DIMTOOL: A Platform for Determining Worst Case Latencies in Switched Queuing Networks

Emanuel Heidinger*†, Stefan Burger*, Stefan Schneele*, Alexander Klein†, Georg Carle†

*EADS Innovation Works
Dept. IW-SI-CO
D-81663 München, Germany
Email: {emanuel.heidinger,stefan.schneele,
stefan.burger}@eads.net

†Technische Universität München
Institut für Informatik, I-8
Boltzmannstraße 3
D-85748 Garching b. München, Germany
Email: {klein,carle}@net.in.tum.de

***Abstract*—Performance evaluation in computer networks requires consistent traffic and topology models to deliver comparable results. In this work we present our platform that provides a consistent interface by encapsulating standard parameters as scheduling strategy, link speed, processing delay, and propagation delay. In contrast to existing solutions, we provide several performance estimation techniques within a single toolbox making the identified performance results comparable. The functionality of our developed toolbox is demonstrated by employing it to real-world scenarios in avionics, which is the Ethernet based Cabin Management System and the communication network inside the cabin server.**

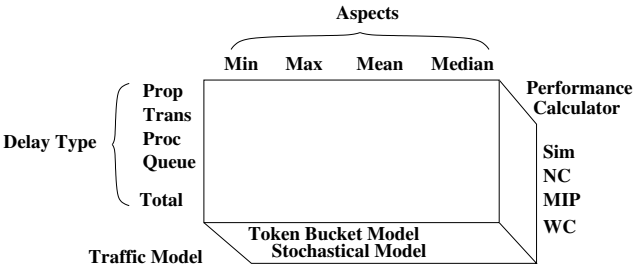

Fig. 1. Performance Calculation with DIMTOOL

I. Introduction

Automotive and aeronautics industry have to cope with evolved communication networks in vehicles and airplanes. However, there are good reasons to benefit from recent progress in the field of switched Ethernet. In aeronautics, a clear cost driver is the weight of the communication infrastructure, and so, any additional redundant link results additional weight. Transmitting more traffic over the network while possibly merging different safety domains would bring clear advantage compared to current systems. In automotive industry, the cost drivers in terms of network technology are connectors, cable and their shielding, as well as the price for the network chips. The objectives of both means of transportation differ somewhat, but the intention is clear: less weight, lower price, higher bandwidth, and yet no loss in terms of safety. Certainly, these objectives influence each other and are difficult to fulfill. Often an existing and expensive certification process does not encourage the introduction of novel communication technologies. In terms of safety, an important point that be addressed is the certification process, which differs in the certain areas of transportation. The evaluations in those processes are often hand-crafted or semi-automated. Our toolchain tries to fill that gap by providing a consistent platform for performance evaluation and calculation such that certification processes can be completed in shorter time. Figure 1 shows the aspects of our performance calculation toolbox.

The task of performance estimation and calculation yielded several approaches in the last decades. There are discrete event simulations based on Monte Carlo studies, methods that aim at the calculation of queuing delays based on queuing theory as well as its alternative Network Calculus (NC). The calculated results often fall in the category of latency, memory utilization, and link utilization. To the best of our knowledge, these tools deliver results, that cannot easily be compared since they may assume different models or constraints. In this work we present the network calculation platform DIMTOOL, which provides such a consistent view on network performance calculations.

This paper provides the following key contributions: In contrast to previous tools, we provide several performance estimators under one umbrella, thus making performance results of different methods transparent and comparable. We employ DIMTOOL to verify latency constraints in the switched aircraft cabin and show that the NC FIFO bound does not hold by Worse Case Simulation (WCS). In contrast, we did not observe the necessity of the non-FIFO bound with the established Monte Carlo method— even after several random runs.

This paper is structured as follows: Section II shows related work in the field of performance calculators. Section III provides an overview of DIMTOOL, its objectives, its architecture and the provided back-ends. The experimental results are discussed in Section IV. Section V summarizes the work and outlines further steps.

II. Related Work

In this section we provide a comprehensive overview of network performance calculation solutions. Throughout this work, network performance calculators are classified by the employed technology. More precisely, we cover network simulation based on Monte Carlo Simulation (SIM), NC analysis,

VALUETOOLS 2012, October 09-12, Cargèse, France

DOI 10.4108/valuetools.2012.250296

Model Checking (MC) approaches as well as WCS. Table I gives a comprehensive, not necessarily complete overview of available products and the underlying technology.

Comparison of results achieved from those network performance estimators is hardly possible because the traffic and network models often differ from each other, and small distinctions may have significant impact on the results. This aspect is addressed by the work shown in this paper. For this, a generic interface and wrappers are provided, that act as backends to simplify comparison of the determined performance bounds. Furthermore, we provide a Matlab front-end to the Network Calculation tools.

Computer networks are usually too complex to identify the worst case through network simulation. Popular network simulators are ns-2 [1], ns-3 [2], SSF [3], OMNeT++ [4] and OPNET [5]. The main idea behind such a Monte Carlo simulation is to do individual simulation runs that are based on different random seeds. However, it cannot be guaranteed to observe the worst case in one of these runs, such that advanced models and techniques are required. Our previous work [6] presented a method that shifts bounds achieved in a Monte Carlo simulation towards the worst case observed by analytical models. This approach is refered to as WCS.

Recent approaches for performance evaluation are based on real-time scheduling analysis tools, e. g., ChronVal [7] or SymTA/S [8] to determine maximum and minimum performance bounds in communication networks. These tools are usually employed for determining worst case schedules of processes in CPUs, but when those tools are aware of handling non-preemptive processes, this technique can be used to model worst case schedules of frames in a switched network. The model checker Uppaal [9] was used to determine performance bounds in the field of Ethernet networks [10], which is in line with the recently mentioned real-time scheduling analyzers, because they exhaustively enumerate the search space. In the later we will use the term MC to refer to the recently mentioned approaches. The approach given in [11] (and briefly presented in Section III-C) also falls into this category.

