The DISCO Stochastic Network Calculator Version 1.0 – When Waiting Comes to an End

Michael A. Beck
University of Kaiserslautern
Distributed Computer Systems Lab (DISCO)
Germany
beck@cs.uni-kl.de

Jens Schmitt
University of Kaiserslautern
Distributed Computer Systems Lab (DISCO)
Germany
jschmitt@cs.uni-kl.de

ABSTRACT

The stochastic network calculus (SNC) is a recent methodology to analyze queueing systems in terms of probabilistic performance bounds. It complements traditional queueing theory and features support for a large set of traffic arrivals as well as different scheduling algorithms. So far, there had been no tool support for SNC analyses. Therefore, we present the DISCO Stochastic Network Calculator (DISCO-SNC) version 1.0, a Java library supporting the modelling and analysis of feedforward queueing networks using the SNC. The DISCO-SNC allows to calculate probabilistic delay and backlog bounds given a feedforward topology consisting of work-conserving servers and a set of flows traversing the network. While the DISCO-SNC is still in its infancy it is designed in a modular fashion to allow for an easy extension of, e.g., traffic types and scheduling algorithms; furthermore, it performs the optimization of free parameters as they usually appear during SNC analyses due to the application of the Chernoff bound or Hölder inequality. Apart from this core functionality, the DISCO-SNC also provides a flexible GUI to make the SNC accessible even for SNC-unexperienced users.

General Terms

Theory

Keywords

network calculus, tool suport, performance bounds

1. INTRODUCTION

The stochastic network calculus (SNC) emerged as a useful alternative methodology to analyze queueing networks [11, 2, 6, 8]. The SNC can circumvent technical difficulties in scenarios involving, e.g., non-Poisson traffic or complex statistical dependencies by resorting to inequalities instead of striving for exact results. Thus, it allows to compute rigorous probabilistic bounds on performance measures such as delay, backlog, and throughput, or, even the capacity of networks.

Mathematically, it is related to the theory of effective bandwidth [9] and its older "brother", the deterministic network calculus (DNC) [10]; in fact, it could also be viewed as their combination in a uniform (and generalized) framework. The SNC can deal with a large set of different arrival models (Poisson, Markov-modulated fluids, fractional Brownian motion, to name a few) and also allows to track per-flow performance under different scheduling disciplines such as FIFO, SPQ, GPS, and EDF. Furthermore, and may be most outstanding, it allows for an elegant end-to-end analysis for what is called convolution-form networks [4], i.e., queueing networks for which the end-to-end service can be expressed in terms of a (min-plus) convolution of the service processes at each server. Convolution-form networks hold the promise to be characterize a larger set of queueing networks than the classical product-form networks known from traditional queueing theory.

The main approaches to formulate the SNC can be found in [11, 2, 6, 8]. Recently, the dust has settled and the relation between these different flavours of SNC has been understood better [5]. The SNC turned out to be very useful in many applications; here we name a few less obvious use cases: overloaded FIFO queues [3], power grid [16], airplane cabin network [12].

Despite all this progress and development of the SNC, there has been no publicly available tool support for the SNC so far. The only tools available are for the DNC [1] [14, 1, 13, 15]. This is somewhat surprising as SNC analyses can benefit even more from tool support, because besides the analysis of larger scenarios which cannot be done manually any more there is also the issue of numerical optimization of free parameters. These free parameters are the result from numerous invokations of "approximations" like the Chernoff bound or Hölder inequalities. Indeed, their optimization has largely been neglected in literature so far. To do so, however, requires an SNC tool again. Thereofore, we present the first publicly available SNC tool: the DISCO Stochastic Network Calculator (DISCO-SNC) version 1.0.

ValueTools'13, December 10 – 12 2013, Turin, Italy

[1] A slight, but notable exception to this statement can be found on the Properbounds project webpage: `http://www.ikt.uni-hannover.de/properbounds.html`; here, one can find two interesting programs: the Network Analyzer, a demo Java applet, and the Backlog Calculator which allows for a single node backlog analysis. While interesting, both are stand-alone programs and are not conceived as tool support yet.

2. BASICS OF SNC

As stated above there exist different formulations of the stochastic network calculus. We decided on the $\sigma(\theta), \rho(\theta)$-calculus as presented in [2, 6]. Due to its compact formulation, it allows an elegant algorithmic approach to complex scenarios.

