Fan, X., M. Jonsson, and J. Jonsson, "Guaranteed real-time communication in packet-switched networks with FCFS queuing," Computer Networks, vol. 53, no. 3, pp. 400-417, Feb. 2009.

Guaranteed Real-Time Communication in Packet-Switched Networks with FCFS Queuing

Xing Fan[1], Magnus Jonsson[1], and Jan Jonsson[2]

1. CERES, Centre for Research on Embedded Systems
School of Information Science, Computer and Electrical Engineering, Halmstad University, Halmstad, Sweden, Box 823, SE-301 18, Sweden. {xing.fan, magnus.jonsson}@hh.se

2. Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden, janjo@ce.chalmers.se

Abstract

In this paper, we propose a feasibility analysis of periodic hard real-time traffic in packet-switched networks using First Come First Served (FCFS) queuing but no traffic shapers. Our work constitutes a framework that can be adopted for real-time analysis of switched low-cost networks like Ethernet without modification of the standard network components. Our analysis is based on a flexible network and traffic model, e.g., variable-sized frames, arbitrary deadlines and multiple switches. The correctness of our real-time analysis and the tightness of it for network components in single-switch networks are given by theoretical proofs. The performance of the end-to-end real-time analysis is evaluated by simulations. Moreover, our conceptual and experimental comparison studies between our analysis and the commonly used Network Calculus (NC) shows that our analysis can achieve better performance than NC in many cases.

Keywords: real-time communication, packet-switching, First Come First Served (FCFS)

1. Introduction

Many applications, especially networked embedded real-time applications such as radar applications and control systems, require periodic hard real-time communication, meaning that every frame in the traffic stream should be 100% guaranteed to meet imposed timing requirements. Meanwhile, there is the trend of implementing embedded networks with packet-switched technologies. However, providing guarantees of timely delivery in packet-switched networks is a complicated problem because we must consider the problem of deriving the worst-case delay across multiple hops in the network.

Many approaches for solving this problem rely on adding packet scheduling, e.g., Earliest Deadline First (EDF), and having an admission control mechanism to verify that the specified requirements can be met [1-4]. However, packet scheduling may result in added cost and modification of the implementation, since many standard packet-switching network components only support FCFS. Consequently, standard components with FCFS queuing have been considered by many researchers. A method for calculating the worst-case packet delay in switched Ethernet with FCFS queuing has been proposed [5]. However, their method can only be applied to a limited range of applications due to assumptions on minimal-sized frames and specific traffic characteristics. Moreover, the correctness of the method has not been formally proven.

A widely accepted analytical technique, Network Calculus (NC), enables an approach for calculating the worst-case delay for FCFS queuing [6-8] and has been applied on packet-switched networks [9-14]. However, all these NC-based solutions require modification when applied to a network with standard components such as switched Ethernet, e.g., implementing traffic shapers in the source nodes [12] [13] or supporting priorities to logical real-time channels [15], which significantly increase the cost and implementation complexity. Tight end-to-end delay bounds for FCFS sink-tree networks have been derived using NC [16]. However, the analysis is not generalized to common network topology. In addition, NC cannot be used directly for periodic traffic, unless the periodic model is transformed into the NC traffic model. Unfortunately, such model transformation will introduce pessimism [17].

The work in this paper is motivated by (*i*) the need for cost-effective real-time communication solutions and (*ii*) the lack of real-time analysis of periodic real-time traffic for FCFS queuing. To that end, we study how to predict the worst-case delay for periodic hard real-time traffic over packet-switched networks with only FCFS queuing.

* Corresponding author.
Email addresses: xing.fan@hh.se (X. Fan)

The choice of not modifying network components rises the question of how to handle burstiness and jitter in the analysis. Burstiness is the variance in traffic rate and jitter is the variation of a time metric. While it is possible to model the incoming traffic at the second hop, it is much more difficult to achieve accurate models of the traffic flows after the second hop (in networks with multiple switches) because of the difficulties in predicting aggregated jitter introduced by the previous hops. *Hence, we face the challenge of predicting the traffic interference and re-characterizing the traffic arrival pattern in the intermediate network elements.*

We have published a preliminary analytical framework for single-switch networks [16]. The current paper extends that work with theoretical proofs and analysis for networks with multiple switches. To that end, we have the following detailed contributions.

- We propose a real-time analysis with a flexible model of the network and its traffic, allowing analysis of networks with multiple switches, variable-sized frames, arbitrary deadlines and switches with different bit-rate ports. In contrast, many existing real-time analyses in the literature are only developed for simple cases, for example, deadline being equal to period [5], a single-switch network [12] [13], switches with homogeneous bit rate ports [5] [12-15] or a fixed frame size [5].
- We show the correctness of our analysis by theoretical proofs. In contrast, the work on FCFS analysis in [5] does not provide any formal correctness proof of the worst-case delay calculations.
- We give theoretical proofs for the tightness of our worst-case delay analysis for network components in single-switch networks. In contrast, the delay estimations for such components are less tight in the NC analysis [11-14].
- We derive the maximum required buffer size. In contrast, some real-time analyses assume limited buffer size, which may lead to inefficient link utilization.
- We have conducted a comparison study between our analysis and NC. Our conceptual comparison shows that our analysis is tight for network components in single-switched networks, while NC is not for those cases because of the way that the traffic is modeled.
- We have developed a theory for transforming the periodic model into the rate-and-burstiness-constrained model, which has not been proposed before. Such model transformation provides the option of deriving delays for periodic traffic with NC. In this way, a better analytical scheme for any given system with periodic real-time traffic can be chosen.

Moreover, we have conducted simulations and a comparison study to evaluate the performance of our approach.

The remainder of the paper is organized as follows. Section 2 introduces the network model and the terminology. The real-time analysis for isolated network elements is presented in Section 3. The real-time analysis for a whole network is reported in Section 4. Section 5 describes a comparison study between our analysis and NC. Section 6 presents the simulation evaluation of our analysis. Finally, Section 7 concludes the paper.

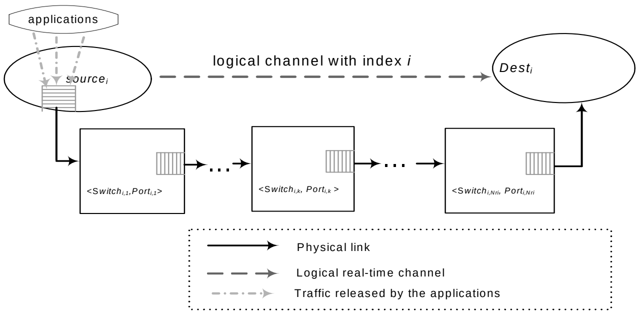

applications
logical channel with index i
$source_i$
$Dest_i$
$<Switch_{i,1},Port_{i,1}>$
$<Switch_{i,k}, Port_{i,k}>$
$<Switch_{i,Nri}, Port_{i,Nri}$
Physical link
Logical real-time channel
Traffic released by the applications

Figure 1. Physical link and logical channel.

2. Network model, traffic model, assumptions and relaxations

2.1 Network model

We consider a network with *Nnode* nodes and *Nswi* switches, which enables the structuring of different network topologies and different configurations, thereby supporting different types of applications. Each node and switch in the network employs FCFS queuing, that is, frames are taken from the queue in the order of arrival.

A *physical link* is a unidirectional transmission link which accepts network traffic from one network element and transfers network traffic to another network element at a constant bit rate. The bit rate of the physical link originating from source node k is denoted as $Rnode_k$ (bits/s) and the bit rate of the physical link originating from the output port p in switch s is denoted as $Rswi_{s,p}$ (bits/s).

2.2 Traffic model

A *logical real-time channel* (with index i), τ_i, is a virtual unidirectional connection from the source node, $Source_i$, to the destination node, $Dest_i$. The network maintains *Nch* multiple simultaneous logical real-time channels. As illustrated in Figure 1, once a logical channel τ_i is established, the route, denoted by $Route_i$, is determined. $Route_i$ is a sequence of physical links each originating from a certain output port in a certain switch and can be expressed as a vector of switch/port pairs:

$$Route_i = (<Switch_{i,k}, Port_{i,k}>),\ \ k = 1,\dots, Nr_i, \qquad (1)$$

where Nr_i indicates the total number of switches on the route, $Switch_{i,k}$ indicates the kth switch on the route and $Port_{i,k}$ indicates the output port in $Switch_{i,k}$ being used. Note that we have chosen to treat the source node link separately from the switch links, because the subsequent real-time analysis is different (as will be shown in Section 3).