Another well-accepted method to determine performance bounds in computer networks is known under the concept of NC [12]–[14], being a competitor to classical queuing theory. Mainly academic research of the last decade brought tools to the community of NC such as DISCO [15], COINC [16], CyNC [17], or RTC [18].

In all those performance calculators, the network and the flows traversing the topology have to be specified explicitly by graphs and traffic patterns. There exist various data formats to describe these environments that are often tool dependent. Besides those tool dependent data formats there exist some standardized data formats as the Common Information Model (CIM) [19] used for network management or BRITE [20] used for topology creation.

Tool	Sim	NC	MC	WCS
DIMTOOL	✓	✓	✓	✓
ns-2 [1]	✓			
ns-3 [2]	✓			
SSF [3]	✓			
OMNeT++ [4]	✓			
OPNET [5]	✓			
ChronVal [7]			✓	
SymTA/S [8]			✓	
Uppaal [9]			✓	
DISCO [15]		✓		
COINC [16]		✓		
CyNC [17]		✓		
RTC [18]		✓		

TABLE I
PERFORMANCE ESTIMATION TOOLS

III. DIMENSIONING TOOL

In this section we introduce the concepts and architecture of DIMTOOL and introduce the covered performance calculators.

A. Objectives and Traffic Models

In communication networks, there exist three major approaches to determine meaningful bounds of real-time applications: (a) analytical methods, (b) network simulation, and (c) measurements.

To evaluate safety critical systems, a mixture of those approaches is commonly used to prove correct functionality. Analytical methods have the drawback, that determined performance bounds may differ significantly from that of a deployed network. Consider a worst case scenario in a queuing network. The worst case will only be provoked when the packets of interest are delayed by all other crossing packets. Proving the correctness of queuing networks, does, however, still contain some unsolved problems — methods known so far either pay by significant overestimation, or by computational effort. While overestimated bounds are sometimes tolerable, the computational effort of tight bounds is usually impractical for larger networks [21]. In fact, the problem of finding tight bounds has been proven to be NP-hard [22], such that an exponential number of solutions has to be compared in order to find the tightest bound when using the known linear optimization based algorithms of [22], [23].

Typically the following delay types are observed in switched Ethernet: (a) Propagation Delay, (b) Transmission Delay, (c) Processing Delay, and (d) Queuing Delay. Current work primarily focused on determining worst case values for transmission and queuing delays. Due to the nature of queuing networks, where arbitrary cross traffic may pass the system, this may lead to high variability. In contrast to transmission and queuing delay, values for propagation and processing delay are relatively stable which allows providing tight bounds. Bounds for the processing delay highly depend on the assumed multiplexing architecture, i. e., whether the switch fabric is implemented in hardware, or as a software solution.

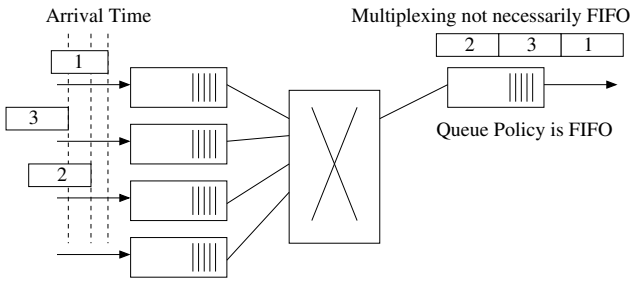

Fig. 2. Arbitrary Multiplexing in Switches

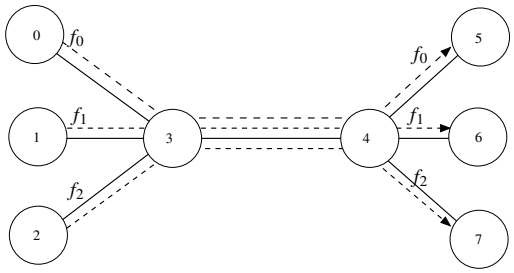

Fig. 3. Tandem Scenario with four Flows

Flow	Burst [bytes]	Rate [kBits/s]
f_1	1024	50
f_2	256	500
f_3	408	1000

TABLE II
TOKEN BUCKET PARAMETERS

```
topo
nodes 8
0,node:10.33.0.102:255.0.0.0,3,switch
3,switch,1,node:10.33.0.103:255.0.0.0
3,switch,2,node:10.33.0.104:255.0.0.0
3,switch,4,switch
4,switch,5,node:10.33.0.105:255.0.0.0
4,switch,6,node:10.33.0.106:255.0.0.0
4,switch,7,node:10.33.0.107:255.0.0.0
#
flows
0, node, { (TokenBucket, 1024, 7.25,
   10.33.0.106, normal(0.202, 0.04)) ) }
1, node, { (TokenBucket,  256, 62.5,
   10.33.0.106, normal(0.202, 0.04)) ) }
2, node, { (TokenBucket, 408, 125,
   10.33.0.107, normal(0.202, 0.04)) ) }
#
```

Listing 1. Example Description of Topology and Flows

We will briefly introduce common approaches to define traffic profiles. In the field of NC, standard token bucket traffic models are used, such as the single token bucket, the dual token bucket or even multiple token bucket curves built of several piecewise linear curves. Common literature (e.g., [24]) characterizes the single token bucket as being (σ,ρ)-*constrained*, while σ specifies the burst and ρ the burst rate. Paying attention to the property of computer networks usually sending the individual frames at wire speed, the second model using a dual token bucket [14] is employed to additionally model peak rate and maximum packet size. The dual bucket model is described as a 4-tuple (p,M,r,b), where p represents the peak rate, M is the maximum packet size, r corresponds to the rate of the token bucket and b is the burst of the token bucket. In network simulation, there exist several models for traffic description, e.g., Markov on-off processes, distributions over wait times and packets sizes, self-similar processes [25]–[27], as well as traces from traffic measurements.

