Towards Content-Centric Control Plane Supporting Efficient Anomaly Detection Functions

Hoang-Long MAI[1,2] Guillaume Doyen[2] Wissam Mallouli[1]
Edgardo Montes-de-Oca[1] Olivier Festor[3]

[1]Montimage, Paris - France, [2]Troyes University of Technology, France,
[3]Lorraine Research Laboratory in Computer Science and its Applications, France

Journées Cloud 2019, Toulouse, France
To appear in the 15th International Conference on Network and Service Management - CNSM 2019

Outline

Introduction

Context & Problematic

- Large number of low resource IoT devices & complex security functions
- Distributed Anomaly detection in IoT, Fog Computing

Hypothesis

- Each node executes identical security functions
- Metrics do not change frequently, results are repeated gradually over time
- Results are probably already executed in neighbor nodes

⇒ Hypothesis verification: *In a normal condition of Named Data Networking, 87% of computational security operations are repeated.*

Contribution

- Leveraging Named Function Networking (NFN) as an execution environment for anomaly detection (AN)
- Consider Bayesian Network (BN) inference as an AN framework since it stands for a representative function that numerous security components

Outline

Background on ICN - NDN - NFN

Information Centric Networking (ICN) - Named Data Networking (NDN)

- Name each content object instead of using IP address
- In-network caches for better delivery performance
- Among all ICN proposals, Named Data Networking (NDN) is the most promising one

Named function networking (NFN)

- Naming a function defined by its name and its parameters, and using λ-expressions as name resolution
- In-network caches results of calculation

Background on Bayesian Networks

Terminology

- A random variable, called X_i, is a set of possible values of a random phenomenon.
- An *evidence* $E = e$ is a subset $E = (X_{e_1}, \ldots, X_{e_m})$ of random variables standing for the observed phenomenon.
- A *factor* ϕ is defined as a function from a set of random variables $Val(X_1, \ldots, X_n)$ to $\mathbb{R}$.

Table: A factor example

A	B	C	$\phi(A, B, C)$
1	1	1	0.25
1	1	2	0.35
1	2	1	0.08
1	2	2	0.16
2	1	1	0.05
2	1	2	0.07
2	2	1	0
2	2	2	0
3	1	1	0.15
3	1	2	0.21
3	2	1	0.09
3	2	2	0.18

Background on Bayesian Networks

Bayesian Inference

- A Bayesian Network (BN) is a probabilistic graphical model that consists of nodes and directed edges.
- The *inference* designates an algorithm which consists in calculating the conditional probability of $P(X_q|E=e)$:

$$P(X_q) = \frac{1}{Z}\sum_{X_n}\phi_n\cdot(\ldots(\sum_{X_2}\phi_3\cdot(\sum_{X_1}\phi_2\cdot\phi_1))) \quad (1)$$

$$\textit{where} \quad Z = \sum_{X_1,\ldots,X_n}\prod_{i=1}^{n}\phi_i(X_i, Pa(X_i)) \quad (2)$$

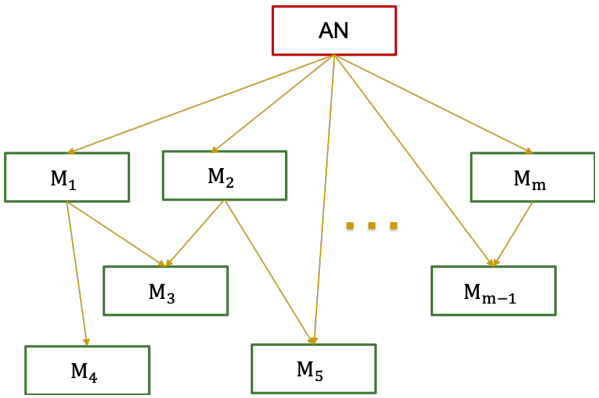

Figure: Example of a Bayesian network

Background on Bayesian Networks

Variable Elimination (VE) algorithm

Input: *initial factors* (Φ) and *evidence* (E=e)
Output: Conditional probability $P(X_q|E=e)$

```
foreach φ_i ∈ Φ do
    φ_i ← φ_i(E = e) // Factor reduction
end
Select Elimination Order (σ);
foreach x_i ∈ σ do
    foreach φ_j ∈ Φ do
        if x_i ∈ Scope[φ_j] then
            ψ_i ← ψ_i * φ_j // Factor product
        end
    end
    φ_i ← Σ_{X_i} ψ_i // Factor marginalization
end
foreach φ ∈ Φ do
    φ ← φ * φ // Factor product
end
Z ← Σ_{X_1...X_n} φ
P ← φ/Z // Factor normalization
```

