Tightening Network Calculus Delay Bounds by Predicting Flow Prolongations in the FIFO Analysis

Fabien Geyer[1,2] **Alexander Scheffler**[3] **Steffen Bondorf**[3]

IEEE RTAS 2021

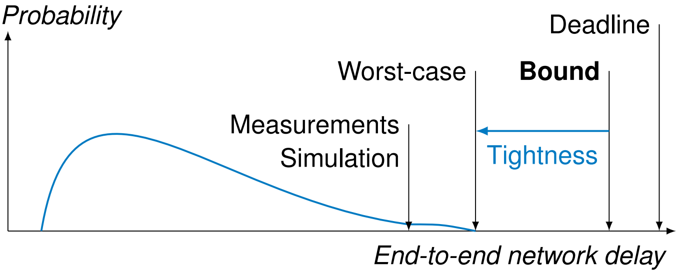
[1] Chair of Network Architectures and Services
Technical University of Munich, Germany

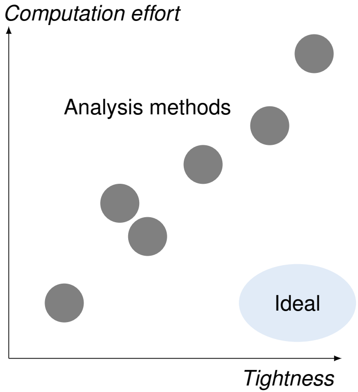
[2] Airbus Central R&T
Munich, Germany

[3] Faculty of Mathematics, Center of Computer Science
Ruhr University Bochum, Germany

Motivation

Worst-Case End-to-End Performance Analysis

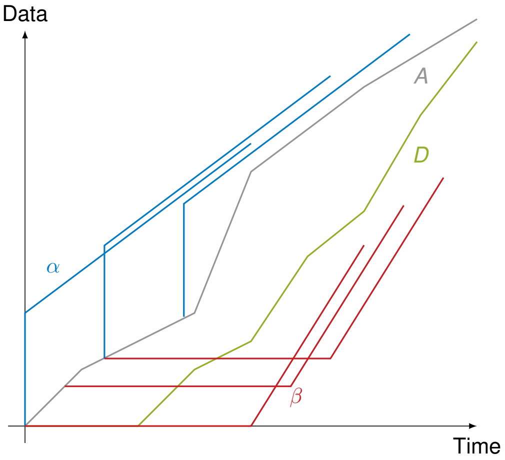




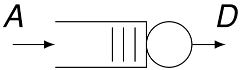
- Trade-off between computational effort and tightness
- **This talk: network analysis method with good tightness and fast execution**

Background

Network Calculus – Basics



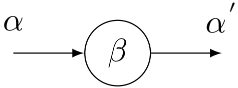
Basis: Cumulative arrivals and services [Cruz, 1991]



Arrival curve α: $A(t) - A(t-s) \leq \alpha(s), \forall t \leq s$

Service curve β: If the service by system $\mathcal{S}$ for a given input A results in an output D, then $\mathcal{S}$ offers a service curve $\beta \in \mathcal{F}_0$ iff

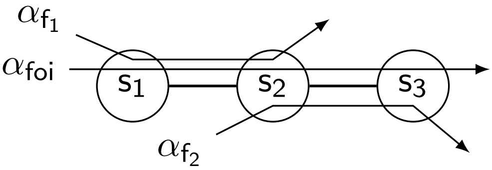
$$\forall t : D(t) \geq \inf_{0 \leq d \leq t} \{A(t-d) + \beta(d)\}.$$



Background

Network Calculus – FIFO Analysis

How to derive an end-to-end delay bound?



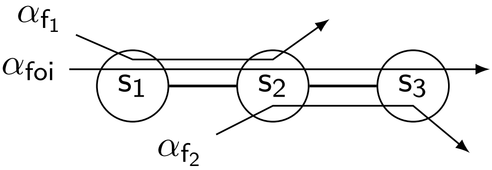
LUDB – Least Upper Delay Bound
[Bisti et al., 2008, Bisti et al., 2012]

Step 1: Compute the nesting tree
Step 2: Compute an end-to-end service curve by removing cross-flows step by step
Step 3: Compute the end-to-end delay bound

Background

Network Calculus – FIFO Analysis

How to derive an end-to-end delay bound?