Network performance evaluation tools differ between FIFO bound and non-FIFO bound. Figure 2 shows the relevance of this distinction in a simple switching scenario, and why we require a non-FIFO bound even if queuing policy is FIFO in the output queue. In fact non-FIFO bounds would not be required for performance evaluation, if the implemented switch guaranteed FIFO forwarding [28]. The implementation would have to store the concrete arrival time of each packet for guaranteeing true FIFO behavior. Despite the fact, that the concrete implementation details are not published by hardware vendors, it is assumed that most switch implementations do not guarantee FIFO forwarding. Instead maximum bipartite matching algorithms should have been implemented to achieve high throughput in the crossbar [29].

B. Topology and Flow Description

In fact there are several data formats available to express network topologies which highly depend on the exact field of application and the consequent information content. Some data formats mentioned earlier use XML representation while others use Comma Separated Value (CSV). Our toolbox follows the latter approach and describes topologies via CSV. More precisely we use two sections, one expressing the topology of the network, the other expressing the data flows traversing the network. Listing 1 gives an idea of the format we use in this toolchain to express topology and traffic generation.

This description covers the tandem scenario given in Figure 3; the token bucket rates are given in Table II. The topology section contains the number of nodes and a list of bidirectional edges. The description of each node may contain further information such as IP address/mask and type of switch. We defined the node types *node* as IP endpoint and *switch* as an Ethernet compliant switch. The flow section contains a list of arrival curves that are to occur in the specified nodes. The first two entries specify the described node with numerical id and the type of the node. The remainder contains the arrival curve descriptions, which conforms to a *TokenBucket* arrival curve in this example. Additionally, the destination IP address, as well as the offset of the arrival curve (being normally distributed) are given. We provide the following arrival curve descriptions: *TokenBucket*, *DualTokenBucket*, *Trace*, and *MMOnOff*. Some of those are better suited for use in specific back-ends than others. In particular, it is not easy to generate reasonable traffic in a Monte Carlo simulation from a pure token bucket

curve due to the fact that the start times do not have a direct counterpart in the token bucket model. On the other side it is difficult to abstract an NC arrival curve from a Markov-modulated on/off process, since the token bucket peak rate might be very high compared to the average rate arising from the Markov-modulated on/off process, unless the parametrized wait times are constant.

C. Performance Calculation Back-Ends

The proposed toolchain has the ability to address different back-ends that in turn determine performance bounds. At the time of writing, there exist the following back-ends: Network simulation with OPNET, NC analysis using the DISCO network analyzer [30], the rare event simulation as given in [6], and the optimization based approach given in [11].

1) Network Simulation Back-End: Network simulation with Monte Carlo methods is a well-accepted technique to estimate performance bounds within a reasonable time, especially when network topology and use case hardly allow a real setup. This could be due to the network consisting of a vast number of participating network nodes or due to the constraints that have to be applied. In general, establishing a realistic testbed of the scenario of interest might be too time-consuming and cost-intensive.

We employ Discrete-Event Simulation based on the Monte-Carlo method to simulate networks. In the Monte-Carlo method, the experiment is carried out several times to derive more meaningful results. Since simulation with computer systems implies a deterministic calculation of the result, processes usually depend on random number generators, that are initialized with different random seeds.

There are popular network simulators that are based on this approach, e.g., SSF, OMNeT++ and OPNET to name a few. The suggested back-end is based on the OPNET network simulator and uses External Model Access (EMA) to control OPNET simulations from Matlab. The EMA architecture can be used to create and simulate network scenarios. For this, OPNET provides an interface via shared object/DLL to allow access to the OPNET simulation runtime. Since computation of several experiments requires high computation time, this back-end is run in offline mode, which means, that Matlab gets control back while the simulation is still running. We provide methods to check, whether the back-end has already finished and whether results are available from the back-end.

2) Network Calculus Analysis Back-End: NC is an analytical approach to determine worst case values for delay and backlog. This approach is based on the work of Cruz [12], [13] and was enhanced by Le Boudec who wrapped this early approach into a mathematical framework based on the (min,+)-algebra [14]. In the NC, we define traffic envelopes of flows rather than handling the actual arrival and departure processes known from queuing theory. The key operations in NC are convolution and deconvolution as given by Definition 3.1 and Definition 3.2:

Definition 3.1: MIN-PLUS CONVOLUTION [14].

$$(f \otimes g)(t) = \inf_{0 \le s \le t} \{f(t-s) + g(s)\}. \quad (1)$$

Definition 3.2: MIN-PLUS DECONVOLUTION [14].

$$(f \oslash g)(t) = \sup_{u \ge 0} \{f(t+u) - g(u)\}. \quad (2)$$

The Min-Plus Convolution is primarily used for the concatenation of servers, the Min-Plus Deconvolution is primarily used for determining the output arrival curve of a forwarding server.

In order to determine worst case values for backlog and delay, we require the definitions of Vertical Deviation (Definition 3.3) and Horizontal Deviation (Definition 3.4).

Definition 3.3: VERTICAL DEVIATION [14].

$$v(f,g) = \sup_{t \ge 0} \{f(t) - g(t)\} \quad (3)$$

Definition 3.4: HORIZONTAL DEVIATION [14].

$$h(f,g) = \sup_{t \ge 0} \{\inf\{d \ge 0 \text{ such that } f(t) \le g(t+d)\}\} \quad (4)$$

The Vertical Deviation is used to determine worst case values for the backlog, the Horizontal Deviation is used to determine worst case values for the delay.

The suggested NC back-end uses the DISCO network analyzer. With this toolbox, we obtain a valuable network analyzer class which provides standard network algorithms as: (a) Dijkstra's shortest path algorithm, (b) an implementation of the turn prohibition algorithm [31], and (c) a topology converter that converts network graphs to server graphs. The developers of the toolbox made a lot of progress in last years in terms of tight NC bounds [23], [32], such that standard tightening approaches as Pay Bursts Only Once (PBOO) and Pay Multiplexing Only Once (PMOO) are supported by this toolbox.