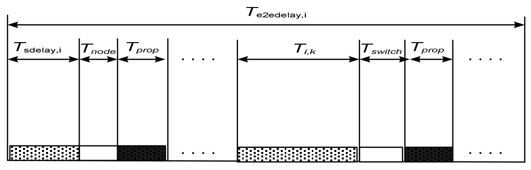


Figure 2. Timing diagram of the worst-case end-to-end delay over the whole network.

A *periodic logical real-time channel* τ_i is one which releases messages regularly with a constant interval. It is characterized by the period, *Tperiod*$_i$, which is the time interval between the message releases, the traffic volume, C_i (in bits), which is the maximum amount of traffic including data and protocol header per period, and the end-to-end relative deadline, *Tdl*$_i$, which is the time constraint for τ_i specified by the application. For convenience, a periodic logical real-time channel will be called a *real-time channel* in the rest of the paper.

To determine whether a real-time channel meets its timing requirement, we need to find out whether or not the end-to-end worst-case delay, *Te2edelay*$_i$, exceeds the deadline. A real-time channel τ_i is said to be *schedulable* if *Te2edelay*$_i$ $\leq$ *Tdl*$_i$. A *feasible link* is a physical link for which the set of real-time channels allocated to it are schedulable and a *feasible network system* is one for which every link is feasible.

As illustrated in Figure 2, *Te2edelay*$_i$, consists of:

- *Tsdelay*$_i$, the worst-case delay at the source node *i*.
- $T_{i,k}$, the worst-case delay at each switch/port <*Switch*$_{i,k}$, *Port*$_{i,k}$>.
- *Tnode*, the worst case latency for a frame in the head of the queue to leave a source node.
- *Tswitch*, the worst case latency for a frame in the head of the queue to leave a switch/port.
- *Tprop*, the propagation delay over a physical link.

Tnode and *Tswitch* are caused by the non-preemptive transmission, because we cannot interrupt the transmission of frames already being stored in the Network Interface Card (NIC) or being transmitted on a physical link. All timing parameters are expressed in seconds.

2.3 Assumptions

- The real-time channels are independent in that there is no shared resources other than the physical links.
- There is no switch processing overhead cost. Neither routing delay, nor error handling delay nor other delays associated with performing switching functions is considered in our analysis. It is, however, easy to add a worst-case constant for such delays in the analysis
- The network employs deadlock free routing.

2.4 Relaxations

- The deadline for a real-time channel is not related to its period. This means that deadlines may be shorter than periods or longer than periods.
- Real-time channels do not necessarily release their first messages at the same time. In fact, any release pattern can be assumed.
- A message is a sequence of frames, possibly of varying sizes.
- Our analysis supports switches with homogeneous bit-rate ports as well as switches with different bit-rate ports.

3. Real-time Analysis for Isolated Network Elements

Estimation of the worst-case delay at every hop in the route of a real-time channel is of critical importance to estimate the end-to-end worst-case delay over the channel. It can be shown that the delay is caused by the traffic interference among the real-time channels and the FCFS queuing policy [17]. With the knowledge of the periodic model at the source nodes, it is possible to model the incoming traffic at the second hop. However, it is much more difficult to achieve accurate models of the traffic after the second hop because of the difficulties in predicting aggregated jitter introduced by the previous hops.

The different levels of knowledge of the traffic characteristics in the intermediate network elements suggest us to develop three separate analytical schemes (see [17] for an extended argumentation):

1) for a source node,
2) for a switch only receiving RT traffic from source nodes, and
3) for a switch receiving traffic from source nodes as well as other switches.

In the following sub-sections, we provide detailed descriptions of how to derive the worst-case delay for these three different cases. The link propagation delays and the delay caused by non-preemptive packet transmission are not included in this analysis. Instead, we will include them in the end-to-end worst-case delay analysis in Section 4.

3.1 Case 1: Source node receiving traffic from applications

To analyze the schedulability of a real-time channel τ_i, it is important to find the *critical instant,* the message release pattern of all the real-time channels that leads to the worst-case delay of τ_i. If the channel does not miss its deadline in the case of the critical instant, it will not miss the deadline in any other case.

Our analysis strategy for Case 1 is as follows: we first find the critical instant (Lemma 1), and then proceed to analyze the worst-case delay (Theorem 1) and finally derive the maximum required buffer size for the source node (Corollary 1).

Proof idea of Lemma 1

In Lemma 1, we will prove that the critical instant for an FCFS-scheduled channel set is the *synchronous pattern*, the scenario in which all real-time channels release their first messages at the same time. The proof is inspired by the strategy used for analyzing EDF scheduling [19] [20]. However, in contrast to EDF, an arriving frame to an FCFS queue will always have to wait for the completion of the transmission of the already-queued frames. Therefore, the queuing delay corresponds to the amount of traffic in the output queue at the arrival time, referred to as queuing population and expressed in bits. Obviously, the worst-case delay corresponds to the maximum queuing population. The goal of Lemma 1 is to prove that, given any message release pattern, the peak value of the queuing population is not higher than caused by the synchronous pattern.

We now introduce some definitions that will be used throughout our analysis.

Definition 1 The *cumulative workload*, $W_k(t_1, t_2)$ (in bits), for a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ originating from source node k is the sum of the traffic volume of messages released by the real-time channels during time interval $[t_1, t_2), (\forall i,\ t_1 \le r_i)$, that is,

$$W_k(t_1,t_2)=\sum_{i=1}^{n}\max\left(\left(\left\lfloor\frac{t_2-r_i}{Tperiod_i}\right\rfloor+1\right)C_i,0\right),$$

where r_i is the time instant when τ_i releases its first message.

Definition 2 The *busy-period* is the first interval of continuous link utilization time in the schedule of a synchronous periodic channel set. The length of the synchronous busy-period, $BP(\Gamma)$, is the length of the synchronous busy-period of the channel set $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ allocated to one physical link (with link rate R expressed in bit/s), which is calculated by the following iterative computation

$$\begin{cases} BP^0(\Gamma)=W(0,0)=\sum_{i=1}^{i}C_i; \\ BP^i(\Gamma)=W(0,\dfrac{BP^{i-1}(\Gamma)}{R}),\ i\ge 1; \end{cases},$$

and

$$BP(\Gamma)=BP^i(\Gamma), if\ BP^i(\Gamma)=BP^{i-1}(\Gamma).$$

Lemma 1. Assume that FCFS queuing is used to schedule a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ on the physical link originating from the source node k. Then the critical instant for any channel τ_i is the synchronous pattern of Γ.

Proof. Given any message release pattern, assume that the peak value of the queuing population occurs at time instant t ($t \ge 0$). Obviously, t is in a busy-period. Let t' be the end of the last link idle period before t ($0 \le t' \le t$), as illustrated in Figure 3.

If the given message release pattern is not the synchronous pattern, t' must still be the message release time of at least one real-time channel. Without loss of generality, assume

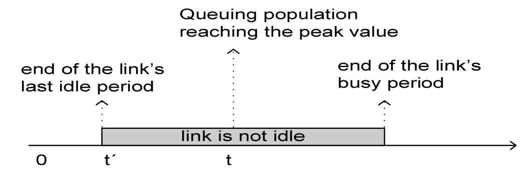

Queuing population reaching the peak value
end of the link's last idle period
end of the link's busy period
link is not idle
0
t'
t

Figure 3. Timing figures used in the proof of Lemma 1.

that τ_1 releases its first message at t' and τ_i ($2\le i \le n$) releases its first message at $t' + \Delta r_i$, ($\Delta r_i \ge 0$). If we shift the first message of every other real-time channel for which $\Delta r_i > 0$ to t' (preserving their periodicity), then the cumulative workload during the time interval $[t', t)$ will not be decreased. Consequently, the queuing population at any time instant during $[t', t)$ will not be decreased. Also, the peak value of the queuing population after the shifting will not be less than the previous peak value, and it will occur at or before t.

Since there was no link idle time during the time interval $[t', t)$, there will be no link idle time during the time interval $[t', t)$ after the shifting, due to the non-decreased workload.

