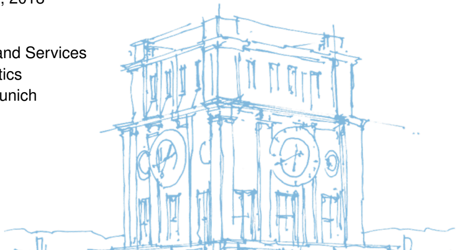
Deterministic Network Calculus Analysis of Multicast Flows

Fabien Geyer – `fgeyer@net.in.tum.de`
Work done in collaboration with Steffen Bondorf

4th Workshop on Network Calculus (WoNeCa-4)

Wednesday 28th February, 2018

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich



Motivation

Why studying multicast flows (one source to many destinations)?

- Some network architectures and protocols are heavily based on multicast communication (eg. AFDX)
- The traditional view in network calculus is that flows are unicast

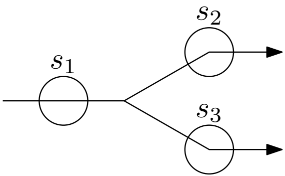


Figure 1: Illustrative example

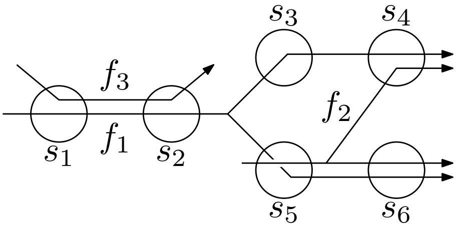


Figure 2: Illustrative example

Naming convention

- A multicast flow is made of multiple trajectories, one for each destination
- Locations where packets are duplicated are called forks.

Motivation

Unicast-based methods – Unicast Feed-Forward Analysis

- Option 1: Transform all trajectories in individual flows

$\rightarrow$ Unnecessary resource utilization at each server

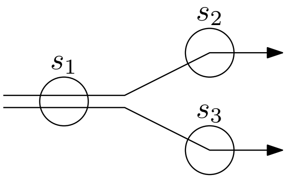


Figure 3: Illustrative example

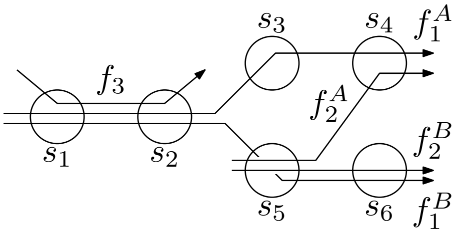


Figure 4: Illustrative example

Motivation

Unicast-based methods – Explicit Intermediate Bounds (EIB)

- Option 2: Split flows in multiple subflows according to their trajectories

→ Not a true end-to-end analysis

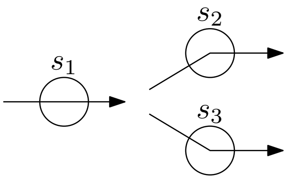


Figure 5: Illustrative example

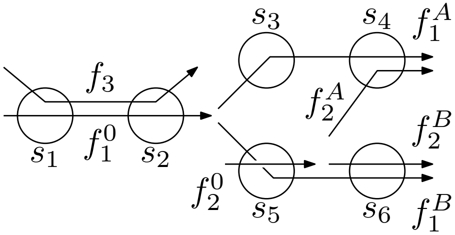


Figure 6: Illustrative example

Outline

Multicast Feed-Forward Analysis

Goals

- Have a true end-to-end method
- Do not require flow or network transformation
- Benefit from PMOO and PBOO principles

$\rightarrow$ Welcome **mcastFFA**: Multicast Feed-Forward Analysis

General idea

- Reduce the network to relevant servers as well as (partial) flows and multicast flow trajectories

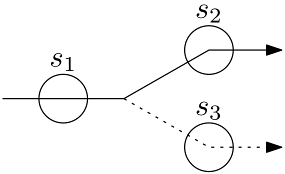


Figure 7: Illustrative example

Multicast Feed-Forward Analysis

Principles

Concepts

- Iterate over all *n* trajectory of interest and execute separate analyses
- Identify flows' inter-dependencies by traversing the network in the opposite direction of links
- Derive the sub-network relevant to a specific trajectory
- Use standard feed-forward techniques on this sub-network