When addressing the end-to-end performance bounds in switched Ethernet, an essential topic is the mapping of the store-and-forward delay of switches, since fluid flow models do not have a direct counterpart of variable-sized packets. More precisely, we have the following options known from standard NC literature (e.g., [14]). The alternatives discrete burst (DB) and rate latency (RL) were already summarized in [11].

- The packet is described as a DB that is already active at time 0. A staircase or (σ,ρ)-constrained arrival curve can be used to model this traffic.
- A RL service curve of the form $\beta_{C,l_{max}/C}$ is applied with C being the capacity of the discussed link and l_{max} being the packet size of a maximum sized frame that is able to delay the flow of interest.
- A packetizer (PKT) is introduced that acts as an additional delay element which releases the whole packet if completely received (cf. [14]).

The PKT is also a concept of the early NC. However, in order to safely apply PKT to an edge-by-edge analysis, having shown more accuracy (cf. [23]), the PKT is usually realized by a RL service curve, a service curve that delays flows by a maximum sized frame. Unfortunately an additional delay of a maximum sized frame at each server yields more pessimistic bounds compared to the original packetizer developed for the node-by-node analysis. An approach that captures this inaccuracy is given by the mixed integer program (MIP) back-end.

3) MIP Analysis Back-End: In order to determine the exact worst case in switched Ethernet, the recently introduced packetizer has to know the packetization sequence a priori. Especially when links run at different wire speed, the presented analytical methods cannot model the exact worst case. This observation is similar to the pitfall of Pay Multiplexing Only Once SFA (PMOO-SFA) as given in [32] and also outlined in [33]. Due to the commutativity of the (min,+)-convolution, the burstiness of the traffic is always paid for at the rate of the slowest server.

Consider the following example: Three packets (size 125, 250 and 500 bytes) are transmitted over the two subsequent servers $S1$ and $S2$, where $S2$ has a faster service rate than server $S1$. Furthermore, we are interested in the worst case latency of the 125 byte packet. The results of the cumulative arrivals are given in Figure 4 and Figure 5.

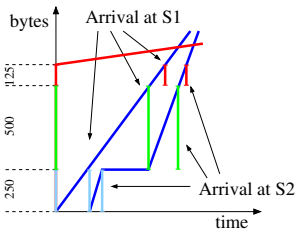

Fig. 4. 250 byte packet prior to 500 byte packet

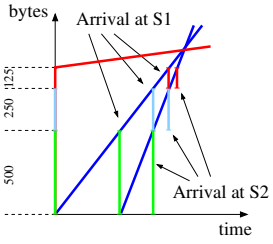

Fig. 5. 500 byte packet prior to 250 byte packet

We observe the following situation: The packet of interest is of size 125 bytes and is delayed by one 500 byte packet and one 250 byte packet. Depending on whether the 250 byte packet or the 500 byte packet is transmitted at the first place, we observe different worst cases with the packetization approach. Since any packet sequence is conceivable, the highest worst case latency of all sequences has to be taken into account. Without having the information of the packetization sequence yielding the exact worst case, an additional delay of a maximum sized frame has to be assumed (RL approach).

The given MIP back-end is able to capture this situation accurately and was introduced in [11]. The presented packetized traffic model describes the worst case schedules in a single MIP instance and thus benefits from the advantage of both edge-by-edge analysis and packetized model, yet with additional effort. The exhaustive enumeration of schedules is similar to techniques known from model checking and allows the calculation of worst case bounds. The basic idea is to model the case that one packet is delayed by another packet with a boolean (integer) variable. If this variable is true, i.e., one, the worst case delay is increased by the transmission delay of the previous packet, otherwise it is increased by zero. Compared to the techniques from NC, this model allows better bounds in terms of tightness yet requiring additional computational effort since we solve an NP-hard problem.

4) Rare Event Simulation Back-End: The technique of performing a rare event simulation from token bucket NC arrival curves was shown in [6]. Within this back-end, we try to simulate the rare event that packets are delayed by all other packets and use the technique of importance sampling. In the pure network simulation back-end, the traffic generating nodes have their start time set according to random number generators. However, the method of how start times are chosen is extremely important because worst case queuing situations of token bucket shaped traffic are only provoked, if the offset to each other is at the worst position. For arrival curves in token bucket form, quite a few simulation runs are required to provoke worse case queuing, since offsets are usually not aligned in the most pessimistic way. In this back-end, we determine pessimistic offsets according to the network topology and the traversing traffic flows. When having these pessimistic offset, we run a parametrized simulation that allows shifting the worst observed latencies determined by the simulation towards the exact worst case.

D. DIMTOOL Architecture

Figure 6 shows the architecture of the DIMTOOL. On the right we see the tools based around the topology description. In order to provide a convenient graphical user interface, we built some import routines for the network editor *Network Notepad* [34] as well as for an only internally used program. Within this tool we can easily assemble cabin network configurations that match with different airplane types and setups. The *Topology Generator* is mainly used for research and advance development in the field of the aircraft cabin and allows the simulation of various cabin scenarios. This is of special interest when new techniques shall be elaborated, such as new scheduling algorithms in the switches or synchronization mechanisms. The *Configuration* contains the actual bandwidth reservation, VLAN rules or routing entries. The configuration can then be distributed with standardized management protocols as Simple Network Management Protocol (SNMP) and Web Based Enterprise Management (WBEM). The *Topology Description* is a system independent format based on XML that contains the actual topology as well as the devices hosted in the network. The *Topology Description* is exported to a CSV format that contains both the topology and the flows through the system. Currently, all mentioned export routines and transformations are provided by a C++ class library based on the Ultimate++ framework [35] that showed good portability. The CSV-based *Topology and Flow Description* is passed to the DIMTOOL that creates reports from performance calculator back-ends. These reports are either in diagram form or represented by text files that are in turn required by certification processes.