If we now consider the message release pattern from time t' on, we obtain the synchronous pattern and the worst-case situation (maximum queuing population) occurs during the first link busy period. □

Proof idea of Theorem 1

Lemma 1 suggests studying the synchronous pattern of an FCFS-scheduled channel set in the first busy period. In Theorem 1, we will exploit this to calculate the worst-case delay and prove that the achieved result is tight (the minimum bound without any overestimation). First, we derive the queuing population of source node k expressed as a function of time, $QP_k(t)$, by calculating the difference between the cumulative workload arriving before t and the amount of traffic being removed before t. Recall that to find the worst-case delay, we need to find the maximum queuing population. Thus, in the next step, we find $max\{QP_k(t),\ t \ge 0\}$. Finally, we show that the obtained worst-case delay is also tight.

Theorem 1 Assume that FCFS queuing is used for a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ in the same source node k. If the link utilization

$$Unode_k=\sum_{i=1}^{n}\frac{C_i}{Rnode_k Tperiod_i}\le 1,$$

then

$$Tsdelay_k=\sum_{j=1}^{n}\frac{C_j}{Rnode_k}.$$

In addition, the worst-case delay occurs at the beginning of the busy-period.

Proof. According to Lemma 1, the critical instant is the synchronous pattern. Without loss of generality, we therefore assume all real-time channels to release their first messages at time 0.

For $\forall t, 0 \le t \le BP(\Gamma)$, the cumulative workload of messages arriving before t is:

$$W_k(0,t) = \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{t}{Tperiod_j} \right\rfloor \right) C_j \tag{2}$$

Since $\sum_{j=1}^{n} \frac{C_j}{Rnode_k Tperiodj} \le 1$, we have:

$$\sum_{j=1}^{n} \frac{C_j}{Tperiod_j} \le Rnode_k . \tag{3}$$

Hence,

$$\sum_{j=1}^{n} \frac{C_j}{Tperiod_j} - Rnode_k \le 0 \tag{4}$$

Since t is in the first busy period, $Rnode_k \cdot t$ bits are transmitted during $[0, t)$ (removed from the queue). Consequently, according to Equation 2 and Equation 4, the amount of bits remaining to be transmitted at time t, $QP_k(t)$, is:

$$\begin{aligned} QP(t) &= W_k(0,t) - t \cdot Rnode_k \\ &= \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} \left(\left\lfloor \frac{t}{Tperiod_j} \right\rfloor \right) C_j - t \cdot Rnode_k \\ &\le \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} \left(\frac{t}{Tperiod_j} \right) C_j - t \cdot Rnode_k \\ &= \sum_{j=1}^{n} C_j + t \cdot \left(\sum_{j=1}^{n} \frac{C_j}{Tperiod_j} - Rnode_k \right) \\ &\le \sum_{j=1}^{n} C_j \end{aligned} \tag{5}$$

This shows that the maximum queuing population is:

$$\max\{QP_k(t), t \ge 0\} = \sum_{j=1}^{n} C_j \tag{6}$$

With the maximum queuing population, the worst-case delay, *Tsdelay_i*, is calculated as:

$$Tsdelay_i = \sum_{j=1}^{n} \frac{C_j}{Rnode_k}, (\forall \mathrm{i} \in [1, \mathrm{n}]) \tag{7}$$

In order to show that $Tsdelay_i$ is the tight worst-case bound without any overestimation, we calculate (using Equation 5) the queuing population at time 0,

$$\begin{aligned} QP_k(0) &= W_k(0,0) - 0 = \sum_{j=1}^{n} \left(1 + \left\lfloor \frac{0}{Tperiod_j} \right\rfloor \right) C_j - 0 \\ &= \sum_{j=1}^{n} C_j = \max\{QP_k(t), t \ge 0\}. \end{aligned} \tag{8}$$

We can now see that

$$\sup\{QP_k(t), t \ge 0\} = QP_k(0), \tag{9}$$

which proves that the maximum queuing population does occur, and it occurs at time 0 (the beginning of the first busy-period according to our assumptions). In other words, the tight bound for the worst-case delay is $\sum_{j=1}^{n} \frac{C_j}{Rnode_k}$.

This concludes the proof. □

Using Theorem 1, we are now able to calculate the maximum required buffer size in the source node.

Corollary 1 Assume that FCFS queuing is used for a set of real-time channels $\Gamma = \{\tau_1, \tau_2, .., \tau_n\}$ in the same source node k. If the link utilization

$$UNnode_k = \sum_{j=1}^{n} \frac{C_j}{Tperiod_j \cdot Rnode_k} \le 1,$$

then the maximum required buffer size at source node k is

$$BN_k = \sum_{j:Source_j = k} C_j .$$

Proof. See Equation 6 in Theorem 1. □

3.2 Case 2: Switch only receiving traffic from source nodes

In contrast to the source node case, the worst-case delay at an output port of a switch does not always occur at the beginning of the first busy-period. The reason is that the burstiness of the incoming traffic is limited by the incoming physical link. Nevertheless, the strategy for the worst-case delay analysis is still to first find the critical instant (Lemma 2 and Theorem 2), and then to calculate the worst-case delay and the maximum required buffer size (Algorithm 1).

Proof idea of Lemma 2

Let the second hop be represented by <*s*, *p*>, output port *p* at switch *s*. In Lemma 2, we will prove that, for the case of the synchronous pattern at the source node, the cumulative traffic from that node to <*s*, *p*> is not less than that for any other cases of release patterns. To that end, we analyze the *cumulative traffic volume, Traffic_k*(t', t), from a source node k to <*s*, *p*> during [t', t). The cumulative traffic volume is the amount of traffic that has been delivered from a source node to <*s*, *p*> during a certain time interval, and it affects the workload at <*s*, *p*> which may include traffic from other source nodes as well. It is proven that the

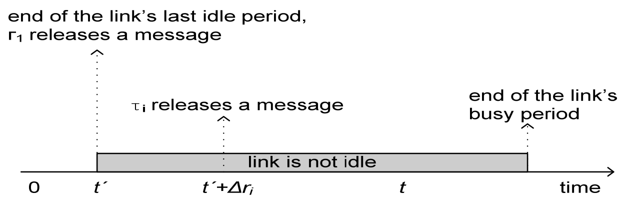

end of the link's last idle period,
τ_1 releases a message
τ_i releases a message
end of the link's busy period
link is not idle
0
t'
$t'+\Delta r_i$
t
time

Figure 4. Timing figures used in the proofs of Lemma 2.

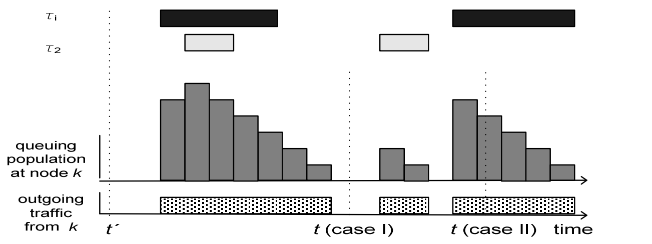


Figure 5. Illustration of workload and outgoing traffic from node k.

cumulative traffic volume under the synchronous pattern is not less than any other message release pattern by considering two cases.

Lemma 2 Assume that FCFS queuing is used to schedule a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ on the physical link originating from source node k. Also, assume that the outgoing link from port p of switch s is busy at time instant t and t' is the end of the last link idle period before t ($0 \le t' \le t$). Then, the volume of cumulative traffic from source node k to port p of switch s during $[t', t)$ under the synchronous pattern (the first messages are released at time t' at node k) is not less than that under any other release pattern ($0 \le t' \le t$).

Proof. Given any message release pattern, without loss of generality, assume that τ_1 releases its first message at t' and τ_i ($2 \le i \le n$) releases its first message at $t' + \Delta r_i$, ($\Delta r_i \ge 0$), as illustrated in Figure 4.

The cumulative workload for the incoming link from node k during $[t', t)$, $W_k(t', t)$ is derived as:

$$W_k(t',t)=\sum_{i=1}^{n}\left(1+\left\lfloor\frac{t-t'-\Delta r_i}{Tperiod_i}\right\rfloor\right)C_i. \tag{10}$$

We will now analyze $Traffic_k(t', t)$ by considering the two cases, illustrated in Figure 5, where at time instant t the link from k is idle (Case I) and busy (Case II), respectively. Note that the link from node k to switch s may be idle at some time instants when the outgoing link from $<s, p>$ is busy during $[t', t)$, because there can be traffic from other incoming links. Let t_{idle} represent the accumulated length of all the link idle periods during $[t', t)$.

