, in this thesis, based on the IEEE standard 802.15.4, we propose two protocols: *DV-hop protocol* and *Class-1 protocol*. Then we combine these two protocols into our adaptive range-free localization protocol.
- In our *DV-hop protocol*, we design new data payload formats, and a new access method E-CSMA/CA to improve the performance of non-slotted CSMA/CA. In addition, several parameters such as timers and maximum number of received anchors are proposed to end each step of DV-hop based algorithms. Our *DV-hop protocol* can be used to implement the DV-hop based algorithms, including our *Checkout DV-hop* algorithm and our *Selective 3-Anchor DV-hop* algorithm.
- In our *Class-1 protocol*, normal nodes broadcast their localization request to neighbor nodes, and then their neighbor anchors respond by sending back anchors' positions. In this protocol, our E-CSMA/CA method is also used to reduce frame collisions. Our *Class-1 protocol* can be used to implement the class-1 algorithms including *Mid-perpendicular*.
- The *DV-hop protocol* has much higher overhead than *Class-1 protocol*. The overhead can be quantized by the metric "number of transmitted frames". The approximate value of number of transmitted frames for *DV-hop protocol* is $2 \times num^2 \times "ratio\ of\ anchors"$, while that for *Class-*

1 protocol is $num \times (1 - "ratio\ of\ anchors") \times m$, where *m* is the average number of neighbor anchors for one normal node, and *num* is the total number of nodes (include anchors and normal nodes). So, given the ratio of anchors, the network administrator can estimate the network overhead for both protocols. Thus, the maximum acceptable network overhead has its corresponding maximum ratio of anchors, which is defined as the threshold of ratio of anchors "RA_{thres}". When the ratio of anchors is lower than RA_{thresh}, *DV-hop protocol* needs to be used; but when the ratio of anchors is higher than RA_{thresh}, in order to avoid a large number of network traffic, *Class-1 protocol* should be used. This is the basic principle of our adaptive range-free protocol.

- Our protocols are implemented using the simulator WSNet in the IEEE 802.15.4 wireless network. The comparative network simulation results are presented and analyzed in terms of localization accuracy, overhead, node mobility, and node synchronization. Results show that, globally, our new algorithms have better accuracy than the existing typical range-free algorithms. We can also note that, in term of overhead, the DV-hop based algorithms have much higher network overhead than the class-1 algorithms like Centroid and CPE, because DV-hop based algorithms require the broadcasts throughout the network. In terms of mobility, node mobility can have a bigger influence on the accuracy of DV-hop based algorithms than that of the class-1 algorithms, because DV-hop based algorithms need longer localization period to support the broadcasts through the network. Finally, it should be noted that, node synchronization is not necessary for our algorithms and protocols.

5.2 Perspectives

In the future, we want to continue our work from the following aspects.

We will investigate the performance of class-1 algorithms in real radio propagation scenarios. The class-1 algorithms, such as Centroid, CPE and *Mid-perpendicular*, have a common assumption: the radio range of nodes is identical and spherical. However, in reality, the radio propagation is irregular, especially for indoor scenarios. The range of nodes varies with the factors such as environment, antenna of node and battery of node. Thus, we need to take the radio variation into consideration and to improve the localization algorithms.

We will try to find the best values for some parameters in our algorithms and protocols. In our proposals, we have defined some parameters, such as the additional random waiting time in our E-CSMA/CA method, the maximum number of received anchors to end each step of DV-hop, the localization period of our protocols, and the threshold of ratio of anchors in our adaptive range-free protocol. We have set some available values for these parameters in the simulations in this thesis. But these values are not certainly the best for the parameters. Thus, in the future simulations, we would like to find those best values.

We will make sure our adaptive range-free localization protocol can work without network administrator. In this thesis, at the beginning of localization period, the network administrator chooses Class-1 protocol or DV-hop protocol for all normal nodes. This mechanism has low network overhead. But it is not flexible, because the network can change with node movement, node failure, and new node entry. Thus, we will find a new mechanism to let each node choose its proper protocol, without much increase of network overhead.

We will be interested in the combination of range-based and rang-free algorithms. Compared with range-free algorithms, the range-based algorithms usually have much better accuracy. However, the range-based algorithms still require a normal node has at least 3 anchors at range. On the contrary, DV-hop algorithm doesn't have this requirement. Thus, when we want to localize a class-2 normal node with high accuracy, we can combine DV-hop algorithm with range-based algorithms. For example, the distances between neighbor nodes can be measured precisely by range-based algorithms, and then these precise distance values can be used to replace the estimated distances in DV-hop algorithm. Thus, the accuracy of DV-hop algorithm can be improved. In fact, not only improve the accuracy, the combination of DV-hop and range-based algorithms can also improve the stability of DV-hop algorithm. In the thesis, in terms of confidence interval, we have noted that the accuracy of DV-hop algorithm varies a lot when the distribution of nodes changes. This variation will be improved if the precise distances are integrated into DV-hop algorithm.

We are also interested in the implementation of our proposals into prototypes. Though the realization into prototypes, we can finally obtain the performance of our algorithms and protocols in real environment. This can help us to further improve our proposals.

References

[802_11] IEEE 802.11 Standards a/b/g/n/ac. http://grouper.ieee.org/groups/802/11/

[802_15_1] IEEE 802.15.1 Standards 2002, 2005. http://grouper.ieee.org/groups/802/15/

[802_15_4] IEEE 802.15.4 Standards 2006, 2007, 2009. http://grouper.ieee.org/groups/802/15/

[ACG 09] K. ALAGHA, G. CHALHOUB, A. GUITTON, E. LIVOLANT, S. MAHFOUDH, P. MINET, M. MISSON, J. RAHME, T. VAL, A. VAN DEN BOSSCHE, "Cross-layering in an industrial wireless sensor network: case study of OCARI" Journal of Networks : JNW Vol. 4 - issue 6, Collection special issue: Wireless Sensor Networks: Theory and Practice, Academy Publisher ISBN 1796-2056 (2009).

[AK 09] I. Amundson, and X. Koutsoukos, "A survey on localization for mobile wireless sensor networks". In Proceedings of the 2nd international Conference on Mobile Entity Localization and Tracking in GPS-Less Environments, Orlando, FL, USA, Sep. 30, 2009, pp. 235-254.

[ALW 03] K. Alzoubi, X. Li, Y. Wang, et al. "Geometric Spanners for Wireless Ad Hoc Networks". IEEE Transactions on Parallel and Distributed Systems, vol. 14, no. 4, Apr. 2003, pp. 408-421.

[ARH 10] A. Ali, M. Razak, M. Hidayab, et al. "Investigation of indoor WIFI radio signal propagation", IEEE Symposium on Industrial Electronics & Applications, Penang, 2010, pp. 117-119.

[AWP 06] X. An, J. Wang, R.V. Prasad, and I. G. M. M. Niemegeers. "OPT: online person tracking system for context-awareness in wireless personal network". In REALMAN '06: Proceedings of the 2nd International Workshop on Multi-hop Ad Hoc Networks: from Theory to Reality, pages 47-54, New York, USA, 2006. ACM.

[BCS 98] S. Basagni, I. Chlamtac, V. Syrotiuk, et al. "A distance routing effect algorithm for mobility (DREAM)", In Proceedings of the 4th Annual ACM/IEEE international Conference on Mobile Computing and Networking (MobiCom '98), Dallas, Texas, USA, Oct. 25-30, 1998, pp. 76-84.

[BHE 00] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low-Cost Outdoor Localization for Very Small Devices, IEEE Personal Communications, 7(5) (2000)28-34.

[BFA 05] F. Benbadis, T. Friedman, M. de Amorim, S. Fdida, "GPS-free positioning system for wireless sensor networks". In Proceedings of the 2nd IFIP International Conference on Wireless and Optical Communications Networks (WOCN '05), Mar. 2005, pp. 541-545.

[BP 00] P. Bahl, V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system". In Proceedings of the 9th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '00), Tel-Aviv, Israel, March 2000, pp. 775-784.

[BT 05] J. Bachrach, and C. Taylor, "Chapter 9: Localization in Sensor Networks". In Hand-book of Sensor Networks - Algorithms and Architectures. Edited by: Stojmenovic, I., 2005 John Wiley & Sons.

[CHA 06] V. Chandrasekhar, et al. "Localization in underwater sensor networks: survey and challenges". In Proceedings of the 1st ACM international Workshop on Underwater Networks (WUWNet '06), Los Angeles, CA, USA, Sep. 25, 2006, pp. 33-40.

[CHR 05] Y. Chraibi. Localization in wireless sensor networks. Master's thesis, Royal Institute of Technology, Sweden, November 2005.

[DGV 11] R. Dalcé, L. Gui, T. Val, A. Van Den Bossche, A. Wei, "Localisation par Méthodes Range-based et Range-free de Stations Mobiles Communicantes dans un Réseau sans Fil", Colloque Francophone sur l'Ingénierie des Protocoles (CFIP 2011), Sainte-Maxime, France, May 2011.

[DPG 01] L. Doherty, K.S.J. Pister, and L.E. Ghaoui. "Convex position estimation in wireless sensor networks", in: Proceedings of IEEE INFOCOM '01, Alaska, USA, Apr.2001, vol. 3, pp. 1655-1663.

[DPS 08] S. Duangsuwan, S. Promwong, N. Sukutamatanti, "Measurement and Modeling of RFID Propagation Channel with in an Indoor Environment", International Conference on Advanced Computer Theory and Engineering, Phuket, pp. 393-397, 2008.

[DVB 11] R. Dalce, T. Val, A. Van den Bossche. “Comparison of Indoor Localization Systems based on Wireless Communications.” Wireless Engineering and Technology, Scientific Research Publishing, Irvine - USA, vol. 2, no. 4, octobre 2011.

[EGH 99] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, “Next century challenges: scalable coordination in sensor networks”. In Proceedings of the 5th Annual ACM/IEEE international Conference on Mobile Computing and Networking (MobiCom ‘99), Seattle, Washington, USA, Aug. 15-19, 1999, pp. 263-270.

[ELM 04] Elnahrawy, E., Li, X., Martin, R. P. “The limits of localization using signal strength: a comparative study”. In Proceedings of 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 04), Santa Clara, CA, USA, Oct. 4-7, 2004, pp. 406-414.

[ERO 12] A.G.F. Elias, J.J.P.C. Rodrigues, L.M.L. Oliveira, B.B. Zarpelao, "A Ubiquitous Model for Wireless Sensor Networks Monitoring", Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp: 835-839, 2012.

[FAL 09] Kevin Fall et al. “Chapter 18: Radio propagation models”. The NS manual, May 2009. Available at http://www.isi.edu/nsnam/ns/doc/index.html.

[FG 02] R. Fontana, S. Gunderson, “Ultra-wideband precision asset location system”. In Proceedings of the 2002 IEEE Conference on Ultra Wideband Systems and Technologies (ICUWB ’02), Baltimore, Maryland, USA, May 20-23, pp. 147-150.

[FRA 08] Jackson FRANCOMME, "Propositions pour un protocole déterministe de contrôle d'accès et de routage avec économie d'énergie dans les réseaux ZigBee", Thèse soutenue le 24 Juin 2008.

[GAR 07] V. Garg, “Wireless Communications & Networking”, Publisher: Morgan Kaufmann, USA, June 2007.

[GKS 88] D.R. Guarino, R.P. Kruger, S. Sayre, et al. "DARPA sensor national testbed: hardware and software architecture", 2nd Symposium on the Frontiers of Massively Parallel Computation, pp:295-301, 1988.

[GKW 02] D. Ganesan, B. Krishnamachari, A. Woo, et al. “Complex behavior at scale: An experimental study of low-power wireless sensor networks”. Technical Report UCLA/CSD-TR 02-0013, UCLA Computer Science, 2002.

[GLN 09] Y. Gu, A. Lo, I. Niemegeers, "A survey of indoor positioning systems for wireless personal networks," IEEE Communications Surveys & Tutorials, vol.11, no.1, pp.13-32, First Quarter 2009.

[GVW 11] L. Gui, T. Val, A. Wei, “A Novel Two-Class Localization Algorithm in Wireless Sensor Networks”, Network Protocols and Algorithms, vol. 3, no. 3 (2011).

[GVW 12] L. Gui, T. Val, A. Wei, R. Dalcé, “Improvement of Range-free Localization Systems Realized by a DV-hop Protocol in Wireless Sensor Networks”, Ad Hoc & Sensor Wireless Networks, in review, submitted on 5 July 2012.

[GVW 13] L. Gui, T. Val, A. Wei, “A New Adaptive Range-free Localization Protocol for Wireless Sensor Networks”, based on Chaptre 4 of this thesis, to be submitted before March 2013.

[GWV 10] L. Gui, A. Wei, T. Val, “A two-level range-free localization algorithm for wireless sensor networks,” IEEE Conference on Wireless Communications Networking and Mobile Computing, pp. 1-4, Chengdu, September 2010.

[GWV 11] L. Gui, A. WEI, T. VAL, “Improving Localization Accuracy Using Selective 3-Anchor DV-hop Algorithm”, IEEE Vehicular Technology Conference (VTC 2011-fall), pp.1-5, San Francisco, September 2011.

[GWV 12] L. Gui, A. Wei, T. Val, “A Range-Free Localization Protocol for Wireless Sensor Networks”, International Symposium on Wireless Communications Systems (ISWCS 2012), pp. 496-500, Paris, August 2012.

[HB 01] J. Hightower and G. Borriello. “A survey and taxonomy of location systems for ubiquitous computing”. Technical Report UW-CSE, University of Washington, Computer Science and Engineering, Seattle, WA, USA, August 2001.

[HCG 08] E. Hamida, G. Chelius, J. Gorce, "On the Complexity of an Accurate and Precise Performance Evaluation of Wireless Networks using Simulations", In proceedings of the 11th ACM-IEEE International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM 2008), October 2008.

[HHB 03] T. He, C. Huang, B. Blum, et al. “Range-free localization schemes for large scale sensor networks”. In Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom ‘03), San Diego, CA, USA, Sep. 14-19, 2003, pp. 81-95.

[HPJ 03] Y. Hu, A. Perrig, D. Johnson, “Packet leashes: a defense against wormhole attacks in wireless networks”, In the Proceeding of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM ‘03), San Francisco, CA, USA, Mar.30-Apr. 3, 2003, pp. 1976-1986.

[HWL 97] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, “Global Positioning System: Theory and Practice”, Fourth Edition, Springer Wien, New York, 1997.

[HZL 10] S. Hou, X. Zhou, and X. Liu, “A novel DV-hop localization algorithm for asymmetry distributed wireless sensor networks,” 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), vol. 4, pp. 243-248, 2010.

[KAP 96] E. Kaplan, “Understanding GPS: principles and applications”, Artech House, 1996.

[KGK 05] Y. Kim, R. Govindan, B. Karp, et al. “Geographic routing made practical”, In Proceedings of the 2nd Conference on Symposium on Networked Systems Design and Implementation (NSDI ‘05), May 02-04, 2005, Berkeley, CA, USA, pp. 217-230.

[KH 05] L. Kovavisaruch, and K. Ho, “Alternate source and receiver location estimation using TDOA with receiver position uncertainties,” IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '05), Pennsylvania, USA, pp. iv/1065 - iv/1068, March 2005.

[KK 00] B. Karp, H. Kung, “GPSR: greedy perimeter stateless routing for wireless networks”, In Proceedings of the 6th Annual international Conference on Mobile Computing and Networking (MobiCom ‘00), Boston, Massachusetts, USA, Aug. 06-11, 2000, pp. 243-254.

[KKP 99] J. M. Kahn, R. H. Katz and K. S. J. Pister, "Mobile Networking for Smart Dust", ACM/IEEE Intl. Conf. on Mobile Computing and Networking (MobiCom 99), Seattle, WA, August 17-19, 1999.

[KR 06] T. Karalar, and J. Rabaey, “An RF ToF based ranging implementation for sensor Networks”. IEEE International Conference on Communications (ICC ‘06), Istanbul, Turkey, Jun. 2006, pp. 3347-3352.

[KRV 09] P. Kumar, L. Reddy, S. Varma, “Distance measurement and error estimation scheme for RSSI based localization in Wireless Sensor Networks,” Fifth IEEE Conference on Wireless Communication and Sensor Networks (WCSN), Allahabad, India, pp. 1-4, Dec. 2009.

[LC 09] Y. Lee, W. Chung, “Wireless sensor network based wearable smart shirt for ubiquitous health and activity monitoring”, Sensors and Actuators B: Chemical, vol: 140, issue: 2, pp: 390-395, 16 July 2009.

[LCK 10] J. Lee, W. Chung, E. Kim, and I. Hong, “Robust DV-hop algorithm for localization in wireless sensor network”, International Conference on Control Automation and Systems, pp. 2506-2509, 2010.

[LCM 10] G. López, V. Custodio, J. Moreno, "LOBIN: E-Textile and Wireless-Sensor-Network-Based Platform for Healthcare Monitoring in Future Hospital Environments," IEEE Transactions on Information Technology in Biomedicine, vol.14, no.6, pp.1446-1458, Nov. 2010.

[LDG 09] Z. Li, W. Dehaene, G. Gielen, "A 3-tier UWB-based indoor localization system for ultra-low-power sensor networks," IEEE Transactions on Wireless Communications, vol.8, no.6, pp.2813-2818, June 2009.

[LH 05] N. Li , J. Hou, “Localized topology control algorithms for heterogeneous wireless Networks”, IEEE/ACM Transactions on Networking, vol. 13, no.6, Dec. 2005, pp. 1313-1324.

[LLP 06] S. Lanzisera, D. Lin, K. Pister, “RF time of flight ranging for wireless sensor network localization”. International Workshop on Intelligent Solutions in Embedded Systems, Vienna, Austria , Jun. 2006, pp. 1-12.

[LP 05] L. Lazos, R. Poovendran, “SeRLoc: Robust localization for wireless sensor networks”, ACM Transactions on Sensor Networks, vol. 1, no. 1, pp. 73-100, Aug. 2005.

[LS 02] L. Lee, R. Scholtz, “Ranging in a dense multipath environment using an UWB radio link”. IEEE Journal on Selected Areas in Communications, vol. 20, no. 9, Dec. 2002, pp. 1677-1683.

[LSS 07] J. Lee, Y. Su, C. Shen, "A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi", 33rd Annual Conference of the IEEE Industrial Electronics Society, pp: 46-51, 2007.

[LY 07] J. Li, M. Yu, “Sensor coverage in wireless ad hoc sensor networks”, International Journal of Sensor Networks, vol. 2, no. 3/4, pp. 218-229, Jun. 2007.

[LYW 10] Y. Liu, Z. Yang, X. Wang, et al. “Location, Localization, Localizability”. Journal of Computer Science and Technology, vol. 25, no. 2, Mar. 2010, pp. 247-297.

[MA 12] D. Matolak, R. Apaza, "Peer-to-peer VHF propagation path loss in the airport surface area", Integrated Communications, Navigation and Surveillance Conference (ICNS), Herndon 2012, pp. G1-1 - G1-7.

[MDF 00] D. McCrady, L. Doyle, H. Forstrom, et al. “Mobile ranging using low accuracy clocks”. IEEE Transactions on Microwave Theory and Techniques, vol. 48, no. 6, Jun. 2000, pp. 951-958.

[MFA 07] G. Mao, B. Fidan, B. Anderson, “Wireless sensor network localization techniques”. Computer Networks, vol. 51, no. 10, Jul. 2007, pp. 2529-2553.

[MLR 04] D. Moore, J. Leonard, D. Rus, and S. Teller, “Robust distributed network localization with noisy range measurements”. In Proceedings of the 2nd international Conference on Embedded Networked Sensor Systems (SenSys ‘04), Baltimore, MD, USA, Nov. 03-05, 2004. pp. 50-61.

[MZF 08] E. Miluzzo, X. Zheng, K. Fodor, and A. Campbell, “Radio characterization of 802.15.4 and its impact on the design of mobile sensor networks”. In Proceedings of the 5th European Conference on Wireless Sensor Networks (EWSN ‘08), Bologna, Italy, Jan. 30-Feb. 01, 2008, pp. 171-188.

[NN 03] D. Niculescu, and B. Nath, “DV based Positioning in ad hoc networks”, Journal of Telecommunication Systems, 22(1/4)(2003), pp. 267-280.

[NS_2] NS-2 http://www.isi.edu/nsnam/ns/

[OPNET] OPNET http://www.opnet.com/

[OWW 10] R. Ouyang, A. Wong, K. Woo, "GPS Localization Accuracy Improvement by Fusing Terrestrial TOA Measurements", IEEE International Conference on Communications (ICC), pp. 1-5, May 2010.

[PLM 02] K. Pahlavan, X. Li, J. Makela, “Indoor geolocation science and technology”. IEEE Communications Magazine, vol. 40, no. 2, Feb. 2002, pp. 112-118.

[POL 12] H. Park, S. Oh, E. Lee et al. "Selective wakeup discipline for continuous object tracking in grid-based wireless sensor networks", IEEE Wireless Communications and Networking Conference (WCNC), pp: 2179-2184, 2012.