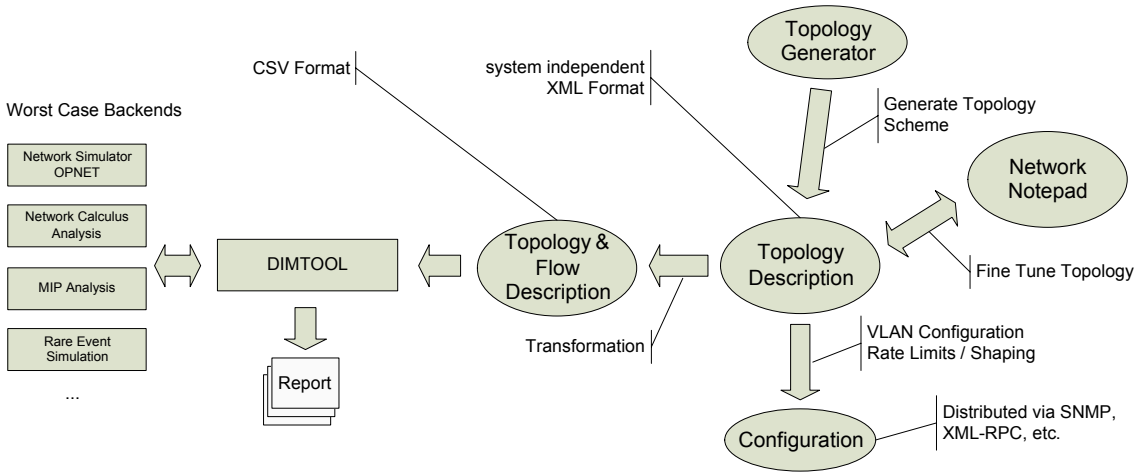

Fig. 6. Toolbox DIMTOOL

NetworkCalculation

```
+max(in aggregation:Aggregation): NetworkCalculation
+min(in aggregation:Aggregation): NetworkCalculation
+avg(in aggregation:Aggregation): NetworkCalculation
+median(in aggregation:Aggregation): NetworkCalculation
+optimizeTopo()
+optimizeFlows()
+use(in topo:Topology,in flows:Flows)
+use(filename:String)
+printDelay()
+printBacklog()
+printReport()
+printVersion()
+getFirstDelay(): CurveData
+getNextDelay(): CurveData
+getFirstBacklog(): CurveData
+getNextBacklog(): CurveData
+using(in method:Integer): NetworkCalculation
+of(in flows:Flows): NetworkCalculation
+run()
+method(in amethod:Integer): NetworkCalculation
+fifo(afifo:Integer): NetworkCalculation
+speed(aspeed:Double): NetworkCalculation
+proc(in aproc:Double): NetworkCalculation
+isFinished(): Boolean
+finalize()
```

Fig. 7. Class Network Calculation

The remainder of this section shows the available methods of the DIMTOOL as given in the NetworkCalcution class shown in Figure 7. These methods address the following tasks: Computational Methods, Execution Methods, Finalization Methods, Result Methods, Global Parametrization Methods and Optimization Methods.

Computational Methods: The expected parameter *aggregation* of the Computational Methods thereby defines how the determined delay bounds will concretely be assembled, e.g., whether to identify the jitter, the delay difference versus variation (in case of multicast flows) or the sum. More precisely we introduce the following methods: (a) *min*, (b) *max*, (c) *avg*, and (d) *median*. The respective functions determines minimum, maximum, average and median values for delay and backlog.

Execution Methods: The execution methods are used to control performance calculation, i.e., which back-end to choose, which topology and cross traffic flows to use, which flows to investigate. The run method actually triggers the back-end and performs computation or simulation. We introduce the following methods: (a) method *use* specifies topology and flows to be used, (b) method *of* defines the addressed flows, (c) method *using* chooses the used back-end, and finally (d) method *run*, which triggers the actual computation/simulation.

Finalization Methods: Some back-ends provide instantaneous performance calculation as being provided by the NC back-end, i.e., returning from the *run* method call implies that NC results are available. The other back-ends currently use an offline computation, since the computation is very intensive compared to the NC back-end such that the calling Matlab instance would be blocked. For the back-ends that run in offline mode, we provide the finalization methods to check whether the back-end has already finished so that results can be read from the back-end. We provide the following methods: (a) method *isFinished* returns true when simulation/analysis is finished, otherwise *isFinished* returns false, and (b) method *finalize* which finalizes the back-end. When *finalize* was called on a finished back-end, the Result Methods can be executed to read the results from the back-end.

Result Methods: The result methods are used to process information as detected by the back-ends. These methods can be called when *isFinished* from the previously stated Finalization Methods returns 1. In order to process the results, we provide print methods and get methods. Print methods create textual representations of the results while the returned objects of the get methods can be used for plotting in Matlab. More precisely we introduce the following methods: (a) method *printDelay* prints the delay report, (b) method *printBacklog* prints the backlog report, (c) method *printReport* prints both delay and backlog report, (d) methods *getFirstDelay*/*getNextDelay* allows iteration over delay curves, and (e) methods *getFirstBacklog*/*getNextBacklog* allows iteration over backlog curves.

Global Parametrization Methods: The global parametrization methods are used to set parameterizable topology options globally. This allows the central parametrization of the FIFO assumption, parametrization of the link speed and setting

processing delay in intermediate switch nodes. We provide the following methods: (a) method *fifo* sets FIFO processing order globally, (b) method *speed* sets link speed globally, and (c) method *proc* sets processing delay globally.

Optimization Methods: The optimization methods act as entry point for implementing optimization algorithms for flows and topology. These optimization algorithms are intended to invoke the provided back-ends in order to optimize aspects of topology and flows. In the current version, the Optimization Methods *optimizeTopo* for topology optimization and *optimizeFlows* for flow optimization are provided.