Case I. The link from node k is idle at time instant t. Hence, the amount of link capacity during $[t', t)$ is not used 100%, that is,

$$\begin{aligned}Traffic_k(t',t)&=Rnode_k(t-t'-t_{idle})\\&\le Rnode_k(t-t')\end{aligned}. \tag{11}$$

Also, the link being idle at time instant t indicates that the cumulative workload during $[t', t)$ has been delivered to the next hop, that is:

$$Traffic_k(t',t)=W_k(t',t). \tag{12}$$

Case II. The link from k is busy at time instant t. Note that the link from node k may have one or several idle periods during $[t', t)$. Thus, we still have:

$$\begin{aligned}Traffic_k(t',t)&=Rnode_k(t-t'-t_{idle})\\&\le Rnode_k(t-t')\end{aligned}, \tag{13}$$

And the remaining traffic at node k indicates:

$$Traffic_k(t',t)\le W_k(t',t). \tag{14}$$

If we shift the first message of every other real-time channel for which $\Delta r_i > 0$ to t' (preserving their periodicity), the workload of the link originating from node k after the shifting, $W_k(t', t)$, will not be decreased, that is:

$$W'_k(t',t)\ge W_k(t',t). \tag{15}$$

Let $Traffic'_k(t', t)$ represent the amount of cumulative traffic from source node k to $<s, p>$ during $[t', t)$ after the shifting. Now, we will analyze $Traffic'_k(t', t)$ and prove that $Traffic'_k(t', t) \ge Traffic_k(t', t)$ holds for each of the cases. Let t'_{idle} represent the cumulated length of all the link idle periods during $[t', t)$ after the shifting.

The analysis of Case I.

a) If the link is still idle at time t after the shifting, then the workload during $[t', t)$ has been consumed. That is, $Traffic'_k(t', t) = W'_k(t', t)$. According to Equation 12 and Equation 15, we have $Traffic'_k(t', t) \ge Traffic_k(t', t)$.

b) Otherwise, if the link is busy at time t after the shifting, there may still be link idle time during $[t', t)$. Therefore, we have:

$$\begin{aligned}Traffic'_k(t',t)&=Rnode_k(t-t'-t'_{idle})\\&\le Rnode_k(t-t')\end{aligned}. \tag{16}$$

According to Equation 15, the non-decreased workload after the shifting indicates a non-increased length of idle time, that is:

$$t'_{idle}\le t_{idle}. \tag{17}$$

Consequently, $Traffic'_k(t', t) \ge Traffic_k(t', t)$ holds.

The analysis of Case II.

a) If the link is idle at time t after the shifting, then the workload during $[t', t)$ has been consumed. That is, $Traffic'_k(t', t) = W'_k(t', t)$. Consequently, according to Equation 14 and Equation 15, we have $Traffic'_k(t', t) \ge Traffic_k(t', t)$.

b) Otherwise, if the link is busy at time t after the shifting, we can use the same arguments as used in the sub-case b) of Case I. The non-decreased workload leads to a non-increased length of idle time. Hence, $Traffic'_k(t', t) \ge Traffic_k(t', t)$ still holds.

The above analysis yields:

$$Traffic_k'(t',t) \geq Traffic_k(t',t). \tag{18}$$

This concludes Lemma 2. □

Proof idea of Theorem 2

Lemma 2 provides us with two useful findings. First, it gives a hint on how the critical instant at <*s*, *p*> can be found by studying the synchronous pattern at the source node. Second, the use of multiple cases and sub-cases in the proof highlights the difficulty of deriving a simple analytical expression for the worst-case delay at <*s*, *p*>. This indicates a need for an algorithm to calculate it.

In Theorem 2, we will prove that the critical instant at <*s*, *p*> is still a consequence of the synchronous pattern at the source nodes by looking at the aggregated incoming traffic from all the source nodes. Our proof idea of Theorem 2 is as follows. First, we partition the channel set into subsets according to their origins and calculate the cumulative traffic volume from each source node to <*s*, *p*>. Second, we calculate the queuing population for <*s*, *p*> expressed as a function of time, $QP_{s,p}(t)$, by calculating the difference between the aggregated traffic from all the incoming links arriving before *t* and the amount of traffic being removed before *t*. Recall that, by finding the maximum queuing population, we also get the worst-case delay. Thus, as the third step, we find the critical instant.

Theorem 2 Assume that FCFS queuing is used to schedule a set of real-time channels $\Gamma=\{\tau_i: 1\leq i \leq n\}$, all of which traverse the same output port *p* of switch *s*. Then, the critical instant for any channel τ_i at the output port is a direct consequence of the synchronous pattern of Γ at the source nodes.

Proof. Assume that channel set Γ can be decomposed into several subsets:

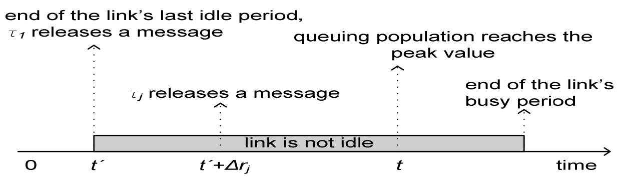
$$\Gamma = \bigcup_{k=1}^{Nnode} \Gamma_k \ , \ \Gamma_k = \{\tau_i : Source_i = k\ ) \ , \tag{19}$$

where Γ_k includes the real-time channels originating from the same source node *k*.

Given any message release pattern, assume that the peak value of the queuing population in the queue for <*s*, *p*> occurs at time instant *t* ($t \geq 0$). Obviously, *t* is in a busy-period. Let *t'* be the end of the last link idle period before *t* ($0 \leq t' \leq t$).

Even if the message release pattern is not the synchronous pattern at the source nodes, *t'* must be the message release time of at least one real-time channel. Without loss of generality, assume that τ_1 is one of the channels that releases a message at *t'*, and that any other channel τ_j ($2 \leq j \leq n$) releases a message at $t' + \Delta r_i$, ($\Delta r_i \geq 0$), as illustrated in Figure 6.

The workload for the outgoing physical link of <*s*, *p*> during [*t'*, *t*), $W_{s,p}(t', t)$, is the aggregated traffic from the incoming links, that is:



end of the link's last idle period,
τ_1 releases a message
queuing population reaches the peak value
τ_j releases a message
end of the link's busy period
link is not idle
0
t'
$t'+\Delta r_j$
t
time

Figure 6. Timing figures used in the proofs of Theorem 2.

$$W_{s,p}(t',t) = \sum_{k=1}^{Nnode} Traffic_k(t',t) \tag{20}$$

Hence, the amount of bits that remains to be transmitted at time *t*, $QP_{s,p}(t)$, is:

$$QP_{s,p}(t) = W_{s,p}(t',t) - (t-t')Rswi_{s,p}\ , \tag{21}$$

Where $Rswi_{s,p}$ is the rate of the physical link at output port *p* in switch *s* (bits/s)

Now, shift the first message of every other real-time channel for which $\Delta r_i > 0$ to *t'* (preserving their periodicity). Then, let $Traffic'_k(t', t)$ represent the amount of cumulative outgoing traffic from source node *k* to <*s*, *p*> during [*t'*, *t*) after the shifting. According to Lemma 2,

$$Traffic'_k(t', t) \geq Traffic_k(t', t). \tag{22}$$

Similarly, the workload for the outgoing physical link from <*s*, *p*> during [*t'*, *t*) after the shifting, $W'_{s,p}(t', t)$, is:

$$W'_{s,p}(t',t) = \sum_{k=1}^{Nnode} Traffic'_k(t',t). \tag{23}$$

Hence, according to Equations 20, 22 and 23, we have:

$$W'_{s,p}(t',t) \geq W_{s,p}(t',t). \tag{24}$$

Consequently, the queuing population after the shift, $QP'_{s,p}(t)$, will not be decreased, because:

$$\begin{aligned} QP'_{s,p}(t) &= W'_{s,p}(t',t) - (t-t')Rswi_{s,p} \\ &\geq W_{s,p}(t',t) - (t-t')Rswi_{s,p} = QP_{s,p}(t). \end{aligned} \tag{25}$$

Since the outgoing link from <*s*, *p*> is not idle during [*t'*, *t*) before the shifting, there will be no link idle time during [*t'*, *t*) after the shifting. If we now consider the message release pattern from time *t'* on, we have obtained the synchronous pattern at the source nodes and the maximum queuing population (the worst case) at the switch port with no link idle period prior to it.