LUDB – Least Upper Delay Bound
[Bisti et al., 2008, Bisti et al., 2012]

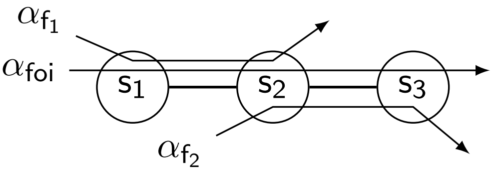
Step 1: Compute the nesting tree
Step 2: Compute an end-to-end service curve by removing cross-flows step by step
Step 3: Compute the end-to-end delay bound

Nesting: A sequence of servers ("tandem") is called nested if any two flows have disjunct paths or one flow is completely included in the path of the other flow.

Background

Network Calculus – FIFO Analysis

How to derive an end-to-end delay bound?



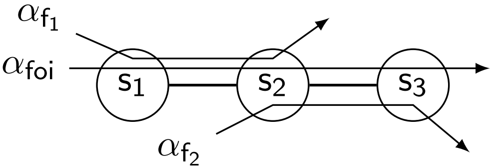
LUDB – Least Upper Delay Bound
[Bisti et al., 2008, Bisti et al., 2012]

***Step 1:* Derive all cuts creating nested subtandems**
Step 2: Compute the nesting trees
Step 3: Compute an end-to-end service curves
Step 3a: by removing cross-flows step by step and
***Step 3b:* by concatenating the intermedite service curves**
Step 4: Compute the end-to-end delay bound

Background

Network Calculus – FIFO Analysis

How to derive an end-to-end delay bound?

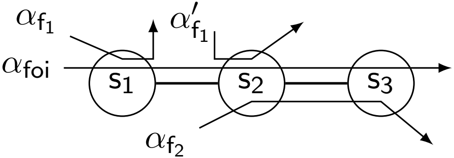


LUDB – Least Upper Delay Bound
[Bisti et al., 2008, Bisti et al., 2012]

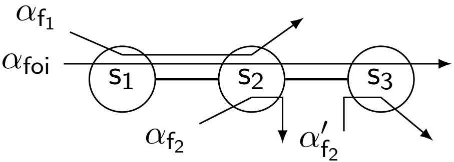
***Step 1:* Derive all cuts creating nested subtandems**
Step 2: Compute the nesting trees
Step 3: Compute an end-to-end service curves
  Step 3a: by removing cross-flows step by step and
  ***Step 3b:* by concatenating the intermedite service curves**
Step 4: Compute the end-to-end delay bound

Where to cut?

Cutting alternative 1:



Cutting alternative 2:

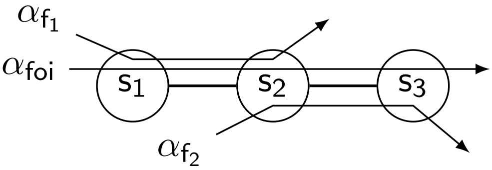


Neither alternative is strictly better than the other

Background

Network Calculus – FIFO Analysis

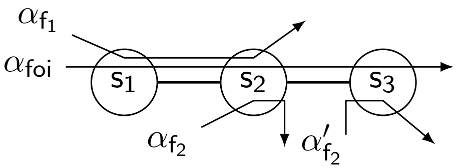
How to derive an end-to-end delay bound?



LUDB – Least Upper Delay Bound
[Bisti et al., 2008, Bisti et al., 2012]

***Step 1:* Derive all cuts creating nested subtandems**
Step 2: Compute the nesting trees
Step 3: Compute an end-to-end service curves
    Step 3a: by removing cross-flows step by step and
    ***Step 3b:* by concatenating the intermedite service curves**
Step 4: Compute the end-to-end delay bound

What's the problem with cutting alternative 2?