[PSG 12] J. A. Palazon, M. Sepulcre, J. Gozalvez, G. Prieto, "Experimental RSSI-based Localization System using Wireless Sensor Networks", Proceedings of the 17th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'12), September 2012.

[PAT 04] R. Patro, “Localization in wireless sensor network with mobile beacons,” Proceedings of 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, pp. 22-24, Sept. 2004.

[PPG 09] O. Postolache, J. Pereira, P. Girao, "Smart Sensors Network for Air Quality Monitoring Applications," IEEE Transactions on Instrumentation and Measurement, vol.58, no.9, pp.3253-3262, Sept. 2009.

[RAP 01] T. Rappaport, “Wireless Communications: Principles and Practices”. 2nd Edition, Prentice Hall, 2001.

[RMN 11] D. Roth, J. Montavont, T. Noel, "MOBINET: Mobility management across different wireless sensor networks", IEEE Wireless Communications and Networking Conference, Quintana Roo 2011, pp. 351-356.

[RS 06] P. Rong, and M. Sichitiu, “Angle of arrival localization for wireless sensor networks,” Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks, USA, 2006, vol.1, pp.374-382.

[RT 06] F. Reichbach and D. Timmermann, “Indoor Localization with Low Complexity in Wireless Sensor Networks,” IEEE International Conference on Industrial Informatics, Singapore, August 2006, pp. 1018-1023.

[SB 09] A. Sanjeev, B. Boaz, “Computational Complexity: A Modern Approach”. Published by Cambridge, 2009.

[SDT 06] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, “Understanding the causes of packet delivery success and failure in dense wireless sensor networks”. In Proceedings of the 4th international Conference on Embedded Networked Sensor Systems (SenSys ‘06), Boulder, Colorado, USA, Oct. 31 - Nov. 03, 2006, pp. 419-420.

[SK 08] C. Suh, Y. Ko, "Design and implementation of intelligent home control systems based on active sensor networks," IEEE Transactions on Consumer Electronics, vol.54, no.3, pp.1177-1184, August 2008.

[SCH 08] J.P. Sheu, P.C. Chen, and C.S. Hsu. A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement, IEEE Transactions on Mobile Computing, 7(9)(2008), pp. 1110-1123.

[SLH 06] Jang-Ping Sheu; Jian-Ming Li; Chih-Shun Hsu, “A Distributed Location Estimating Algorithm for Wireless Sensor Networks”, IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC’06), vol.1, pp.1-8, June 2006.

[SHS 01] A. Savvides, C. Han, and M. Strivastava, “Dynamic fine-grained localization in Ad-Hoc networks of sensors”. In Proceedings of the 7th Annual international Conference on Mobile Computing and Networking (MobiCom ’01), Rome, Italy, July 2001, pp. 166-179.

[SPS 02] A. Savvides, H. Park, and M. Srivastava, “The bits and flops of the n-hop multilateration primitive for node localization problems”. In Proceedings of the 1st ACM international Workshop on Wireless Sensor Networks and Applications (WSNA ’02), Atlanta, Georgia, USA, Sep. 28, 2002, pp. 112-121.

[SRL 02] C. Savarese, J. Rabaey, and K. Langendoen, “Robust Positioning Algorithms for Distributed Ad-Hoc Wireless Sensor Networks”. In Proceedings of the General Track of the Annual Conference on USENIX Annual Technical Conference, Berkeley, CA, USA, Jun. 10-15, 2002, pp. 317-327.

[SSA 01] J.L. da Silva, J. Shamberger, M.J. Ammer, et al. "Design methodology for PicoRadio networks", Conference and Exhibition on Design, Automation and Test in Europe, pp: 314-323, 2001.

[SSS 01] Spread Spectrum Scene. “An introduction to indoor radio propagation”. http://sssmag.com/indoor.html, June 2001.

[ST 11] A. Silva, G. Tucci, "On the Spectral Characteristics of Ad Hoc Networks and their Mobility Properties", Seventh International Conference on Mobile Ad-hoc and Sensor Networks, Beijing 2011, pp. 215-222.

[SUR 12] N.K. Suryadevara, "Wireless Sensor Network Based Home Monitoring System for Wellness Determination of Elderly", IEEE Sensors Journal, Volume: 12, Issue: 6, Page(s): 1965-1972, 2012.

[TAM 06] A. Terzis, A. Anandarajah, K. Moore, et al. “Slip surface localization in wireless sensor networks for landslide prediction”, In Proceedings of the 5th international Conference on information Processing in Sensor Networks (IPSN ‘06), Nashville, Tennessee, USA, April 19-21, 2006, pp. 109-116.

[TAY 10] A.K. Talukdar, H. Ahmed, R. Yavagal, "Second Edition: Mobile Computing", Published by Tata McGraw-Hill Education, 2010.

[VAL 02] Thierry VAL, "Contribution à l'ingénierie des systèmes de communication sans fil", HDR, UT2, ICARE, Spécialité : Génie Informatique, automatique et traitement du signal, le 11 Juillet 2002.

[VAN 07] Adrien VAN DEN BOSSCHE, "Proposition d'une nouvelle méthode d'accès déterministe pour un réseau personnel sans fil à fortes contraintes temporelles", Thèse soutenue le 6 juillet 2007.

[VCV 08] Thierry VAL, Eric CAMPO, Adrien VAN DEN BOSSCHE, "Technologie ZigBee / 802.15.4 : Protocoles, topologies et domaines d'application", Traité "Réseaux et télécommunications" Vol. TE7508, Collection TEA, Techniques de l'Ingénieur, Mai 2008.

[VH 04] P. Voltz, D. Hernandez, "Maximum likelihood time of arrival estimation for real-time physical location tracking of 802.11a/g mobile stations in indoor environments," Position Location and Navigation Symposium (PLANS), California, USA, pp. 585-591, April 2004.

[VVC 11] Adrien VAN DEN BOSSCHE, Thierry VAL, Eric CAMPO, "Technologie sans fil 802.15.4 - Son héritage protocolaire et ses applications", Traité "Réseaux et télécommunications" Vol. TE7509, Collection TEA, Techniques de l'Ingénieur, Nov 2011.

[WAL 06] G. Werner-Allen, K. Lorincz, et al. "Deploying a wireless sensor network on an active volcano". IEEE Internet Computing, vol. 10, no. 2, pp. 18-25, Mar. 2006.

[WIK 802.11ac] http://en.wikipedia.org/wiki/IEEE_802.11ac

[WIN 06] J. Winters, "Smart antenna techniques and their application to wireless ad hoc Networks". IEEE Wireless Communications, vol. 13, no. 4, Aug. 2006, pp. 77-83.

[WKC 07] Whitehouse, K., Karlof, C., and Culler, D. "A practical evaluation of radio signal strength for ranging-based localization". SIGMOBILE Mobile Computing and Communications Review, vol. 11, no. 1 Jan. 2007, pp. 41-52.

[WKP 07] Widyawan, M. Klepal, and D. Pesch. "Influence of predicted and measured fingerprint on the accuracy of RSSI-based indoor location systems. In Proceedings of the 4th Workshop on Positioning, Navigation and Communication (WPNC 2007), Hannover, Germany, March 2007.

[WKW 05] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler, "The effects of ranging noise on multihop localization: an empirical study". In Proceedings of the 4th international Symposium on information Processing in Sensor Networks (IPSN '05), Los Angeles, California,USA, Apr. 24-27, 2005, pp. 73-80.

[WS 98] M. Win, R. Scholtz, "Impulse radio: how it works". IEEE Communications Letters, vol. 2, no. 2, Feb. 1998, pp. 36-38.

[WSNET] WSNET. http://wsnet.gforge.inria.fr/

[XHE 01] Y. Xu, J. Heidemann, D. Estrin, "Geography-informed energy conservation for Ad Hoc routing". In Proceedings of the 7th Annual international Conference on Mobile Computing and Networking (MobiCom '01), Rome, Italy, Jul. 16-21, 2001, pp. 70-84.

[XKR 10] Z. Xu, D. Koltsov, A. Richardson, L. Le, M. Begbie, "Design and simulation of a multi-function MEMS sensor for health and usage monitoring," Prognostics and Health Management Conference (PHM '10), pp.1-7, January 2010.

[XRS 10] X. Xu, N. Rao, and S. Sahni, "A computational geometry method for localization using differences of distances". ACM Transactions on Sensor Networks, vol. 6, no. 2, Feb. 2010, pp. 1-25.

[YL 10] Z. Yang, and Y. Liu, "Quality of Trilateration: Confidence-Based Iterative Localization". IEEE Transactions on Parallel and Distributed Systems. vol. 21, no. 5, May 2010, pp. 631-640.

[YYR 06] M. Youssef, A. Youssef, C. Rieger, et al. "PinPoint: an asynchronous time-Based location determination system". In Proceedings of the 4th international Conference on Mobile Systems, Applications and Services (MobiSys '06), Uppsala, Sweden, Jun. 19-22, 2006, pp. 165-176.

[ZAR 04] Cyril ZARADER, "Le protocole sans-fil ZigBee/802.15.4 et ses applications : Les liaisons sans fil dans l'industrie", Revue de l'électricité et de l'électronique (REE), 2004, n10, pp:78-85.

[ZHK 04] G. Zhou, T. He, S. Krishnamurthy, and J. Stankovic, "Impact of radio irregularity on wireless sensor networks". In Proceedings of the 2nd international Conference on Mobile Systems, Applications, and Services (MobiSys '04), Boston, MA, USA, Jun. 06-09, 2004, pp. 125-138.

[ZN 05] P. Zheng and L. Ni. "Smart Phone and Next Generation Mobile Computing". Publisher: Morgan Kaufmann, USA, 2005.

[ZXL 09] Z. Zhang, G. Xu, Y. Li, and S. Huang, "DV-hop based self-adaptive positioning in wireless sensor networks," IEEE Conference on Wireless Communications Networking and Mobile Computing, pp. 1-4, 2009.

Annexes

Annex 1 : Résumé Long en Français

Depuis quelques années, les réseaux de capteurs sans fil sont au centre des activités de recherche de la communauté scientifique, en particulier face aux vastes applications potentielles comme les soins médicaux, les maisons intelligentes, ou la surveillance de l'environnement. Pour ces applications, la localisation des équipements mobiles communicants est une problématique importante.

Les algorithmes de localisation existants peuvent être classés en deux catégories "*range-based*" et "*range-free*".

Le principe de localisation "*range-based*" est de mesurer précisément la distance ou l'angle entre deux nœuds d'un même réseau. Plusieurs technologies permettent cette mesure, comme par exemple : le RSSI (*Received Signal Strength Indicator*), le TOA (*Time of Arrival*), le TDOA (*Time Difference of Arrival*) ou l'AOA (*Angle of Arrival*). Une fois cette mesure effectuée, la position peut alors être obtenue simplement par trilatération ou triangulation. La localisation "*range-based*" a deux inconvénients majeurs. Le premier est lié aux matériels supplémentaires nécessaires pour la mesure. Ces composants matériels de mesure consomment plus d'énergie et augmentent le coût de la solution. L'autre inconvénient repose sur la précision des mesures qui peut varier selon plusieurs paramètres liés à l'environnement du réseau : le taux d'humidité, le bruit électromagnétique, et la propagation multi-chemin ou *multi-path fading* en intérieur en particulier.

La localisation "*range-free*" permet d'éviter grandement ces deux inconvénients. Généralement, les nœuds, fixes ou mobiles, dont on connait la position sont appelés "ancres" ou *anchors*. Les autres nœuds avec une position à déterminer sont appelés "nœuds normaux" ou *normal nodes*. Pour estimer leurs positions, ces nœuds normaux recueillent tout d'abord des informations de connectivité réseaux ainsi que la position des ancres, puis calculent leurs propres positions. Par rapport au principe "*range-based*", la technique "*range-free*" est ainsi plus rentable, parce qu'il n'y a pas besoin de composants matériels supplémentaires pour la mesure et l'évaluation de la distance. Elle peut donc s'adapter à tout type de transmission sans fil. Par conséquence, nous avons focalisé nos travaux de la thèse sur les approches "*range-free*".

Dans la littérature, de nombreux algorithmes de localisation "*range-free*" ont été proposés. Parmi eux, *Centroïde* et *CPE* (*Convex Position Estimation*) nécessitent des nœuds normaux ayant au moins trois ancres voisines à un saut, tandis que *DV-hop* (*Distance Vector-Hop*) n'impose pas cette restriction. Toutefois, les algorithmes "*range-free*" ne sont pas assez précis. De plus, les algorithmes publiés dans la littérature sont généralement étudiés hors contexte réseau sans prendre en compte les aspects protocolaires. Notre objectif est de proposer des algorithmes mais aussi les protocoles associés permettant d'améliorer la précision de localisation de ce type de méthode « *range-free* ».

La suite de ce résumé en français du manuscrit de la thèse est organisée comme suit. La section I donne un aperçu des travaux relatifs au domaine de recherche qui nous concerne. La section II introduit nos propositions de nouveaux algorithmes. La section III présente ensuite les nouveaux protocoles associés que nous proposons. La section IV présente et analyse les résultats de simulations que nous avons effectués pour valider nos propositions. Enfin, nous concluons et présentons nos perspectives de travail.

I. Etat de l'Art

Dans cette section, les travaux de recherche les plus proche de notre problématique sont étudiés et comparés. Certains d'entre eux, tels que *Centroïde* et *CPE*, sont très simples, mais nécessitent que les nœuds normaux disposent au moins de trois ancres voisines. D'autres travaux, comme *DV-hop*, peuvent être utilisés pour tous les nœuds normaux, même ceux qui n'ont pas trois ancres à portée, mais génèrent beaucoup plus de trafic réseau.

Centroïde et *CPE* sont deux algorithmes typiques basées sur les méthodes « *range-free* ». Nous supposons qu'autour du nœud normal N_x, il y a m ancres voisines A_1, A_2... A_m, dont les positions sont respectivement (x_1, y_1) (x_2, y_2). . . (x_m, y_m). Il est aussi supposé que tous les nœuds ont la même portée radio. Cette hypothèse est bien sûr purement théorique mais couramment admise par la communauté scientifique qui contribue sur cet axe de recherche. Le principe de *Centroïde* est comme suit : les ancres diffusent périodiquement leur position ; N_x reçoit alors la position des ancres, et calcule ainsi sa position estimée comme la moyenne des positions des ancres voisines. La position estimée est calculée ainsi : $x_{cen} = (\sum_{i=1}^{m} x_i)/m,\ y_{cen} = (\sum_{i=1}^{m} y_i)/m$.

CPE (Convex Position Estimation) a été initialement proposé par Doherty [DPG 01]. Par cet algorithme, les positions estimées des nœuds normaux sont calculées comme le résultat d'un problème d'optimisation. Puisque les processus d'optimisation sont trop compliqués pour les nœuds dont la capacité de calcul est limitée, l'algorithme *CPE* original est centralisé : un serveur plus puissant s'occupe de tous les calculs, puis diffuse en radio les résultats aux nœuds normaux. De part ce principe, l'algorithme *CPE* original n'est pas très flexible car très centralisé.

Une version simplifiée et répartie de l'algorithme *CPE* a été proposée. Le principe est de définir le rectangle estimé (*ER*), qui borne la zone de recouvrement des portées de A_1, A_2... A_m. Ensuite, le centre du rectangle estimé est considéré comme la position estimée de N_x. Les coordonnées sont calculées ainsi : $x_{ER} = \frac{\min_i x_i + \max_i x_i}{2}$, $y_{ER} = \frac{\min_i y_i + \max_i y_i}{2}$.

Centroïde et *CPE* simplifié ont deux avantages : une légère charge du réseau et une faible complexité de calcul. Mais leurs précisions ne sont pas très bonnes. Par exemple, basés sur l'algorithme *Centroïde*, les expériences dans [BHE 00] montrent que l'erreur de localisation est d'environ 1,83 mètre, lorsque la portée radio de nœuds capteurs est de 8,94 mètres.

Les algorithmes ci-dessus ne fonctionnent que pour les nœuds normaux ayant au moins 3 ancres voisines. Toutefois, si la densité des ancres n'est pas suffisamment élevée dans le réseau, certains nœuds normaux peuvent parfois se retrouver avec moins de 3 ancres voisines. Dans ce cas, il est possible d'utiliser les algorithmes basés sur *DV-hop*.

DV-hop a été initialement proposé par Niculescu [NN 03]. Le système de localisation basé sur *DV-hop* peut localiser des nœuds normaux quand ils ont moins de 3 ancres voisines, alors que *Centroïde* et *CPE* ne peuvent pas résoudre ce cas de figure. Malheureusement, ceci est obtenu au prix d'un plus grand trafic et des calculs plus nombreux et complexes. Sur la figure 1, bien que le nœud normal N_x n'ait une seule ancre voisine à sa portée radio, N_x peut utiliser *DV-hop* pour se localiser. *DV-hop* se compose des trois étapes suivantes.

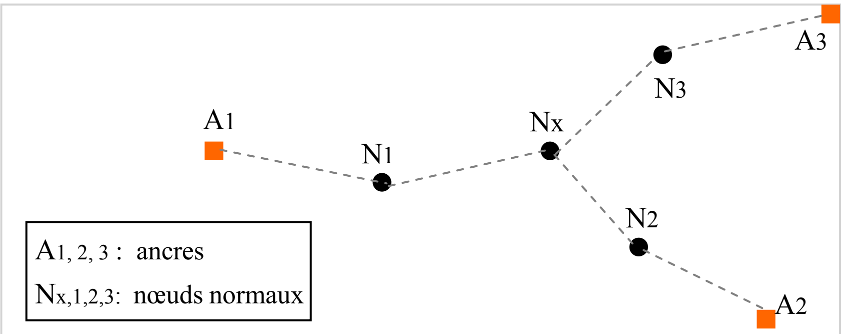


Figure 1 : Un exemple de la topologie réseau

Étape n°1 : tout d'abord, chaque ancre A_i diffuse au travers du réseau une trame contenant sa position et le nombre de sauts initialisé à 0. Cette valeur de nombre de sauts augmente pendant la diffusion de cette trame. Cela signifie que, dès que la trame est reçue par un nœud, la valeur du nombre de sauts sera incrémentée lors de son relai. A la première réception de cette trame, chaque nœud N (ancre ou nœud normal) enregistre la position de A_i, et initialise hop_i comme la valeur du nombre de sauts dans la trame. Ici, hop_i est le nombre de sauts minimum entre N et A_i. Ensuite, si N reçoit la même trame, N maintient le champ hop_i : si la trame reçue contient une valeur de nombre de sauts inférieure à hop_i, N met à jour hop_i avec cette valeur et relayera cette trame ; si la value du nombre de sauts dans la trame est supérieure à hop_i, N va ignorer cette trame. Grâce à ce mécanisme, tous les nœuds du réseau peuvent obtenir les nombres de sauts minimum à chaque ancre.

Étape n°2 : quand chaque ancre A_i a reçu les positions des autres ancres ainsi que les nombres de sauts minimaux aux autres ancres, A_i peut calculer sa distance moyenne par saut, notée dph_i. Le détail sur le calcul de dph_i peut être trouvé dans [NN 03]. Par la suite, dph_i sera diffusé à tous les nœuds du réseau par A_i.

Étape n°3 : lors de la réception de dph_i, le nœud normal N_x multiplie $hop_{i,\,Nx}$ (son nombre de sauts à A_i) par dph_i, ainsi N_x peut obtenir sa distance à chaque ancre A_i, notée $d_{i,\,Nx}$. Ici, $i \in \{1,2,\ldots m_d\}$, où m_d est le nombre d'ancres totales dans le réseau. Ensuite, chaque nœud normal peut calculer sa position estimée $N_{DV\text{-}hop}$ par trilatération. Le détail des calculs de $N_{DV\text{-}hop}$ peut être trouvé dans [NN 03].

Bien que l'algorithme *DV-hop* puisse localiser les nœuds normaux qui ont moins de trois ancres voisines, sa précision de localisation doit être améliorée elle aussi. Ainsi, de nombreux algorithmes basés sur *DV-hop* ont été proposés ces dernières années par la communauté scientifique.

DDV-hop : cet algorithme change l'étape n°2 et l'étape n°3 de l'algorithme *DV-hop*. Dans l'étape n°2 de *DDV-hop*, chaque ancre A_i diffuse non seulement sa *distance-per-hop* dph_i au travers du réseau, mais diffuse également l'erreur différentielle de dph_i. La définition et le calcul de cette erreur différentielle peut être trouvée dans [HZL 10]. Dans l'étape n°3, *DDV-hop* et *DV-hop* diffèrent entre eux par le calcul de la distance estimée entre un nœud normal N_x et chaque ancre A_i. Dans l'algorithme *DDV-hop*, N_x utilise sa propre distance par saut notée dph_{Nx} pour remplacer dph_i (la distance par saut de A_i). Ici, dph_{Nx} est obtenue comme la somme pondérée des distances par saut de toutes les ancres. Les facteurs de pondération sont décidés par l'erreur différentielle des distances par saut des ancres.