This concludes Theorem 2. □

The idea of Algorithm 1

Lemma 2 shows the difficulty of finding a simple analytical expression for calculating the worst-case delay and the maximum buffer size. For this reason, we have developed a utility function (Algorithm 1) to find the maximum value by checking the queuing population at different time instants.

First, the algorithm does not have to calculate the worst-case delay for every real-time channel traversing <*s*, *p*>, since the delay is the same for every channel, that is,

```
Algorithm 1 Algo_1($s, p$);
  Input ($s, p$);
  Output ( $Dport_{s,p}$, $BS_{s,p}$)

1. Initialization.
   Input ($s, p, Nchs, Nnode, Nswi, Nport[1..Nswi], Rnode[1..Nnode], Rswi[1..Nswi][1..\ Nport[1..Nswi]]$);
   $t=0; tstep=0; Q=0;\ Dport_{s,p} =0; Pnode=zeros[1..Nnode]$;
2. Calculate the queuing population, $Q$, for $<s,p>$ at time instant $t$.
   2.1 Keep track of the worst-case queuing delay, $Dport_{s,p}$.
       if ( $Q > Dport_{s,p}$ )  then  $Dport_{s,p} = Q$;  end if
   2.2 Calculate how many bits that are on the way for each incoming physical link queued in each source node.
       for $i = 1..Nchs$
           if $<s, p> = <Switch_{i,1}, Port_{i,1}>$ && $mod(t, T_{period,i})==0$
              then $Pnode[Source_j] = Pnode[Source_j] + C_j$ ;
           end if
       end for
   2.3 Send bits from each incoming link to the output queue in the switch.
       for $i=1..Nports$
           if ($Pnode[i] >0$)
               then $Pnode[i]= Pnode[i] - Rnode_j\ tstep$;
                    $Q = Q + Rnode_j\ tstep$;
           end if
       end for
   2.4 Remove bits from the output buffer.
       $Q = Q - Rnode_j\ tstep$;
   2.5 Calculate the next check time instant, either the time when the queue at an end node is empty or the time when a new period of one logical real-time channel starts.
       $tstep = \min\left( \bigcup_{i=1}^{Nnode} \frac{Pnode[i]}{Rnode_i} \cup \bigcup_{i=1}^{Nchs} \left( T_{period,i} - \mathrm{mod}(t, T_{period,i}) \right) - \{0\} \right);$
   2.6 Increase $t$ and reset $tstep$.
       $t=t+tstep$; $tstep=0$;
3. If it is not the end of the first busy period, check the queuing population at the next time instant.
   if ( $t < BP(\Gamma)$ )  then  repeat step 2;  end if
4. Return ( $Dport_{s,p} =Dport_{s,p} / Rswi_{s,p}$ , $BS_{s,p} =Dport_{s,p}$).
```

$$T_{i,1} = T_{j,1} = Dport_{s,p}$$
$$if\quad \langle switch_{i,1}, port_{i,1} \rangle = \langle switch_{j,1}, port_{j,1} \rangle = \langle s, p \rangle, \tag{26}$$

where $Dport_{s,p}$ is the worst-case delay for the frames through $<s, p>$.

Second, Theorem 2 suggests that the examined interval can be limited to the first busy-period, which reduces the time and memory complexity. Without loss of generality, the start of the busy-period is assumed to be at time instant 0 in Algorithm 1.

Third, making queuing population checks at all possible time instants suffers from intractable computational complexity. In our analysis, checking the queuing population will therefore be event driven (see below for a detailed motivation). We then simulate the transmission over the physical links in Algorithm 1 by keeping track of the amount of incoming traffic after each event. The simulator works as follows. After each event, we transfer a certain amount of bits from each incoming physical link to the output queue in the switch. Meanwhile, if the output queue in the switch is not empty, a certain amount of bits are sent to the outgoing link.

Last, the output of Algorithm 1 is the tight worst-case delay because we do not miss the peak value and we do not overestimate according to the previous discussion.

The idea of event driven checking in Algorithm 1

In Algorithm 1, the queuing population will be checked when 1) a new period of any related logical real-time channel starts or 2) the output queue from any source node becomes empty. Our rationale for this checking strategy is as follows.

According to Equation 21, the task of checking the queuing population corresponds to checking the value of $Traffic_k(t', t)$. We will now derive the guidelines of how to estimate $Traffic_k(t', t)$ by considering the two cases, as illustrated in Figure 5.

As long as t remains in the same idle period (Case I), the value of $Traffic_k(t', t)$ does not change. Therefore, we only need to keep track of its value at the beginning of this idle period, that is, the time instant when the output queue at the source node becomes empty.

If the link from k is busy at time instant t (Case II), as long as t remains in the same busy period, the value of $Traffic_k(t', t)$ increases monotonically. Therefore, we only need to keep track of its value at the beginning and at the end of this busy period. Note that a busy period is caused by a message release of at least one real-time channel. Thus, we need to recalculate the value of $Traffic_k(t', t)$ when any related real-time channel starts a new period.

The above observations suggest us to check the queuing population only at such time instants at which a new period of any related logical real-time channel starts or at which the output queue from any source node becomes empty. It should be noted that the queuing population of an output queue in the switch is either constant, monotonically increasing or monotonically decreasing between two such adjacent time instants. This guarantees that we do not exclude the peak values of the queuing population. After checking all such critical time points during the first busy-period of the synchronous release pattern, we consequently get the maximum queuing population.

3.3 Case 3: Switches receiving traffic from source nodes as well as other switches

Similar to the analyses for the previous cases, we first find the critical instant and then calculate the maximum queuing population and worst-case delay.

Recall that the difficulties in predicting aggregated jitter leads to a decreased accuracy in the traffic models after the second hop. Without an accurate traffic model, there is no well-established way of computing the *tight* worst-case delay. However, it is still possible to obtain a guaranteed worst-case delay after the second hop, but at the price of some pessimism in the estimation.

Similar to Case 2 (Lemma 2), the critical instant causing the worst-case delay at the considered hop is the scenario in which the first messages of all the relevant channels (those traversing the considered switch/port) arrive at their previous hops at the same time. Hence, we can deal with the worst-case delay analysis in a similar way as we did for Case 2: we first derive the worst-case delay in the first busy-period and then use a utility function to calculate the queuing population.

Unfortunately, a new problem that does not occur in Case 2 complicates the delay analysis after the second hop. Figure 7 illustrates this issue regarding the delay at <*s*, *p*>. Here, τ_1 and τ_2 are the relevant channels, while τ_3 is an interfering channel. τ_1 originates from node 1, traverses port <*s'*, *p'*>, port <*s*, *p*> and finally arrives at node 2. τ_2 is

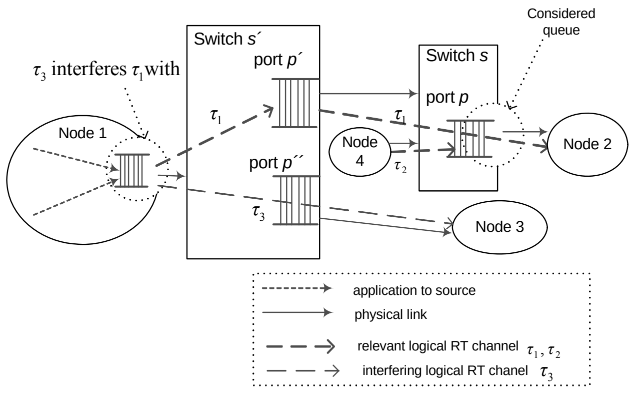


Figure 7. Traffic interference after the second hop.

relevant because it also traverses port <*s*, *p*>. τ_3 originates from node 1, traverses port <*s'*, *p''*> and finally arrives at node 3. Although τ_3 does not traverse port <*s*, *p*>, it does interfere with τ_1 at source node 1. Therefore, even at the beginning of the busy period of the outgoing link of port <*s*, *p*>, when the messages of τ_1 and τ_2 leave at their previous hops at the same time, not-yet-transmitted frames belonging to τ_1 might remain in the output queue of the previous port <*s'*, *p'*>, due to interference with frames of τ_3 at node 1 during the previous busy period.