For presentation, we briefly present the basic concepts and results of the $\sigma(\theta), \rho(\theta)$-calculus.

Definition 1. We call a sequence of non-negative, real random variables $(a_n)_{n\in\mathbb{N}}$ a *flow* and introduce its *cumulative arrivals* up to time n by:

$$A(n) := \sum_{i=1}^{n} a(i)$$

A flow is $\sigma(\theta), \rho(\theta)$-bounded for some θ, if the following moment-generating functions exist and are bounded by

$$\mathbb{E}(e^{\theta(A(n)-A(m))}) \leq e^{\theta\rho(\theta)(n-m)+\theta\sigma(\theta)}$$

for all $0 \leq m \leq n$.

Similarly to the arrivals the service is described by a bound on its moment generating function:

Definition 2. A doubly indexed stochastic process S with $0 \leq S(m,n)$ and $S(m,n) \leq S(m,n')$ for all $0 \leq m \leq n \leq n'$ is said to be a *dynamic S-server*, if for all arrivals-flows A and all $n \in \mathbb{N}_0$ holds

$$D(0,n) \geq \min_{0\leq k\leq n}\{A(0,k) + S(k,n)\},$$

where D describes the output of the dynamic S-server.

A dynamic S-server is $\sigma(\theta), \rho(\theta)$-bounded for some θ, if the following moment-generating functions exist and are bounded by

$$\mathbb{E}(e^{-\theta S(m,n)}) \leq e^{\theta\rho(\theta)(n-m)+\theta\sigma(\theta)}$$

for all $0 \leq m \leq n$.

For analysing feed-forward networks, it is necessary to calculate leftover service and output bounds. For the former, imagine a service element to handle two flows. We can compute a $\sigma(\theta), \rho(\theta)$-bound for the service the low-priority arrivals receive:

THEOREM 1. *Assume two flows A_1 and A_2 and a dynamic S-server handling both flows at the same time. This service element is a dynamic S_l-server for A_2 with*

$$S_l(m,n) := S(m,n) - A_1(m,n)$$

In the case of A_1 and S being stochastically independent, we further have a $\sigma_{S_l}(\theta), \rho_{S_l}(\theta)$-bound on S_l with:

$$\sigma_{S_l}(\theta) := \sigma_{A_1}(\theta) + \sigma_S(\theta), \quad \rho_{S_l}(\theta) := \rho_{A_1}(\theta) + \rho_S(\theta)$$

Calculating bounds on the output is crucial, since it allows to treat them again as arrival flows.

THEOREM 2. *Let A be fed into a dynamic S-server. Assume A and S to be stochastically independent and $\rho_S(\theta) < -\rho_A(\theta)$. Then the output D is $\sigma(\theta), \rho(\theta)$-bounded with*

$$\sigma(\theta) := \sigma_A(\theta) + \sigma_S(\theta) - \frac{1}{\theta}\log(1 - e^{\theta\rho_A(\theta)+\rho_S(\theta)})$$

$$\rho(\theta) := \rho_A(\theta)$$

Note that in both the above theorems we did assume stochastic independence between arrivals and the dynamic S-server. In fact one can drop this assumption, but in this case Hölder parameters must be introduced. For some $\frac{1}{p} + \frac{1}{q} = 1$ the bound on the leftover service reads:

$$\sigma_{S_l} = \sigma_{A_1}(p\theta) + \sigma_S(q\theta), \quad \rho_{S_l} = \rho_{A_1}(p\theta) + \rho_S(q\theta)$$

Similarly, we get in the stochastically dependent case for the output bound:

$$\sigma(\theta) = \sigma_A(p\theta) + \sigma_S(q\theta), \quad \rho(\theta) = \rho_A(p\theta) + \rho_S(q\theta)$$

One more theorems is needed, providing us the performance bounds we are looking for:

THEOREM 3. *Let A be $\sigma_A(\theta), \rho_A(\theta)$-bounded and S be $\sigma_S(\theta), \rho_S(\theta)$-bounded. Denote by D the output of S corresponding to A. In case A is stochastically independent of S we have:*

$$\mathbb{P}(q(n) > x) \leq e^{\theta(\sigma_A(\theta)+\sigma_S(\theta)-x)} \cdot (1 - e^{\theta(\rho_A(\theta)+\rho_S(\theta))})^{-1}$$

$$\mathbb{P}(d(n) > N) \leq e^{\theta(\sigma_A(\theta)+\sigma_S(\theta)+\rho_S(\theta)N)}(1-e^{\theta(\rho_A(\theta)+\rho_S(\theta))})^{-1}$$

Here $q(n), d(n)$ denote the backlog and the virtual delay at time n, respectively. In the case of stochastic dependence between A and S one can introduce Hölder parameters. Note that the above bounds are dependent on θ and potentially on several Hölder parameters. For a reasonable bound, optimization over these parameters is needed.