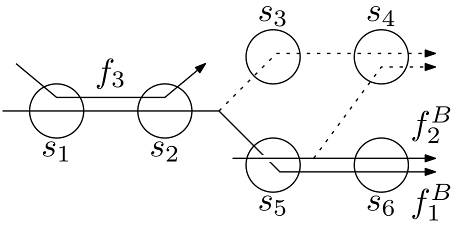


Figure 8: Illustrative example – First trajectory

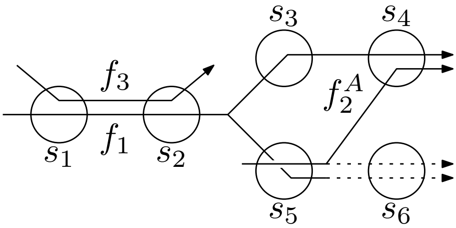


Figure 9: Illustrative example – Second trajectory

Application to f_2 – First trajectory

PBOO:

$$\beta^{\text{l.o.}f_2^B} = \beta^{\text{l.o.}f_2^B}_{\langle 5,6\rangle} \qquad \text{(cut enforced by SFA, no single-tandem analysis)}$$

$$= \beta^{\text{l.o.}f_2^B}_5 \otimes \beta^{\text{l.o.}f_2^B}_6 = \left(\beta_5 \ominus \alpha_5^{f_1^B}\right) \otimes \left(\beta_6 \ominus \alpha_6^{f_1^B}\right)$$

$$= \left(\beta_5 \ominus \left(\alpha^{f_1} \oslash \beta^{\text{l.o.}f_1}_{\langle 1,2\rangle}\right)\right) \otimes \left(\beta_6 \ominus \left(\alpha^{f_1} \oslash \beta^{\text{l.o.}f_1^B}_{\langle 1,2,5\rangle}\right)\right)$$

PMOO:

$$\beta^{\text{l.o.}f_2^B} = \beta^{\text{l.o.}f_2^B}_{\langle 5,6\rangle} \qquad \text{(there is no enforced cut)}$$

$$= (\beta_5 \otimes \beta_6) \ominus \alpha_5^{f_1} = (\beta_5 \otimes \beta_6) \ominus \left(\alpha^{f_1} \oslash \beta^{\text{l.o.}f_1}_{\langle 1,2\rangle}\right)$$

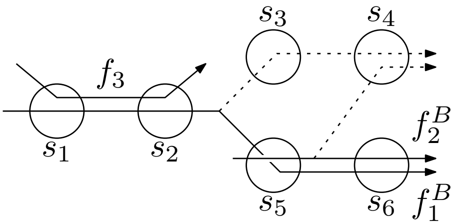


Figure 10: Illustrative example – First trajectory

with $(\beta \ominus \alpha)(d) = \sup\{(\beta - \alpha)(u) \mid 0 \leq u \leq d\}$ denoting the non-decreasing upper closure of $(\beta - \alpha)(d)$

Multicast Feed-Forward Analysis

Application to f_2 – Second trajectory

PBOO:

$$\beta^{\text{l.o.}f_2^A} = \beta^{\text{l.o.}f_2^A}_{\langle 5,4\rangle} \quad \text{(only single-hop interference so cutting is fine)}$$

$$= \beta_5^{\text{l.o.}f_2^A} \otimes \beta_4^{\text{l.o.}f_2^A} = \left(\beta_5 \ominus \alpha_5^{f_1^B}\right) \otimes \left(\beta_4 \ominus \alpha_4^{f_1^A}\right)$$

$$= \left(\beta_5 \ominus \left(\alpha^{f_1} \oslash \beta^{\text{l.o.}f_1}_{\langle 1,2\rangle}\right)\right) \otimes \left(\beta_4 \ominus \left(\alpha^{f_1} \oslash \beta^{\text{l.o.}f_1^A}_{\langle 1,2,3\rangle}\right)\right)$$

PMOO:

A cut of $\beta^{\text{l.o.}f_1^A}_{\langle 1,2,3\rangle}$ into $\beta^{\text{l.o.}f_1}_{\langle 1,2\rangle} \otimes \beta_3^{\text{l.o.}f_1^A}$ was needed in the EIB analysis, meaning that PMOO could not be implemented.