This observation suggests that the delay analysis should also consider the remaining frames in the intermediate switches. Specifically, in order to find the worst-case delay at such a port, we also need to find the upper-bound of the additional delay introduced by such remaining frames in the previous port. For Case 2, it is possible to obtain a tight upper bound of the additional delay because of the exact knowledge of traffic characteristics at the source node. However, for Case 3, we cannot predict for how long time the messages have remained in the buffer since we choose to not use traffic regulators in the switches. This leads to the challenge of predicting the maximum volume of remaining frames. Fortunately, the problem can still be solved utilizing the analysis for the previous hops. To that end, we conceive an idea of reducing the problem to easily solvable cases.

The idea of Algorithm 2

Before we present our algorithm, we use the example in Figure 7 to explain our idea. For example, to analyze the delay at port <*s*, *p*>, we need to calculate the maximum population of the remaining frames of each relevant channel at the previous hop if the previous hop is a switch/port (<*s'*, *p'*> in this example). To that end, we will assume that the maximum population of the remaining frames at <*s'*, *p'*> equals the maximum queuing population at <*s'*, *p'*> $BS_{s',p}$.

By taking the above discussions into account, we can now propose an iterative way to solve the problem. The iterative algorithm is described in Algorithm 2. To calculate the worst-case delay for channel i at its jth switch/port, $T_{i,j}$,

```
Algorithm 2   Algo_2 $(i, j)$
  Input $(i, j)$
  Output ( $T_{i,j}$ )

1  for $(k=1; k<=j; k++)$
       Call Algo_3 to calculate the worst-case delays at
       port $<switch_{i,k}, port_{i,k}>$;
   end for
2  $s=switch_{i,j}, p=port_{i,j}$
3  $T_{i,j}= Dport_{s,p}$
4  return ( $T_{i,j}$ )
```

we begin with the calculation at the first switch/port, then proceed to the next hop by taking the remaining frames into account, finally finishing when the delay for the *j*th switch/port has been calculated. The worst-case delay and buffer size calculation for each switch/port is calculated by Algorithm 3, which is an extension of Algorithm 1, in the sense that we use the similar idea of keeping track of the queuing population during the first busy period but taking the remaining frames at the previous ports into account. However, the worst-case delay calculated by Algorithm 3 may not be tight due to the aggregated jitter (jitter is the variation in arrival periodicity).

Note that the iterative idea behind Algorithm 2 implies an essential demand on deadlock free routing, meaning that the network ensures that the route for each logical real-time channel is individually loop free and that the routes for all logical real-time channels do not interact in a way that can create deadlocks.

4. Real-time analysis for the whole network

With the worst-case delays at isolated network elements, we are able to obtain the end-to-end worst-case delay and carry out a real-time analysis for the whole communication network in the section. Our analysis is developed for networks with multiple switches, which can of course be used for single-switch networks.

The time constraints are guaranteed by the addition of real-time channels. The test is divided into two steps: checking the utilization constraint and the delay constraint.

First we must check the utilization constraint. It is obvious that the utilization for the physical link from any source node k to the switch, $UNode_k$, must be less than or equal to the maximum value, 1:

$$UNode_k = \sum_{i:Source_i=k} \frac{C_i}{Tperiod_i Rnode_k} \le 1 . \qquad (27)$$

Likewise, for the physical link from any switch/port pair $<s, p>$, $Uswi_{s,p}$, must be less than or equal to the maximum value, 1:

$$Uswi_{s,p} = \sum_{i:\langle s,p\rangle \in Route_i} \frac{C_i}{Tperiod_i Rswi_{s,p}} \le 1 . \qquad (28)$$

Furthermore, we need to verify whether or not the worst-case end-to-end delay of each channel does not exceed its deadline, that is,

$$Te2edelay_i \le Tdl_i . \qquad (29)$$

As illustrated in Figure 2, the end-to-end worst case delay is,

$$\begin{aligned} Te2edelay_i &= Tsdelay_i + \sum_{j=1}^{Nr_i} T_{i,j} \\ &+ (Nr_i + 1)Tprop + Tnode + Nr_i Tswitch, \end{aligned} \qquad (30)$$

by taking the delays from Theorem 1 and Algorithm 2.

If the above utilization constraint is met for all the physical links and the delay constraint is met for all the logical real-time channels, we accept the real-time channels and guarantee that their deadlines are met.

5. Relation to Network Calculus

This section relates our analysis method to NC-LH [12] [13], a real-time switched Ethernet solution based on NC. The two approaches are based on the same network architecture (switched Ethernet) and queuing mechanism (FCFS), but use different traffic models and analytical schemes to calculate the worst-case end-to-end delays [16]. We will first describe the delay estimation used in NC-LH, and then introduce a theory for transforming the strictly periodic traffic model used in our analysis into the rate-and-burstiness-constrained model used in NC-LH. Using this theory, we will be able to derive the delays for periodic traffic with NC analysis and thereby be able to choose the better analytical scheme for any given system with periodic real-time traffic.

5.1 Delay Estimation Using NC

For the convenience of the reader, we use the notations defined in the previous sections as a common framework to explain both analyses.

In the NC-LH analysis, the delay and buffer bounds of a switch/port depend on two parameters: the *arrival curve* (the incoming traffic at the switch/port) and the *service curve* (the availability of the switch/port to send that data). These parameters are defined below for single-switch Ethernet networks [12].

Definition 3 The arrival curve, $\alpha_k(t)$, is the accumulated traffic volume from source node k during the time interval $[0, t]$. All $\alpha_k(t)$ satisfies $\alpha_k(t)=\min(Rnode_k \cdot t + Tef, r_k \cdot t + b_k)$, where r_k determines an upper-bound to the long term average rate of the traffic flow, b_k expresses the maximum burstiness of the outgoing traffic and Tef expresses the time for transferring a full-sized Ethernet frame.

```
**Algorithm 3   Algo_3** ($s, p$);
  **Input** ($s, p$);
  **Output** ( $Dport_{s,p}$, $BS_{s,p}$)

1 Initialization.
  **Input** ($Nchs, Nnode, Nswi, Nport[1.. Nswi], Rnode[1.. Nnode], Rswi[1.. Nswi, 1.. Nports[1.. Nswi]]_s\ s, p,$);
  $t=0;\ tstep = 0;\ Q = 0;\ Dport_{s,p} = 0;\ PNbuffer = zeros[1.. Nnode];\ PSbuffer = zeros[1.. swi, 1.. Nport[1.. Nswi]];$

2   Add the former remaining bits in each previous switch.
      **for** $i = 1..Nchs$
        **if** ( $\{s,p\}$ is the $k$th element of $Route_i$ ) && ($k>1$)
          **then**
               Call Algo_4_2 to calculate the worst-case delays at port $<switch_{i,k-1}, port_{i,k-1}>$;
               $PSbuffer[switch_{i,k-1}, port_{i,k-1}] = BS[switch_{i,k-1}, port_{i,k-1}]$;
        **end if**
      **end for**

3  Calculate the queuing population, $Q$, for $<s, p>$ at time $t$.
   3.1 Keep track of the worst-case queuing delay, $Dport_{s,p}$, at output port $p$ in the switch $s$.
        **if** ( $Q > Dport_{s,p}$ )    **then**    $Dport_{s,p} = Q$    **end if**
   3.2 Find out the amount of bits on the way queued for each incoming link.
     **for** $i = 1..Nchs$
        **if** ( $\{s,p\}$ is the $k$th element of $Route_i$ ) && $mod(t, T_{period,j})$==0  **then**
           **if**  $k == 1$
              **then**   $PNbuffer[Source_i] = PNbuffer[Source_i] + C_i$;
              **else**   $PSbuffer[switch_{i,k-1}, port_{i,k-1}] = PSbuffer[switch_{i,k-1}, port_{i,k-1}] + C_i$;   **end if**
        **end if**
      **end for**

   3.3 Transfer bits from the previous port to the considered port.
      **for** $i = 1..Nnode$
         **if** ($PNbuffer[i] > 0$)   **then**  $PNbuffer[i] = PNbuffer[i] - Rnode_i\ tstep$;
                                              $Q = Q + Rnode_j\ tstep$;          **end if**
      **end for**
      **for** $i = 1..Nswi$
         **for** $j = 1..Nport_i$
            **if** ($PSbuffer[i,j] > 0$) && ((i!=s) || (j!=p))  **then**  $PSbuffer[i,j] = PSbuffer[i,j] - Rswi_{i,j}\ tstep$;
                                                                  $Q = Q + Rswi_{i,j}\ tstep$;      **end if**
         **end for**
      **end for**

   3.4  Remove bits from the output buffer.
        $Q = Q - Rswi_{s,p}\ tstep$;

   3.5 Calculate the next check time instant, either the time when one incoming link is empty or the time when a new
period of one logical real-time channel starts.
$tstep = \min\left( \bigcup_{i=1}^{Nnode} \frac{PNbuffer[i]}{Rnode_i} \cup \left( \bigcup_{(i=1)\&(i \neq s)}^{Nswi} \bigcup_{(j=1)\&(j \neq p)}^{Nport_i} \frac{PSbuffer[i,j]}{Rswi_i} \right) \cup \bigcup_{i=1}^{Nchs} \left( T_{period,i} - \mathrm{mod}(t, T_{period,i}) \right) - \{0\} \right)$;
   3.6 Increase $t$ and reset $tstep$.
        $t = t + tstep;\ tstep = 0;$

4. If it is not the end of the first busy period, check the queuing delay at the next check time instant.
   **if** ( $t < BP(\Gamma)$ )   **then**    repeat step 3;   **end if**

5. **Return** ( $Dport_{s,p} = Dport_{s,p} / Rswi_{s,p}$, $B_{s,p} = Dport_{s,p}$).
```