3. WHAT DISCO-SNC CAN DO FOR YOU

DISCO-SNC shall allow for an easy extension to change it to its users' needs. Hence, DISCO-SNC follows a modular design, allowing to easily implement, for example, a new arrival model, without the need of dealing with the other parts of the tool. For the same reason large parts of the DISCO-SNC work on a symbolic level, with (mathematical) functions, instead of numbers. To give some insights on the DISCO-SNC, we present what modules exist and how they work together.

3.1 Arrivals

In MGF-calculus, arrivals are described by their $\sigma(\theta), \rho(\theta)$-bounds and are hence represented in the DISCO-SNC as a pair of mathematical functions. These functions are implemented symbolically. For example, a constant rate arrival is represented by $\sigma(\theta) = 0$ and $\rho(\theta) = c$. Hence adding a new arrival flow simply requires to initialize Arrival objects with the corresponding $\sigma(\theta)$ and $\rho(\theta)$. The Arrival class itself provides methods to evaluate the expression $e^{\theta\rho(\theta)(n-m)+\theta\sigma(\theta)}$ as well as constructing the $\sigma(\theta)$ and $\rho(\theta)$ of an output, given an arrival and service description, i.e., it is able to perform Theorem 2. Further an Arrival object keeps track of its stochastic dependencies to other network-elements and updates them during usage of the Theorems 1 and 2.

```
Arrival(function s, function r){
    this.r = r;
    this.s = s;}
double evaluate(t,n,m){
    return exp(t*s.getValue(t)
          +t*r.getValue(t)*(n-m));}
Arrival output(arrival, service){
```

```
apply theorem 2;}
```

Code 1: Arrival Class in Pseudocode

3.2 Service

The description of service elements parallels the one of arrivals. They are also represented by two functions $\sigma(\theta)$ and $\rho(\theta)$. The service-class provides a method to perform theorem 1 (again, while keeping track of stochastic dependencies).

3.3 Analysis

To calculate a specific performance measure at some point in the network (i.e., a backlog- or delay-bound for a specific flow at a specific node) two steps must be performed: First, a symbolic analysis is performed, resulting in expressions as seen in 3, which afterwards - in a second step - must be optimized over their free parameters. In general, the computation of several leftover service and output bounds must be performed, before 3 can be applied. We plainly call this first step *analysis*. For the sake of brevity, we do not go into the details of different analysis techniques.

The analysis step is represented by its own (abstract) class and users can modify or extend it for their own type of analysis. So far, the DISCO-SNC alreday provides a simple, yet effective analysis. The pseudo-code of it can be seen in code 2. It works as follows: Output- and service-bounds are successively computed until Theorem 3 can be used. For this a stack is used, containing all service elements, for which descriptions to all incoming arrivals are available. Service elements are popped and "serve" their arrival with the highest priority. This means output- and service bounds are calculated, which might lead to new service elements getting pushed onto the stack. This procedure is continued until the flow of interest and the service of interest is known or the stack is empty. In the former case the performance bound can be calculated. In the latter case an analysis with the methods presented in theorems 1-3 is not possible because the network/flow combination is not feed-forward. Note that the analysis takes place on a symbolic level and is just a manipulation of the σ- and ρ-functions of the involved arrivals and service elements.

```
Arrival analyze(VoI,FoI){
//Initializes the stack of vertices
for(entry : vertices){
 if(entry.canServe()) stack.push(entry);
 }
successful=false;
//Serve until FoI and SoI are characterized
while(!stack.isEmpty()){
 current_v = stack.pop();
 flowID = current_v.whoHasPriority();
 next_v = successor(current_v,flowID);
//Checks for SoI and FoI
 if(current_v == VoI && flowID == FoI){
  bound = calcBound(flowID, current_v);
  successful = true;
  break;
 }
//Calculates output leftover service
 output = current_v.serve();
 next_v.learnArrival(output);
//Pushes next vertex
  if(next_v.canServe()) stack.push(next_v);
//Pushes current vertex if needed
  if(current_v.canServe())
  stack.push(current_v);
}
return bound;
}
```