Algorithm 1: Variable Elimination algorithm

Outline

Naming scheme and data structure - Factor

Properties

- The *factor* packet includes three parts: the list of the names of random variables, their dimensions, the list of all values of ϕ
- Types of factors: the initial factors and the temporary factors

Naming scheme

- */data/fac/initial/<name of variables>*
- */data/fac/temporary/<hash(factor)>*

Data structure

Structure	List of variables' name				List of variables' dimension				List of values									
Detail	X1	X2	...	Xn	dim(X1)	dim(X2)	...	dim(Xn)	$\phi(X1=1, X2=1, \ldots, Xn=1)$		$\phi(X1=1, X2=1, \ldots, Xn=2)$		...	$\phi(X1=dim(X1), X2=dim(X2), \ldots, Xn=dim(Xn))$				
Example	A	B	C	3	2	2	0.25	0.35	0.08	0.16	0.05	0.07	0	0	0.15	0.21	0.09	0.18

Figure: Data structure of a *factor*

Naming scheme and data structure - Evidence

Properties

- The packet of *evidence* encompasses three parts: the name of variables, their dimensions, and values of the *evidence*.
- An *evidence* is designed to be computed with a *factor* in *factor reduction*
- The value of the *evidence* for a variable in a packet consists of a chain of 0's and 1's.

Naming scheme

- */data/evi/<name of variables>/<values of evidence>*

Data structure

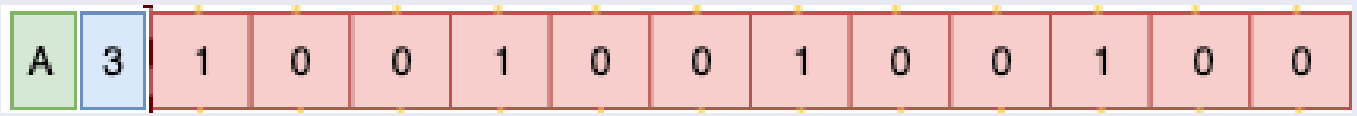

Figure: Data structure of an *evidence*

Transformation of functions into λ-calculus

Principle functions

- Factor reduction: */func/reduce/(/data/fac/...,/data/evi/...)*
- Factor product: */func/product/(/data/fac/...,/data/fac/...)*
- Factor marginalization: */func/marginalize/(/data/fac/...,variable)*
- Factor normalization: */func/normalize/(/data/fac/...)*

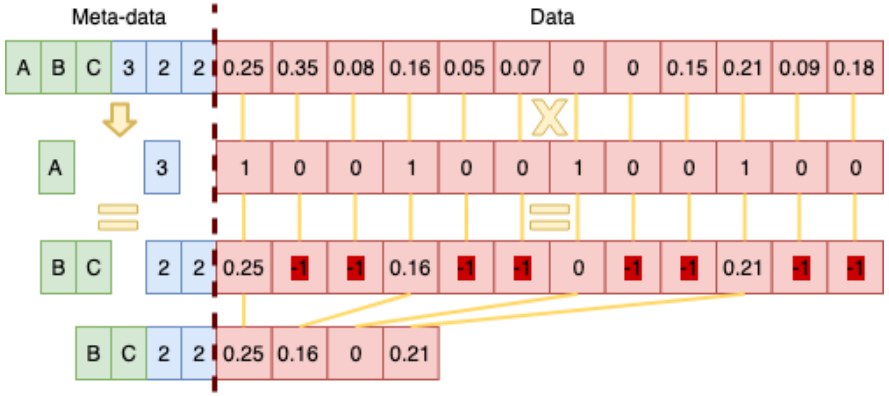

Figure: Example of function factor reduction

```
Function reduceFactor(fac,evi)
    return getVarDim(fac,getVarDimEvi(evi)) +
      SEPARATOR_CHAR
    serialize(filter(lambda x:x >-1,
      map(lambda x,y: -1 if y == 0 else x*y,
        deserialize(fac), reduce(lambda x,y:
          productEviVal(x,y),getValEvi(evi)))))
```