Self-adaptative DV-hop : cet algorithme se compose de deux méthodes complémentaires. Comme la seconde méthode a besoin d'informations de type *RSSI*, on ne considère généralement que la première méthode de *Self-adaptative DV-hop*. Cet algorithme induit une même charge réseau que *DV-hop*, mais modifie légèrement l'étape n°3. Dans l'étape n°3, lorsqu'un nœud normal N_x calcule sa distance estimée à A_i, N_x utilise sa propre distance par saut notée dph_{adp} pour remplacer dph_i (la distance par saut de A_i). dph_{adp} est ainsi obtenue par la somme pondérée des distances par saut de toutes les ancres. Dans cet algorithme, quand on calcule dph_{adp}, le facteur de pondération de dph_i est décidé en fonction du nombre de sauts entre N_x et A_i. Plus il y a de sauts entre N_x et A_i, plus petite est la valeur attribuée au facteur de pondération de dph_i.

Robust DV-hop : l'algorithme *Robust DV-hop* (*RDV-hop*) est proposé dans [LCK 10]. Il différe des deux algorithmes ci-dessus, car il remplace dph_i (la distance par saut de A_i). *RDV-hop* définit la valeur de *distance-par-hop* entre N_x et A_{i}, notée $dph_{\text{Nx,i}}$. $dph_{\text{Nx,i}}$ est calculé comme la somme pondérée des distances par saut entre A_i et les autres ancres. Ici, la distance par saut entre A_i et A_k est notée $dph_{\text{i,k}}$. Dans le calcul de $dph_{\text{Nx,i}}$, le facteur de pondération de $dph_{\text{i,k}}$ a le facteur maximal si N_x est un nœud sur le chemin le plus court entre A_i et A_k.

Tous les algorithmes ci-dessus basés sur *DV-hop* utilisent d'une méthode de pondération afin de déterminer une distance par saut pour chaque nœud normal. Toutefois, afin d'obtenir une distance par saut plus précise, des informations supplémentaires sont parfois nécessaires, tels que l'erreur différentielle dans *DDV-hop* et les nombre de sauts à toutes les ancres dans *Robust DV-hop*. La diffusion de ces informations supplémentaires augmente le trafic réseau. Il faut aussi noter que les résultats de simulations de ces algorithmes étudiés et présentés ci-dessus ne sont pas convaincants, car les distributions de nœuds dans les simulations sont particulières et spécifiques, plutôt que des distributions choisies au hasard. Motivé par ces constatations, notre objectif a été d'obtenir une meilleure précision sans augmenter la charge du réseau, nous avons donc proposé de nouveaux algorithmes plus performants.

II. Proposition de nouveaux algorithmes *Range-free*

Suite à l'analyse précédente sur les algorithmes typiques liés à la méthode « *range-free* », lorsqu'un nœud normal a au moins 3 ancres voisines, il peut se localiser en utilisant des algorithmes tels que *Centroïde* ou *CPE* (c'est surtout la version simplifiée de *CPE qui est utilisée car non centralisée et donc moins contraignante*). En revanche, quand un nœud normal a moins de 3 ancres voisines, il doit utiliser les algorithmes basés sur DV-hop.

Afin de permettre à chaque nœud normal de choisir son propre algorithme de localisation suivant la topologie environnante, nous avons séparé les nœuds normaux en deux classes : les nœuds de la première classe ont au moins 3 ancres voisines (à 1 saut ou à portée radio), alors que les nœuds de la deuxième classe ont moins de trois ancres voisines. Pour les nœuds normaux de chaque classe, nous avons proposé nos propres nouveaux algorithmes.

II.A Nouvel algorithme *Range-free* pour les nœuds de la classe 1

Pour les nœuds normaux de la classe 1, *Centroïde* et *CPE* sont des méthodes couramment utilisées en raison de leur faible coût de calcul et du faible trafic réseau engendré. Néanmoins, leur précision de localisation n'est pas très performante. Notre nouvelle méthode "*Mid-perpendicular*" va être capable d'atteindre une meilleure précision. Son principe est de cherche à trouver le centre de la zone de recouvrement des cellules radio des ancres voisines.

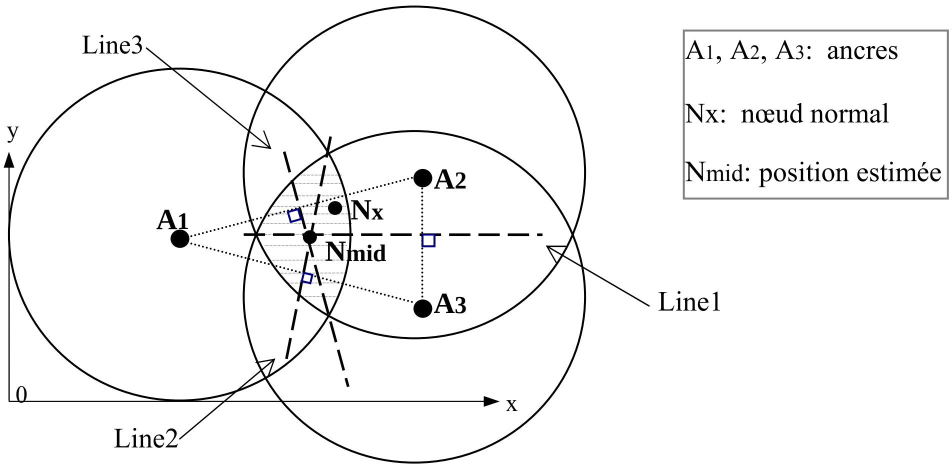


Figure 2 : *Mid-Perpendicular* dans le cas avec 3 ancres voisines

Tout d'abord, nous avons étudié le cas où un nœud normal n'a que 3 ancres voisines. Comme montré dans la figure 2, autour du nœud normal N_x, il existe dans ce cas étudié 3 ancres voisines A_1, A_2 et A_3. N_x se situe donc dans la zone de recouvrement des cellules de A_1, A_2 et A_3. Cette figure montre aussi comment calculer le centre de la zone de recouvrement. La droite A_2A_3 relie les ancres A_2 et A_3. La bissectrice à la droite A_2A_3 est "*Line1*". Par symétrie, *Line1* traverse le centre de la zone de recouvrement. Les droites A_1A_3 et A_1A_2 ont aussi leur bissectrice, respectivement *Line2* et *Line3*. Chaque bissectrice traverse le centre de la zone de recouvrement. Ainsi, l'intersection des trois bissectrices, notée N_{mid}, peut être considérée comme le centre de la zone de recouvrement. En effet, afin de calculer la position de l'intersection N_{mid}, seulement deux bissectrices sont nécessaires, par exemple, *Line1* et *Line2*. Si les coordonnées des trois ancres A_1 A_2 A_3 sont respectivement (x_1, y_1), (x_2, y_2), et (x_3, y_3), la position de N_{mid} peut être finalement calculée ainsi :

$$\begin{cases} x_{mid} = \dfrac{(x_1^2 - x_2^2)(y_3 - y_1) + (x_1^2 - x_3^2)(y_1 - y_2) + (y_1 - y_2)(y_2 - y_3)(y_3 - y_1)}{2\left[y_1(x_2 - x_3) + y_2(x_3 - x_1) + y_3(x_1 - x_2)\right]} \\ y_{mid} = \dfrac{(y_1^2 - y_2^2)(x_3 - x_1) + (y_1^2 - y_3^2)(x_1 - x_2) + (x_1 - x_2)(x_2 - x_3)(x_3 - x_1)}{2\left[x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)\right]} \end{cases}$$

On note qu'il y a une condition pour la dérivation ci-dessus : si les 3 ancres voisines de N_x forment un triangle aigu, où tous les angles sont inférieurs à 90 degrés. Pourtant, si les 3 ancres voisines forment un triangle rectangle ou un triangle obtusangle, le calcul de N_{mid} sera plus simple : cette fois, N_{mid} est le centre du côté le plus long du triangle.

Par la suite, nous avons étudié le cas où un nœud normal a plus de 3 ancres voisines. Supposons qu'il existe *m* ancres voisines autour du nœud normal N_x, et $m > 3$. Nous avons trouvé la zone de recouvrement des cellules de toutes les *m* ancres qui est obtenue principalement par trois ancres. Dans la figure 3, nous donnons un exemple de 4 ancres voisines. Sur cette figure, on peut voir que la zone de recouvrement de cellules des 4 ancres est en fait obtenue par la contribution des trois ancre A_1 A_2 et A_4. Ces trois ancres ont les caractéristiques suivantes : (1) deux d'entre eux ont la plus longue distance qui les sépare, par rapport aux distances entre les quatre ancres. C'est parce que les deux ancres les plus éloignées forment le plus petit recouvrement. Dans l'exemple, les deux ancres les plus éloignées sont A_1 et A_4. (2) La troisième ancre est la plus éloignée de la ligne reliant les deux ancres mentionnées précédemment. Dans cet exemple, comme les deux ancres les plus éloignées sont A_1 et A_4, les autres ancres sont A_2 et A_3. Par rapport à A_3, A_2 a une distance plus longue à la ligne reliant A_1 et A_4. Ainsi, la troisième ancre est A_2.

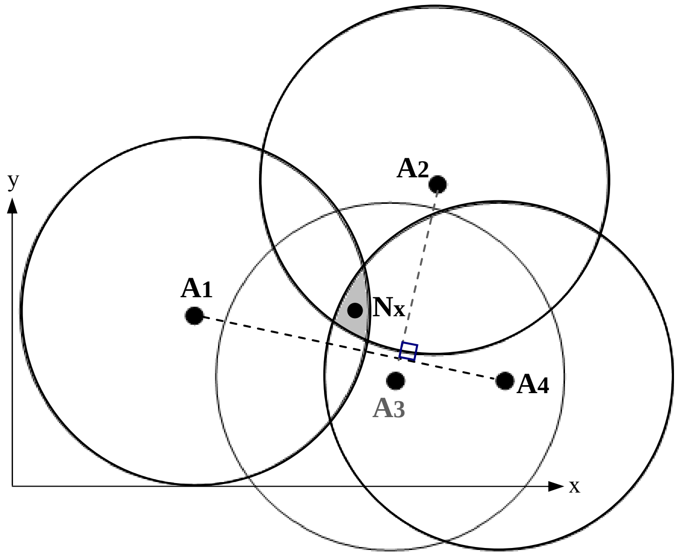


Figure 3 : Un exemple avec 4 ancres voisines

On sait maintenant comment trouver les trois ancres qui forment la zone de recouvrement des cellules de *m* ancres voisines. Tout d'abord, N_x calcule la distance entre tous les couples de deux ancres. Comme il y a *m* ancres voisines, il y aura C_m^2 distances au total à calculer. En comparant ces distances, N_x peut trouver les deux ancres les plus éloignées, notées A_i et A_k. Ensuite, parmi toutes les autres ancres excluant A_i et A_k, N_x trouve la troisième ancre qui a la plus longue distance à la ligne reliant A_i et A_k. Cette ancre est notée A_j. Ainsi, A_i, A_j et A_k sont les trois ancres qui forment la zone de recouvrement des cellules de toutes les *m* ancres. Enfin, N_x peut calculer le centre de la zone de recouvrement des cellules de A_i, A_j et A_k, le nœud obtient alors sa position estimée.

Les résultats des simulations que nous avons réalisé avec MATLAB montrent que, en moyenne, ”Mid-perpendicular” offre une meilleure précision que Centroïde et CPE.

II.B Nouveaux algorithmes *Range-free* pour les nœuds de la classe 2

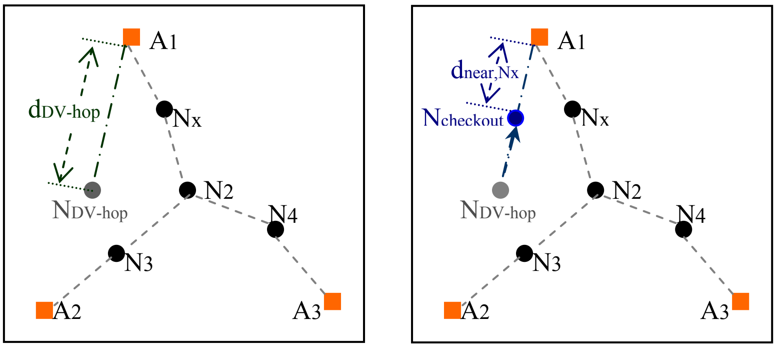
En générale, dans un réseau, il y a toujours peu d'ancres et beaucoup de nœuds normaux. Par conséquence, la plupart de nœuds normaux appartiennent à la classe 2, ayant moins de 3 ancres voisines. L'algorithme *DV-hop* est fréquemment utilisé pour localiser les nœuds de la classe n°2. Cependant, sa précision n'est pas suffisante. Pour améliorer la précision, nous avons proposé deux nouveaux algorithmes ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”.

(i) *Checkout DV-hop*

Comme nous venons de la voir, pour les nœuds normaux de la classe n°2, *DV-hop* est une méthode de localisation « *range-free* » fréquemment utilisée. L'idée clé de *DV-hop* est de calculer la distance approximative entre le nœud normal N_x et chaque ancre A_i, en multipliant le nombre de sauts minimal par la distance moyenne par saut. Cela signifie que $d_{i,\mathrm{Nx}} = hop_{i,\mathrm{Nx}} \times dph_i$, où $d_{\mathrm{i,Nx}}$ est la distance approximative entre N_x et A_{i}. $hop_{\mathrm{i,Nx}}$ est le nombre de sauts minimum entre N_x et A_{i}. dhp_{i} est la distance approximative moyenne par saut sur A_{i}. Ici, i appartient à l'ensemble [1, 2 ...m], si le nombre total des ancres est m.

Comme $d_{\mathrm{i,Nx}}$ est un paramètre fondamental pour le calcul de la position du nœud normal N_x, il a une influence considérable sur la précision de *DV-hop*. On note la distance réelle entre N_x et A_{i} : $d_{\mathrm{i,NxTrue}}$, et la différence entre $d_{\mathrm{i,NxTrue}}$ et $d_{\mathrm{i,Nx}}$: $\Delta d_{\mathrm{i,Nx}}$. Évidemment, $\Delta d_{\mathrm{i,Nx}}$ influence directement l'imprécision de *DV-hop*. On note la différence entre dhp_{i} et sa valeur réelle : Δdhp_{i}. Puis, nous obtenons : $\Delta d_{i,\mathrm{Nx}} = hop_{i,\mathrm{Nx}} \times \Delta dph_i$. Donc, lorsque $hop_{\mathrm{i,Nx}}$ augmente, Δdhp_{i} augmente également, et la précision de *DV-hop* devient plus faible. Si A_{near} est l'ancre la plus proche de N_x parmi toutes les ancres possibles, corrélativement, $hop_{near,Nx}$ est la plus petite valeur. Finalement, $\Delta d_{near,\mathrm{Nx}}$ est la plus petite erreur de distance possible. Ainsi, par rapport aux autres ancres, l'évaluation de la distance entre N_x et A_{near}, notée $d_{near,\mathrm{Nx}}$, est plus précise. En fonction de cette déduction, notre algorithme *Checkout DV-hop* tente de profiter au maximum de $d_{near,\mathrm{Nx}}$, qui est la valeur relativement la plus fiable.

Dans la figure 4, nous illustrons le principe de *Checkout DV-hop*. Notre méthode n'ajoute qu'une étape à *DV-hop*. En utilisant *DV-hop*, le nœud normal N_x obtient sa position estimée notée $N_{\mathrm{DV\text{-}hop}}$ avec les coordonnées (x', y'). Ensuite, N_x calcule la distance entre $N_{\mathrm{DV\text{-}hop}}$ et A_{near}, notée $d_{\mathrm{DV\text{-}hop}}$. Notons que N_x a évalué sa distance à A_{near}, notée $d_{near,\mathrm{Nx}}$. Par la suite, N_x exécute l'étape « *checkout* ». Le but de cette étape est de déplacer la position estimée de $N_{\mathrm{DV\text{-}hop}}$ vers une nouvelle position $N_{checkout}$, dont la distance à A_{near} est $d_{near,\mathrm{Nx}}$. Pour aboutir à cet objectif, la méthode la plus facile et la plus rapide est de déplacer la position le long de la droite reliant $N_{\mathrm{DV\text{-}hop}}$ et A_{near}. $N_{checkout}$ est sur cette droite ; la distance entre $N_{\mathrm{DV\text{-}hop}}$ et A_{near} est $d_{near,\mathrm{Nx}}$.



(a) *DV-hop* (b) *Checkout DV-hop*

Figure 4 : Principe de *Checkout DV-hop*

Dans [GWV 10], en utilisant MATLAB, nous avons éffectué plusieurs simulations avec différents scénarii où les nœuds sont aléatoirement distribués dans l'espace de localisation. Les résultats montrent que notre algorithme *Checkout DV-hop* atteint une précision 15% plus élevée que l'algorithme *DV-hop*.

(ii) *Selective 3-Anchor DV-hop*

En utilisant la distance estimée entre le nœud normal et sa plus proche ancre, "*Checkout DV-hop*" ajuste le résultat de la localisation de *DV-hop*. Bien que "*Checkout DV-hop*" n'ajoute qu'une étape simple à *DV-hop*, son amélioration sur la précision n'est pas très remarquable. Ainsi, nous avons proposé un autre nouvel algorithme "*Selective 3-Anchor DV-hop*", qui peut obtenir une meilleure précision au prix d'une augmentation plus importante de la complexité de calcul. Le principe basique de cet algorithme est le suivant : le nœud normal sélectionne toutes les trois ancres possibles afin de former des « *3-anchor groups* », puis il calcule les positions estimées grâce à ces « *3-anchor groups* ». Enfin, en fonction de la relation entre les positions estimées et les connectivités, le nœud normal choisit la position la plus précise.

Considérons un réseau avec m_d ancres A_1 A_2 ... A_{md}. En utilisant l'algorithme *DV-hop*, un nœud normal N_x peut calculer sa position estimée $N_{\mathrm{DV\text{-}hop}}$ en fonction de ses distances estimées aux ancres. Ainsi, la précision de ces distances estimées a une influence importante sur la précision de *DV-hop*.

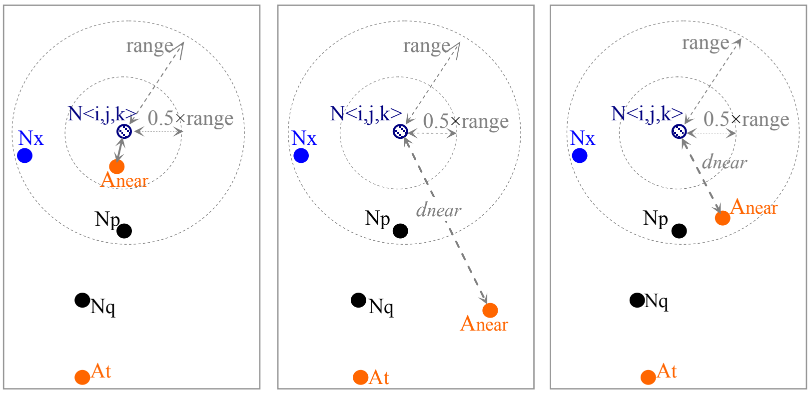
En fait, au lieu d'utiliser toutes les m_d distances estimées, trois distance sont suffisantes pour N_x pour calculer sa position. Par exemple, nous pouvons utiliser $d_{i,\mathrm{Nx}}$, $d_{j,\mathrm{Nx}}$, $d_{k,\mathrm{Nx}}$, qui sont les trois distances estimées entre N_x et les trois ancres A_i, A_j, A_k. Puis, en se basant sur la méthode MLE (*Maximum Likelihood Estimation*), nous pouvons obtenir une « *3-anchor estimated position* » de N_x, notée as $N_{<i,j,k>}$.

Le principe de *Selective 3-Anchor DV-hop* est de sélectionner la plus précise "*3-anchor estimated position*". Ici, le critère de sélection est la connectivité. Dans l'algorithme *DV-hop*, la connectivité de N_x est définie comme les nombres de sauts minimum entre N_x et des ancres. Par exemple, si dans un réseau, il y a m_d ancres au total, et si le nombre de sauts minimum entre N_x et chaque ancre A_i est $hop_{i,Nx}$, alors la connectivité de N_x est [$hop_{1,Nx}$, $hop_{2,Nx}$ ... $hop_{md,Nx}$]. Dans [GWV 11], nous avons donné la relation entre la connectivité et de la distance : plus petite est la différence de connectivité entre deux nœuds, plus petite est la distance entre eux. Selon cette relation, la "*3-anchor estimated position*" ayant la connectivité la plus similaire à N_x devait être la plus proche de N_x. Ainsi, le principe basique de notre algorithme *Selective 3-Anchor DV-hop* est de choisir la "3-ancre position estimée" qui a la connectivité la plus semblable à N_x.

