Demo Abstract: Worst-Case Performance Analysis with the Disco Deterministic Network Calculator

Alexander Scheffler, Steffen Bondorf, Jens B. Schmitt
Distributed Computer Systems (DISCO) Lab, TU Kaiserslautern, Germany

***Abstract*—Network calculus can derive worst-case bounds on performance metrics in communication networks. That is, the end-to-end delay experienced by a flow as well as the buffer requirement of the crossed servers. This knowledge proved to be useful for different purposes where determinism is decisive, e.g., certification of the safety-critical avionics systems. We provide a tool for automated network calculus analysis, the Disco Deterministic Network Calculator (DiscoDNC). In this demonstration, we provide a representative workflow illustrating its capabilities.**

I. Introduction

Deterministic Network Calculus (DNC) provides an algebraic framework for accurate formal worst-case performance analysis of data networks [1], [2]. It can derive upper bounds on two key values: 1) the end-to-end delay of a flow crossing the network and 2) the backlog building up in crossed servers' queues. The former bounds are used to validate against reporting deadlines in response-time critical systems, the latter enables dimensioning of buffers in order to prevent packet loss due to overflows. Most prominently, DNC was applied in the avionics sector to model and analyze AFDX (Avionics Full Duplex Switched Ethernet) and to certify AFDX backbone networks as found in the Airbus A380 or Boeing 787. To facilitate network calculus-based evaluations, we have developed and maintain an open-source tool, the Disco Deterministic Network Calculator (DiscoDNC) [3], [4], [5].

II. Deterministic Network Calculus Background

The basic concepts underlying DNC are curves that bound resource demand and availability as well as algebraic operations that transform these curves. To be precise, arrival curves $\alpha(d)$ upper bound the worst-case cumulative arrivals caused by a data flow during any observation of duration d. Service curves $\beta(d)$ complement this concept by lower bounding the guaranteed forwarding service provided by a server. Networks of servers are crossed by data flows whose arrivals are usually only bounded at their point of entry to the network. Therefore, DNC provides $(\min,+)$-algebraic operations that transform arrival curves and service curves, e.g., to aggregate multiple data flows or to derive the minimum service available to a specific flow. A brief overview operations is given in Table I, a detailed treatment can be found in [1].

Curve transformations within a network are subject to the entanglement of the involved flows. Although the network needs to be well-defined and cycle-free, there still are degrees of freedom that permit different orders of operations. The objective of a DNC analysis is thus to find an order that computes a valid and accurate delay or backlog bound. There are various algebraic DNC analyses provided by the literature. Most notable are Separate Flow Analysis (SFA) and Pay Multiplexing Only Once analysis (PMOO) (none of which is strictly best) as well as the recent generalization to Tandem Matching Analysis (TMA) [2] that enumerates all permissible orders of operations and is therefore strictly the best. These analyses strive to derive a single flow of interest's end-to-end left-over service curve $\beta^{\text{l.o.}f}_{\text{e2e}}$ that is used to compute its delay bound, subject to its cross-traffic entanglement.

Table I: Basic network calculus operations.

Operation		Description
$\beta_1 \otimes \beta_2$	Convolution	Concatenate two servers into a single one – assuming worst combined service provisioning
$\alpha_1 \oslash \beta_1$	Deconvolution	Bound the output of a system – create the largest output arrival curve for flow 1
$\beta_1 \ominus \alpha_1$	Left-over Service Curve	Derive server 1's minimum residual service curve as available to other arriving flows – assuming worst possible multiplexing behavior
$\alpha_1 + \alpha_2$	Aggregation	Aggregate flows to a single one – assuming worst entanglement of data arrivals

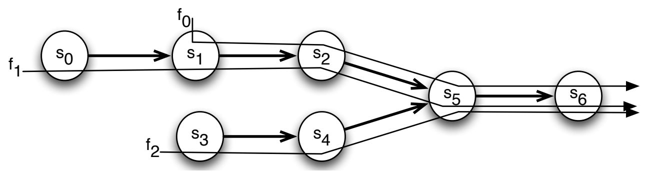


Figure 1: Sink-tree network.

Complexity and Effort: The DiscoDNC provides implementations of all of the analyses mentioned above, necessity of which we demonstrate in this section. The complexity of a DNC analyses depends on the analyzed network's size and the flows crossing it. The rather simple SFA end-to-end left-over service curve for flow f_1 in Figure 1's sink-tree is

$$\begin{aligned}\beta^{\text{l.o.}f_1}_{\text{e2e}} \quad = \quad & \beta_{s_0} \otimes (\beta_{s_1} \ominus \alpha^{f_0}) \otimes (\beta_{s_2} \ominus (\alpha^{f_0} \oslash \beta_{s_1})) \otimes \\ & (\beta_{s_5} \ominus (((\alpha^{f_0} \oslash \beta_{s_1}) \oslash \beta_{s_2}) + ((\alpha^{f_2} \oslash \beta_{s_3}) \oslash \beta_{s_4}))) \otimes \\ & (\beta_{s_6} \ominus ((((\alpha^{f_0} \oslash \beta_{s_1}) \oslash \beta_{s_2}) + ((\alpha^{f_2} \oslash \beta_{s_3}) \oslash \beta_{s_4})) \oslash \beta_{s_5}))\end{aligned}$$

where α^{f_n} denotes the arrival curve of flow f_n, $n \in \{0, 1, 2\}$, and β_{s_i}, $i \in \{0, \ldots 6\}$, is the service curve of server s_i. All these are assumed to be known. Even in this small sink-tree network with a rather simple entanglement of cross-flows f_0 and f_2, 20 $(\min,+)$-algebraic operations are required. The TMA [2] and further additions to DNC [6], [7], [8] that are implemented in the DiscoDNC even increase this number. Thus, fast and up-to-date tool support is essential to DNC.