Definition 4 The service curve, $\beta(t)$, of an Ethernet switch/port <*s*, *p*> is described by the following function

$$\beta(t) = \begin{cases} Rswi_{s,p}(t - tswitch), & t \geq tswitch \\ 0, & t < tswitch \end{cases}, \quad (31)$$

where *tswitch* is the switch-specific parameter describing the maximum delay (without queuing effects) after which the switch starts to transmit a frame once it is received.

For simplicity, *tswitch* is assumed in our analysis to be zero.

In NC-LH, the worst-case delay of a frame for the output port, *p*, at the switch, *s*, $Dport_{s,p}$, is

$$Dport_{s,p} = \sum_{k=1}^{Nnode_i} \frac{b_k}{Rswi_{s,p}} - g_{\max}\left(1 - \sum_{k=1}^{Nnode_i} \frac{r_k}{Rswi_{s,p}}\right) + tswitch, \quad (32)$$

where g_{max} is:

$$g_{\max} = \max_{k=1}^{Nnode}\left(\frac{b_k - Tef}{Rswi_{s,p} - r_k}\right). \quad (33)$$

Moreover, the maximum buffer size, or the amount of memory needed to store the queued frames, for the considered output port is given by

$$BS_{s,p} = Rswi_{s,p} Dport_{s,p} = \sum_{k=1}^{N} b_k - g_{\max}(Rswi_{s,p} - \sum_{k=1}^{Nnode} r_k) + Rswi_{s,p} tswitch \cdot \quad (34)$$

5.2 Model Transformation

The traffic models used in our analysis and in NC-LH are different. We will describe the (*r*, *b*) model and then develop a theory to transform the periodic model into the (*r*, *b*) model.

The traffic model used in the NC analytical scheme satisfies certain constraints on the average rate and burstiness. This model, called the (*r*, *b*) model [6] [8] (or T-SPECs model [12]), is defined as follows.

Definition 5 An arrival curve $\alpha_k(t)$ is said to follow the (*r*, *b*) model if $for \forall t \geq 0, \alpha_k(t) \leq r_k t + b_k \quad (r_k \geq 0, b_k \geq 0)$, where r_k determines an upper bound to the long term average rate of traffic flow and b_k expresses the maximum burstiness of the traffic.

While the (*r*, *b*) model has a flexibility that allows it to be used to model a large variety of traffic, it cannot be used directly for periodic traffic. This is because the (*r*, *b*) model assumes that the long term average rate is bounded, while periodic traffic guarantees that the momentary rate is bounded. Hence, in order to analyze the delay for periodic traffic using NC, we need to transform the periodic model into the (*r*, *b*) model. Such a transformation is given in Theorem 3.

Our idea for proving Theorem 3 is as follows. We will first derive the long term average rate and the maximum burstiness for a given set of periodic channels, and then we prove that the resulting traffic flow satisfies the (*r*, *b*) model.

Theorem 3. Assume a set of periodic real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ transmitting on the physical link from source node *k* to a switch/port. Then the (*r*, *b*) equivalent of the outgoing traffic flow from source node *k* is calculated as follows: $r_k = \sum_{i=1}^{n}(C_i / Tperiod_i), b_k = \sum_{i=1}^{n} C_i$.

Proof. The long term average rate, r_k can be calculated as:

$$r_k = \sum_{i=1}^{n}(C_i / Tperiod_i). \quad (35)$$

Following the definition of a periodic real-time channel, we can see that the maximum volume of the traffic released per period by channel τ_i is C_i. This means that channel τ_i has a maximum burstiness of C_i. Therefore, the maximum burstiness of the aggregated traffic flow is

$$b_k = \sum_{i=1}^{n} C_i, \quad (36)$$

It is known that the cumulative outgoing traffic, $a_k(t)$, from source node *k* is less than $W_k(0, t)$, the cumulative workload of Γ at source node *k* during time interval [0, *t*). In other words,

$$a_k(t) \leq W_k(0,t). \quad (37)$$

Given that $W_k(0,t) = \sum_{i=1}^{n}(\lfloor t / Tperiod_i \rfloor + 1) \cdot C_i$, Equation 37 can now be rewritten as:

$$\begin{aligned} a_k(t) &\leq \sum_{i=1}^{n}(\lfloor t / Tperiod_i \rfloor) \cdot C_i + \sum_{i=1}^{n} C_i \\ &\leq \sum_{i=1}^{n}(C_i / Tperiod_i) \cdot t + \sum_{i=1}^{n} C_i \\ &= r_k \cdot t + \sum_{i=1}^{n} C_i = r_k \cdot t + b_k \end{aligned} \quad (38)$$

Equation 40 shows that the cumulative incoming traffic from source node *k* to the port satisfies the (*r*, *b*) model.

This concludes the proof of Theorem 3.

In summary, Theorem 3 demonstrates that traffic released by a given periodic channel set can be transformed into the (r, b) model that is used in NC. Hence, this model transformation enables us to use the delay estimation for a periodic channel set using NC.

5.3 Comparison

Our theoretical proof presented in Section 3 shows the tightness of our worst-case delay analysis for network components in a single-switch network. In contrast, the maximum delay and buffer size derived using the NC may not be tight, as has been observed by other researchers [8]. We will make two important comparisons of the analysis methods in the following paragraphs.

- It has been observed that the output traffic of a switch is generally less bursty than the input traffic to it, because the outgoing flow is shaped by the physical link [8]. This observation is crucial for avoiding over-estimation of the worst-case delay at the switch ports, and we take advantage of this in our analysis. In contrast, the worst-case delay bound derived by NC is based on the assumption that all input streams may accumulate their burst arrival rates to cause a 'worst case' in which a switch may suffer from simultaneous burstiness arrivals of many real-time flows. This means that NC analysis introduces pessimism in finding delay bounds by iteratively applying output burst bounds hop-by-hop [8].
- In transforming the periodic model to the (r, b) model, some detailed information about the traffic model will be lost, such as the time instances at which the burstiness occurs. Specifically, by using a transformed NC traffic model to analyze a synchronous periodic channel set, unavoidable pessimism will exist in the calculation of the arrival curve, $a_k(t)=\min(Rnode_k \cdot t + Tef, r_k \cdot t + b_k)$. In contrast, using our proof in Section 3 for the case of the switch only receiving traffic from source nodes, we are able to use a value of $a_k(t)$ that can be less than $\min(Rnode_k \cdot t + Tef, r_k \cdot t + b_k)$ because we can detect (using Algorithm 1) that the incoming link to a switch/port may not always be busy. That is, we are able in our analysis to keep track of the exact amount of incoming traffic volume to a switch/port, without any overestimation. Consequently, the maximum delay and buffer size derived by Equations 32 and 34 will be more pessimistic than those derived by our analysis for the case of the switch only receiving traffic from source nodes.

With the transformation, it is now possible to find a better analysis method to use for periodic traffic. For example, if a single-switch network is used, these results indicate that our analysis should be chosen for schedulability analysis. In other cases, it is not clear as to which method is better. Therefore, simulations can be used to identify which method is better.