Cependant, la connectivité de "*3-anchor estimated position*" $N_{<i,j,k>}$ est encore inconnue. Par conséquent, nous avons proposé la méthode suivante pour calculer le nombre de sauts entre $N_{<i,j,k>}$ et chaque ancre, noté $hop_{<i,j,k>,t}$. On note la distance entre $N_{<i,j,k>}$ et chaque ancre A_t : $d_{<i,j,k>,t}$. Ainsi, si N_x connaît la distance par saut entre $N_{<i,j,k>}$ et A_t notée $dph_{<i,j,k>,t}$, alos N_x peut calculer le nombre de sauts entre $N_{<i,j,k>}$ et A_t comme étant $hop_{<i,j,k>,t} = \frac{d_{<i,j,k>,t}}{dph_{<i,j,k>,t}}$. Il faut donc trouver comment estimer $dph_{<i,j,k>,t}$.

En fait, N_x connaît seulement les distances par saut de chaque ancre: dph_1, dph_2, ..., dph_{md}, y compris la distance par saut de A_t notée dph_t. Ainsi, on doit estimer $dph_{<i,j,k>,t}$ basée sur dph_1, dph_2, ..., dph_{md}. Pour cela, trois types de relation entre $N_{<i,j,k>}$ et sa plus proche ancre A_{near} sont considérés, en fonction de leur distance. Dans le premier cas, la distance entre $N_{<i,j,k>}$ et A_{near} est si petite que nous pouvons utiliser la distance par saut sur A_{near} (notée dph_{near}) comme une approximation de $dph_{<i,j,k>,t}$. En revanche, dans le deuxième cas, la distance entre $N_{<i,j,k>}$ et A_{near} est si grande que nous ne pouvons utiliser que dph_t comme l'approximation de $dph_{<i,j,k>,t}$. Le troisième cas est entre les deux cas ci-dessus, donc, la valeur de $dph_{<i,j,k>,t}$ peut être définie comme la moyenne des dph_{near} et dph_t. Ces trois cas sont présentés dans la figure 5.

La procédure de l'algorithme *Selective 3-Anchor DV-hop* se résume comme suit. Les première et seconde étapes sont les mêmes que *DV-hop*. Dans la troisième étape, N_x calcule d'abord ses "3-ancre positions estimée", puis N_x calcule la connectivité de chaque "3-ancre position estimée". Enfin, N_x choisit la meilleure "3-ancre position estimée" qui a la connectivité la plus semblable avec lui.



5(a) d_{near}<range/2 5(b) d_{near} >range 5(c) range/2< d_{near} <range

Figure 5: Trois types de relation entre $N_{<i,j,k>}$ et A_{near}

Les résultats des simulations réalisées et présentées dans [GWV 11] montrent que notre algorithme *Selective 3-Anchor DV-hop* atteint une meilleure précision que plusieurs autres algorithmes existants. L'amélioration de la précision peut être de 20% à 57%, par rapport aux différents algorithmes et aux différentes scénarii.

III. Proposition de nouveaux protocoles de localisation

Lors de la vérification de nos trois nouveaux algorithmes présentés ci-dessus, nous avons constaté que la plupart des algorithmes existants sont étudiés par la communauté scientifique en utilisant uniquement des simulateurs algorithmiques tels que MATLAB. Les problèmes liés aux réseaux et les influences des protocoles sont généralement négligés comme la collision des trames au niveau MAC et la synchronisation des nœuds. Nous avons alors proposé deux protocoles : ”*DV-hop protocol*” et ”*Classe-1 protocol*”. Par la suite, nous avons combiné ces deux protocoles pour obtenir notre ”*adaptive range-free localization protocol*”.

III.A *DV-hop protocol*

Notre ”*DV-hop protocol*” peut être utilisé pour mettre en œuvre les algorithmes basés sur *DV-hop*, notamment ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”. Dans ”*DV-hop protocol*”, nous avons défini des formats de trames adaptés, une nouvelle méthode d'accès ”E-CSMA/CA” pour améliorer les performances de la couche MAC classique ”*non-slotted CSMA/CA*”, et adapté plusieurs paramètres pour clore chaque étape de protocole.

Deux formats de trames sont proposés pour les deux premières étapes de *DV-hop protocol*. Ils sont conforment au format général défini dans la norme IEEE 802.15.4. À l'étape n°1, chaque ancre A_i diffuse sur le réseau une trame "*frame_pos*$_i$", afin que tous les nœuds (en les ancres et les nœuds normaux) puissent connaître la position de A_i et le nombre de sauts minimum à A_i. À l'étape n°2, A_i diffuse à travers le réseau une trame "*frame_dph*$_i$" qui contient la distance moyenne par saut sur A_i.

Nous avons aussi proposé une nouvelle méthode d'accès ”E-CSMA/CA” pour réduire les collisions de trames. Les collisions peuvent se produire lorsque les ancres diffusent simultanément leurs trames. Lorsque chaque ancre A_i diffuse une trame, selon le principe du *non-slotted CSMA/CA*, A_i attend d'abord une courte période aléatoire, puis si le canal est toujours libre, la trame est envoyée immédiatement. Dans la norme, la courte période aléatoire est choisie au hasard parmi les 8 valeurs: 0, t_{bo}, $2\times t_{bo}$, …, $7\times t_{bo}$, où t_{bo} est la période de *back-off*. Conformément à la norme IEEE 802.15.4, si le débit de données est de 250kbps, la durée t_{bo} est de 320 ms, et la valeur maximale de cette période aléatoire est de 7×320µs=2.24ms. Avec une telle période d'attente aussi courte, lorsque les trames sont diffusées simultanément au travers du réseau, les collisions se produisent trop fréquemment.

La solution que nous proposons pour réduire les collisions est d'ajouter une autre plus longue durée aléatoire avant *CSMA/CA*. Ainsi, la probabilité de collision est réduite. Au début de l'étape n°1 du *DV-hop protocol*, chaque ancre A_i attend une durée aléatoire notée t_{wpi}. Puis, A_i effectue le *non-slotted CSMA/CA* et envoie sa trame *frame_pos*$_i$. De même, au début de l'étape n°2, A_i attend une durée aléatoire notée t_{wdi}. Puis, A_i effectue le classique CSMA/CA et envoie sa trame *frame_dph*$_i$.

Pour clore chaque étape du *DV-hop protocol*, nous avons proposé quelques paramètres spécifiques. Tout d'abord, *num_wait_pos* est la valeur pour clore l'étape n°1. Tant qu'un nœud n'a reçu ce nombre de positions d'ancres, il ne peut pas terminer l'étape n°1. Egalement, $T^0_i+t_{s1}$ a été proposé comme étant le délai pour terminer l'étape n°1. Même si un nœud n'a encore reçu pas au moins *num_wait_pos* positions d'ancres, il doit terminer étape n°1 si ce délai expire. De plus, *num_wait_dph* est le nombre de distances par saut pour finaliser l'étape n°2. Enfin, nous avons proposé de la même façon $T^0_i+t_{s1}+t_{s2}$ qui est le délai pour terminer étape n°2. On va présenter et analyser les résultats de simulations avec le simulateur WSNet sur *DV-hop protocol* dans la section IV.

III.B *Class-1 protocol*

Notre "*Classe-1 protocol*" peut être utilisé pour mettre en œuvre les algorithmes tels que *Centroïde*, *CPE* et *Mid-perpendicular*. Il inclut 3 étapes, le principe basique du *Class-1 protocol* est présenté ci-dessous.

Tout d'abord, N_x diffuse une trame à ses voisins pour une demande de localisation. Cette trame est notée *frame_req*. Lors de la diffusion de *frame_req*, notre méthode E-CSMA/CA doit être utilisée pour réduire les collisions, parce que plusieurs nœuds normaux peuvent être simultanément prêts à envoyer leurs trames.

Deuxièmement, si une ancre voisine de N_x reçoit la demande de N_x, cette ancre envoie sa position à N_x. Ici, E-CSMA/CA est aussi recommandé pour réduire les collisions, parce qu'il y a peut-être de nombreuse ancres autour de N_x (6 par exemple). Simultanément, toutes les ancres reçoivent *frame_req* et sont prêtes à envoyer leurs positions.

Enfin, si pendant une période t_{recv}, N_x a reçu des positions en provenance d'au moins 3 ancres voisines, N_x peut calculer sa position en utilisant les algorithmes tels que *Centroïde*, *CPE* et *Mid-perpendiculer*. On va présenter et analyser les résultats de simulations sur *Class-1 protocol* dans la section IV.

III.C *Adaptive Range-free Localization Protocol*

Les deux protocoles présentés ci-dessus ont chacun leurs avantages et inconvénients. Le *Classe-1 protocol* est simple, mais il nécessite les nœuds normaux a au moins 3 ancres voisines. *DV-hop protocol* peut être utilisé par tous les nœuds normaux, mais il induit une charge du réseau importante.

Afin de profiter des avantages de ces deux protocoles, leur combinaison est considérée comme notre « *adaptative range-free localisation protocole* ».

Le choix entre *Classe-1 protocol* et *DV-hop protocol* est décidé par l'administrateur du réseau. Nous avons besoin de fixer un seuil pour le ratio d'ancres, noté RA_{thresh}. Si le ratio d'ancres est inférieur à RA_{thresh}, l'administrateur choisit *DV-hop protocol* car la plupart des nœuds normaux ont moins de 3 ancres voisines. Mais, si le ratio d'ancres est supérieur à RA_{thresh}, afin d'éviter un grand trafic, *Classe-1 protocol* doit être utilisé de préférence.

La valeur de RA_{thresh} est choisie par l'administrateur en fonction du trafic maximum qu'on peut accepter et de sa connaissance sur le nombre d'ancres dans le réseau. Une petite valeur de RA_{thresh} indique qu'on peut accepter une faible charge du réseau. Mais la valeur de RA_{thresh} ne peut pas être trop basse car dans ce ca-là de nombreux nœuds normaux ayant moins de 3 ancres voisines ne peuvent pas être localisés.

IV. Simulations et Evaluations

Afin d'évaluer la précision des algorithmes «*range-free*», des simulations ont été effectuées en utilisant MATLAB. On note que la distribution des nœuds a une influence importante sur la précision des algorithmes. En général, la précision de notre algorithme *Mid-perpendiculer* est de 15% supérieure à celle des algorithmes *Centroid* et *CPE*. Notre algorithme *Checkout DV-hop* a une précision de localisation d'environ 15% meilleure que *DV-hop*. Cependant, notre algorithme *Selective 3-Anchor DV-hop* est 45% plus précis que *DV-hop*.

Nous avons aussi évalué la complexité théorique de calcul des algorithmes. *Centroid* et *CPE* ont une faible complexité de l'odre de $O(m)$, alors que *Mid-perpendicular* entraine une complexité aussi élevée que $O(m^2)$. La complexité de *DV-hop* et *Checkout DV-hop* reste au niveau de $O(m_d)$, mais *Selective 3-Anchor DV-hop* entraine une complexité la plus élevée de l'odre de $O(m_d^{\,3})$. Ici, m est le nombre d'ancres voisines autour un nœud normal, tandis que m_d représente le nombre de toutes les ancres dans le réseau.

Nos protocoles ont été modélisés et simulés en utilisant le simulateur *WSNet* dans le contexte de réseaux de capteurs conformes au standard IEEE 802.15.4. Les résultats montrent que globalement, nos nouveaux algorithmes associés aux protocoles adéquats sont plus précis que les algorithmes classiques. Par rapport à la charge du réseau, les protocoles basés sur *DV-hop* sont beaucoup plus lourds que les protocoles de la classe 1, parce que *DV-hop* nécessite des diffusions globales dans le réseau. Eu égard de la mobilité des nœuds, l'influence sur la précision des protocoles basés sur *DV-hop* est plus importante que celle des protocoles de la classe 1, parce que *DV-hop* nécessite une durée plus longue pour les diffusions globales. Finalement, nous avons aussi montré que la synchronisation des nœuds entre les différentes étapes n'est pas nécessaire pour nos protocoles.

V. Conclusion et Perspectives

Dans le contexte de réseaux de capteurs sans fil, la technique de localisation "*range-free*" est plus efficiente, par rapport au principe "*range-based*". Par conséquence, nous avons focalisé nos travaux de cette thèse sur les techniques "*range-free*", une thèse complémentaire dans la même équipe travaillant elle sur les méthodes « *range-based* ».

Afin de permettre à chaque nœud normal de choisir son propre algorithme de localisation suivant la topologie environnante, nous avons proposé un mécanisme adapté en séparant les nœuds normaux en deux classes : les nœuds de la première classe ont au moins 3 ancres voisines, alors que les nœuds de la deuxième classe ont moins de trois ancres voisines.

Pour les nœuds normaux de la classe 1, nous avons proposé un nouvel algorithme ”*Mid-perpendicular*”, qui cherche à trouver un centre de la zone de recouvrement des cellules radio des ancres voisines. Les résultats des simulations par MATLAB montrent que, en moyenne, ”*Mid-perpendicular*” offre une meilleure précision que Centroïde et CPE.

Pour les nœuds normaux de la classe 2, nous avons proposé deux nouveaux algorithmes ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”. Les résultats des simulations montrent que, *Checkout DV-hop* a une précision de localisation d'environ 15% supérieure à *DV-hop*, tandis que la précision de *Selective 3-Anchor DV-hop* est 45% meilleure que *DV-hop*.

Lors de la vérification par simulation de nos trois nouveaux algorithmes, nous avons remarqué que la plupart des algorithmes existants sont étudiés en utilisant uniquement des simulateurs algorithmiques tels que MATLAB, les problèmes liés aux réseaux et les influences des protocoles ont été généralement négligés comme la collision des trames et la synchronisation des nœuds. Nous avons pris soin de proposer deux protocoles associés : ”*DV-hop protocol*” et ”*Classe-1 protocol*”. Par la suite, nous avons combiné ces deux protocoles pour obtenir notre ”*adaptive range-free localization protocol*”. Pour ces protocoles, nous avons défini des formats de trames adaptés, et une nouvelle méthode d'accès ”E-CSMA/CA” pour améliorer les performances de la couche MAC classique ”*non-slotted CSMA/CA*”. D'un côté, notre ”*DV-hop protocol*” peut être utilisé pour mettre en œuvre les algorithmes basés sur DV-hop, notamment ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”. De l'autre côté, notre ”*Classe-1 protocol*” peut être utilisé pour mettre en œuvre les algorithmes tels que *Centroïde*, CPE et ”*Mid-perpendicular*”.

Basé sur nos protocoles, en utilisant le simulateur WSNet, nous avons simulé différents algorithmes ”range-free” dans le contexte de réseaux de capteurs conformes au standard IEEE 802.15.4. Les résultats ont été présentés et analysés en termes de la précision de la localisation, charge du réseau, mobilité des nœuds, et synchronisation de ces derniers.

En perspectives, nous proposons d'étudier la performance des algorithmes en utilisant un modèle de couche radio réel. Il serait également intéressant de combiner des algorithmes ”*range-based*” et ”*range-free*”. Enfin, la dernière perspective envisagée porte sur la mise en œuvre de nos algorithmes et protocoles sur des prototypes réels.