III. The Disco Deterministic Network Calculator

The DiscoDNC provides a Java implementation of the algebraic DNC framework presented in Section II – from simple curves to complex analyses. We chose to restrict our implementation to the practical class of piece-wise linear curves that can represent token-bucket traffic regulation, rate-latency service guarantees, or approximate measured traffic [9] accurately. We provide implementations of the DNC operations of Table I tailored to this restrictions on curve shapes. We also provide the classes to create a network and automated procedures for the analyses mentioned above (SFA, PMOO and TMA). Thus, a user only needs to model the network and select the analysis to be executed. Expert knowledge about permissible orders of DNC operations is not required. Yet, as algebraic DNC is a modular performance analysis where intermediate results potentially possess valuable information, we also offer access to them. That enables, e.g., to identify performance bottlenecks like saturated servers. The Disco Deterministic Network Calculator is published under a permissive open source license and its progress can be traced online [10].

Tool Usage Demonstration

The DiscoDNC provides a set of supplementary, well-documented tests that demonstrate creation and analysis of networks. For instance, it includes the network shown in Figure 1. The following code snippets illustrate the general proceeding when creating this network manually and computing f_1's SFA delay bound (including $\beta_{\text{e2e}}^{\text{l.o.}f_1}$ from above) automatically:

1) Code 1 depicts the network creation.
2) Code 2 provides the steps to run the analysis: Create an instance of the SFA for the network and start it for flow f_1.
3) Code 3 shows how to access the derived delay bound.

```
Network network = new Network();

Server s0 = network.addServer(service_curve);
Server s1 = network.addServer(service_curve);
...
Server s6 = network.addServer(service_curve);

network.addLink(s0,s1);
network.addLink(s1,s2);
...
network.addLink(s5,s6);

                        // shortest path routing from source to sink
Flow f0 = network.addFlow(arrival_curve,s1,s6);
Flow f1 = network.addFlow(arrival_curve,s0,s6);
Flow f2 = network.addFlow(arrival_curve,s3,s6);
```

Code 1: Creating the network shown in Figure 1.

```
sfa = new SeparateFlowAnalysis(network);
sfa.performAnalysis(f1);
```

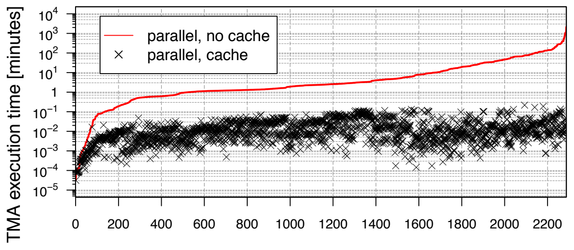
Code 2: Starting an analysis.

```
double delay_bound = sfa.getDelayBound();
```

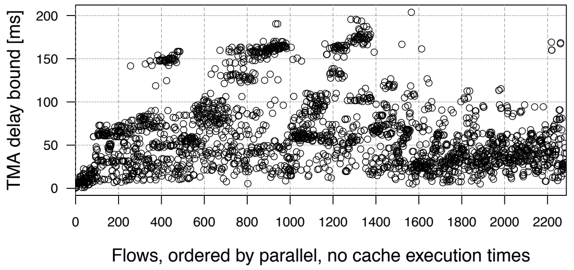
Code 3: Accessing the delay bound.

The main purpose of this tool demonstration is, however, to demonstrate the degree of automation we can achieve as well as the reproducibility of results mentioned in [5]. To that end, we present a workflow to visualize the delay bounds that were computed during the run-time measurements presented in [5]. It consists of instantiating an existing network taken from [2], automatically running multiple analyses, optionally paired with various recently proposed feature additions [7], [8], applied to multiple flows, storing all the computed delay bounds to the file system, and visualizing the results with GNU R. A similar workflow to visualize the run-time measurements is demonstrated as well. Results are shown in Figure 2 and reveal that length of analysis run-times and magnitude of resulting delay bounds do not necessarily correlate.

Requirements: The DiscoDNC workflow can be executed on a standard computer with Java 8, Java IDE and GNU R.



(a) Flow analysis run-times when executing the TMA.



(b) Flow delay bounds when executing the TMA.

Figure 2: Example GLP network from [2] with 160 devices, 572 servers and 2288 flows. Code is available online [10].

References

[1] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer, 2001.

[2] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Quality and cost of deterministic network calculus – design and evaluation of an accurate and fast analysis," *Proc. ACM Meas. Anal. Comput. Syst. (POMACS)*, vol. 1, no. 1, pp. 16:1–16:34, 2017.

[3] S. Bondorf and J. B. Schmitt, "The DiscoDNC v2 – a comprehensive tool for deterministic network calculus," in *Proc. of EAI ValueTools*, 2014.

[4] [Online]. Available: http://discodnc.cs.uni-kl.de/

[5] A. Scheffler, M. Fögen, and S. Bondorf, "The deterministic network calculus analysis: Reliability insights and performance improvements," in *Proc. of IEEE CAMAD*, 2018.

[6] S. Bondorf and J. B. Schmitt, "Calculating accurate end-to-end delay bounds – you better know your cross-traffic," in *Proc. of EAI ValueTools*, 2015.

[7] S. Bondorf, "Better bounds by worse assumptions - improving network calculus accuracy by adding pessimism to the network model," in *Proc. of IEEE ICC*, 2017.

[8] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Catching corner cases in network calculus – flow segregation can improve accuracy," in *Proc. of GI/ITG MMB*, 2018.

[9] D. E. Wrege and J. Liebeherr, "Video traffic characterization for multimedia networks with a deterministic service," in *Proc. of IEEE INFOCOM*, 1996.

[10] [Online]. Available: https://github.com/NetCal/