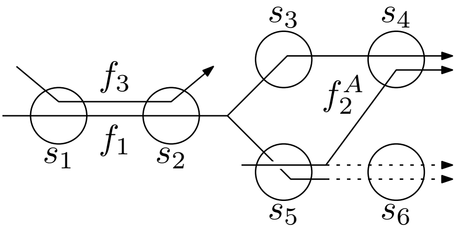


Figure 11: Illustrative example – Second trajectory

Comparison to (Non-)Network Calculus Approaches

Comparison against Trajectory Approach (TA) [1] and Forward End-To-End Delay Approach (FA) [3]

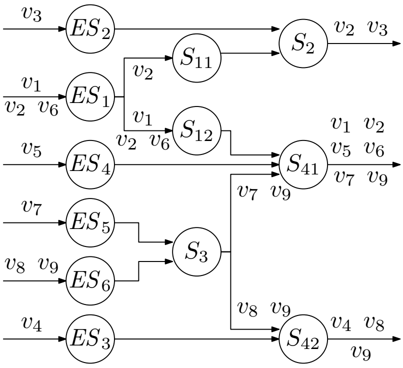


Figure 12: Evaluated AFDX network

Flow	[3] TA	[3] FA	u. trans. PMOO	EIB TFA	EIB SFA	EIB PMOO	**mcastFFA** SFA	**mcastFFA** PMOO
v_1	142	192	142	182	182	142	182	**122**
$v_2(S_2)$	**122**	**122**	142	**122**	**122**	**122**	**122**	122
$v_2(S_{41})$	**142**	192	**142**	182	182	162	182	**142**
v_3	66	**56**	**56**	**56**	**56**	**56**	**56**	**56**
v_4	**56**	66	**56**	**56**	**56**	**56**	**56**	**56**
v_5	106	106	**96**	**96**	**96**	**96**	**96**	**96**
v_6	142	192	142	182	182	142	182	**122**
v_7	-	152	142	142	142	142	142	**132**
v_8	**92**	122	102	112	112	102	112	**92**
$v_9(S_{41})$	-	162	142	152	152	142	152	**132**
$v_9(S_{42})$	**92**	122	102	112	112	102	112	**92**

Table 1: Delay bounds (values given in μs, best in bold).

Evaluation

An Industry-scale AFDX Data Network

Evaluation on AFDX-like network of 650 multicast flows with 1112 trajectories in total.

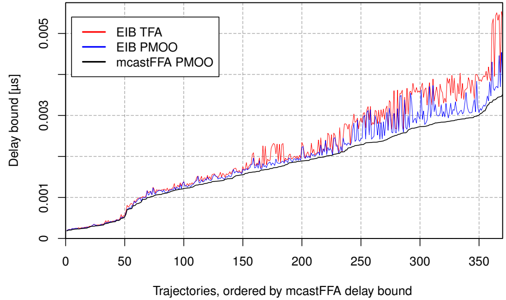


Figure 13: Delays on evaluated AFDX network

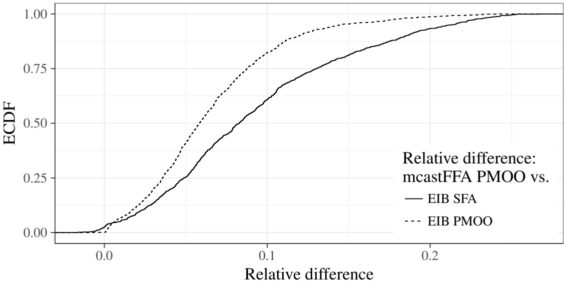


Figure 14: Comparison between methods on evaluated AFDX network

Conclusion

Contributions

- Analysis of existing restrictions in network calculus due to unicast flow model
- Proposition of mcastFFA: analysis of multicast flows with deterministic network calculus
- Implementation in DiscoDNC (thanks to Bruno Cattelan)

Numerical evaluation

- Match or better results compared to related work
- Promising gains on realistic AFDX use-case

Note: Talk based on contribution at Valuetools 2016 [2] and further extensions

Bibliography

[1] H. Bauer, J. L. Scharbarg, and C. Fraboul.
Applying and optimizing trajectory approach for performance evaluation of afdx avionics network.
In *IEEE Conference on Emerging Technologies Factory Automation, 2009. ETFA 2009*, pages 1–8, Sept. 2009.

[2] S. Bondorf and F. Geyer.
Generalizing Network Calculus Analysis to Derive Performance Guarantees for Multicast Flows.
In *Proceedings of the 10th International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS 2016, Oct. 2016.

[3] G. Kemayo, N. Benammar, F. Ridouard, H. Bauer, and P. Richard.
Improving AFDX End-to-End Delays Analysis.
In *Emerging Technologies Factory Automation (ETFA), 2015 IEEE 20th Conference on*, pages 1–8, Sept. 2015.