Annex 2 : MATLAB Source Code for DV-hop Based Algorithms

```
%Set Parameters
Hsize = 100; % Size of one side of the square area
Crange = 20; % communication range
Dintl = 1; % Distance of interval in one side
Nintl = Hsize / Dintl; % divide the Hsize into Nintl intervals
Narea = Nintl^2; % Narea: number of intervals in the total area
Numnodes = 100; % number of nodes
%randomly distribute all nodes
Nanc=[5,10,20,30,40,50,60,70,80,90]; %ratio of anchors
Nrat=size(Nanc);
Nrat=Nrat(2); % number of ratios
Rnum_dis=ones(1,Nrat)*20; % number of random times for nodes distribution
Rnum=ones(1,Nrat)*100; % number of random times for anchors selection
locerr = zeros(Nrat,6);   % location error
locnum = zeros(Nrat,6);
%Simulation begins
for countrat=1:Nrat %different anchors ratio
    cntRnumdis=countrat
    Nanctemp=Nanc(countrat)/100*Numnodes;
    for disrandtimes=1:Rnum_dis(cntRnumdis)
        Pnode = zeros(Numnodes,2*6+Nanc(Nrat));
        Pnode(:,13:(2*6+Nanc(Nrat)))=Pnode(:,13:(2*6+Nanc(Nrat)))-1;
        % intial hop number is -1
        %intialization: 100 sensor nodes randomly distruted
        nodechosen=[1:Narea];
        ctime = datestr(now, 30);
         tseed = str2num(ctime((end - 5):end))+disrandtimes;
        rand('seed', tseed) % in order to get a true random, not a fake random
        Rseq = randint(Numnodes,1,[1,Narea]);  %100 random numbers between [1:Narea]
        for seqcount=1:Numnodes %exchange,because maybe two same numbers in Rseq
            nodetemp=nodechosen(seqcount);
            nodechosen(seqcount)=nodechosen(Rseq(seqcount));
            nodechosen(Rseq(seqcount))=nodetemp;
        end
        Rseq=nodechosen(1:Numnodes)';   %now any two numbers in Rseq are different
        Rseq = sort(Rseq);
        Pnode(:,1)=(0.5+floor((Rseq-1)/Nintl))*Dintl;
        Pnode(:,2)=(0.5+mod(Rseq-1,Nintl))*Dintl;
        %%second iteration is anchors selection
        for Rcount=1:Rnum(countrat) % random times for anchors
            locerrmaxtemp=zeros(1,6);   locerrmintemp=ones(1,6)*1000000;
            Pnode(:,13:(2*6+Nanc(Nrat))) = zeros(Numnodes,Nanc(Nrat));
            % intial hop number is -1, maximum anchor number is Nanc(Nrat)
            Pnode(:,13:(2*6+Nanc(Nrat)))=Pnode(:,13:(2*6+Nanc(Nrat)))-1;
            %%%%%%%%%%%%choose anchors (randomly)
            ctime = datestr(now, 30);
             tseed = str2num(ctime((end - 5) : end))+Rcount;
            rand('seed', tseed) % in order to get a true random, not a fake random
            anchors= randint(Nanctemp,1,[1,Numnodes]);
            anchosen=[1:Numnodes];
            for anccount=1:Nanctemp
                anctemp=anchosen(anccount);
                anchosen(anccount)=anchosen(anchors(anccount));
                anchosen(anchors(anccount))=anctemp;
            end
            anchors=anchosen(1:Nanctemp);
            anchors=sort(anchors);
            for Acount=1:Nanctemp
                Pnode(anchors(Acount),12+Acount)=0;
            end
            %%for every node, calculate its smallest number of hops to each anchor
            for Acount=1:Nanctemp
                nodepass=[]; nodepass=[nodepass;anchors(Acount)];
                Vnum=0;     % number of voisins(neighour)
                Pvoi=[]; % indice of voisins
                for Ncount=1:Numnodes % find all neighbour nodes to this anchor
NdisA=(Pnode(Ncount,1:2)-Pnode(anchors(Acount),1:2))*(Pnode(Ncount,1:2)-Pnode(anchors(Acount),1:2))';
                    if NdisA>0 && NdisA<Crange*Crange   %this is a neighbour
                        Vnum=Vnum+1;
                        Pvoi=[Pvoi;Ncount];
                        Pnode(Ncount,12+Acount)=0+1;  %one-hop to this anchor
                    end
                end
                Tvoi=Pvoi;    % Tvoi:temparely used for indices of voisins
                Pvoi=[];
                nodepass=[nodepass;Tvoi];
                sigfin=prod(Pnode(:,12+Acount)+1);
                %signal shows the end of this, means all nodes get the hop numbers
                while sigfin == 0
```

```
                    for Vcount=1:Vnum
                        for Ncount=1:Numnodes
                            if prod(nodepass-Ncount)~=0
                                NdisA=(Pnode(Ncount,1:2)-
Pnode(Tvoi(Vcount),1:2))*(Pnode(Ncount,1:2)-Pnode(Tvoi(Vcount),1:2))';
                                if NdisA>0 && NdisA<Crange*Crange % this is a neighbour
                                    Pvoi=[Pvoi;Ncount];
                                    nodepass=[nodepass;Ncount];
           if Pnode(Ncount,12+Acount)==-1 || Pnode(Ncount,12+Acount)>Pnode(Tvoi(Vcount),12+Acount)+1
                    Pnode(Ncount,12+Acount)=Pnode(Tvoi(Vcount),12+Acount)+1;
                                    end
                                end
                            end
                        end
                    end
                    Tvoi=[];
                    Tvoi=Pvoi;
                    Pvoi=[];
                    Vnum=size(Tvoi);
                    Vnum=Vnum(1);
                    sigfin=prod(Pnode(:,12+Acount)+1);
                    if Vnum == 0
                        sigfin=1;
                    end
                end
            end
            %%calculate distance per hop between any two anchors
            disperhop=[];
            for Acount1=1:Nanctemp
                distemp=[];
                for Acount2=1:Nanctemp
                    if Acount2 ~= Acount1
       distemp=[distemp;sqrt((Pnode(anchors(Acount1),1:2)-
Pnode(anchors(Acount2),1:2))*(Pnode(anchors(Acount1),1:2)-Pnode(anchors(Acount2),1:2))')];
                    end
                end
            disperhop=[disperhop,sum(distemp)/sum(Pnode(anchors(Acount1),13:(12+Nanctemp)))];
            end
            distemp=[];
            %%%now to localize each normal node
            for Ncount=1:Numnodes
                if prod(Ncount-anchors)~=0
                    hopnumtemp=Pnode(Ncount, 13:(12+Nanctemp));
                    Pnode(Ncount,13:(12+Nanctemp))=Pnode(Ncount,13:(12+Nanctemp)).*disperhop; %
distance to each anchor
                    est_dis=Pnode(Ncount,13:(12+Nanctemp)); %estimated distance
                    hop_cnt=hopnumtemp;

                    %DV-hop
                    matrixA=zeros(Nanctemp-1,2);  matrixB=zeros(Nanctemp-1,1);
    elemB=(Pnode(anchors(Nanctemp),1:2))*(Pnode(anchors(Nanctemp),1:2))'-Pnode(Ncount,12+Nanctemp)^2;
                    for matcount=1:(Nanctemp-1)
                  matrixA(matcount,:)=-2*[Pnode(anchors(matcount),1:2)-Pnode(anchors(Nanctemp),1:2)];
                  matrixB(matcount,:)=Pnode(Ncount,12+matcount)^2-
(Pnode(anchors(matcount),1:2))*(Pnode(anchors(matcount),1:2))'+elemB;
                    end
                    Pnode(Ncount,3:4)=(inv(matrixA'*matrixA)*matrixA'*matrixB)';
          errtemp=sqrt((Pnode(Ncount,3:4)-Pnode(Ncount,1:2))*(Pnode(Ncount,3:4)-Pnode(Ncount,1:2))');
                    locnum(countrat,1)=locnum(countrat,1)+1;
                    locerr(countrat,1)=locerr(countrat,1)+errtemp;
                    %Selective DV-hop
                    %choose any three estimated distance to form a group
                    bon_group=zeros(3,3); comp_temp=100000*ones(1,3); groupres=zeros(3,1);
                    tmpdisperhop=zeros(1,Nanctemp);
                    for cnt1=1:(Nanctemp-2)
                       est_dis1=est_dis(cnt1);
                       for cnt2=(cnt1+1):(Nanctemp-1)
                           est_dis2=est_dis(cnt2);
                           for cnt3=(cnt2+1):Nanctemp
                               est_dis3=est_dis(cnt3);
                               matrixA=zeros(2,2); matrixB=zeros(2,1);
                        elemB=(Pnode(anchors(cnt3),1:2))*(Pnode(anchors(cnt3),1:2))'-est_dis(cnt3)^2;
                               matrixA(1,:)=-2*[Pnode(anchors(cnt1),1:2)-Pnode(anchors(cnt3),1:2)];
                               matrixA(2,:)=-2*[Pnode(anchors(cnt2),1:2)-Pnode(anchors(cnt3),1:2)];
           matrixB(1,:)=est_dis(cnt1)^2-(Pnode(anchors(cnt1),1:2))*(Pnode(anchors(cnt1),1:2))'+elemB;
           matrixB(2,:)=est_dis(cnt2)^2-(Pnode(anchors(cnt2),1:2))*(Pnode(anchors(cnt2),1:2))'+elemB;
                          if det(matrixA)>1.0e-8
                             res_tmp=(inv(matrixA)*matrixB)'; mindisonehop=10000; mindisanc=0;
                                   for anc_cnt=1:Nanctemp
      tempdisonehop=sqrt((res_tmp-Pnode(anchors(anc_cnt),1:2))*(res_tmp-Pnode(anchors(anc_cnt),1:2))');
                                       if tempdisonehop<mindisonehop
                                           mindisonehop=tempdisonehop;
```

```
                                        if tempdisonehop < Crange
                                            mindisanc=anc_cnt;
                                        end
                                    end
                                end
                                if mindisanc~=0
                                    for anc_cnt=1:Nanctemp
                                        if anc_cnt~=mindisanc
                                            dishopmindisanc=sqrt((Pnode(anchors(anc_cnt),1:2)-
Pnode(anchors(mindisanc),1:2))*(Pnode(anchors(anc_cnt),1:2)-
Pnode(anchors(mindisanc),1:2))')/Pnode(anchors(mindisanc),12+anc_cnt);
                                            if mindisonehop<Crange/2
                                                tmpdisperhop(anc_cnt)=dishopmindisanc;
                                            else
                                     tmpdisperhop(anc_cnt)=(dishopmindisanc+disperhop(anc_cnt))/2;
                                            end
                                        end
                                    end
                                    tmpdisperhop(mindisanc)=100;
                                else
                                    tmpdisperhop=disperhop;
                                end
                                err_temp=0;hopcnt_temp=zeros(1,Nanctemp); finerrtmp=0;
                                for cnt_anc=1:Nanctemp
         dis_temp=sqrt((res_tmp-Pnode(anchors(cnt_anc),1:2))*(res_tmp-Pnode(anchors(cnt_anc),1:2))');
                                    hopcnt_temp(cnt_anc)=dis_temp/tmpdisperhop(cnt_anc);
                                    if dis_temp<=Crange
                                        hopcnt_temp(cnt_anc)=1;
                                    end
                                    err_temp=err_temp+abs(hopcnt_temp(cnt_anc)-hop_cnt(cnt_anc));
                                end
                                thisgroup=[cnt1,cnt2,cnt3];
                                if err_temp<comp_temp(1)
                                    comp_temp(1)=err_temp;
                                    bon_group(1,:)=thisgroup;
                                end
                            end
                        end
                    end
                end
                if bon_group(1)==0
                    Pnode(Ncount,5:6)=Pnode(Ncount,3:4);
                else
                    cnt1=bon_group(1,1);cnt2=bon_group(1,2);cnt3=bon_group(1,3);
                    matrixA=zeros(2,2); matrixB=zeros(2,1);
                    elemB=(Pnode(anchors(cnt3),1:2))*(Pnode(anchors(cnt3),1:2))'-est_dis(cnt3)^2;
                    matrixA(1,:)=-2*[Pnode(anchors(cnt1),1:2)-Pnode(anchors(cnt3),1:2)];
                    matrixA(2,:)=-2*[Pnode(anchors(cnt2),1:2)-Pnode(anchors(cnt3),1:2)];
           matrixB(1,:)=est_dis(cnt1)^2-(Pnode(anchors(cnt1),1:2))*(Pnode(anchors(cnt1),1:2))'+elemB;
           matrixB(2,:)=est_dis(cnt2)^2-(Pnode(anchors(cnt2),1:2))*(Pnode(anchors(cnt2),1:2))'+elemB;
                    restemp4=(inv(matrixA)*matrixB)';
                    Pnode(Ncount,5:6)=restemp4;
                end
          errtemp=sqrt((Pnode(Ncount,5:6)-Pnode(Ncount,1:2))*(Pnode(Ncount,5:6)-Pnode(Ncount,1:2))');
                locnum(countrat,6)=locnum(countrat,6)+1;
                locerr(countrat,6)=locerr(countrat,6)+errtemp;
                %DDV-hop
                diff_err=zeros(1,Nanctemp);
                for cntemp=1:Nanctemp
                    dis_err=0;
                    for cntemp2=1:Nanctemp
                        if cntemp2 ~= cntemp
                            distemp=sqrt((Pnode(anchors(cntemp),1:2)-
Pnode(anchors(cntemp2),1:2))*(Pnode(anchors(cntemp),1:2)-Pnode(anchors(cntemp2),1:2))');
                dis_err=dis_err+abs(disperhop(cntemp)-distemp/Pnode(anchors(cntemp),12+cntemp2));
                        end
                    end
                    diff_err(cntemp)=dis_err/(Nanctemp-1);
                end
                diff_err=diff_err/sum(diff_err);
                avghopsize=sum(diff_err.*disperhop);
                dis_n_a=hopnumtemp*avghopsize;
                matrixA=zeros(Nanctemp-1,2);  matrixB=zeros(Nanctemp-1,1);
elemB=(Pnode(anchors(Nanctemp),1:2))*(Pnode(anchors(Nanctemp),1:2))'-dis_n_a(Nanctemp)^2;
                for matcount=1:(Nanctemp-1)
              matrixA(matcount,:)=-2*[Pnode(anchors(matcount),1:2)-Pnode(anchors(Nanctemp),1:2)];
                    matrixB(matcount,:)=dis_n_a(matcount)^2-
(Pnode(anchors(matcount),1:2))*(Pnode(anchors(matcount),1:2))'+elemB;
                end
                if det(matrixA'*matrixA)>1.0e-4
                    Pnode(Ncount,5:6)=(inv(matrixA'*matrixA)*matrixA'*matrixB)';
                else
```

```
                    Pnode(Ncount,5:6)=Pnode(Ncount,3:4);
                end
          errtemp=sqrt((Pnode(Ncount,5:6)-Pnode(Ncount,1:2))*(Pnode(Ncount,5:6)-Pnode(Ncount,1:2))');
                locnum(countrat,2)=locnum(countrat,2)+1;
                locerr(countrat,2)=locerr(countrat,2)+errtemp;
                %Self-Adaptive DV-hop
                weightcoef=zeros(1,Nanctemp);
                sumhop=sum(hopnumtemp);
                weightcoef=(sumhop-hopnumtemp)/sumhop/(Nanctemp-1);
                avghopsize2=sum(weightcoef.*disperhop);
                dis_n_a=hopnumtemp*avghopsize2;
                matrixA=zeros(Nanctemp-1,2);  matrixB=zeros(Nanctemp-1,1);
            elemB=(Pnode(anchors(Nanctemp),1:2))*(Pnode(anchors(Nanctemp),1:2))'-dis_n_a(Nanctemp)^2;
                for matcount=1:(Nanctemp-1)
              matrixA(matcount,:)=-2*[Pnode(anchors(matcount),1:2)-Pnode(anchors(Nanctemp),1:2)];
                    matrixB(matcount,:)=dis_n_a(matcount)^2-
(Pnode(anchors(matcount),1:2))*(Pnode(anchors(matcount),1:2))'+elemB;
                end
                Pnode(Ncount,7:8)=(inv(matrixA'*matrixA)*matrixA'*matrixB)';
          errtemp=sqrt((Pnode(Ncount,7:8)-Pnode(Ncount,1:2))*(Pnode(Ncount,7:8)-Pnode(Ncount,1:2))');
                locnum(countrat,3)=locnum(countrat,3)+1;
                locerr(countrat,3)=locerr(countrat,3)+errtemp;
                %Robust DV-hop
                diff_err=zeros(1,Nanctemp);
                for cntemp=1:Nanctemp
                    weightcoef2=[]; avgsizetemp=[];
                    for cntemp2=1:Nanctemp
                        if cntemp2 ~= cntemp
             weightcoef2=[weightcoef2,1/(hopnumtemp(cntemp)+hopnumtemp(cntemp2)-
Pnode(anchors(cntemp),12+cntemp2)+1)];
             avgsizetemp=[avgsizetemp, sqrt((Pnode(anchors(cntemp),1:2)-
Pnode(anchors(cntemp2),1:2))*(Pnode(anchors(cntemp),1:2)-
Pnode(anchors(cntemp2),1:2))')/Pnode(anchors(cntemp),12+cntemp2)];
                        end
                    end
                    diff_err(cntemp)=sum(weightcoef2.*avgsizetemp)/sum(weightcoef2);
                end
                dis_n_a=hopnumtemp.*diff_err;
                matrixA=zeros(Nanctemp-1,2);  matrixB=zeros(Nanctemp-1,1);
            elemB=(Pnode(anchors(Nanctemp),1:2))*(Pnode(anchors(Nanctemp),1:2))'-dis_n_a(Nanctemp)^2;
                for matcount=1:(Nanctemp-1)
              matrixA(matcount,:)=-2*[Pnode(anchors(matcount),1:2)-Pnode(anchors(Nanctemp),1:2)];
                    matrixB(matcount,:)=dis_n_a(matcount)^2-
(Pnode(anchors(matcount),1:2))*(Pnode(anchors(matcount),1:2))'+elemB;
                end
                Pnode(Ncount,5:6)=(inv(matrixA'*matrixA)*matrixA'*matrixB)';
          errtemp=sqrt((Pnode(Ncount,5:6)-Pnode(Ncount,1:2))*(Pnode(Ncount,5:6)-Pnode(Ncount,1:2))');
                locnum(countrat,4)=locnum(countrat,4)+1;
                locerr(countrat,4)=locerr(countrat,4)+errtemp;
                %Checkout DV-hop
                %%%%%% find the nearest anchor
                nearthree=zeros(1,3); %indices of the nearest anchors
                hoptemp=Pnode(Ncount, 13:(12+Nanctemp)); %hop counts from normal node to anchors
                sorttemp=sort(hoptemp);
                findtemp=find(Pnode(Ncount, 13:(12+Nanctemp))==sorttemp(1));
                nearthree(1)=findtemp(1);
                %%%%%%%%%%%%%%%%%%
                currentneardis=sqrt((Pnode(Ncount,3:4)-
Pnode(anchors(nearthree(1)),1:2))*(Pnode(Ncount,3:4)-Pnode(anchors(nearthree(1)),1:2))');
                originneardis=Pnode(Ncount,12+nearthree(1));
                difference=currentneardis-originneardis;
                Pnode(Ncount,5:6)=Pnode(Ncount,3:4)-
difference/currentneardis*(Pnode(Ncount,3:4)-Pnode(anchors(nearthree(1)),1:2));
          errtemp=sqrt((Pnode(Ncount,5:6)-Pnode(Ncount,1:2))*(Pnode(Ncount,5:6)-Pnode(Ncount,1:2))');
                locnum(countrat,5)=locnum(countrat,5)+1;
                locerr(countrat,5)=locerr(countrat,5)+errtemp;
                end
            end
        end
    end
    end
    locerr(countrat,1:6)=locerr(countrat,1:6)./locnum(countrat,1:6);
    locerr(countrat,1:6)=locerr(countrat,1:6)/Crange*100;
end
```

Annex 3 : MATLAB Source Code for Class-1 Algorithms (Centroid, CPE, Mid-perpendicular)

```
%Set Parameters
Hsize = 40;   % Size of one side in square area
Crange = 20;   % communication range
Dintl = 0.5; % Distance of interval of one side
Nintl = Hsize / Dintl;  % divide the Hsize into Nintl intervals
Narea = Nintl^2;  % Narea: number of intervals in the total area
Rnum=2000000;   % times of simulations temporary
mindist = 10;
Nanc=5; % number of anchors
Pnom = [20, 20];   % true position of normal node
locerr = zeros(1,4);   % location error, 4 algos for class 1
locnum = 0; result1=[]; result2=[]; result3=[]; result4=[];
Rcount=1;  % initial: number of simulations
while Rcount < Rnum
    Panc = zeros(Nanc,2);
    %intialization: Nanc anchors randomly distruted
    nodechosen=[1:Narea];
    Rseq = randint(Nanc,1,[1,Narea]);
     % Nanc random numbers between [1:Narea],but not surely different from each other
    for seqcount=1:Nanc  %exchange, because may exist two same numbers in Rseq
         nodetemp=nodechosen(seqcount);
         nodechosen(seqcount)=nodechosen(Rseq(seqcount));
         nodechosen(Rseq(seqcount))=nodetemp;
    end
    Rseq=nodechosen(1:Nanc)';  %now sure that any two numbers in Rseq are different
    Rseq = sort(Rseq);
    Panc(:,1)=(0.5+floor((Rseq-1)/Nintl))*Dintl;
    Panc(:,2)=(0.5+mod(Rseq-1,Nintl))*Dintl;
  %whether the normal node has 3 neighbor anchors
  onehopnum =0;
  for Ncount=1:Nanc
      disanctemp = sqrt((Pnom-Panc(Ncount,1:2))*(Pnom-Panc(Ncount,1:2))');
            if disanctemp <= Crange
                onehopnum=onehopnum+1;
            end
  end
  onehopcount1=1;
  while onehopcount1 < Nanc
       ancpos1=Panc(onehopcount1,1:2);
       onehopcount2 = onehopcount1 + 1;
       while onehopcount2 <= Nanc
           ancpos2=Panc(onehopcount2,1:2);
           distanc1to2=sqrt((ancpos1-ancpos2)*(ancpos1-ancpos2)');
           if distanc1to2 < mindist
               onehopnum = 0;
               onehopcount2=Nanc+1;
               onehopcount1=Nanc+1;
           end
           onehopcount2 = onehopcount2 + 1;
       end
       onehopcount1 = onehopcount1 + 1;
  end
   %%%%%% performe all algorithms
        %first need to detect whether there are Nanc neighbour anchors
        if onehopnum==Nanc
            locnum=locnum+1;
            %%%%%%%%%performe Centroid
            Pnomtemp=mean(Panc,1);
            errtemp=sqrt((Pnomtemp-Pnom)*(Pnomtemp-Pnom)');  errtemp1=errtemp;
            locerr(1,1)=locerr(1,1)+errtemp;
            result1=[result1;errtemp/Crange*100];
            %%%%%%%%%performe CPE
            CPEleft=max(Panc(:,1));  CPEright=min(Panc(:,1));
            CPEup=min((Panc(:,2)));    CPEdown=max((Panc(:,2)));
            Pnomtemp=[(CPEleft+CPEright)/2, (CPEup+CPEdown)/2];
            errtemp=sqrt((Pnomtemp-Pnom)*(Pnomtemp-Pnom)'); errtemp2=errtemp;
            locerr(1,2)=locerr(1,2)+errtemp;
            result2=[result2;errtemp/Crange*100];
            %%%%%%%%% performe direct mid-perpendicular
            allresult=[];
            for onehopcount1=1:(onehopnum-2)
                ancpos1=Panc(onehopcount1,1:2);
                for onehopcount2=(onehopcount1+1):(onehopnum-1)
                    ancpos2=Panc(onehopcount2,1:2);
                    for onehopcount3=(onehopcount2+1):onehopnum
                        ancpos3=Panc(onehopcount3,1:2);
                 xa=ancpos1(1);xb=ancpos2(1);xc=ancpos3(1);ya=ancpos1(2);yb=ancpos2(2);yc=ancpos3(2);
```

```
                        dist1=sqrt((ancpos1-ancpos2)*(ancpos1-ancpos2)');
                        dist2=sqrt((ancpos2-ancpos3)*(ancpos2-ancpos3)');
                        dist3=sqrt((ancpos3-ancpos1)*(ancpos3-ancpos1)');
                        if  dist1^2>=dist2^2+dist3^2
                            resultemp=mean([ancpos1;ancpos2],1); % temporary result for position
                        elseif dist2^2>=dist1^2+dist3^2
                            resultemp=mean([ancpos2;ancpos3],1);
                        elseif dist3^2>=dist1^2+dist2^2
                            resultemp=mean([ancpos3;ancpos1],1);
                        else
                            resultemp(1)=((xa^2-xb^2)*(yc-ya)+(xa^2-xc^2)*(ya-yb)+(ya-yb)*(yb-
yc)*(yc-ya))/((xa-xb)*yc+(xc-xa)*yb+(xb-xc)*ya)/2;
                            resultemp(2)=((ya^2-yb^2)*(xc-xa)+(ya^2-yc^2)*(xa-xb)+(xa-xb)*(xb-
xc)*(xc-xa))/((ya-yb)*xc+(yc-ya)*xb+(yb-yc)*xa)/2;
                        end
                        allresult=[allresult; resultemp];
                    end
                end
            end
            Pnomtemp=mean(allresult,1);
            errtemp=sqrt((Pnomtemp-Pnom)*(Pnomtemp-Pnom)');
            errtemp3=errtemp;
            locerr(1,3)=locerr(1,3)+errtemp;
            result3=[result3;errtemp/Crange*100];
            %%%%%%%%% performe simplified mid-perpendicular
            longanc=zeros(1,3);  % longest line connecting any two anchors
            for onehopcount1=1:(onehopnum-1)
                ancpos1=Panc(onehopcount1,1:2);
                for onehopcount2=(onehopcount1+1):onehopnum
                    ancpos2=Panc(onehopcount2,1:2);
                    distanc1to2=sqrt((ancpos1-ancpos2)*(ancpos1-ancpos2)');
                    if distanc1to2 > longanc(3)
                       longanc(3)=distanc1to2;
                       longanc(1)=onehopcount1;
                       longanc(2)=onehopcount2;
                    end
                end
            end
            ancpos2=Panc(longanc(1),1:2);
            ancpos3=Panc(longanc(2),1:2);
            dist2=longanc(3);
            xb=ancpos2(1); xc=ancpos3(1);  yb=ancpos2(2); yc=ancpos3(2);
            longanc(3)=0;
            for onehopcount1=1:onehopnum
                if onehopcount1 ~= longanc(1) && onehopcount1 ~= longanc(2)
                    ancpos1=Panc(onehopcount1,1:2);
                    distemp = abs(ancpos1(1)*(yb-yc)+ancpos1(2)*(xc-xb)+yc*(xb-xc)+xc*(yc-yb))/dist2;
                    if distemp > longanc(3)
                        longanc(3)=distemp;
                        anctemp = onehopcount1;
                    end
                end
            end
            ancpos1=Panc(anctemp,1:2);
            xa=ancpos1(1); ya=ancpos1(2);
            dist1=sqrt((ancpos1-ancpos2)*(ancpos1-ancpos2)');
            dist3=sqrt((ancpos3-ancpos1)*(ancpos3-ancpos1)');
            if dist2^2>=dist1^2+dist3^2
                resultemp=mean([ancpos2;ancpos3],1);
            else
                resultemp(1)=((xa^2-xb^2)*(yc-ya)+(xa^2-xc^2)*(ya-yb)+(ya-yb)*(yb-yc)*(yc-ya))/((xa-
xb)*yc+(xc-xa)*yb+(xb-xc)*ya)/2;
                resultemp(2)=((ya^2-yb^2)*(xc-xa)+(ya^2-yc^2)*(xa-xb)+(xa-xb)*(xb-xc)*(xc-xa))/((ya-
yb)*xc+(yc-ya)*xb+(yb-yc)*xa)/2;
            end
            Pnomtemp=resultemp;
            errtemp=sqrt((Pnomtemp-Pnom)*(Pnomtemp-Pnom)');
            errtemp4=errtemp;
            locerr(1,4)=locerr(1,4)+errtemp;
            result4=[result4;errtemp/Crange*100];
        end
        Rcount = Rcount +1;
        if locnum == 5000
            Rcount = Rnum +1;
        end
end
if locnum~=0
    locerr=locerr/locnum;
    locerr=locerr/Crange*100;  % percentage of radio range
end
```