$$h(\alpha_{foi}, ((\beta_1 \otimes (\beta_2 \ominus \alpha_2)) \ominus \alpha_1) \otimes (\beta_3 \ominus (\alpha_2 \oslash (\beta_2 \ominus ((\alpha_{foi} + \alpha_1) \oslash \beta_1)))))$$

with

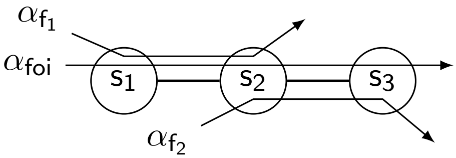
$$\text{convolution: } (f \otimes g)(d) = \inf_{0 \leq u \leq d} \{f(d-u) + g(u)\}$$

$$\text{deconvolution: } (f \oslash g)(d) = \sup_{u \geq 0} \{f(d+u) - g(u)\}$$

Contribution

Network Calculus – LUDB and Flow Prolongation

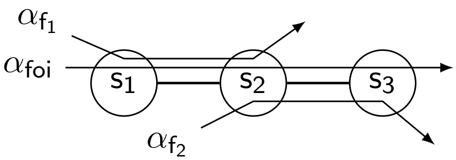
What can we do about it?



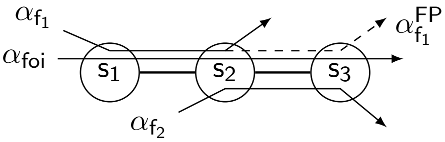
Contribution

Network Calculus – LUDB and Flow Prolongation

What can we do about it?



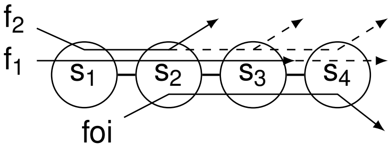
Create a nested tandem in a different way, before we face the cutting-problem!



$$h(\alpha_{foi} + \alpha_1, \beta_1 \otimes ((\beta_2 \otimes \beta_3) \ominus \alpha_2))$$

Does it Scale?

Flow prolongation in general does not [Bondorf, 2017], e.g., see:

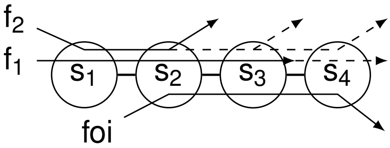


Contribution

Network Calculus – LUDB and Flow Prolongation

Does it Scale?

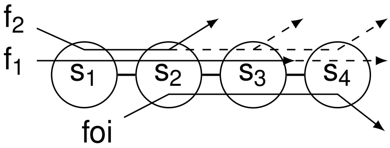
Flow prolongation in general does not [Bondorf, 2017], e.g., see:



Not if you try to search exhaustively,
not even when considering the objective to convert non-nested tandems to nested tandems.

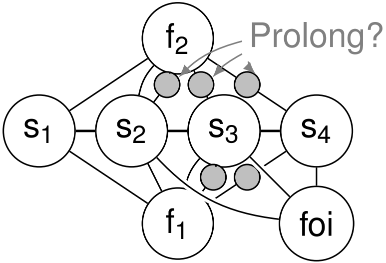
Does it Scale?

Flow prolongation in general does not [Bondorf, 2017], e.g., see:



Not if you try to search exhaustively,
not even when considering the objective to convert non-nested tandems to nested tandems.

Thus, we converted the tandem into a Graph Neural Network:



We call the new analysis *DeepFP*.

Heuristic based on Graph Neural Networks

Graph Neural Networks – Introduction

Graph Neural Networks [Scarselli et al., 2009] and related architectures are able to process general graphs and predict feature of nodes $\mathbf{o}_v$

Principle

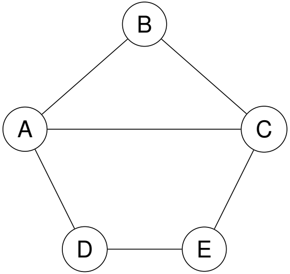
- Each node has a *hidden* vector $\mathbf{h}_v \in \mathbb{R}^k$
- . . . computed according to the vector of its neighbors
- . . . and are propagated through the graph

Algorithm

- Initialize $\mathbf{h}_v^{(0)}$ according to features of nodes
- for $t = 1, \ldots, T$ do
  - $\mathbf{a}_v^{(t)} = AGGREGATE\left(\left\{\mathbf{h}_u^{(t-1)} \mid u \in Nbr(v)\right\}\right)$
  - $\mathbf{h}_v^{(t)} = COMBINE\left(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\right)$
- return $READOUT\left(\mathbf{h}_v^{(T)}\right)$

