Abstract—

The question of how to calculate latency and buffer bounds in complex networks is becoming ever more relevant in today’s interconnected world. A number of approaches to this problem have been developed; this paper provides an introduction to the concepts as well as the application of stochastic network calculus and shortly introduces two alternative methods for calculating relevant bounds in networks: deterministic network calculus and classic queuing theory. This paper also provides an overview of various open-source tools that use these different approaches and, using two different reference topologies, compares them regarding important factors such as the tightness of bounds. While the tested tools perform similarly in uncongested networks, specific tools, such as DISCO SNC, can provide tighter bounds and greater functionality. The results of one tool, DISCO DNCv2, could not be reliably compared with those of the others.

Index Terms—stochastic network calculus, queuing theory, network performance evaluation, scheduling

1. Introduction

The increasing dependence on fast, reliable networks in academia and elsewhere necessitated the formalization of those networks. Quality of Service (QoS) guarantees regarding delay, throughput, and packet loss are especially important when measuring the performance of networks and ensuring their QoS [1]. The required guarantees vary depending on the needs of end-users and the contracts they have with their respective network provider. A user who, for example, is interested in real-time communication with other users would be less interested in the lower bounds on throughput and more interested in the upper bounds on delay; on the other hand, a user who must send large files would be more concerned with the lower bounds on throughput.

Network calculus was first introduced in 1991 by Cruz in two related papers, [2] and [3], as a new way to determine the relevant bounds inside a communication network. Since this initial publication, network calculus has developed in two branches: stochastic and deterministic network calculus. In short, deterministic network calculus provides the worst-case bounds for a given network while stochastic network calculus provides bounds based on statistical distributions and a certain level of acceptable exceedance of required bounds [4]. Deterministic network calculus is often used to model networks in which there is no tolerance for packet loss or long delays (e.g. critical infrastructure). Stochastic network calculus can be employed to model networks which can tolerate a certain amount of loss or delay in order to more closely mirror the real-life requirements of many networks and users. It can, therefore, be used to more accurately and tightly find bounds / guarantees in networks that are inherently statistical in nature [4].

2. Background

This section provides an overview of the fundamentals of (stochastic) network calculus and its similarities and differences with other theories.

2.1. Network Calculus

In order to be effectively used for network analysis (i.e. deriving relevant bounds), a theory must be characterized by the following five properties [4]:

1) Service Guarantees - Stochastic service guarantees (e.g. backlog, delay) can be derived for single nodes using a specific traffic model and a specific server model.

2) Output Characterization - The output of a server can be modeled using the same traffic model and the input.

3) Concatenation - The concatenation (i.e. convolution) of multiple servers can be modeled using the same server model.

4) Leftover Service - The service available to a traffic flow can be modeled using the same server model if multiple flows are simultaneously using the service.

5) Superposition - The superposition of multiple traffic flows can be modeled using the same traffic model.

These properties, especially the third and fourth properties, allow for the concatenation of multiple service and arrival flows; this, in turn, reduces the necessary calculations and can significantly improve the results when compared to node-to-node analysis [4].

Network calculus characterizes networks using two curves: service (server) and arrival (traffic). The arrival curve describes the traffic sent using its upper bound while the service curve defines a lower bound that a server provides [4]. Often an “envelope process” is used to describe these curves; the envelope process simply
refers to a function which deterministically bounds the
process but is not necessarily tight. One of the main
advantages of network calculus is that service curves can
be concatenated (i.e. convoluted) using min-plus algebra
and thus more effectively analyzed.

2.2. Mathematical Basics & Notation

This section provides an overview of the notation used
and, partially, the fundamental mathematical concepts as
described and used in [4] and [5].

\( \mathbf{F} \) is used to denote the set of non-negative wide-sense
increasing functions in \( \{ a(i) : \forall 0 \leq x \leq y, 0 \leq a(x) \leq a(y) \} \)
and for which it holds \( \forall x < 0 : a(x) = 0 \). This
set of functions is used to characterize arrival and service
curves as a function of time \( t \).

Min-plus algebra is used to perform operations on
flows. An important algebraic structure when using min-
plus algebra is

\[
(\mathbf{F} \cup \{+\infty\}, +, \land)
\]

which is a commutative diode with the zero element \(+\infty\)
and the identity element 0 for all \( x \geq 0 \) and \(+\infty\) otherwise
[6]. The min-plus convolution and deconvolution of two
functions, \( a \) and \( b \), are respectively defined as follows:

\[
(a \otimes b)(x) = \inf_{0 \leq y \leq x} [a(x + y) - b(y)]
\]

\[
(a \otimes b)(x) = \sup_{y \geq 0} [a(x + y) - b(y)]
\]

\( A(t) \) refers to the (cumulative) arrival process and \( A^*(t) \) to the (cumulative)
departing traffic process of a (lossless) server. The backlog \( B(t) \) is then defined as
\( A(t) - A^*(t) \). \( A_i \) and \( A^*_i \) refer respectively to the arrival
and departure models of flow \( i \) in network element \( h \).