Annex 4 : WSNet Source Code for DV-hop Protocol

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <stdlib.h>
#include <include/modelutils.h>
/* Defining module informations*/
model_t model =  {
    "localisationclasse2mobile",
    "Linqing GUI",
    "0.1",
    MODELTYPE_APPLICATION,
    {NULL, 0}
};
/* Defining node type */
#define NORMAL 0
#define ANCHOR 1
/* Node private data */
struct nodedata {
  int *overhead;  int type;   uint64_t birth;  int seq;  uint64_t period;
  /* for stats */
  int packet_tx;   int packet_rx;  uint64_t timecurrent;  uint64_t timenext;
  int tasknext;  uint64_t timemaxattentdhp;  uint64_t timemaxattentpos;
   int numhop; int numdhp; int numdhphop; int numancretotal;
   // the default number of maximum accepted anchors
   int hopcnt[30]; int idanchop[30]; int idancdhp[20]; int indicehop[20]; int indicedhp[20];
  // totally can obtain 30 anchors at step 1 and 20 anchors at step 2
  float xancre[30]; float yancre[30]; float dhp[20]; float pos_x; float pos_y; int range;
};
/* Data header */
struct packet_header {
  int  source;  int  seq;
};
// data payload
struct datainpacket {
  int typeofnode; int hopcnt; float pos_x; float pos_y;
};
int callmeback(call_t *c, void *args);
void rx(call_t *c, packet_t *packet);
int init(call_t *c) { return 0; }
int destroy(call_t *c) {
    return 0;
}
/* Here we read the node variables from the xml config file*/
int setnode(call_t *c, void *params) {
    struct nodedata *nodedata = malloc(sizeof(struct nodedata));
    int i = get_entity_links_down_nbr(c);
    param_t *param;
    /* default values */
    nodedata->period=1000000000; nodedata->type=NORMAL; nodedata->birth=0;
    nodedata->seq=0; nodedata->packet_tx = 0; nodedata->packet_rx = 0;
    nodedata->numhop = 0; nodedata->numdhp = 0; nodedata->timecurrent = 0;
    nodedata->tasknext = 0;
    //0 initialization, 1 diffuse-position(anchor) or calculation(normal node), 2 diffuse dhp(anchor)
    nodedata->numancretotal = 20;
    nodedata->range = 20;
    /* reading the "default" markup from the xml config file */
    das_init_traverse(params);
    while ((param = (param_t *) das_traverse(params)) != NULL) {
        if (!strcmp(param->key, "type")) {
            if (get_param_integer(param->value, &(nodedata->type))) {
                goto error;
            }
        }
        if (!strcmp(param->key, "period")) {
            if (get_param_time(param->value, &(nodedata->period))) {
                goto error;
            }
        }
         if (!strcmp(param->key, "birth")) {
            if (get_param_integer(param->value, &(nodedata->birth))) {
                goto error;
            }
        }
    }
    /* alloc overhead memory */
    if (i) {
        nodedata->overhead = malloc(sizeof(int) * i);
    } else {
        nodedata->overhead = NULL;
```

```
    }
    set_node_private_data(c, nodedata);
    nodedata->timemaxattentdhp = nodedata->period/2;
    nodedata->timemaxattentpos = nodedata->period/8*7;
    nodedata->timecurrent = nodedata->birth;
    int cntemp=0;
    for (cntemp=0;cntemp<30;cntemp++)
         nodedata->hopcnt[cntemp] = -1;
    for (cntemp=0; cntemp<20; cntemp++ )
         nodedata->indicedhp[cntemp] = -1;
    if (c->node==0){
         FILE *posfile; posfile = fopen("position_results_classe2mobile.txt", "w");
         if (posfile == NULL)  { printf("Error! Problem occurs when creating the result file!\n"); }
         else { fclose(posfile); }
     }
    FILE *posfile;  posfile=fopen("position_results_classe2mobile.txt", "a");
    if (posfile!= NULL) {
 fprintf(posfile,"%s","position of ");fprintf(posfile, "%d", c->node);fprintf(posfile, "%s", " is ");
 fprintf(posfile,"%lf",get_node_position(c->node)->x); fprintf(posfile, "%s", " ");
 fprintf(posfile,"%lf",get_node_position(c->node)->y); fprintf(posfile, "%s", "\n"); fclose(posfile);
      }
    return 0;
 error:
    free(nodedata);
    return -1;
}

int unsetnode(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    /* we print number of transmitted frames before exit */
    if (nodedata->packet_tx > 0 || nodedata->packet_rx > 0) {
      if (nodedata->packet_tx > 0) {
           FILE *posfile;  posfile=fopen("position_results_classe2mobile.txt", "a");
           if (posfile!= NULL)
           {fprintf(posfile,"%d",nodedata->packet_tx); fprintf(posfile, "%s", " "); fclose(posfile);}
        }
    }
    if (nodedata->overhead) {
        free(nodedata->overhead);
    }
    free(nodedata);
    return 0;
}

/* ************************************************** */
int bootstrap(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    int i = get_entity_links_down_nbr(c);
    entityid_t *down = get_entity_links_down(c);
  while (i--) {
        call_t c0 = {down[i], c->node};
        if ((get_entity_type(&c0) != MODELTYPE_ROUTING)
            && (get_entity_type(&c0) != MODELTYPE_MAC)) {
            nodedata->overhead[i] = 0;
        } else {
            nodedata->overhead[i] = GET_HEADER_SIZE(&c0);
             // printf("overhead size is %d\n", sizeof(nodedata->overhead[i]));
         }
    }
    nodedata->timecurrent = get_time();
    /* we schedule a new callback */
    scheduler_add_callback(nodedata->timecurrent, c, callmeback, NULL);
    return 0;
}
int ioctl(call_t *c, int option, void *in, void **out) {
    return 0;
}
int callmeback(call_t *c, void *args) {
    struct nodedata *nodedata = get_node_private_data(c);
    entityid_t *down = get_entity_links_down(c);   call_t c0 = {down[0], c->node};
    destination_t destination = {BROADCAST_ADDR, {-1, -1, -1}};
    packet_t *packet = packet_alloc(c, nodedata->overhead[0] + sizeof(struct
packet_header)+sizeof(struct datainpacket));
    struct packet_header *header = (struct packet_header *) (packet->data + nodedata->overhead[0]);
    struct datainpacket *datapacket = (struct datainpacket *) (packet->data + nodedata->overhead[0]
+ sizeof(struct packet_header));
    int timepass = ( get_time() - nodedata->timecurrent )/1000000; //time between two callmeback
    int cntemp=0; int cntemp1; int sumhopcnt; float dhptemp;
    if ( timepass == 0  ) {     //beginning of a period
         if (nodedata->type == 1) { // for anchor, next step is broadcasting position
              srand(time(NULL)+c->node*10);    // first-numbered nodes set to anchors
```

```
            nodedata->timenext = nodedata->timecurrent+ 500000000/(5*nodedata->numancretotal)*(rand()%(5*nodedata->numancretotal)+1); /*max is 500ms*/
        }
        else {  // for normal node, next step is calculating its position by DV-hop
            nodedata->timenext = nodedata->timecurrent + nodedata->timemaxattentpos;
        }
        nodedata->tasknext = 1;
        scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
    }
    if ( get_time() == nodedata->timenext && nodedata->tasknext == 1 ) {
     // this step, for anchors broadcast pos, for normal nodes calculate pos
        if ( nodedata->type == 1) {
        nodedata->pos_x=get_node_position(c->node)->x;  nodedata->pos_y=get_node_position(c->node)->y;
        header->source=c->node; header->seq=++nodedata->seq; datapacket->typeofnode=nodedata->type;
        datapacket->hopcnt=0; datapacket->pos_x=nodedata->pos_x; datapacket->pos_y=nodedata->pos_y;
             if (SET_HEADER(&c0, packet, &destination) == -1) {
                   packet_dealloc(packet);
                  return -1;
               }
             TX(&c0, packet);
             /* for stats */
             nodedata->packet_tx++; nodedata->last_seq_pos++;
            if (nodedata->numhop == 30-1) {
//if this anchor already gather all other anchors' positions,wait then diffuse dhpi
              srand(time(NULL)+c->node*10);
              uint64_t timedelay = (rand()%100+1)*5000000; //wait random time
              nodedata->timenext = nodedata->timenext + timedelay;
            }
            else { //anchor not received all others' positions, wait attentemaxdhp then diffuse dhpi
              nodedata->timenext = nodedata->timecurrent + nodedata->timemaxattentdhp;
            }
            nodedata->tasknext = 2;
            scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
        }
        else {
            if(nodedata->numdhp <= nodedata->numancretotal && nodedata->numdhp >= 3){
                    // it is time for the position calculation
                    //first find those hopcount and dhp who are from the same anchors
                     nodedata->numdhphop = 0;
                    for ( cntemp=0; cntemp<nodedata->numdhp; cntemp++ ) {
                        for ( cntemp1=0; cntemp1<nodedata->numhop; cntemp1++ ) {
                            if ( nodedata->idanchop[cntemp1] == nodedata->idancdhp[cntemp] ) {
                                 nodedata->indicehop[nodedata->numdhphop] = cntemp1;
                                 nodedata->indicedhp[nodedata->numdhphop] = cntemp;
                                nodedata->numdhphop++;
                           cntemp1=nodedata->numhop+1; //go out of second loop, already find hopcount
                            }
                        }
                    }
                    if ( nodedata->numdhphop >= 3 ) {
                       float distemp[20]; float matAx[20]; float matAy[20];
                        float matB[20]; float matAA[4]; float abstemp;
                       for ( cntemp=0; cntemp<nodedata->numdhphop; cntemp++ )
                            distemp[cntemp] = nodedata->dhp[nodedata->indicedhp[cntemp]] * nodedata->hopcnt[nodedata->indicehop[cntemp]];
                       for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                            matAx[cntemp] = (nodedata->xancre[nodedata->indicehop[cntemp]]-nodedata->xancre[nodedata->indicehop[nodedata->numdhphop-1]])*(-2);
                            matAy[cntemp] = (nodedata->yancre[nodedata->indicehop[cntemp]]-nodedata->yancre[nodedata->indicehop[nodedata->numdhphop-1]])*(-2);
                            matB[cntemp] = distemp[cntemp]*distemp[cntemp] - distemp[nodedata->numdhphop-1]*distemp[nodedata->numdhphop-1] - nodedata->xancre[nodedata->indicehop[cntemp]]*nodedata->xancre[nodedata->indicehop[cntemp]] + nodedata->xancre[nodedata->indicehop[nodedata->numdhphop-1]]*nodedata->xancre[nodedata->indicehop[nodedata->numdhphop-1]] -
nodedata->yancre[nodedata->indicehop[cntemp]]*nodedata->yancre[nodedata->indicehop[cntemp]] +
nodedata->yancre[nodedata->indicehop[nodedata->numdhphop-1]]*nodedata->yancre[nodedata->indicehop[nodedata->numdhphop-1]];
                        }
                       matAA[0]=0; matAA[1]=0; matAA[2]=0; matAA[3]=0;
                       for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                            matAA[0] = matAA[0] + matAy[cntemp]*matAy[cntemp];
                            matAA[1] = matAA[1] + matAx[cntemp]*matAy[cntemp];
                            matAA[3] = matAA[3] + matAx[cntemp]*matAx[cntemp];
                        }
                       abstemp = matAA[0]*matAA[3] - matAA[1]*matAA[1];
                       matAA[0]= matAA[0]/abstemp; matAA[3]= matAA[3]/abstemp;
                        matAA[1]= matAA[1]/abstemp*(-1); matAA[2] = matAA[1];   //matAA=(A'A)-1
                       for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                            matAA[2] = matAA[0]*matAx[cntemp] + matAA[1]*matAy[cntemp];
                           matAy[cntemp] = matAA[3]*matAy[cntemp] + matAA[1]*matAx[cntemp];
                           matAx[cntemp] = matAA[2];
                       }
```

```
                        nodedata->pos_x = 0; nodedata->pos_y = 0;
                        for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                             nodedata->pos_x = nodedata->pos_x + matAx[cntemp]*matB[cntemp];
                             nodedata->pos_y = nodedata->pos_y + matAy[cntemp]*matB[cntemp];
                         }
                        float pos_err = sqrt((nodedata->pos_x - get_node_position(c->node)-
>x)*(nodedata->pos_x - get_node_position(c->node)->x) + (nodedata->pos_y - get_node_position(c-
>node)->y)*(nodedata->pos_y - get_node_position(c->node)->y));
                       //checkout DV-hop
                        float pos_x_chec; float pos_y_chec; float mindistemp=100000; float dis_dvhop;
                         float pos_err_chec; int indice_nearanc;
                        for ( cntemp1=0; cntemp1<nodedata->numdhphop; cntemp1++) {
                          if ( distemp[cntemp1] < mindistemp ) {
                           mindistemp=distemp[cntemp1]; indice_nearanc=nodedata->indicehop[cntemp1]; }
                        }
                        dis_dvhop = sqrt((nodedata->pos_x - nodedata-
>xancre[indice_nearanc])*(nodedata->pos_x - nodedata->xancre[indice_nearanc]) + (nodedata->pos_y -
nodedata->yancre[indice_nearanc])*(nodedata->pos_y - nodedata->yancre[indice_nearanc]));
                        pos_x_chec = mindistemp/dis_dvhop*(nodedata->pos_x-nodedata-
>xancre[indice_nearanc])+nodedata->xancre[indice_nearanc];
                        pos_y_chec = mindistemp/dis_dvhop*(nodedata->pos_y-nodedata-
>yancre[indice_nearanc])+nodedata->yancre[indice_nearanc];
                        pos_err_chec = sqrt((pos_x_chec - get_node_position(c->node)->x)*(pos_x_chec -
get_node_position(c->node)->x) + (pos_y_chec - get_node_position(c->node)->y)*(pos_y_chec -
get_node_position(c->node)->y));
                       // Selective DV-hop
                        float pos_x_sel; float pos_y_sel;float pos_err_sel; float pos_xtmp;
                         float pos_ytmp; float mindiferr_hopcnt=50000;
                         int indice_nearanc_sel=-1;  int cntemp2; int cntemp3; int cntemp4;
                         for ( cntemp1=0; cntemp1<nodedata->numdhphop-2; cntemp1++) {
                             for ( cntemp2=cntemp1+1; cntemp2<nodedata->numdhphop-1; cntemp2++) {
                                   for ( cntemp3=cntemp2+1; cntemp3<nodedata->numdhphop; cntemp3++)  {
                                        // first obtain the estimated position by this 3-anchors group
                                        matAA[0] = (nodedata->xancre[nodedata->indicehop[cntemp1]]-
nodedata->xancre[nodedata->indicehop[cntemp3]])*(-2);
                                        matAA[2] = (nodedata->xancre[nodedata->indicehop[cntemp2]]-
nodedata->xancre[nodedata->indicehop[cntemp3]])*(-2);
                                        matAA[1] = (nodedata->yancre[nodedata->indicehop[cntemp1]]-
nodedata->yancre[nodedata->indicehop[cntemp3]])*(-2);
                                        matAA[3] = (nodedata->yancre[nodedata->indicehop[cntemp2]]-
nodedata->yancre[nodedata->indicehop[cntemp3]])*(-2);
                                        matB[0] = distemp[cntemp1]*distemp[cntemp1] -
distemp[cntemp3]*distemp[cntemp3] - nodedata->xancre[nodedata->indicehop[cntemp1]]*nodedata-
>xancre[nodedata->indicehop[cntemp1]] + nodedata->xancre[nodedata->indicehop[cntemp3]]*nodedata-
>xancre[nodedata->indicehop[cntemp3]] - nodedata->yancre[nodedata->indicehop[cntemp1]]*nodedata-
>yancre[nodedata->indicehop[cntemp1]] + nodedata->yancre[nodedata->indicehop[cntemp3]]*nodedata-
>yancre[nodedata->indicehop[cntemp3]];
                                        matB[1] = distemp[cntemp2]*distemp[cntemp2] -
distemp[cntemp3]*distemp[cntemp3] - nodedata->xancre[nodedata->indicehop[cntemp2]]*nodedata-
>xancre[nodedata->indicehop[cntemp2]] + nodedata->xancre[nodedata->indicehop[cntemp3]]*nodedata-
>xancre[nodedata->indicehop[cntemp3]] - nodedata->yancre[nodedata->indicehop[cntemp2]]*nodedata-
>yancre[nodedata->indicehop[cntemp2]] + nodedata->yancre[nodedata->indicehop[cntemp3]]*nodedata-
>yancre[nodedata->indicehop[cntemp3]];
                                        abstemp = matAA[0]*matAA[3] - matAA[1]*matAA[2];
                                        if ( abstemp > 0.000001 )   {
                                            pos_xtmp = (matAA[3]*matB[0]-matAA[1]*matB[1])/abstemp;
                                            pos_ytmp = (matAA[0]*matB[1]-matAA[2]*matB[0])/abstemp;
                                            //find nearest anchor to this 3-anchor estimated position
                                            indice_nearanc_sel = -1; matAA[0] = 1000;
                                               //matAA[0] recycled to be used as the reference distance
                                            for ( cntemp4=0; cntemp4<nodedata->numdhphop; cntemp4++ ) {
               matAA[1] = sqrt((pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp
- nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]]));
                                      // here matAA[1] is recycled to be used as the temporary distance
                                                 if ( cntemp4==0 )
                                                     matAA[0] = matAA[1]+1;
                                                 if ( matAA[1]<matAA[0] ) {
                                                      matAA[0] = matAA[1];
                                                      if ( matAA[1]<nodedata->range )
                                                          indice_nearanc_sel = cntemp4;
                                                  }
                                             }
                                            matAA[2] =0;  //matAA[2] used as tempoary hop-count diff
                                            for ( cntemp4=0; cntemp4<nodedata->numdhphop; cntemp4++ ) {
                                               if(indice_nearanc_sel>-1){ //nearest anchor within 1 hop
                                                    if ( cntemp4 != indice_nearanc_sel )  {
                                                     if ( matAA[0] <= nodedata->range/2 )  // half hop
                                                                  matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp
- nodedata->xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp - nodedata->xancre[nodedata-
>indicehop[cntemp4]]) + (pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]])*(pos_ytmp -
```