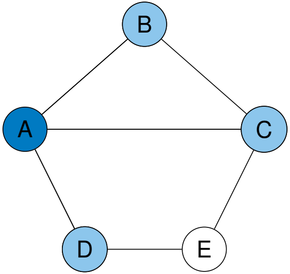
Heuristic based on Graph Neural Networks

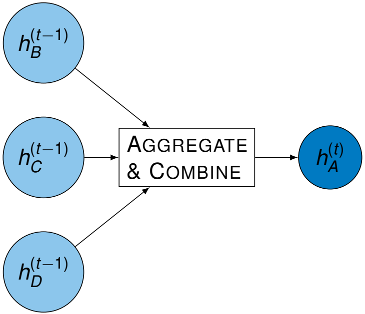
Graph Neural Networks – Illustration



Heuristic based on Graph Neural Networks

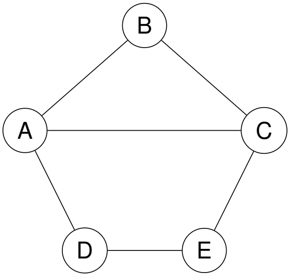
Graph Neural Networks – Illustration

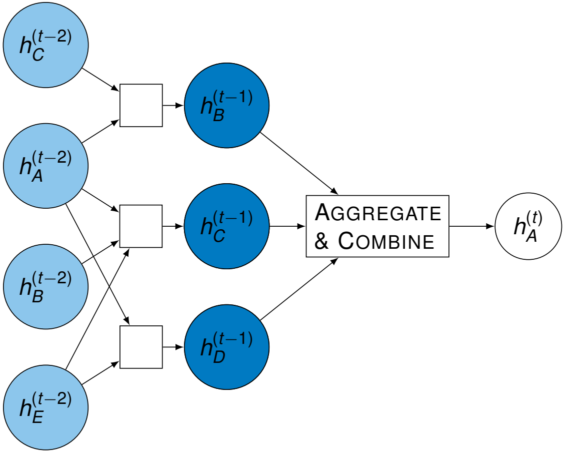




Heuristic based on Graph Neural Networks

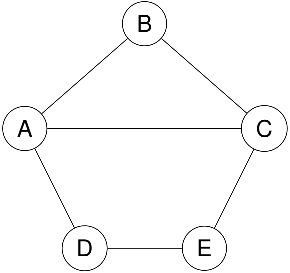
Graph Neural Networks – Illustration

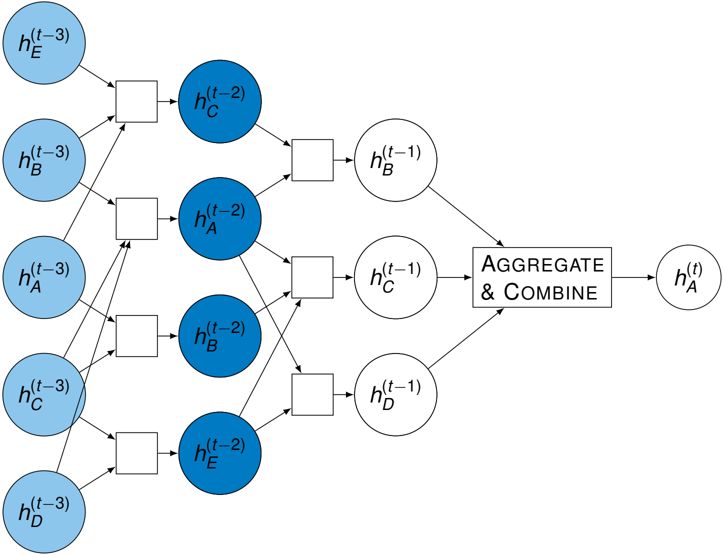




Heuristic based on Graph Neural Networks

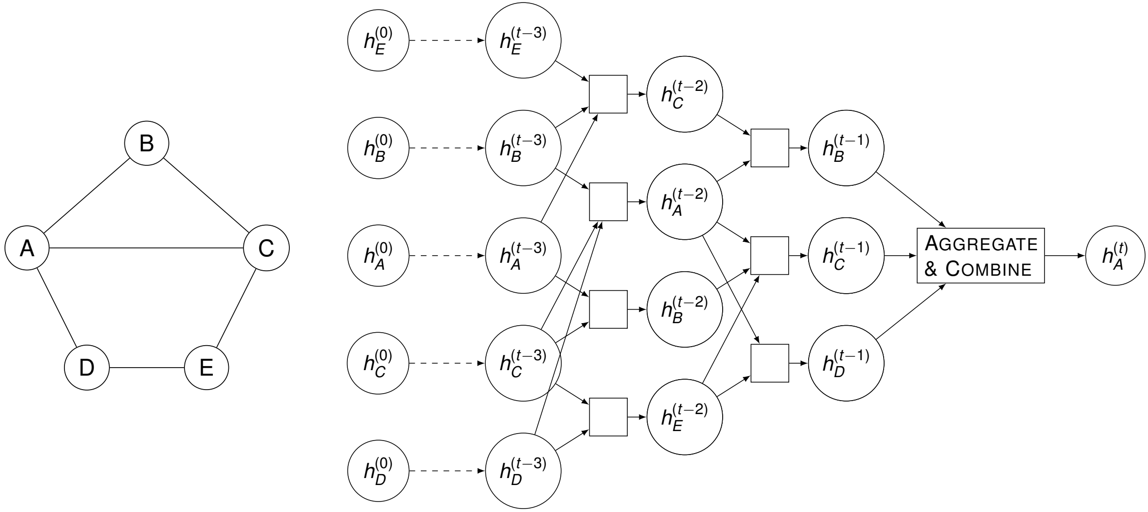