Figure: Transformation of factor reduction into λ-calculus

Outline

Numerical Results

Use case: the Content Poisoning Attack

- Alter content by inserting Bad Data into the cache of routers
- CPA has a subtle impact on several of metrics

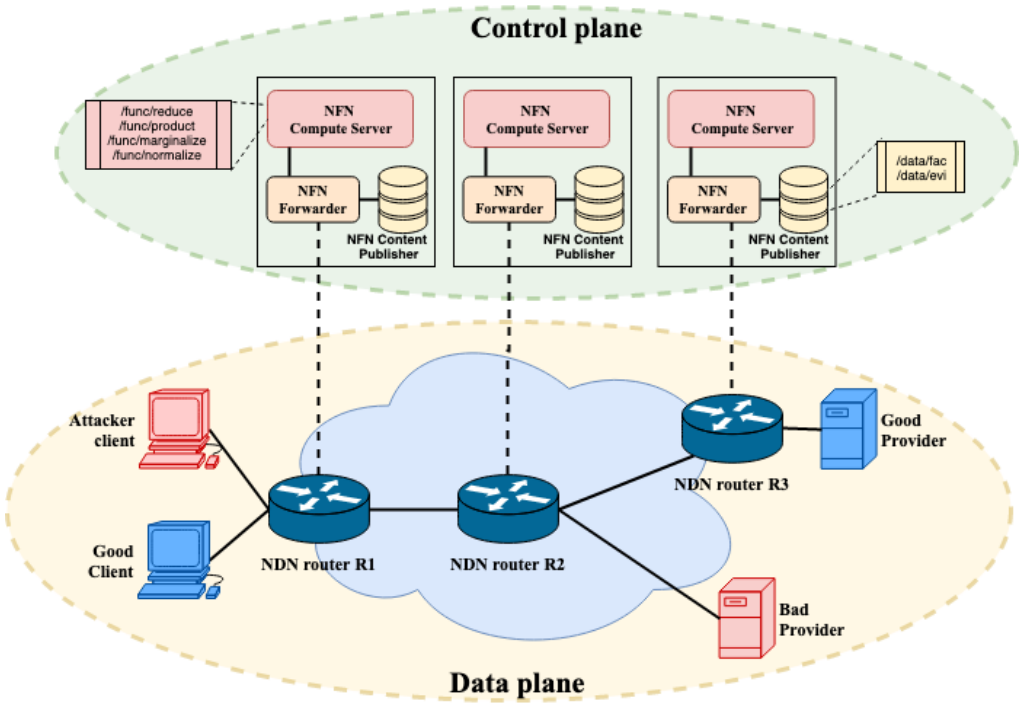

Figure: Experiment topology

Implementation

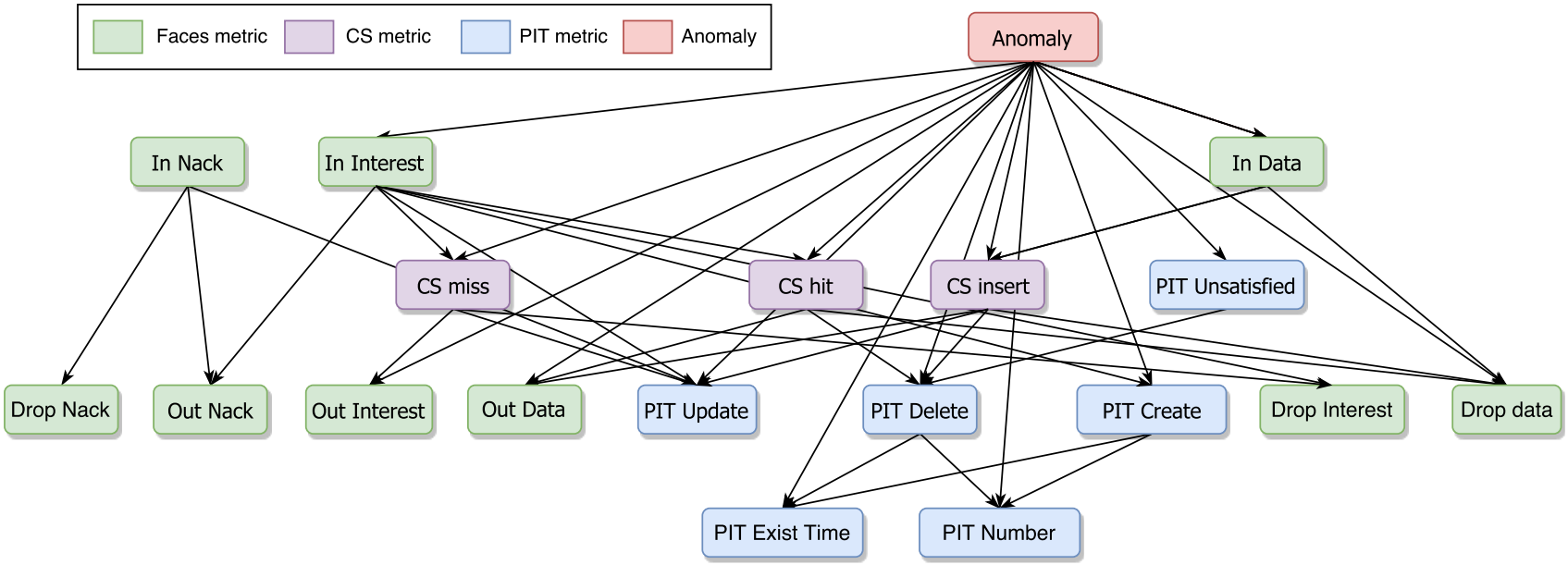

Figure: Bayesian Network to detect CPA

Implementation

Tool	Description	Link
NFN	The newest version of NFN written in python	https://github.com/cn-uofbasel/PiCN
NFD	The Networking Forwarding Daemon of NDN	http://named-data.net/doc/NFD/current/
jNDN	A NDN client library for Java	`https://github.com/named-data/jndn`
pgmpy	Standard library for Bayesian Network	https://github.com/pgmpy/pgmpy

Table: List of tools used in the implementation

Verification of proposed algorithm

The results of the proposed algorithm are 100% identical to the results of the standard library for BN - *pgmpy*.