```
nodedata->yancre[nodedata->indicehop[cntemp4]])) / nodedata->dhp[nodedata-
>indicedhp[indice_nearanc_sel]] - nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                                                            else

matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp -
nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]])) / (nodedata-
>dhp[nodedata->indicedhp[indice_nearanc_sel]]+nodedata->dhp[nodedata->indicedhp[cntemp4]])*2 -
nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                                                    }
                                                    else
                                                         matAA[2]=matAA[2]+abs(1-nodedata-
>hopcnt[nodedata->indicehop[cntemp4]]);
                                                }
                                               else // there is no one-hop anchor
matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp -
nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]])) / nodedata-
>dhp[nodedata->indicedhp[cntemp4]] - nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                                           }
                                           if ( matAA[2] < mindiferr_hopcnt )  {
                                               pos_x_sel = pos_xtmp;  pos_y_sel = pos_ytmp;
                                               mindiferr_hopcnt = matAA[2];
                        }    }   }   }   }
                        if ( mindiferr_hopcnt == 50000 ) {
                           pos_x_sel = nodedata->pos_x; pos_y_sel = nodedata->pos_y;
                        }
                       pos_err_sel = sqrt((pos_x_sel - get_node_position(c->node)->x)*(pos_x_sel -
get_node_position(c->node)->x) + (pos_y_sel - get_node_position(c->node)->y)*(pos_y_sel -
get_node_position(c->node)->y));
                       FILE *posfile;  posfile=fopen("position_results_classe2mobile.txt", "a");
                       if (posfile!= NULL)
                       { fprintf(posfile, "%d", c->node);fprintf(posfile, "%s", " ");
                         fprintf(posfile, "%d", nodedata->numdhp);fprintf(posfile, "%s", " ");
                         fprintf(posfile, "%lf", pos_err); fprintf(posfile, "%s", " ");
                         fprintf(posfile, "%lf", pos_err_chec); fprintf(posfile, "%s", " ");
                         fprintf(posfile, "%lf", pos_err_sel); fprintf(posfile, "%s", "\n");
                         fclose(posfile);
                       }
               else
               {printf("Error! Problem on adding Centroid result to the file!\n"); return 0;}
             }
           }
            // every node inializes those nodedata parameters about the DV-hop
            nodedata->numhop = 0;
             nodedata->numdhp = 0;
             for (cntemp=0;cntemp<30;cntemp++)
                 nodedata->hopcnt[cntemp] = -1;
             for (cntemp=0; cntemp<nodedata->numancretotal; cntemp++ )
                  nodedata->indicedhp[cntemp] = -1;
             nodedata->timenext = nodedata->timecurrent + nodedata->period;
             nodedata->timecurrent = nodedata->timecurrent + nodedata->period;
             nodedata->tasknext = 0;
             scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
        }
    }
     if ( get_time() == nodedata->timenext && nodedata->tasknext == 2 ) {
     // this step, for anchors broadcast dhp
        if ( nodedata->timenext == nodedata->timecurrent + nodedata->timemaxattentdhp ) {
            srand(time(NULL)+c->node*10);
            uint64_t timedelay = (rand()%100+1)*5000000;  //random wait time
           nodedata->timenext = nodedata->timenext + timedelay;
            scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
        }
        else {
           if ( nodedata->numhop <= 30 && nodedata->numhop > 0 ) {
                 // calculate distance per hop
                 dhptemp = 0;  int numhoptmp;
                numhoptmp = nodedata->numhop;
                 for (cntemp=0;cntemp<numhoptmp;cntemp++)
                        dhptemp = dhptemp + sqrt((nodedata->pos_x - nodedata->xancre[cntemp]) *
(nodedata->pos_x - nodedata->xancre[cntemp]) + (nodedata->pos_y - nodedata->yancre[cntemp]) *
(nodedata->pos_y - nodedata->yancre[cntemp]));
                 sumhopcnt = 0;
                 for (cntemp=0;cntemp<numhoptmp;cntemp++)
                        sumhopcnt = sumhopcnt + nodedata->hopcnt[cntemp];
                 nodedata->dhp[0] = dhptemp/sumhopcnt;
                 // diffuse this distance per hop
                 header->source = c->node;
                 header->seq    = ++nodedata->seq;
                 datapacket->typeofnode = nodedata->type;
                 datapacket->hopcnt = -10;
```

```
                datapacket->pos_x = nodedata->dhp[0]; datapacket->pos_y = -1;
                if (SET_HEADER(&c0, packet, &destination) == -1) {
                       packet_dealloc(packet);
                       return -1;
                }
                TX(&c0, packet);
                /* for stats */
                nodedata->packet_tx++; nodedata->last_seq_dhp++;
          }
        // every node inializes those nodedata parameters about the DV-hop
        nodedata->numhop = 0;   nodedata->numdhp = 0;
         for (cntemp=0;cntemp<30;cntemp++)
              nodedata->hopcnt[cntemp] = -1;
         for (cntemp=0; cntemp<nodedata->numancretotal; cntemp++ )
               nodedata->indicedhp[cntemp] = -1;
       nodedata->timenext = nodedata->timecurrent + nodedata->period;
       nodedata->timecurrent = nodedata->timecurrent + nodedata->period;
       nodedata->tasknext = 0;
       scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
       }
    }
    return 0;
}

void rx(call_t *c, packet_t *packet) {    // receive frames
    struct nodedata *nodedata = get_node_private_data(c);
    struct packet_header *header = (struct packet_header *) (packet->data + nodedata->overhead[0]);
    struct datainpacket *datapacket = (struct datainpacket *) (packet->data + nodedata->overhead[0]
+ sizeof(struct packet_header));
    nodedata->packet_rx++;
    entityid_t *down = get_entity_links_down(c);
    call_t c0 = {down[0], c->node};
    int cntemp=0; int signforw;
    if ( datapacket->typeofnode == 1 && header->source != c->node) {
     // if the sender is an anchor and the receiver is not the sender itself
         signforw = 0;
         if ( datapacket->hopcnt >= 0 && nodedata->numhop<30 ) {
        //this packet is for broadcasting the position and hop counts
              if ( nodedata->numhop == 0) {
                   nodedata->hopcnt[0] = datapacket->hopcnt + 1;
                   nodedata->idanchop[0] = header->source;
                   nodedata->xancre[0] = datapacket->pos_x;
                   nodedata->yancre[0] = datapacket->pos_y;
                   nodedata->numhop++;
                   signforw = 1;  // to forward this packet
               }
              else {
                   for ( cntemp=0; cntemp<nodedata->numhop; cntemp++ ) {
                     if (cntemp==nodedata->numhop-1 && nodedata->idanchop[cntemp]!= header->source){
                       // a new anchor for this node
                             nodedata->numhop=nodedata->numhop+1;
                             nodedata->hopcnt[nodedata->numhop-1]=datapacket->hopcnt+1;
                             nodedata->idanchop[nodedata->numhop-1] = header->source;
                              nodedata->xancre[nodedata->numhop-1] = datapacket->pos_x;
                              nodedata->yancre[nodedata->numhop-1] = datapacket->pos_y;
                              signforw = 1;
                              cntemp = nodedata->numhop + 1; //no need search any more              }
                         if ( nodedata->idanchop[cntemp] == header->source ) {
                              if ( nodedata->hopcnt[cntemp] > datapacket->hopcnt + 1 ) {
                                   nodedata->hopcnt[cntemp] = datapacket->hopcnt + 1;
                                   signforw = 1;
                               }
                              cntemp = nodedata->numhop + 1; //no need search any more
                         }
                    }
               }
              if ( signforw == 1 ) {
                   signforw = 0;
                   datapacket->hopcnt = datapacket->hopcnt + 1;
                   srand(c->node*10);
                   uint64_t timedelay = (rand()%100+1)*500000;  //random wait time
                   unsigned long usecsleep = timedelay/1000;
                   usleep(usecsleep);
                   TX(&c0, packet);
                  /* for stats */
                  nodedata->packet_tx++;
               }
              if ( nodedata->type == 1 && nodedata->numhop == 30 && nodedata->tasknext == 2 ) {
                   srand(c->node*10);
                   uint64_t timedelay = (rand()%100+1)*5000000;  //random wait time,
nodedata->timenext = get_time() + timedelay;
                   scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
```

```
            }
        }
        if ( datapacket->hopcnt < -5 && nodedata->type == 1 ) {
       // when dhp packet is received by another anchor
             signforw = 1;
             if ( nodedata->numdhp > 0 ) {
            // search whether this dhp has already restored in the database
                   int cntemp1;
                   for ( cntemp1=0; cntemp1<nodedata->numdhp; cntemp1++ ) {
                         if ( nodedata->indicedhp[cntemp1] == header->source ) {
                     // in this case, indicedhp == id of anchors with dhp
                              signforw= 0;  cntemp1 = nodedata->numdhp + 2;
                          }
                    }
              }
             if ( signforw == 1 && nodedata->numdhp < nodedata->numancretotal) {
                  nodedata->numdhp++;
                  nodedata->indicedhp[nodedata->numdhp-1] = header->source;
                  uint64_t timedelay = (rand()%100+1)*500000; //random wait time
unsigned long usecsleep = timedelay/1000;
                  usleep(usecsleep);
                  TX(&c0, packet);
                 /* for stats */
                 nodedata->packet_tx++;
             }
        }

        if ( datapacket->hopcnt < -5 && nodedata->type == 0 ) {
       // when dhp packet is received by a normal node
             int cntemp1; int signnotedhp = 1;
             if ( nodedata->numdhp > 0 ) {
                  for ( cntemp=0; cntemp<nodedata->numdhp; cntemp++ ) {
                      if ( nodedata->idancdhp[cntemp] == header->source ) {
                             signnotedhp = 0;
                             cntemp = nodedata->numdhp + 1;
                        }
                   }
              }
             if ( signnotedhp == 1 && nodedata->numdhp < nodedata->numancretotal) {
                  //nodedata->last_seq = header->seq;
                  nodedata->numdhp++;
                  nodedata->idancdhp[nodedata->numdhp-1] = header->source;
                  nodedata->dhp[nodedata->numdhp-1] = datapacket->pos_x;
                  uint64_t timedelay = (rand()%100+1)*500000;
                  unsigned long usecsleep = timedelay/1000;
                  usleep(usecsleep);
                  TX(&c0, packet);
                 /* for stats */
                 nodedata->packet_tx++;
if(nodedata->numdhp==nodedata->numancretotal && nodedata->numhop==30 && nodedata->tasknext==1){
                         nodedata->numdhphop = 0;
                        for ( cntemp=0; cntemp<nodedata->numdhp; cntemp++ ) {
                            for ( cntemp1=0; cntemp1<nodedata->numhop; cntemp1++ ) {
                                if ( nodedata->idanchop[cntemp1] == nodedata->idancdhp[cntemp] ) {
                                    nodedata->indicehop[nodedata->numdhphop] = cntemp1;
                                    nodedata->indicedhp[nodedata->numdhphop] = cntemp;
                                   nodedata->numdhphop++;
                                    cntemp1 = nodedata->numhop + 1; //go out second loop
                                  }
                             }
                         }
                         if ( nodedata->numdhphop >= 3 ) {
                            float distemp[20]; float matAx[20]; float matAy[20]; float matB[20];
                            float matAA[4]; float abstemp;
                            for ( cntemp=0; cntemp<nodedata->numdhphop; cntemp++ )
                                distemp[cntemp] = nodedata->dhp[nodedata->indicedhp[cntemp]] *
nodedata->hopcnt[nodedata->indicehop[cntemp]];
                            for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                              matAx[cntemp] = (nodedata->xancre[nodedata->indicehop[cntemp]]-
nodedata->xancre[nodedata->indicehop[nodedata->numdhphop-1]])*(-2);
                              matAy[cntemp] = (nodedata->yancre[nodedata->indicehop[cntemp]]-
nodedata->yancre[nodedata->indicehop[nodedata->numdhphop-1]])*(-2);
                              matB[cntemp] = distemp[cntemp]*distemp[cntemp] - distemp[nodedata-
>numdhphop-1]*distemp[nodedata->numdhphop-1] - nodedata->xancre[nodedata-
>indicehop[cntemp]]*nodedata->xancre[nodedata->indicehop[cntemp]] + nodedata->xancre[nodedata-
>indicehop[nodedata->numdhphop-1]]*nodedata->xancre[nodedata->indicehop[nodedata->numdhphop-1]] -
nodedata->yancre[nodedata->indicehop[cntemp]]*nodedata->yancre[nodedata->indicehop[cntemp]] +
nodedata->yancre[nodedata->indicehop[nodedata->numdhphop-1]]*nodedata->yancre[nodedata-
>indicehop[nodedata->numdhphop-1]];
                            }
                            matAA[0]=0; matAA[1]=0; matAA[2]=0; matAA[3]=0;
                            for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
```

```
                              matAA[0] = matAA[0] + matAy[cntemp]*matAy[cntemp];
                              matAA[1] = matAA[1] + matAx[cntemp]*matAy[cntemp];
                              matAA[3] = matAA[3] + matAx[cntemp]*matAx[cntemp];
                         }
                         abstemp = matAA[0]*matAA[3] - matAA[1]*matAA[1];
                         matAA[0] = matAA[0]/abstemp; matAA[3]= matAA[3]/abstemp;
                         matAA[1] = matAA[1]/abstemp*(-1); matAA[2] = matAA[1]; //matAA=(A'A)-1
                         for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                           matAA[2]=matAA[0]*matAx[cntemp]+matAA[1]*matAy[cntemp];
                        matAy[cntemp]=matAA[3]*matAy[cntemp]+matAA[1]*matAx[cntemp];
                            matAx[cntemp] = matAA[2];
                           }
                         nodedata->pos_x = 0; nodedata->pos_y = 0;
                         for ( cntemp=0; cntemp<nodedata->numdhphop-1; cntemp++ ) {
                              nodedata->pos_x = nodedata->pos_x + matAx[cntemp]*matB[cntemp];
                              nodedata->pos_y = nodedata->pos_y + matAy[cntemp]*matB[cntemp];
                           }
                         float pos_err = sqrt((nodedata->pos_x - get_node_position(c->node)-
>x)*(nodedata->pos_x - get_node_position(c->node)->x) + (nodedata->pos_y - get_node_position(c-
>node)->y)*(nodedata->pos_y - get_node_position(c->node)->y));

                         //checkout DV-hop
                         float pos_x_chec; float pos_y_chec; float mindistemp=100000; float
dis_dvhop; float pos_err_chec; int indice_nearanc;
                         for ( cntemp1=0; cntemp1<nodedata->numdhphop; cntemp1++) {
                             if ( distemp[cntemp1] < mindistemp ) {
                                   mindistemp = distemp[cntemp1];
                                  indice_nearanc = nodedata->indicehop[cntemp1]; }
                         }
                         dis_dvhop = sqrt((nodedata->pos_x - nodedata-
>xancre[indice_nearanc])*(nodedata->pos_x - nodedata->xancre[indice_nearanc]) + (nodedata->pos_y -
nodedata->yancre[indice_nearanc])*(nodedata->pos_y - nodedata->yancre[indice_nearanc]));
                         pos_x_chec = mindistemp/dis_dvhop*(nodedata->pos_x-nodedata-
>xancre[indice_nearanc])+nodedata->xancre[indice_nearanc];
                         pos_y_chec = mindistemp/dis_dvhop*(nodedata->pos_y-nodedata-
>yancre[indice_nearanc])+nodedata->yancre[indice_nearanc];
                         pos_err_chec = sqrt((pos_x_chec - get_node_position(c->node)-
>x)*(pos_x_chec - get_node_position(c->node)->x) + (pos_y_chec - get_node_position(c->node)-
>y)*(pos_y_chec - get_node_position(c->node)->y));
                        // Selective DV-hop
                         float pos_x_sel; float pos_y_sel;float pos_err_sel; float pos_xtmp;float
pos_ytmp; float mindiferr_hopcnt=50000;
                          int indice_nearanc_sel=-1;  int cntemp2; int cntemp3; int cntemp4;
                          for ( cntemp1=0;cntemp1<nodedata->numdhphop-2;cntemp1++){
                            for ( cntemp2=cntemp1+1; cntemp2<nodedata->numdhphop-1; cntemp2++) {
                                 for (cntemp3=cntemp2+1;cntemp3<nodedata->numdhphop; cntemp3++)  {
                                  // first obtain the estimated position by this 3-anchors group
                                      matAA[0] = (nodedata->xancre[nodedata->indicehop[cntemp1]]-
nodedata->xancre[nodedata->indicehop[cntemp3]])*(-2);
                                      matAA[2] = (nodedata->xancre[nodedata->indicehop[cntemp2]]-
nodedata->xancre[nodedata->indicehop[cntemp3]])*(-2);
                                      matAA[1] = (nodedata->yancre[nodedata->indicehop[cntemp1]]-
nodedata->yancre[nodedata->indicehop[cntemp3]])*(-2);
                                      matAA[3] = (nodedata->yancre[nodedata->indicehop[cntemp2]]-
nodedata->yancre[nodedata->indicehop[cntemp3]])*(-2);
                                      matB[0] = distemp[cntemp1]*distemp[cntemp1] -
distemp[cntemp3]*distemp[cntemp3] - nodedata->xancre[nodedata->indicehop[cntemp1]]*nodedata-
>xancre[nodedata->indicehop[cntemp1]] + nodedata->xancre[nodedata->indicehop[cntemp3]]*nodedata-
>xancre[nodedata->indicehop[cntemp3]] - nodedata->yancre[nodedata->indicehop[cntemp1]]*nodedata-
>yancre[nodedata->indicehop[cntemp1]] + nodedata->yancre[nodedata->indicehop[cntemp3]]*nodedata-
>yancre[nodedata->indicehop[cntemp3]];
                                      matB[1] = distemp[cntemp2]*distemp[cntemp2] -
distemp[cntemp3]*distemp[cntemp3] - nodedata->xancre[nodedata->indicehop[cntemp2]]*nodedata-
>xancre[nodedata->indicehop[cntemp2]] + nodedata->xancre[nodedata->indicehop[cntemp3]]*nodedata-
>xancre[nodedata->indicehop[cntemp3]] - nodedata->yancre[nodedata->indicehop[cntemp2]]*nodedata-
>yancre[nodedata->indicehop[cntemp2]] + nodedata->yancre[nodedata->indicehop[cntemp3]]*nodedata-
>yancre[nodedata->indicehop[cntemp3]];
                                      abstemp = matAA[0]*matAA[3] - matAA[1]*matAA[2];
                                      if ( abstemp > 0.000001 )   {
                                pos_xtmp = (matAA[3]*matB[0]-matAA[1]*matB[1])/abstemp;
                                pos_ytmp = (matAA[0]*matB[1]-matAA[2]*matB[0])/abstemp;
                               //then find nearest anchor to this 3-anchor estimated position
                                          indice_nearanc_sel = -1; matAA[0] = 1000;
                                           // matAA[0] recycled be used as the reference distance
                                 for(cntemp4=0;cntemp4<nodedata->numdhphop;cntemp4++){
                                               matAA[1] = sqrt((pos_xtmp - nodedata-
>xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]]) +
(pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]]));             //matAA[1] recycled to use as the temporary distance
                                               if ( cntemp4==0 )
                                                   matAA[0] = matAA[1]+1;
                                               if ( matAA[1]<matAA[0] ) {
```

```
                                                      matAA[0] = matAA[1];
                                                      if ( matAA[1]<nodedata->range )
                                                          indice_nearanc_sel = cntemp4;
                                                  }
                                              }
                                             matAA[2] =0;
                                         //matAA[2] recycled used as tempoary hop-count difference
                          for ( cntemp4=0; cntemp4<nodedata->numdhphop; cntemp4++ ) {
                             if ( indice_nearanc_sel > -1 )  {
                                  //the nearest anchor is within one hop
                                  if ( cntemp4 != indice_nearanc_sel )  {
                                    if ( matAA[0] <= nodedata->range/2 )
                    matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp - nodedata->xancre[nodedata-
>indicehop[cntemp4]])*(pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp -
nodedata->yancre[nodedata->indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])) / nodedata->dhp[nodedata->indicedhp[indice_nearanc_sel]] - nodedata-
>hopcnt[nodedata->indicehop[cntemp4]]);
                                     else
                 matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp - nodedata->xancre[nodedata-
>indicehop[cntemp4]])*(pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp -
nodedata->yancre[nodedata->indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])) / (nodedata->dhp[nodedata->indicedhp[indice_nearanc_sel]]+nodedata-
>dhp[nodedata->indicedhp[cntemp4]])*2 - nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                                     }
                                  else
        matAA[2]=matAA[2]+abs(1-nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                                  }
                            else //there is no one-hop anchor
matAA[2]=matAA[2]+fabs(sqrt((pos_xtmp - nodedata->xancre[nodedata->indicehop[cntemp4]])*(pos_xtmp -
nodedata->xancre[nodedata->indicehop[cntemp4]]) + (pos_ytmp - nodedata->yancre[nodedata-
>indicehop[cntemp4]])*(pos_ytmp - nodedata->yancre[nodedata->indicehop[cntemp4]])) / nodedata-
>dhp[nodedata->indicedhp[cntemp4]] - nodedata->hopcnt[nodedata->indicehop[cntemp4]]);
                            }
                if ( matAA[2] < mindiferr_hopcnt )  {
        pos_x_sel = pos_xtmp;  pos_y_sel = pos_ytmp; mindiferr_hopcnt = matAA[2];
                }
                                        }
                                   }
                              }
                           }
                          if ( mindiferr_hopcnt == 50000 ) {
                              pos_x_sel = nodedata->pos_x; pos_y_sel = nodedata->pos_y;
                            }
                          pos_err_sel = sqrt((pos_x_sel - get_node_position(c->node)->x)*(pos_x_sel
- get_node_position(c->node)->x) + (pos_y_sel - get_node_position(c->node)->y)*(pos_y_sel -
get_node_position(c->node)->y));
                          FILE *posfile;  posfile=fopen("position_results_classe2mobile.txt", "a");
                         if (posfile!= NULL)
                         {fprintf(posfile,"%d", c->node); fprintf(posfile,"%s"," ");
                          fprintf(posfile, "%d", nodedata->numdhp);fprintf(posfile, "%s", " ");
                          fprintf(posfile, "%lf", pos_err); fprintf(posfile, "%s", " ");
                          fprintf(posfile, "%lf", pos_err_chec);fprintf(posfile, "%s", " ");
                          fprintf(posfile, "%lf", pos_err_sel);fprintf(posfile, "%s", "\n");
                          fclose(posfile);}
                 else {printf("Error! Problem on adding Centroid result to file!\n");return 0;}    }