Matlab Functions: We embedded the given architecture into Matlab and provide seamless integration through Matlab interfaces. Currently the following functions are supported by our framework:

- *dimtool_init* — initializes DIMTOOL temporarily
- *dimtool_install* — installs DIMTOOL permanently
- *dimloadtf* — loads topology and flow description
- *dimnc* — returns instance of class NetworkCalculation
- *dimbar* — plots bar graph of delay results
- *dimbarbacklog* — plots bar graph of backlog results
- *dimplot* — plots graph of delay results
- *dimplotbacklog* — plots graph of backlog results
- *dimprinttofile* — writes results to file

The entry point of all calculation or simulation runs is an instance of NetworkCalculation, which is returned by *dimnc*. The interface functions can be called directly on this return value. The function *dimloadtf* loads the topology and flow description of the form described in Section III-B. This function takes an optional parameter to add standard values for the interframe gap and preamble seen in switched Ethernet networks. When the NetworkCalculation class has been parameterized, the simulation or analysis can be triggered by calling *nc.run()*. If *nc.isFinished()* returns true, the results are available and can be can be processed further by *dimbar*, *dimplot* and *dimprinttofile* methods.

IV. Experimental Results

In this section we demonstrate the usage of the DIMTOOL in the context of industrial real-time systems. In particular, we address the performance bounds of an Ethernet-based aeronautic cabin network in Section IV-A and Section IV-B.

A. Cabin Server Intra-Communication

This section addresses the performance bounds as observed in the intra-server communication scenario. The communication inside the Cabin Server (CS) is also mandatory when regarding higher safety levels. Besides scheduling and distribution of processes in ARINC 653 [36] partitions, we are interested in hard performance bounds of the inter CPU communication. These worst case delays can then be provided to real time analysis tools like ChronVal [7] and SymTA/S [8] in order to optimize the scheduling and distribution of processes.

Figure 8 shows a discussed architecture of the intra-server communication. The cabin server consists of two dual-core

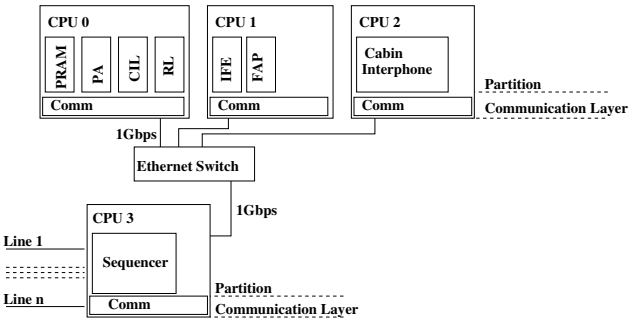

Fig. 8. Intra-Server Communication

CPUs being interconnected by Ethernet. The following functions and use cases are covered by the cabin server: (a) audio announcements like boarding music, safety briefing, or audio announcements, (b) cabin illumination with different light scenarios, (c) reading lights, (d) the entertainment system that provides video, Internet and games as well as (e) the cabin control. Each of those use cases has a direct counterpart in the CS: (a) is covered by process Prerecorded Audio (PRAM) and Passenger Address (PA), (b) is covered by process Cabin Illumination (CIL), (c) is covered by process Reading Lights (RL), (d) is covered by process In-Flight-Entertainment (IFE), and (e) is covered by process Flight Attendant Panel (FAP). Table III briefly lists the expected traffic flows.

Each of these functions is realized as a partition in the ARINC 653 compliant operating system, which run on CPU 0 to CPU 2. The sequencer on CPU 3 controls and manages the communication to the end devices that are connected by Line 1 to Line n. The sequencing unit acts as a gateway to the cabin network, such that the communication between partition and end devices will pass a protocol converter.

Figure 9 outlines the result of our worst case analysis toolchain. We determined the following bounds for the end-to-end delay: Non-FIFO NC bound, FIFO NC bound, and the bound determined by network simulation.We observe that the NC bounds are relatively tight compared to the worst case observed in the network simulation. The reason for that is that the number of traversed servers has direct impact on the tightness of the achieved bounds [37], [38]. In this scenario,

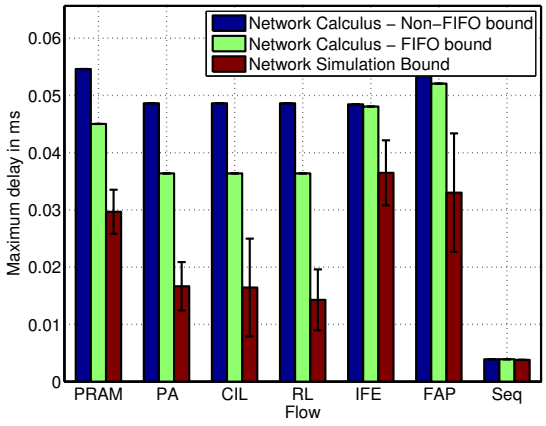

Fig. 9. Analysis of CS Using DIMTOOL

Flow	BW [bytes/ms]	Max [bytes]	Priority
PRAM → Seq	86.40	864	5
PA → Seq	1.60	64	7
CIL → Seq	1.28	64	7
RL → Seq	0.64	64	7
IFE → Seq	3125.00	1522	0
FAP → Seq	125.00	1522	7
Seq → PA	1.60	64	7

TABLE III
PARAMETERS IN INTRA-SERVER COMMUNICATION

```
tf = dimloadtf('intraserver.txt', 0)
nc = dimnc
agg = Factory.getSumAggregation
nc.use(tf.getTopo, tf.getFlows)
nc.max(agg).of(tf.getFlows).using(2).
   method(2).fifo(1).proc(0.012)
nc.run
```

Listing 2. NC Analysis with DIMTOOL, Intra-Server, FIFO

only one server is traversed by the traffic flows.