Evaluation

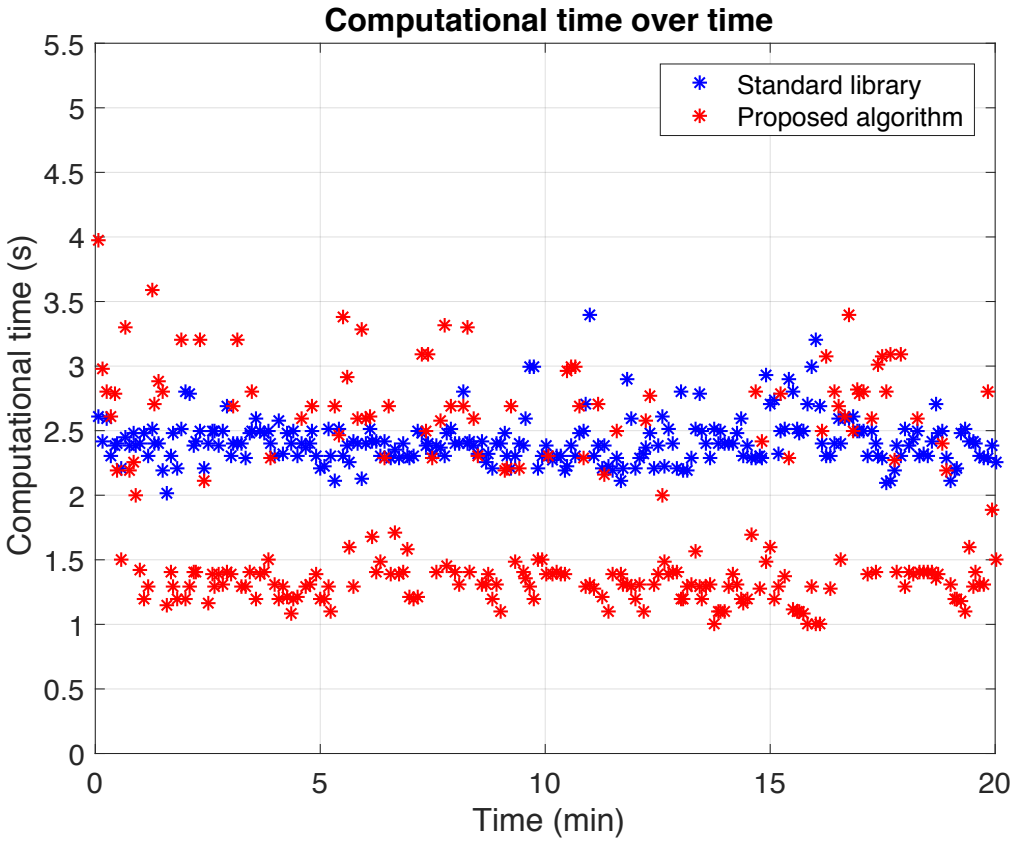

Figure: Snapshot of computational time over time

Evaluation

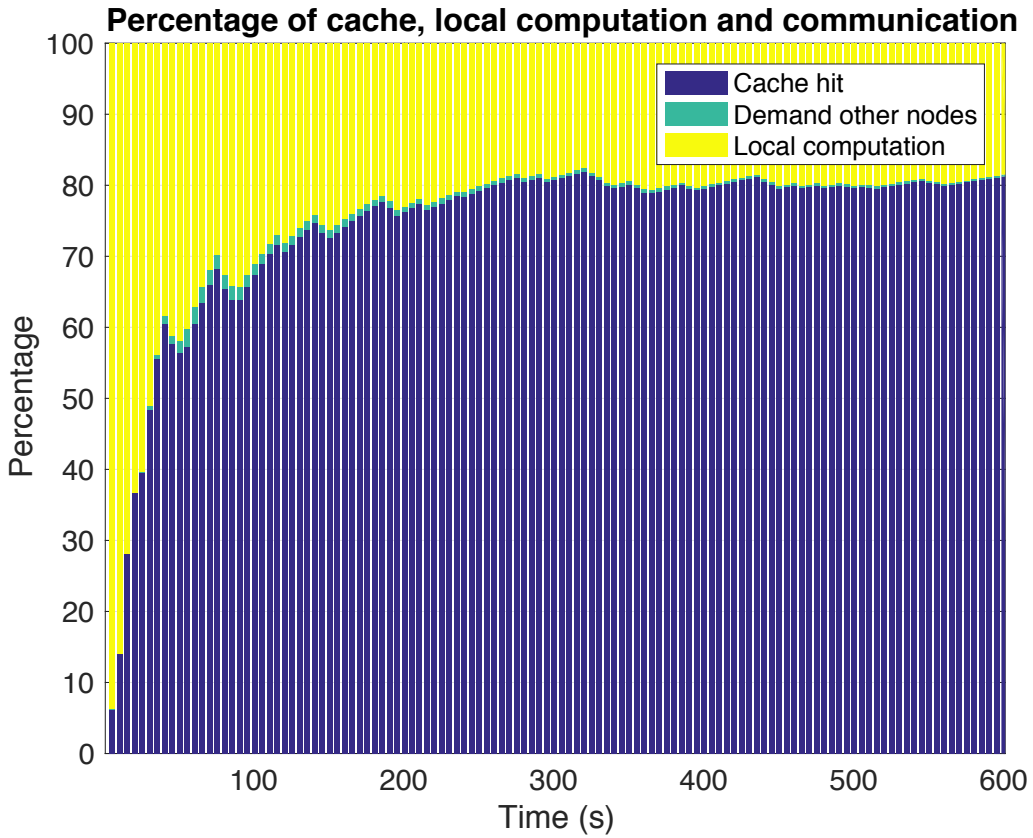

Figure: Evolution of proportion of requests over time

Evaluation

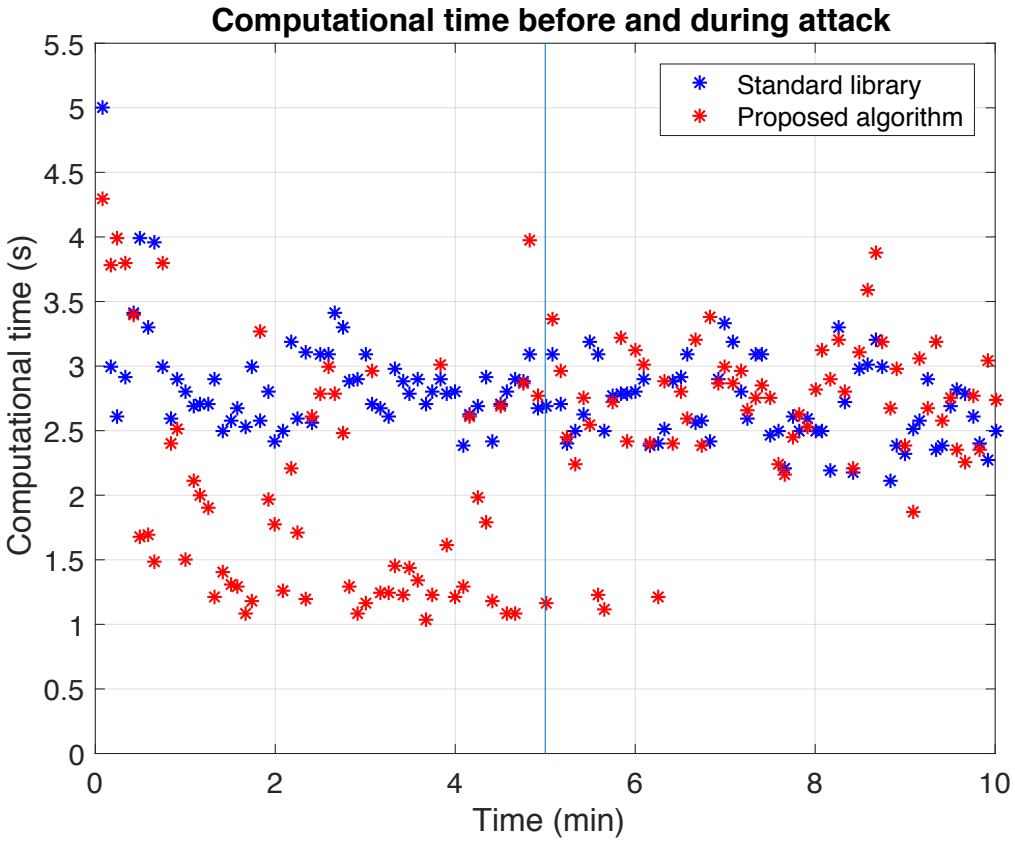

Figure: Snapshot of computational time before and during attack

Evaluation

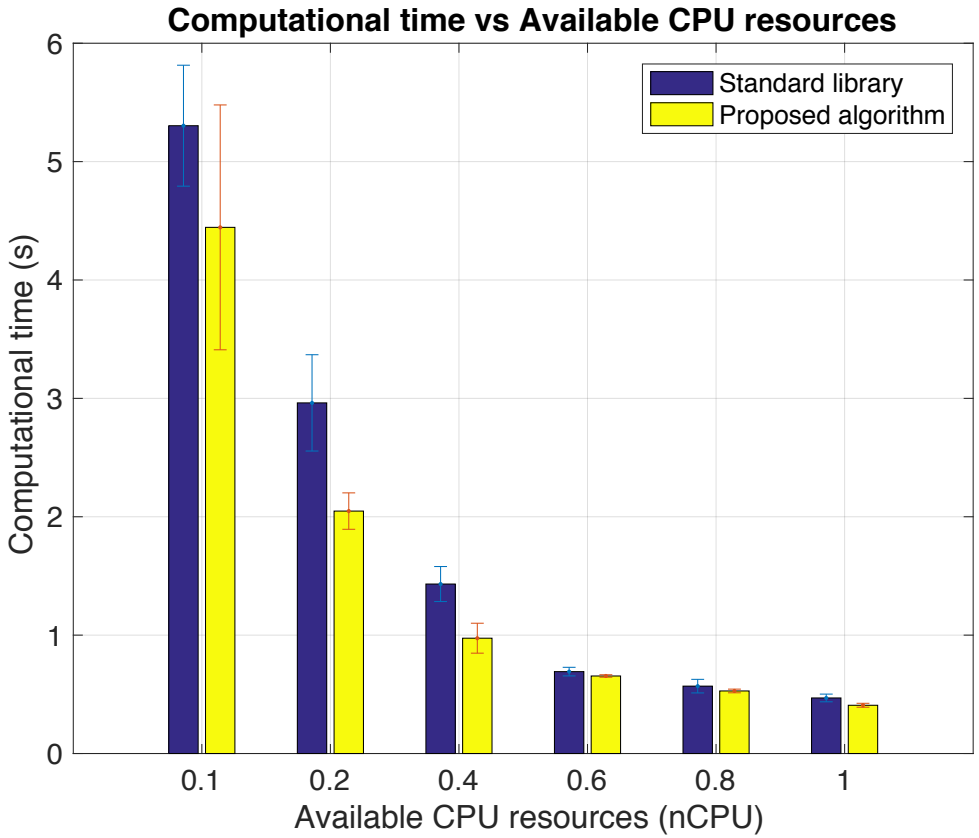