2.3. Traffic Models

The traffic model originally introduced in [2] is the
\( (\sigma, \rho) \) traffic characterization and is deterministic. \( \sigma \) refers
to the burstiness and \( \rho \) to the rate of the traffic flow. A
popular implementation of this traffic flow is referred to as
a token bucket; a traffic flow / arrival curve \( A \) is bounded by the \( (\sigma, \rho) \) model if the following condition holds for
all \( 0 \leq s \leq t \) [4]:

\[
A(s, t) \leq \rho \cdot (t - s) + \sigma
\]

In stochastic network calculus, this model is expanded and depicted using the \( (\sigma(\theta), \rho(\theta)) \) traffic characterization
[4]. Using the moment generating function (MGF) of the
arrival curve, a bound can be derived for this traffic
characterization [4]; an arrival curve \( A \) is bounded by the
\( (\sigma(\theta), \rho(\theta)) \) model for some \( \theta \) if the following condition holds for all \( 0 \leq s \leq t \):

\[
\frac{1}{\theta} \log E[e^{\theta A(s+t)}] \leq \rho(\theta) \cdot (t) + \sigma(\theta)
\]

Stochastic network calculus can then be used to find
and optimize \( \theta \) by defining, for example, the maximum
allowed backlog and the probability with which this bound
must be respected [5]. There are a number of further variations
of the stochastic arrival curve introduced in [4] including
the traffic-amount-centric (t.a.c.) arrival curve, the
virtual-backlog-centric (v.b.c.) arrival curve, and the
maximum-backlog-centric (m.b.c.) arrival curve.

2.4. Similarities and Differences with Queuing Theory

Queuing theory was first introduced in 1909 by A.K.
Erlang and is a branch of mathematics that focuses on
the modelling of the act of waiting in line [7]; queuing
theory was thus not developed with modern communication
/ packet networks in mind. One of the most common
proposals to model arrival and service curves is the M/M/1
model using a Poisson distribution. Although many net-
works can be modeled using queuing theory, multiple of
the properties, especially 3 and 4, described in Section 2.1
cannot, in general, be concluded for queuing theory [4].
In addition, queuing theory focuses on the calculation of the
average case and not the worst case, as in network
calculation [7].

3. Application and Comparison

This section provides an overview of a diverse set of
tools as well as their defining characteristics and defines
two reference topologies. The aforementioned tools are
then used to evaluate both reference topologies; the results
of all evaluations are subsequently analyzed and compared
with one another.

3.1. Tools

The tools which are introduced in this section are all
open-source and were developed primarily for use in
research. Table 1 provides an overview of the supported
network topologies (Tandem, Tree, and Feed-forward) as
well as important network calculus operations (Convolution
and Deconvolution).

<table>
<thead>
<tr>
<th>Topology</th>
<th>Tandem</th>
<th>Tree</th>
<th>Feed-forward</th>
<th>Convolution</th>
<th>Deconvolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISCO SNC</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>DISCO DNCv2</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>SNC MGF Toolbox</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>OMNeT++</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

3.1.1. The DISCO Stochastic Network Calculator.
The DISCO Stochastic Network Calculator (DISCO SNC)
is an open-source tool developed by researchers at the
Distributed Computer Systems (DISCO) chair of the Uni-
versity of Kaiserslautern [8]. It is a modular program
developed primarily in Java for the application of stochas-
tic network calculus and also provides a graphical user
interface (GUI) in order to “make the SNC accessible
even for SNC-inexperienced users” [8]. It supports a
number of traffic characterizations including exponential,
exponentially bounded burstiness, and token bucket.

3.1.2. Stochastic Network Calculus Moment Generating Function Toolbox.
The Stochastic Network Calculus Moment Generating Function Toolbox (SNC MGF Tool-
box) is an open-source tool developed by Paul Nikolaus,
a researcher at the DISCO chair of the University of
Kaiserslautern [9]. It is a library developed in Python for
the application of stochastic network calculus and does
not provide a GUI. It supports a number of traffic characterizations, including Markov modulated on-off traffic, exponentially bounded burstiness, and token bucket.

3.1.3. The DISCO Deterministic Network Calculator v2. The DISCO Deterministic Network Calculator v2 (Disco DNCv2) is the second-generation open-source tool developed by researchers at the DISCO chair of the University of Kaiserslautern for analyzing networks using deterministic network calculus [10].

3.1.4. Objective Modular Network Testbed in C++. Objective Modular Network Testbed in C++ (OMNeT++) is a discrete event simulator developed primarily in C++ and used for, among other things, modeling communication networks [11]. OMNeT++ was first introduced in 1997 and has since been expanded by multiple libraries, including one for the simulation of queuing networks, and a comprehensive GUI.