Code 2: analyze method in Pseudocode

Delta	r_{HJ}	r_S	b_{HJ}	b_S
0.05	$\approx$ 1 sec.	$\approx$ 3 min.	60.3	60.285
0.04	$\approx$ 1 sec.	$\approx$ 7 min.	53.013	52.155
0.03	$\approx$ 1 sec.	$\approx$ 37 min	52.035	52.035

Table 1: Runtimes and delay bounds of HJ heuristic and systematic (S) search at different granularities.

3.4 Optimization

As stated above, the analysis part only performs theorem 3, without actually using any numbers or even optimizing free parameters. This is done separately in a (abstract) class. Again users can implement their own type of optimization (or heuristic) by extending that class.

Sinc we change from a pure symbolic view to a numerical view, this is normally the part where runtime is crucial. A good algorithm can shorten runtime drasticallyat this point. The DISCO-SNC allows to efficiently compare optimization methods and heuristics for their performance in runtime and accuracy. To demonstrate this, we implemented two optimization methods: The first one is a plain systematic search through the entire search space. The second implementation is based on the Hooke and Jeeves (HJ) heuristic [7]: starting at some point in the search space, we check its neighbours' values and move to the best of them. This is repeated until no neighbour offers an improvement over the value of the current point. Although being prone to get stuck in local minima, in some preliminary experiments showed, that the functional structure of the performance bounds (after the symbolic analysis) often allows near-optimal results in a short runtime.

To give some intuition about the gain in runtime, we set up a toy example, consisting of three flows and seven nodes. The solution space consists here of θ and four pairs of Hölder parameters. We present the results delivered by the systematic search and the HJ heuristic in table 3.3. Each line represents a different granularity Δ on the search space, i.e., the difference between two different values for θ. For the granularity of Hölder parameters, note first that they come in pairs, of which one is always ≤ 2 and the other ≥ 2. The granularity describes in this case the difference between two choices of the smaller one of the Hölder parameter-pair.

3.5 Graphical User Interface

Last, but not least the DISCO-SNC features a simple, but useful GUI (see figure 1), allowing users, an easy access to most parts of the DISCO-SNC. It provides information about the service elements in the network as well as the arrival flows traversing it. Adding and removing service elements and arrivals can be easily done within the GUI, as well as calculating performance bounds under the selection

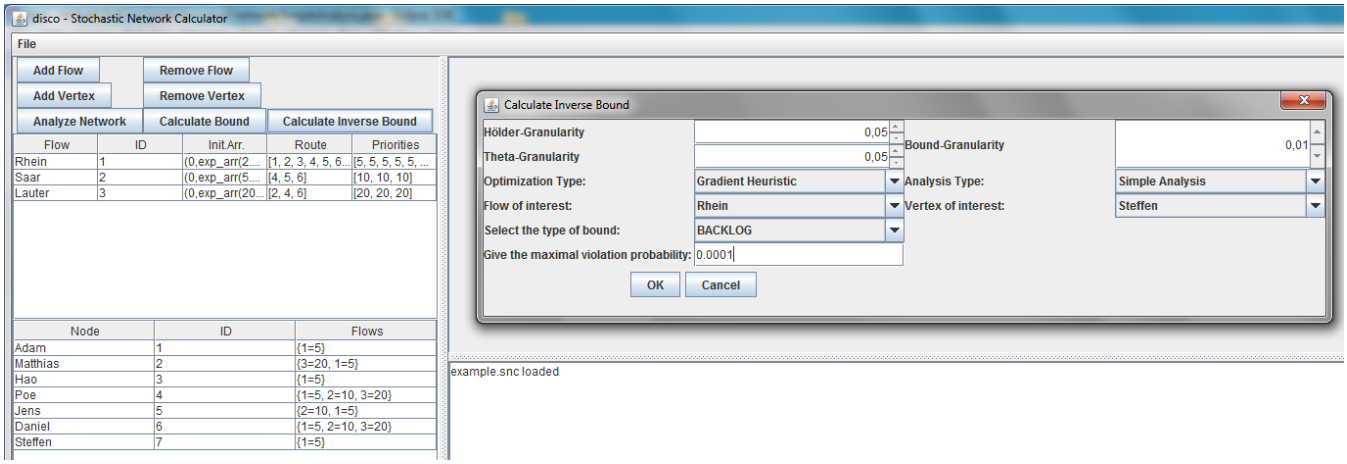

Figure 1: Screenshot of the DISCO-SNC

of the analysis and optimization method. Further loading and saving scenarios is possible.