The results were obtained by executing the code shown in Listing 2 using our DIMTOOL platform: The code invokes Network Calculus Analysis on the Intra-Server communication network. The Matlab calls are forwarded to the wrappers which in turn invoke the chosen back-end, in this case the DISCO network analyzer using the Separated Flow Analysis (SFA) algorithm [30]. We set the processing delay used by DISCO to 12µs and to calculate the FIFO bound.

The remaining curve of Figure 9, i.e., non-FIFO bound and simulation, is determined by Matlab calls similar to that mentioned above. The parameter of the method *using* decides which back-end to choose.

B. Aircraft Cabin Network

In this use case we study the worst case delays of a switched version of the Cabin Management System (CMS) using DIMTOOL. The employed topology is a step towards a commercial-of-the-shelf (COTS)-enabled CMS and was already discussed in [39], where first simulation results were introduced. The basic topology setup is shown in Figure 10. Up to 22 lines are foreseen in a typical airplane with a maximum depth of 15 Ethernet hops. In this example, 105 high priority flows traverse one line of the simplified cabin network with 13 daisy-chained Ethernet switches. One of those flows acts as a multicast flow from the cabin server to the end devices. The others flow from the end devices back to the server. In fact, we connected 7 Service Units (SUs) and 1 handset to each switch employing the traffic patterns shown in Table IV.

Figure 11 shows the results for the back-ends network simulation, WCS, and NC analysis. The delays from the WCS are shifted towards the NC bounds by about 10% compared to the standard network simulation. Furthermore, the FIFO

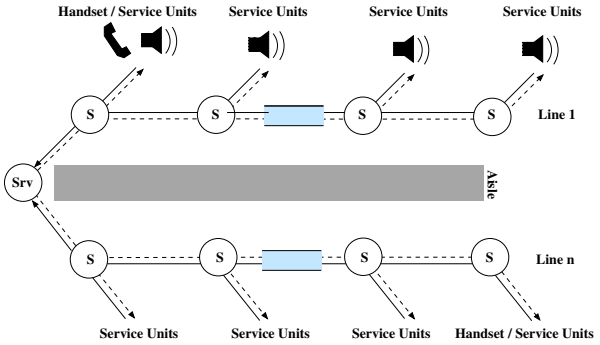

Fig. 10. Cabin Scenario as Sink Tree

bound of the NC analysis does not hold—at least at the first switch. In this scenario we do need the non-FIFO bound to be absolutely sure that delays greater than the analytical bound cannot be observed. In addition, we observe that the worst case bounds from the NC analysis are getting worse with the number of traversed switches. The results confirm the results from previous work such as [37], [38], which means, that we are lacking tightness when traffic flows traverse several servers.

The results were obtained using the proposed DIMTOOL platform. Listing 3 shows the NC analysis of the non-FIFO bound. The remaining curves of Figure 11, i.e., SIM, WCS and FIFO bound, are determined by Matlab calls similar to that mentioned above. The parameter of the method *using* specifies the chosen back-end.

Flow	BW [bytes/ms]	Max [bytes]	Priority
Srv → EndDevices	3456.0	108	7
SU → Srv	25.5	108	7
Handset → Srv	204.0	64	7
Bulk Traffic	1000.0	1518	0

TABLE IV
PARAMETERS IN AIRCRAFT CABIN

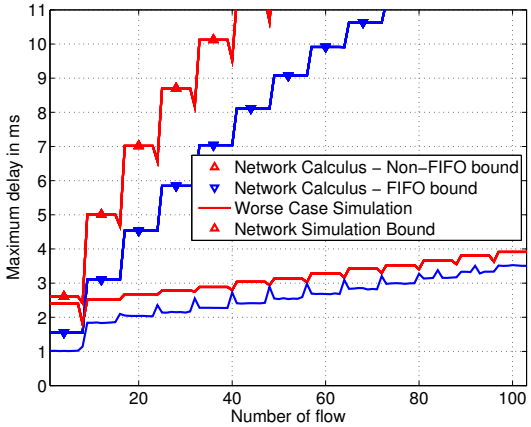

Fig. 11. Analysis of CMS using DIMTOOL

```
tf = dimloadtf('cms.txt', 0)
nc = dimnc
agg = Factory.getSumAggregation
nc.use(tf.getTopo, tf.getFlows)
nc.max(agg).of(tf.getFlows).using(2).
   method(1).fifo(0).proc(0.003)
nc.run
```

Listing 3. NC Analysis, Aircraft Cabin, Non-FIFO

V. Conclusions and Future Work

In this paper we proposed a novel platform to determine performance bounds in computer networks. Compared to previous tools, we provide several performance estimation approaches within a single toolbox. By this we can provide comparable and transparent performance bounds, which in turn helps to find flaws in performance modeling. In the experiments, we investigated an Ethernet-based topology of the aircraft cabin that covers safety-relevant functions, and verified, that the FIFO bound as determined by the NC analysis does not hold. For this we used a WCS, which is — to the best of our knowledge — unique for the platform DIMTOOL. This approach helped us to get a realistic view on worse cases as they are likely to occur in the switched aircraft cabin. For the intra-server communication, we determined dependable worst case bounds, that are provided to process scheduling tools such as chronVAL or SymTA/S. In the next steps we will address further scenarios in the field of aeronautic networks, but also real-world scenarios considering arbitrary computer networks.

References

[1] ns-2. The Network Simulator - ns-2, Accessed 2012-04-24. http://www.isi.edu/nsnam/ns/.

[2] ns-3, Accessed 2012-04-24. http://www.nsnam.org/.

[3] SSFNet. Scalable Simulation Framework, Accessed 2011-07-03. http://www.ssfnet.org/.

[4] OMNeT++. OMNeT++ Network Simulation Framework, Accessed 2011-07-03. http://www.omnetpp.org/.

[5] OPNET. Application and Network Performance with OPNET, Accessed 2011-07-04. http://www.opnet.com/.