Figure: Impact of available CPU resources

Evaluation

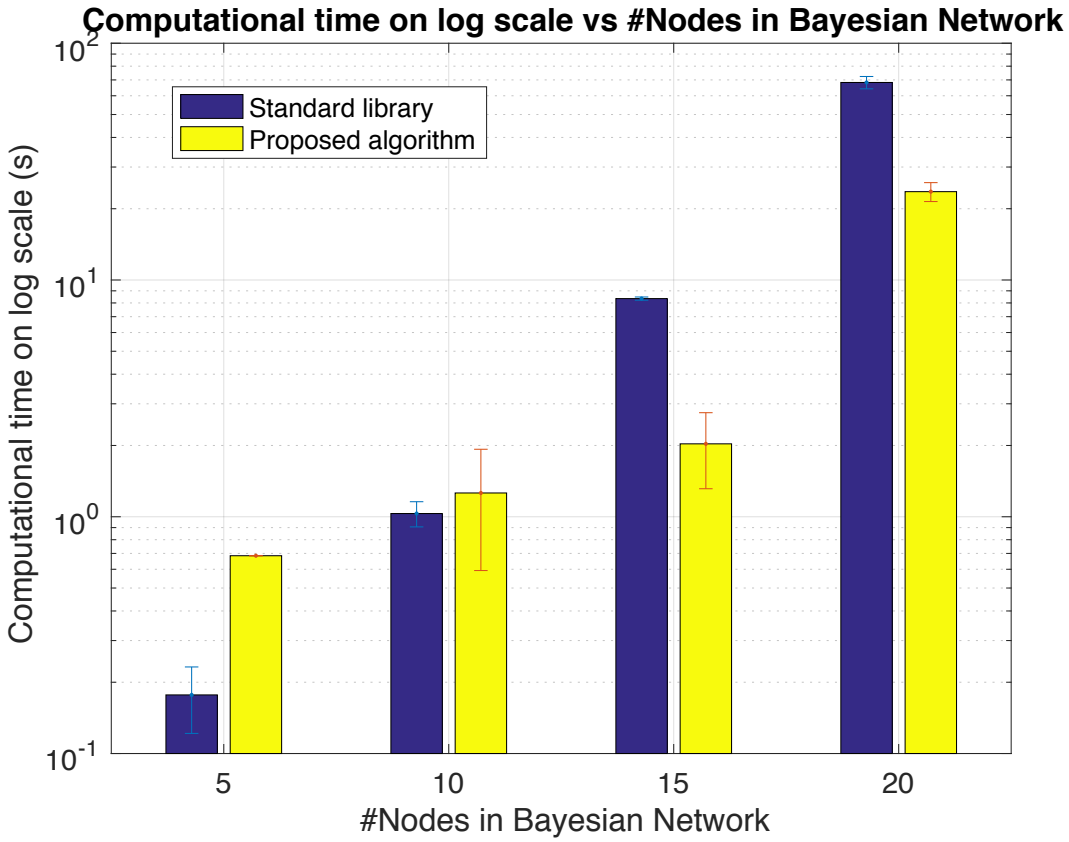

Figure: Impact of #nodes in BN

Evaluation

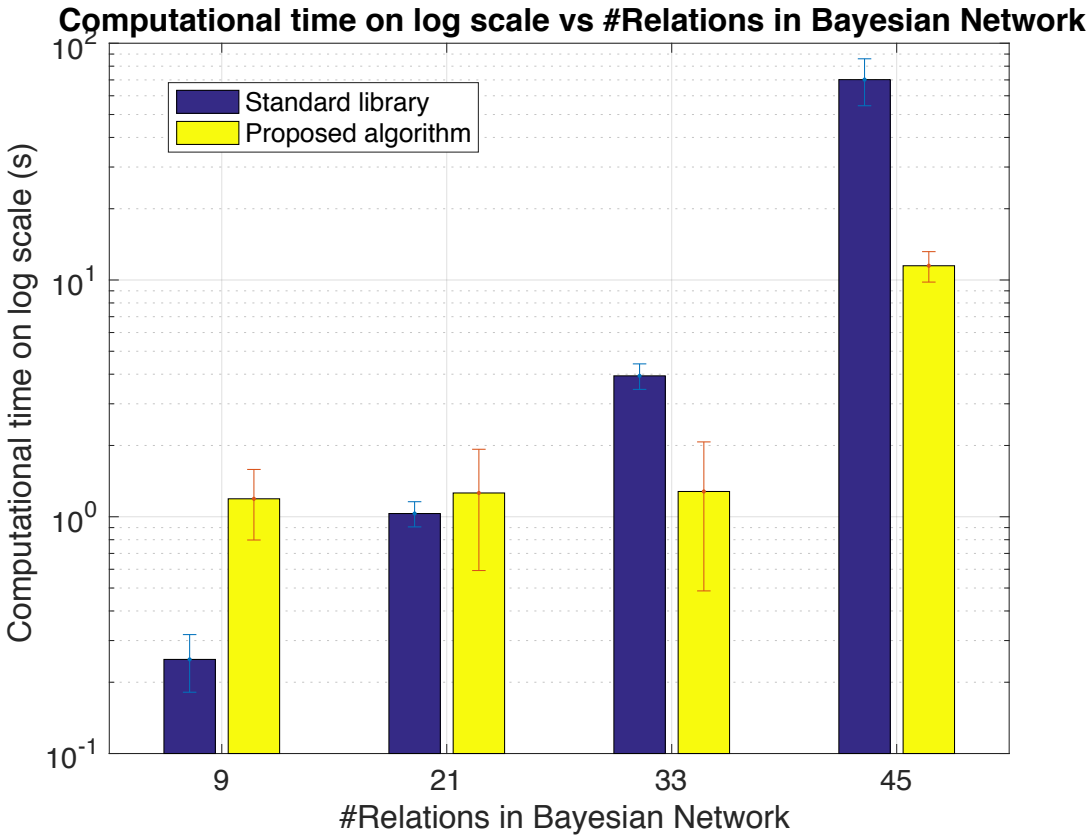

Figure: Impact of #relations in BN

Evaluation

Impacts on computational time

In term of computational time, the proposed approach is efficient when:

- The cache is fed with results from previous executions or neighbor nodes
- In normal traffic but not abnormal traffic
- The computation capacity is limited
- The BN is complex and needs a significant amount of operations

Outline

Conclusion & future work

Conclusion

- The core elements for the design and implementation of an NFN-supported Bayesian Network inference algorithm has been proposed
- Numerical results demonstrated that an NFN-supported BN performs better not only in the case of limited computational resources but also when the BN is complex

Future work

- Focus on further developing the content-oriented control plane by extending the current approach to integrate other methods for anomaly detection.
- Other attacks will also be considered to demonstrate the applicability and generality of the proposed approach