The GUI itself is separated from the core of the program, allowing users to write their own GUI (or extending the existing one).

4. CONCLUSION AND OUTLOOK

We presented the version 1.0 of DISCO-SNC, the first publicly available tool supporting stochastic network calculus analyses. The core functionality was described and exemplified, in particular the optimization module has preliminarily been evaluated with respect to the tradeoff between quality of the bounds vs. run time.

Future work items are abundant: more traffic models, other scheduling disciplines, more sophisticated network analysis methods (similar to what is known from DNC [14]), global optimization over free prarameters, inclusion of other flavours of SNC (tail bounds), and much more. So, while this list is somewhat embarrassingly long, we still believe to have made a promising first step towards a useful tool supporting SNC analyses, hopefully helping other SNC researchers as well as raising interest from so far non-SNC researchers.

5. REFERENCES

[1] A. Bouillard and E. Thierry. An algorithmic toolbox for network calculus. *Discrete Event Dynamic Systems*, 18(1):3–49, Mar. 2008.

[2] C.-S. Chang. *Performance Guarantees in Communication Networks*. Telecommunication Networks and Computer Systems. Springer-Verlag, 2000.

[3] F. Ciucu, O. Hohlfeld, and L. Chen. On the convergence to fairness in overloaded fifo systems. In *Proc. of IEEE INFOCOM*, pages 1988–1996, 2011.

[4] F. Ciucu, J. Schmitt, and W. H. On expressing networks with flow transformations in convolution-form. In *Proc. of IEEE INFOCOM*, pages 1979–1987, 2011.

[5] F. Ciucu and J. B. Schmitt. Perspectives on network calculus: no free lunch, but still good value. *Proc. of ACM SIGCOMM*, Aug. 2012.

[6] M. Fidler. An end-to-end probabilistic network calculus with moment generating functions. In *Proc. of IEEE IWQoS*, pages 261–270, June 2006.

[7] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, Apr. 1961.

[8] Y. Jiang and Y. Liu. *Stochastic Network Calculus*. Springer, 2008.

[9] F. P. Kelly. Notes on effective bandwidths. In F. P. Kelly, S. Zachary, and I. Ziedins, editors, *Stochastic Networks: Theory and Applications*, number 4 in Royal Statistical Society Lecture Notes, pages 141–168. Oxford University Press, 1996.

[10] J.-Y. Le Boudec and P. Thiran. *Network Calculus A Theory of Deterministic Queuing Systems for the Internet*. Number 2050 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany, 2001.

[11] C. Li, A. Burchard, and J. Liebeherr. A network calculus with effective bandwidth. Technical Report CS-2003-20, University of Virginia, Nov. 2003.

[12] J. Scharbarg, F. Ridouard, and C. Fraboul. A probabilistic analysis of end-to-end delays on an afdx avionic network. *Industrial Informatics, IEEE Transactions on*, 5(1):38–49, 2009.

[13] H. Schioler, H. P. Schwefel, and M. B. Hansen. Cync: a matlab/simulink toolbox for network calculus. In *Proc. of ACM Valuetools*, pages 60:1–60:10, 2007.

[14] J. Schmitt and F. Zdarsky. The DISCO Network Calculator - a toolbox for worst case analysis. In *Proc. of VALUETOOLS*. ACM, Nov. 2006.

[15] E. Wandeler and L. Thiele. Real-Time Calculus (RTC) Toolbox. `http://www.mpa.ethz.ch/Rtctoolbox`, 2006.

[16] K. Wang, F. Ciucu, C. Lin, and S. Low. A stochastic power network calculus for integrating renewable energy sources into the power grid. *IEEE Journal on Selected Areas in Communications: Smart Grid Communications Series*, 30:1037–1048, 2012.