[6] Emanuel Heidinger. Rare Events in Network Simulation Using MIP. In *Proc. of the 23rd International Teletraffic Congress. ITC 2011*, pages 314–315, 2011.

[7] INCHRON. chronVAL, Accessed 2012-04-01. http://www.inchron.de/chronval.html.

[8] Symtavision. SymTA/S, Accessed 2012-04-01. http://www.symtavision.com/symtas.html.

[9] UP4ALL. DESIGN VERIFICATION FOR EMBEDDED SYSTEMS, Accessed 2012-04-12. http://www.uppaal.com/.

[10] D. Witsch, B. Vogel-Heuser, J.M. Faure, G. Marsal, et al. Performance analysis of industrial Ethernet networks by means of timed model-checking. In *Proc. of 12th Symposium on Information Control Problems in Manufacturing. INCOM 2006*, pages 1–6, 2006.

[11] E. Heidinger, N. Kammenhuber, A. Klein, and G. Carle. Network Calculus and Mixed-Integer LP Applied to a Switched Aircraft Cabin Network. In *Proc. of the 20th International Workshop on Quality of Service, IWQoS 2012*, pages 1–4, 2012.

[12] R.L. Cruz. A Calculus for Network Delay, Part I, Network Elements in Isolation. *IEEE Transactions on Information Theory*, 37(1):114–131, 1991.

[13] R.L. Cruz. A Calculus for Network Delay, Part II, Network Analysis. *IEEE Transactions on Information Theory*, 37(1):132–141, 1991.

[14] J.Y. Le Boudec. *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Springer-Verlag, 2004.

[15] Distributed Computer System Lab. DISCO Network Calculator, Accessed 2011-01-24. http://disco.informatik.uni-kl.de/content/Downloads.

[16] S. Lagrange. COINC Toolbox, Accessed 2011-05-23. http://www.istia.univ-angers.fr/~lagrange/software.php.

[17] H. Schioler. CyNC, Accessed 2011-05-23. http://www.control.auc.dk/~henrik/CyNC/.

[18] Ernesto Wandeler and Lothar Thiele. Real-Time Calculus (RTC) Toolbox, 2006. http://www.mpa.ethz.ch/Rtctoolbox.

[19] DMTF. Common Information Model, Accessed 2011-07-03. http://www.dmtf.org/standards/cim.

[20] BRITE. Boston university Representative Internet Topology gEnerator, Accessed 2011-07-03. http://www.cs.bu.edu/brite/.

[21] J.-L. Scharbarg and C. Fraboul. Methods and Tools for the Temporal Analysis of Avionic Networks. *New Trends in Technologies: Control, Management, Computational Intelligence and Network Systems*, pages 413–438, 2010.

[22] A. Bouillard, L. Jouhet, and E. Thierry. Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks. In *Proc. of the 29th Conference on Computer Communications. INFOCOM 2010*, pages 1–9, 2010.

[23] J.B. Schmitt, F.A. Zdarsky, and M. Fidler. Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch... In *Proc. of the 27th Conference on Computer Communications. INFOCOM 2008*, pages 1669–1677, 2008.

[24] Y. Jiang. A Basic Stochastic Network Calculus. In *Proc. of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications. SIGCOMM 2006*, pages 123–134, 2006.

[25] M.E. Crovella and A Bestavros. Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5(6):835–846, 1997.

[26] Vern Paxson and Sally Floyd. Wide-Area Traffic: The Failure of Poisson Modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, 1995.

[27] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson. On the Self-Similar Nature of Ethernet Traffic. In *Proc. of SIGCOMM 1993*, pages 183–193, 1993.

[28] G. Rizzo and J.Y. Le Boudec. "Pay bursts only once" does not hold for non-FIFO Guaranteed Rate nodes. *Performance Evaluation*, 62(1-4):366–381, 2005.

[29] N. McKeown, V. Anantharam, and J. Walrand. Achieving 100% throughput in an input-queued switch. In *Proc. of the 15th Conference on Computer Communications. INFOCOM 1996*, volume 1, pages 296–302, 1996.

[30] Jens B. Schmitt and Frank A. Zdarsky. The DISCO Network Calculator: A Toolbox for Worst Case Analysis. In *Proc. of the 1st international conference on Performance evaluation methodolgies and tools. ValueTools 2006*, pages 1–10, 2006.

[31] David Starobinski, Mark Karpovsky, and Lev A. Zakrevski. Application of network calculus to general topologies using turn-prohibition. *IEEE/ACM Transactions on Networking*, 11(3):411–421, 2003.

[32] Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once. In *Proc. of the 14th Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, pages 1–15, 2008.

[33] Andreas Kiefer, Nicos Gollan, and Jens Schmitt. Searching for Tight Performance Bounds in Feed-Forward Networks. In Bruno Mller-Clostermann, Klaus Echtle, and Erwin Rathgeb, editors, *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, volume 5987 of *Lecture Notes in Computer Science*, pages 227–241. Springer-Verlag, 2010. 10.1007/978-3-642-12104-3_18.

[34] Network Notepad. Network Notepad, Accessed 2012-03-09. http://www.networknotepad.com/.

[35] Ultimate++. Ultimate++ framework, Accessed 2012-03-11. http://www.ultimatepp.org/.

[36] Aeronautical Radio, Inc. *ARINC 653: Avionics Application Software Standard Interface*. Annapolis, USA, 2003.

[37] F. Ciucu, A. Burchard, and J. Liebeherr. A network service curve approach for the stochastic analysis of networks. In *ACM SIGMETRICS Performance Evaluation Review*, volume 33, pages 279–290, 2005.

[38] Y. Jiang and Y. Liu. *Stochastic network calculus*. Springer-Verlag, 2008.

[39] E. Heidinger, C. Heller, A. Klein, and S. Schneele. Quality of Service IP Cabin Infrastructure. In *Proc. of the 29th Digital Avionics Systems Conference, DASC 2010*, pages 3.D.4–1–3.D.4–10, 2010.