3.2. Reference Topologies

In order to compare the tightness of the bounds calculated by the aforementioned tools, this section defines two relatively simple network topologies which can be modeled in every tool. Figure 1 models a tandem (chain) network with three service curves; the arrival curve enters the first node and is processed by all three service curves. Figure 2 models a (fat) tree network with two arrival curves entering the two lowest nodes respectively and both departing curves being sent to the root node / service curve. S1, S2, and S3 provide a constant service rate of 5, 4.9, and 4.5 respectively in both topologies. The bounds on the departure curve(s), $A^S_{S1}$ and $A^S_{S3}$, are tested. For both the tree topology, $\lambda < \theta < \lambda$; this can be used to provide a deterministic bound on the stochastic distributions for DISCO DNCv2.

3.3. Configuration of Tools

This section documents the methods used to configure each tool and derive relevant bounds using the tandem topology and backlog bound as an example.

---

**Figure 1: Topology 1 - Tandem**

**Figure 2: Topology 2 - Fat Tree**

**3.3.1. DISCO SNC.** The following configuration can be loaded directly into the DISCO SNC GUI for $\lambda = .3$.

```java
// Configuration of Network
// Interface configuration. Unit Mbps
interfaces.add( "eth0", "MII", "CR", 10, 1000 );
Figure 2: Topology 2 - Fat Tree

Figure 1: Topology 1 - Tandem

**3.3.2. SNC MGF Toolbox.** This Python file for $\lambda = .3$ can be run in the root folder of the project once all dependencies have been added.

```python
if __name__ == '__main__':
    print("Tandem_Performance_Bounds:")
    DELAY_PROB_BOUND = PerformParameter( perform_metric=PerformEnum.DELAY_PROB, value=.05)
    Server1 = ConstantRateServer( rate=5.0 )
    Server2 = ConstantRateServer( rate=4.9 )
    Server3 = ConstantRateServer( rate=4.5 )
    ConvolvedServer = Convolve( Convolve( Server1 , Server2 ) , Server3 )
    TandemTopology = SingleServerMulPerform( arc_list=[ [Server1, 0, 1], [Server2, 1, 2], [Server3, 2, 3], EXPONENTIAL, .3 ]
    DG = ConvolvedServer,
    grid_search( bound_list=[ (0.1, 5.0) ], delta=0.1 )
```

It is important to note that SNC MGF Toolbox does not support FIFO multiplexing / scheduling and instead always uses arbitrary scheduling. Arbitrary scheduling provides a worst-case bound on any scheduler, including FIFO.

**3.3.3. DISCO DNCv2.** This Java file for can be run for $\lambda = .3$ and $\theta = .1$ in the root folder of the project once all dependencies have been added.

```java
public void run() throws Exception
    ServiceCurve service_curve_1 = Curve.getFactory( )
    .createRateLatency(5.0, 0 );
    ServiceCurve service_curve_2 = Curve.getFactory( )
    .createRateLatency(4.9, 0 );
    ServiceCurve service_curve_3 = Curve.getFactory( )
    .createRateLatency(4.5, 0 );
    ServerGraph sg = new ServerGraph( );
    Server s0 = sg.addServer( service_curve_1 , Multiplexing.FIFO );
    Server s1 = sg.addServer( service_curve_2 , Multiplexing.FIFO );
    Server s2 = sg.addServer( service_curve_3 , Multiplexing.FIFO );
    sg.addTurn(0, s1);
    sg.addTurn(s1, s2);
    System.out.println("Tandem.PERFORMANCE.Bounds:")
```
DISCO DNCv2 provides, depending on the choice of $\theta$, a bound of either infinity or 0; this is caused by the lack of burstiness of the arrival curve and latency of the service curves, which cannot be represented in the other programs. The results produced by DISCO DNCv2 are, therefore, not depicted.

### 3.3.4. OMNeT++

The variable “interArrivalTime” of the source refers to the time between generated jobs and is set to exponential($\lambda$). The variable “serviceTime” refers to the time required by the queue to serve a job and is set to $\frac{1}{\mu_1}$, $\frac{1}{\mu_2}$, and $\frac{1}{\mu_3}$ for the three queues respectively. The variable which represents the backlog bound for the entire system is the sum of the variable “queueLength:max” for all queues. The delay bound is represented by the variable “lifeTime:max” of the sink. Each simulation was run for 2,000,000 events.

### 3.4. Results & Analysis

Figures 3 and 4 depict, respectively, the backlog and delay bounds of the calculations and simulations for the tandem topology. Figures 5 and 6 depict, respectively, the backlog and delay bounds of the calculations and simulations for the tree topology.

The data, produced by the most uniform configuration of the heterogeneous tools possible, shows that the values for both the backlog and delay are similar for less congested networks, i.e. networks with larger $\lambda$ values. In more congested networks, however, DISCO SNC almost always provides significantly tighter bounds. The values produced by the OMNeT++ simulations are frequently comparable with those produced by the other tools.

### References