Graph Neural Networks – Illustration





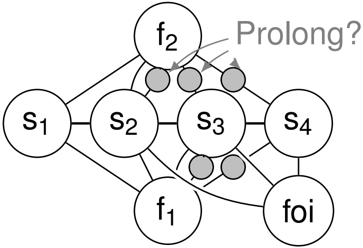
Heuristic based on Graph Neural Networks

Graph Neural Networks – Illustration



Contribution

Network Calculus – LUDB and Flow Prolongation and Predictions

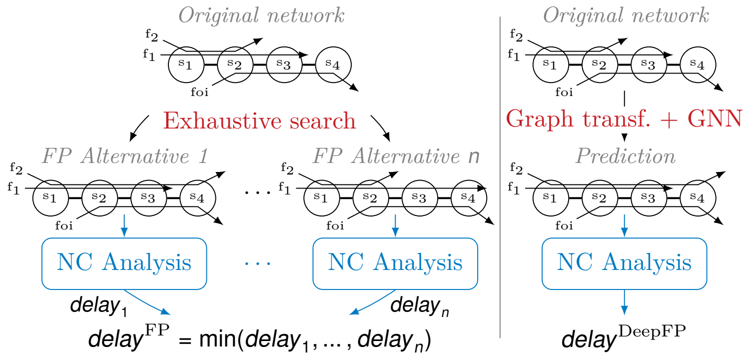


DeepFP$_n$

- Converts the Network Calculus graph into a GNN network
- Predicts a score for each prolongation node, ranking the top prolongation choices
- Let's Network Calculus pick the top $n \geq 1$ combinations of prolongations, to compute n valid delay bounds