                // every node inializes those nodedata parameters about the DV-hop
                       nodedata->numhop = 0;  nodedata->numdhp = 0;
                       for (cntemp=0;cntemp<30;cntemp++)
                           nodedata->hopcnt[cntemp] = -1;
                       for (cntemp=0; cntemp<nodedata->numancretotal; cntemp++ )
                           nodedata->indicedhp[cntemp] = -1;
                       nodedata->timenext = nodedata->timecurrent + nodedata->period;
                       nodedata->timecurrent = nodedata->timecurrent + nodedata->period;
                       nodedata->tasknext = 0;
                       scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
                  }
             }
        }
    }
    else{packet_dealloc(packet);}   /* else dealloc the packet */
}
application_methods_t methods = {rx};
```

Annex 5 : WSNet Source Code for Class-1 Protocol

```
#include <stdio.h>
#include <math.h>
#include <include/modelutils.h>
/* Defining module informations*/
model_t model =  {
    "localisationclasse1mobile",
    "Linqing GUI",
    "0.1",
    MODELTYPE_APPLICATION,
    {NULL, 0}
};
/* Defining node type */
#define NORMAL 0
#define ANCHOR 1
/* Node private data */
struct nodedata {
  int *overhead; int type; int seq; uint64_t birth; uint64_t period;
   /* for stats */
  int packet_tx; int packet_rx;
  float pos_x; float pos_y; uint64_t timecurrent;  uint64_t timenext;
  int numancre; int idanc[20];  float xancre[20];  float yancre[20];
};
/* Data Packet header */
struct packet_header {
  int  source; int  dst;
};
//Packet data payload
struct datainpacket {
  int typeofnode;
  float pos_x; float pos_y;
};
int callmeback(call_t *c, void *args);
void rx(call_t *c, packet_t *packet);
int init(call_t *c) { return 0; }
int destroy(call_t *c) {
    return 0;
}
/* Here we read the node variables from the xml config file*/
int setnode(call_t *c, void *params) {
    struct nodedata *nodedata = malloc(sizeof(struct nodedata));
    int i = get_entity_links_down_nbr(c);
    param_t *param;
    /* default values */
    nodedata->period=300000000; nodedata->type=NORMAL;
    nodedata->seq=0; nodedata->last_seq  = -1;
    nodedata->packet_tx = 0; nodedata->packet_rx = 0;
    nodedata->numancre=0; nodedata->birth=0; nodedata->timecurrent=0;
    /* reading the "default" markup from the xml config file */
    das_init_traverse(params);
    while ((param = (param_t *) das_traverse(params)) != NULL) {
        if (!strcmp(param->key, "type")) {
            if (get_param_integer(param->value, &(nodedata->type))) {
                goto error;
            }
        }
        if (!strcmp(param->key, "period")) {
            if (get_param_time(param->value, &(nodedata->period))) {
                goto error;
            }
        }
         if (!strcmp(param->key, "birth")) {
            if (get_param_integer(param->value, &(nodedata->birth))) {
                goto error;
            }
        }
    }
    /* alloc overhead memory */
    if (i) {
        nodedata->overhead = malloc(sizeof(int) * i);
    } else {
        nodedata->overhead = NULL;
    }
    set_node_private_data(c, nodedata);
    nodedata->timecurrent = nodedata->birth;
    if (c->node==0){
         FILE *posfile; posfile = fopen("position_results_classe1mobile.txt", "w");
         if (posfile == NULL)  {
           printf("Error! Problem occurs when creating the result file!\n"); }
         else { fclose(posfile); }
```

```
    }
    FILE *posfile;  posfile=fopen("position_results_classe1mobile.txt", "a");
    if (posfile!= NULL) {
        fprintf(posfile,"%s","position of "); fprintf(posfile, "%d", c->node);
        fprintf(posfile, "%s", " is "); fprintf(posfile, "%lf", get_node_position(c->node)->x);
        fprintf(posfile, "%s", " "); fprintf(posfile, "%lf", get_node_position(c->node)->y);
        fprintf(posfile, "%s", "\n"); fclose(posfile);
    }
    return 0;
    error:
       free(nodedata);
       return -1;
}
int unsetnode(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    /* we print node stats before exit */
    if (nodedata->packet_tx > 0 || nodedata->packet_rx > 0) {
        if (nodedata->packet_tx > 0) {
           FILE *posfile;  posfile=fopen("position_results_classe1mobile.txt", "a");
            if (posfile!= NULL)
           { fprintf(posfile, "%d", nodedata->packet_tx);
             fprintf(posfile, "%s", " "); fclose(posfile);}
        }
    }
    if (nodedata->overhead) {
        free(nodedata->overhead);
    }
    free(nodedata);
    return 0;
}
int bootstrap(call_t *c) {
    struct nodedata *nodedata = get_node_private_data(c);
    int i = get_entity_links_down_nbr(c);
    entityid_t *down = get_entity_links_down(c);
    while (i--) {
        call_t c0 = {down[i], c->node};
        if ((get_entity_type(&c0) != MODELTYPE_ROUTING)
            && (get_entity_type(&c0) != MODELTYPE_MAC)) {
            nodedata->overhead[i] = 0;
        } else {
            nodedata->overhead[i] = GET_HEADER_SIZE(&c0);
         }
    }
    nodedata->timecurrent = get_time();
    /* if the node type is normal, we schedule a new callback */
    if (nodedata->type == 0) {
      scheduler_add_callback(nodedata->timecurrent, c, callmeback, NULL);
    }
    return 0;
}
int ioctl(call_t *c, int option, void *in, void **out) {
    return 0;
}
int callmeback(call_t *c, void *args) {
    struct nodedata *nodedata = get_node_private_data(c);
    entityid_t *down = get_entity_links_down(c);   call_t c0 = {down[0], c->node};
    packet_t *packet=packet_alloc(c, nodedata->overhead[0]+ sizeof(struct
packet_header)+sizeof(struct datainpacket));
    struct packet_header *header = (struct packet_header *) (packet->data + nodedata->overhead[0]);
    struct datainpacket *datapacket = (struct datainpacket *) (packet->data + nodedata->overhead[0]
+ sizeof(struct packet_header));

     int timepass = ( get_time() - nodedata->timecurrent )/1000000;
     // the time passed between two callmeback
     float xsumtemp; float ysumtemp; float pos_errCen; float pos_errCPE;
     float pos_errMid; float distemp1; float distemp2; float distemp3;
     float pos_xmidtemp; float pos_ymidtemp;
    int cntemp; int cntemp1; int cntemp2;
    if ( nodedata->type == 0 && timepass == nodedata->period/1000000/6*5 ) {
        // it is time for calculation of the position
        if (nodedata->numancre >= 3){
              xsumtemp = 0; ysumtemp = 0;
              for (cntemp = 1; cntemp <= nodedata->numancre; cntemp++){
                   xsumtemp = xsumtemp + nodedata->xancre[cntemp-1];
                   ysumtemp = ysumtemp + nodedata->yancre[cntemp-1];
               }
              nodedata->pos_x = xsumtemp/nodedata->numancre;
              nodedata->pos_y = ysumtemp/nodedata->numancre;
              pos_errCen = sqrt((nodedata->pos_x - get_node_position(c->node)->x)*(nodedata->pos_x -
get_node_position(c->node)->x) + (nodedata->pos_y - get_node_position(c->node)->y)*(nodedata->pos_y
- get_node_position(c->node)->y));
               //CPE
```

```
            xsumtemp = 0; ysumtemp = 1000; pos_xmidtemp = 0; pos_ymidtemp = 1000;
              //temporarily used as the maximum and minimum values
            for (cntemp = 1; cntemp <= nodedata->numancre; cntemp++) {
                 if (nodedata->xancre[cntemp-1] > xsumtemp)
                       xsumtemp = nodedata->xancre[cntemp-1];
                 if (nodedata->xancre[cntemp-1] < ysumtemp)
                       ysumtemp = nodedata->xancre[cntemp-1];
                 if (nodedata->yancre[cntemp-1] > pos_xmidtemp)
                       pos_xmidtemp = nodedata->yancre[cntemp-1];
                 if (nodedata->yancre[cntemp-1] < pos_ymidtemp)
                       pos_ymidtemp = nodedata->yancre[cntemp-1];
            }
            nodedata->pos_x=(xsumtemp+ysumtemp)/2; nodedata->pos_y=(pos_xmidtemp+pos_ymidtemp)/2;
            pos_errCPE = sqrt((nodedata->pos_x - get_node_position(c->node)->x)*(nodedata->pos_x -
get_node_position(c->node)->x) + (nodedata->pos_y - get_node_position(c->node)->y)*(nodedata->pos_y
- get_node_position(c->node)->y));
            //mid-perpendicular
            pos_xmidtemp = 0; pos_ymidtemp = 0; int anc1; int anc2; int anc3;
            xsumtemp = 0; //temporarily used as the maximum distance value
            for (cntemp1=0; cntemp1<=nodedata->numancre-2; cntemp1++) {
                   for (cntemp2=cntemp1+1; cntemp2<=nodedata->numancre-1; cntemp2++) {
                   //first find the longest two anchors
                       distemp1 = (nodedata->xancre[cntemp1] - nodedata->xancre[cntemp2]) *
(nodedata->xancre[cntemp1] - nodedata->xancre[cntemp2]) + (nodedata->yancre[cntemp1] - nodedata-
>yancre[cntemp2]) * (nodedata->yancre[cntemp1] - nodedata->yancre[cntemp2]);
                       if ( distemp1 > xsumtemp ) {
                            xsumtemp = distemp1;
                            anc1 = cntemp1; anc2 = cntemp2;
                       }
                    }
             }
             ysumtemp = 0; //temporarily used as the maximum distance value
             for (cntemp1=0; cntemp1<=nodedata->numancre-1; cntemp1++) {
                 if ( cntemp1 != anc1 && cntemp1 != anc2 ) {
                      distemp1 = fabs(nodedata->xancre[cntemp1]*(nodedata->yancre[anc1]-nodedata-
>yancre[anc2])+nodedata->yancre[cntemp1]*(nodedata->xancre[anc2]-nodedata->xancre[anc1])+nodedata-
>yancre[anc2]*(nodedata->xancre[anc1]-nodedata->xancre[anc2])+nodedata->xancre[anc2]*(nodedata-
>yancre[anc2]-nodedata->yancre[anc1]))/sqrt(xsumtemp);
                      if ( ysumtemp < distemp1 ) {
                           ysumtemp = distemp1; anc3 = cntemp1;
                      }
                  }
              }
             distemp2 = (nodedata->xancre[anc2] - nodedata->xancre[anc3]) * (nodedata->xancre[anc2]
- nodedata->xancre[anc3]) + (nodedata->yancre[anc2] - nodedata->yancre[anc3]) * (nodedata-
>yancre[anc2] - nodedata->yancre[anc3]);
             distemp3 = (nodedata->xancre[anc3] - nodedata->xancre[anc1]) * (nodedata->xancre[anc3]
- nodedata->xancre[anc1]) + (nodedata->yancre[anc3] - nodedata->yancre[anc1]) * (nodedata-
>yancre[anc3] - nodedata->yancre[anc1]);
              if (xsumtemp >= distemp2+distemp3)
              { pos_xmidtemp = (nodedata->xancre[anc1] + nodedata->xancre[anc2])/2;
                pos_ymidtemp = (nodedata->yancre[anc1] + nodedata->yancre[anc2])/2;}
              else {
                pos_xmidtemp = ((nodedata->xancre[anc3]*nodedata->xancre[anc3]-nodedata-
>xancre[anc1]*nodedata->xancre[anc1])*(nodedata->yancre[anc2]-nodedata->yancre[anc3]) + (nodedata-
>xancre[anc3]*nodedata->xancre[anc3]-nodedata->xancre[anc2]*nodedata->xancre[anc2])*(nodedata-
>yancre[anc3]-nodedata->yancre[anc1]) + (nodedata->yancre[anc3]-nodedata->yancre[anc1])*(nodedata-
>yancre[anc1]-nodedata->yancre[anc2])*(nodedata->yancre[anc2]-nodedata->yancre[anc3])) / (nodedata-
>yancre[anc2]*(nodedata->xancre[anc3]-nodedata->xancre[anc1]) + nodedata->yancre[anc1]*(nodedata-
>xancre[anc2]-nodedata->xancre[anc3]) + nodedata->yancre[anc3]*(nodedata->xancre[anc1]-nodedata-
>xancre[anc2])) / 2;
                pos_ymidtemp = ((nodedata->yancre[anc3]*nodedata->yancre[anc3]-nodedata-
>yancre[anc1]*nodedata->yancre[anc1])*(nodedata->xancre[anc2]-nodedata->xancre[anc3]) + (nodedata-
>yancre[anc3]*nodedata->yancre[anc3]-nodedata->yancre[anc2]*nodedata->yancre[anc2])*(nodedata-
>xancre[anc3]-nodedata->xancre[anc1]) + (nodedata->xancre[anc3]-nodedata->xancre[anc1])*(nodedata-
>xancre[anc1]-nodedata->xancre[anc2])*(nodedata->xancre[anc2]-nodedata->xancre[anc3])) / (nodedata-
>xancre[anc2]*(nodedata->yancre[anc3]-nodedata->yancre[anc1]) + nodedata->xancre[anc1]*(nodedata-
>yancre[anc2]-nodedata->yancre[anc3]) + nodedata->xancre[anc3]*(nodedata->yancre[anc1]-nodedata-
>yancre[anc2])) / 2;
             }
             nodedata->pos_x = pos_xmidtemp; nodedata->pos_y = pos_ymidtemp;
             pos_errMid = sqrt((nodedata->pos_x - get_node_position(c->node)->x)*(nodedata->pos_x
- get_node_position(c->node)->x) + (nodedata->pos_y - get_node_position(c->node)->y)*(nodedata-
>pos_y - get_node_position(c->node)->y));
            FILE *posfile; posfile=fopen("position_results_classe1mobile.txt", "a");
             if (posfile!= NULL)
             {  fprintf(posfile, "%d", c->node); fprintf(posfile, "%s", " ");
                fprintf(posfile, "%lf", pos_errCen); fprintf(posfile, "%s", " ");
                fprintf(posfile, "%lf", pos_errCPE); fprintf(posfile, "%s", " ");
                fprintf(posfile, "%lf", pos_errMid); fprintf(posfile, "%s", " ");
                fprintf(posfile, "%d", nodedata->numancre); fprintf(posfile, "%s", "\n");
                fclose(posfile);}
```

```
            else
              {printf("Error! Problem on adding Centroid result to file!\n"); return 0;}
          }
          nodedata->timenext = nodedata->timecurrent + nodedata->period;
          nodedata->timecurrent = nodedata->timecurrent + nodedata->period;
          scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
    }
     if ( nodedata->type == 0 && timepass==0 ) {
     // beginning of period ; it is time to initialize the localisation
        nodedata->numancre = 0;
        /* broadcast request frame: localisation request */
        srand(time(NULL)+c->node*10);
        nodedata->timenext=nodedata->timecurrent+500000000/50*(rand()%50+1);
        scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
    }
   if(nodedata->type==0 && get_time()==nodedata->timenext && timepass<nodedata->period/1000000/6*5){
        destination_t destination = {BROADCAST_ADDR, {-1, -1, -1}};
        header->source=c->node; header->dst=-1;
        datapacket->typeofnode = 4; // 0x04 data request
        datapacket->pos_x = -1; datapacket->pos_y = -1;
        if (SET_HEADER(&c0, packet, &destination) == -1) {
            packet_dealloc(packet);
            return -1;
        }
        TX(&c0, packet);
        /* for stats */
        nodedata->packet_tx++;
        nodedata->timenext = nodedata->timecurrent + nodedata->period/6*5;
        scheduler_add_callback(nodedata->timenext, c, callmeback, NULL);
    }
    return 0;
}
void rx(call_t *c, packet_t *packet) {
    struct nodedata *nodedata = get_node_private_data(c);
    struct packet_header *header = (struct packet_header *) (packet->data + nodedata->overhead[0]);
    struct datainpacket *datapacket = (struct datainpacket *) (packet->data + nodedata->overhead[0]
+ sizeof(struct packet_header));
    nodedata->packet_rx++;
    entityid_t *down = get_entity_links_down(c);
    call_t c0 = {down[0], c->node};
    int notesign = 1; int cntemp;
    // if the receiver is an anchor, and it receives the packet from a normal node
    if (nodedata->type == 1 && datapacket->typeofnode == 4) {
        destination_t destination = {header->source, {get_node_position(header->source)->x,
get_node_position(header->source)->y, get_node_position(header->source)->z}};
        header->dst = header->source; header->source = c->node;
        datapacket->typeofnode = nodedata->type;
        datapacket->pos_x = get_node_position(c->node)->x;
        datapacket->pos_y = get_node_position(c->node)->y;
        if (SET_HEADER(&c0, packet, &destination) == -1) {
          packet_dealloc(packet);
          return -1;
        }
        srand(c->node*10);
        uint64_t timedelay = (rand()%50+1)*200000;  //delay for random time, maximum is 10ms
        unsigned long usecsleep = timedelay/1000;
        usleep(usecsleep);
        TX(&c0, packet);
        /* for stats */
        nodedata->packet_tx++; return;
    }
    // if the receiver is a normal node, and it receives the packet from an anchor
    else if (nodedata->type == 0 && datapacket->typeofnode == 1 && header->dst == c->node) {
        //normal node checks whether it already note down the position of this anchor
        if ( nodedata->numancre > 0 ) {
            for (cntemp = 0; cntemp < nodedata->numancre; cntemp++) {
                if ( nodedata->idanc[cntemp] ==  header->source )
                    notesign = 0;
            }
        }
        if ( notesign == 1 ) {
         nodedata->numancre++; nodedata->idanc[nodedata->numancre-1]=header->source;
           nodedata->xancre[nodedata->numancre-1] = datapacket->pos_x;
           nodedata->yancre[nodedata->numancre-1] = datapacket->pos_y;
           return;
        }
    }
    else{packet_dealloc(packet); /* else dealloc the packet */
}
application_methods_t methods = {rx};
```

Titre de la thèse en français

Amélioration de la Localisation dans les Réseaux de Capteurs sans Fil par Méthodes "*Range-free*"

Résumé de la thèse en français

Dans le contexte des réseaux de capteurs sans fil, la technique de localisation ”*range-free*” est plus efficiente, par rapport au principe ”*range-based*”. Par conséquent, nous avons focalisé nos travaux de cette thèse sur les techniques ”range-free”.

Afin de permettre à chaque nœud mobile ou normal de choisir son propre algorithme de localisation, nous avons proposé un mécanisme adapté en scindant les nœuds normaux en deux classes : les nœuds de la première classe ont au moins 3 ancres voisines, alors que les nœuds de la deuxième classe ont moins de trois ancres voisines. Pour les nœuds normaux de la classe 1, nous avons proposé un nouvel algorithme ”*Mid-perpendicular*”. Pour les nœuds normaux de la classe 2, nous avons proposé deux nouveaux algorithmes ”*Checkout DV-hop*” et ”*Selective 3-Anchor DV-hop*”.

Pour simuler et évaluer la performance de nos trois nouveaux algorithmes dans le contexte protocolaire des réseaux, nous avons pris soin de proposer deux protocoles associés : ”*DV-hop protocol*” et ”*Classe-1 protocol*”. Par la suite, nous avons combiné ces deux protocoles pour obtenir notre ”*adaptive range-free localization protocol*”. Basé sur nos protocoles, en utilisant le simulateur WSNet, nous avons simulé différents algorithmes ”*range-free*” dans le contexte des réseaux de capteurs conformes au standard IEEE 802.15.4. Les résultats ont été présentés et analysés en termes de précision de la localisation, charge du réseau, mobilité des nœuds, et synchronisation de ces derniers.

Mots clés

réseaux de capteurs sans fil, localisation, *range-free*, algorithme, protocole

Titre de la thèse en anglais

Improvement of Range-free Localization Systems in Wireless Sensor Networks

Résumé de la thèse en anglais

In the context of wireless sensor networks, the range-free localization technique is more cost-effective than the range-base scheme. Therefore, in this thesis we focus on the range-free technique.

In order to permit each normal node to choose its suitable localization algorithm, we proposed an adaptive mechanism to categorize normal nodes into two classes: the normal nodes having at least 3 neighbor anchors are class-1 nodes, while others are class-2 nodes. For class-1 normal nodes, we proposed a new algorithm named as *Mid-perpendicular*. For class-2 normal nodes, we proposed two algorithms *Checkout DV-hop* and *Selective 3-Anchor DV-hop*.

In order to simulate and evaluate the performance of our three new algorithms, we proposed two protocols: *DV-hop protocol* and *Class-1 protocol*. Then we combined these two protocols into our *adaptive range-free localization protocol*. Based on our protocols, using the network simulator WSNet, we simulate the concerned range-free localization algorithms in the IEEE 802.15.4 wireless network. The comparative network simulation results are presented and analyzed in terms of localization accuracy, overhead, node mobility, and node synchronization.

Keywords

wireless sensor networks, localization, *range-free*, algorithm, protocol