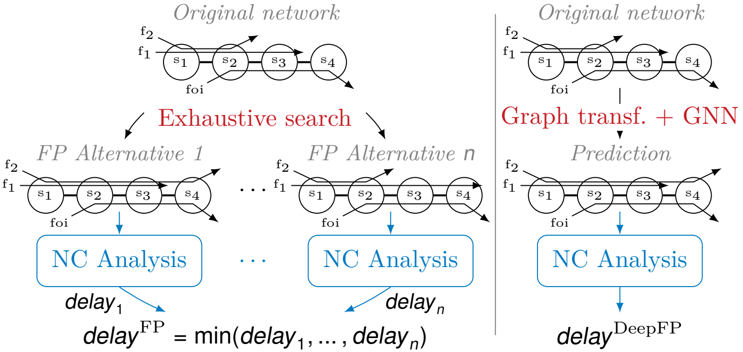
DeepFP Overview

... and Related Work



DeepFP Overview

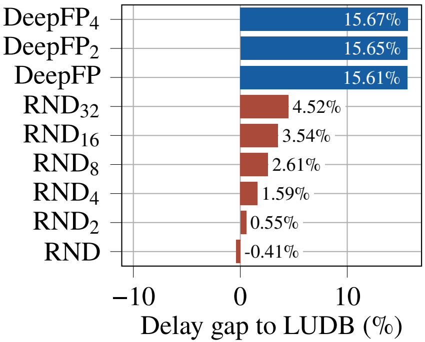
... and Related Work

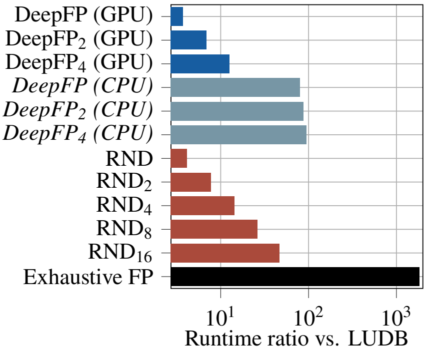


Related Work on NC + GNN:
[Geyer and Carle, 2018,
Geyer and Bondorf, 2019,
Geyer and Bondorf, 2020],
all of which focuses on the
complexities in FIFO systems.

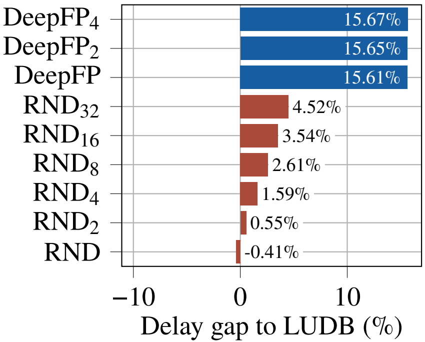
Evaluation

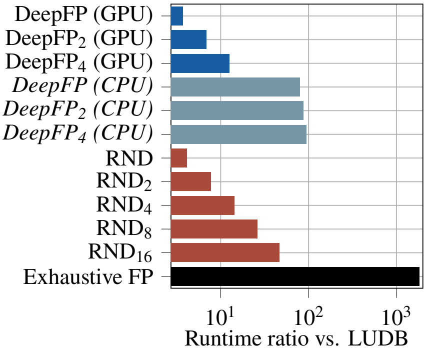
Benchmark to LUDB and Random Heuristic





Evaluation





That's it, thank you for your attention!

Bibliography

[Bisti et al., 2008] Bisti, L., Lenzini, L., Mingozzi, E., and Stea, G. (2008).
Estimating the worst-case delay in FIFO tandems using network calculus.
In *Proc. of ICST ValueTools*.

[Bisti et al., 2012] Bisti, L., Lenzini, L., Mingozzi, E., and Stea, G. (2012).
Numerical analysis of worst-case end-to-end delay bounds in FIFO tandem networks.
Real-Time Systems, 48(5):527–569.

[Bondorf, 2017] Bondorf, S. (2017).
Better bounds by worse assumptions – improving network calculus accuracy by adding pessimism to the network model.
In *Proc. of IEEE ICC*.

[Cruz, 1991] Cruz, R. L. (1991).
A calculus for network delay, part I: Network elements in isolation.
IEEE Trans. Inf. Theory, 37(1):114–131.

[Geyer and Bondorf, 2019] Geyer, F. and Bondorf, S. (2019).
DeepTMA: Predicting effective contention models for network calculus using graph neural networks.
In *Proc. of IEEE INFOCOM*.

[Geyer and Bondorf, 2020] Geyer, F. and Bondorf, S. (2020).
On the robustness of deep learning-predicted contention models for network calculus.
In *Proc. of IEEE ISCC*.

[Geyer and Carle, 2018] Geyer, F. and Carle, G. (2018).
The case for a network calculus heuristic: Using insights from data for tighter bounds.
In *Proc. of NetCal*.

[Scarselli et al., 2009] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009).
The graph neural network model.
IEEE Trans. Neural Netw., 20(1):61–80.