6. Simulation Evaluation

In previous sections, we discussed the pessimism introduced by our analysis and the NC analysis and showed that our analysis does not introduce pessimism in the delay derivation for the single-switch network components. However, it is not clear how much pessimism is introduced by the end-to-end delay aggregation. In this section, we use simulations to measure the pessimism introduced by the real-time analysis.

6.1 Performance metric

The most important performance metric needed to detect pessimism is the *end-to-end packet delay*, which is defined as the time from the time instant at which a packet is ready to be transmitted to the time instant the packet has successfully arrived at the destination.

Definition 6 The *predicted worst-case end-to-end packet delay*, *PD*, for a set of accepted real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ in the network is the longest end-to-end packet delay predicted by the real-time analysis, that is,

$$PD = max\ (Te2edelay_i).$$

Definition 7 The *simulated worst-case end-to-end packet delay*, *SD*, for a set of accepted real-time channels $\Gamma = \{\tau_1, \tau_2, .., \tau_n\}$ in the network is the longest end-to-end packet delay obtained by the packet level simulation.

Definition 8 The *worst-case end-to-end packet delay overestimation ratio*, *DOR*, for a set of accepted real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ in the network is the ratio of the amount of end-to-end delay overestimation to SD, that is, $DOR = (PD - SD) / \mathrm{SD}$.

We will compare the worst-case end-to-end delay predicted by the real-time analysis with the worst-case end-to-end delay assessed by the simulation. The larger the gap, the more pessimistic the analysis is.

Another important performance metric is the *network utilization*, that is, the maximum possible real-time traffic load in the network under the condition of guaranteeing the deadlines for all real-time packets. The reason we study this metric is that a higher maximum possible load generally indicates less pessimism in the worst-case delay estimation.

Definition 9 The *network utilization*, *UNet*, for the set of all the accepted real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ in the network is the average fraction of time that the allocated links are busy, that is,

$$UNet = \frac{\sum_{i=1}^{n}\left(\frac{C_i}{Tperiod_i Rnode_{source_i}} + \sum_{j=1}^{Nr_i}\frac{C_i}{Tperiod_i Rswi_{switch_{i,j},port_{i,j}}}\right)}{Nnode + \sum_{s=1}^{Nswi} Nport_s}.$$

Using the defined performance metrics, we will assess the network scalability and robustness, that is, how well the network can be utilized in different traffic characteristics and network configurations. For example, we will study how the number of real-time channels and tightness of real-time channel deadlines affect the quality of the analysis methods.

6.2 Simulation Set-up

Our simulator was built using Matlab. We have conducted both channel level simulations and packet level simulations, as described below.

Network Architecture

We simulate a switched Ethernet network with a number of full-duplex Ethernet switches and a number of end-nodes. The assumptions made are:

- The number of end-nodes ranges from 8 to 32.
- All buffers are large enough, that is, no packet loss occurs because of buffer overflow.
- The link propagation delay is 500 ns per hop, corresponding to a 100-meter link.
- Each switch has a bit rate of 100 Mbits/s on all ports, unless otherwise stated.
- The number of switches is one, unless otherwise stated.
- A tree topology is chosen for simulations of network with multiple switches.

Traffic Generation

We use a traffic generator for hard real-time channels. To that end, we make the following assumptions:

- Each real-time channel has source and destination nodes that are randomly drawn from the set of available nodes assuming a uniform distribution, unless otherwise stated. We make sure that source and destination nodes are not the same.
- Each real-time channel releases messages periodically.

Other parameters of each real-time channel, such as period, deadline and capacity, will be assigned differently for each particular evaluation study.

Admission Control

In our channel level simulations, we first generate a complete set of channels according to the guidelines above. The generated real-time channels are then checked one by one by an admission controller as to whether or not they can be accepted. The admission control is made using our real-time analyses presented in Section 4.

Packet Transmission

In our packet-level simulations, we study the packet transmissions on the channels that have passed the admission control and observe the end-to-end delay for each individually transmitted packet. To increase the likelihood of detecting the worst-case end-to-end delay, we keep track of the end-to-end delay for each packet for an interval that is 1000 times longer than the longest period among the channels. The time resolution in our simulations are virtual units, and the time resolution corresponds to the model used in our analysis in Section 3, that is, it is event-driven.

6.3 Simulation Evaluation

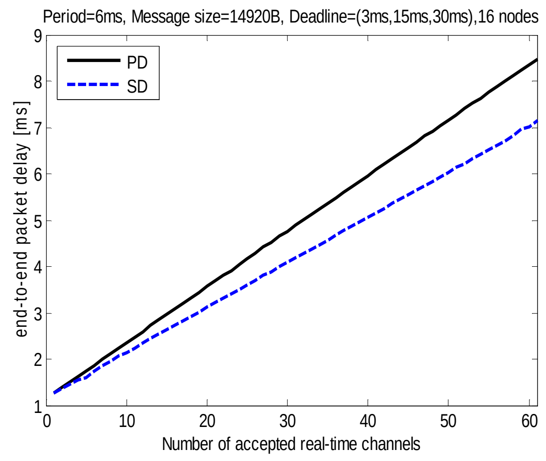
This section evaluates the performance of our analysis method using the metrics presented in Section 6.1. In all plots presented, every reported value is an average of the observed performance measure, taken over the set of simulation runs that are made for each parameter combination. All the results reported in our evaluations are average values. For each evaluation, we repeat the simulations as many times that a 99% confidence level are obtained for a maximum error within 0.5% of the average values reported.

Effect of Traffic Load on Worst-Case End-to-End Delay

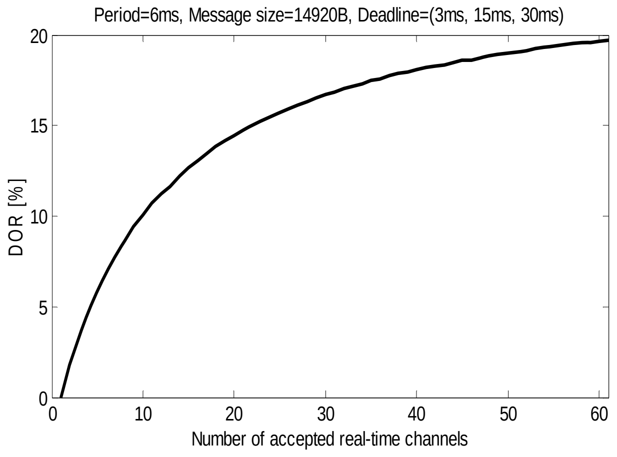
As mentioned in Section 6.1, any pessimism introduced by our real-time analysis will affect the performance. For this reason, we will start our evaluation by determining how the worst-case end-to-end delay is affected by the real-time traffic load (the number of accepted real-time channels).

Figure 8 reports the results of a simulation study that was conducted with every logical real-time channel having a period of 6 ms, a message size of 14 920 bytes (10 Ethernet frames) and end-to-end deadlines randomly chosen from the set {3 ms, 15 ms and 30 ms}. In Figure 8(a), PD and SD are plotted as a function of the number of accepted real-time channels, and the corresponding DOR is plotted in Figure 8(b).

In Figure 8, we first observe that the difference between PD and SD increases with the traffic load. For example, for 10 channels, the predicted delay is 9% higher than the real measured delay, for 30 channels the corresponding figure is 15.6%, and for 60 channels the corresponding figure is 18.5%. This behaviour corroborates our expectations in our previous discussion. The reason for this behaviour is that the increased traffic load leads to higher jitter, which results in more pessimism in the delay estimation, as discussed in Section 3.

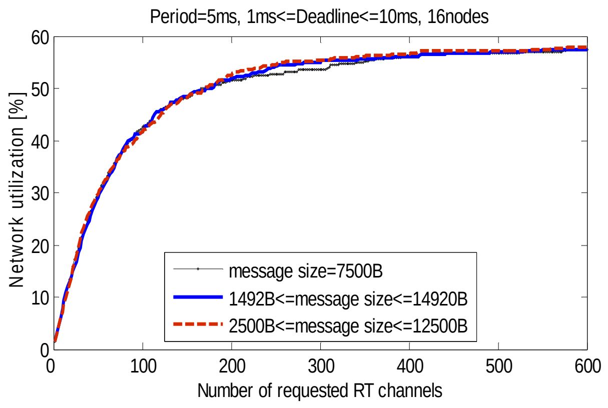


(a)

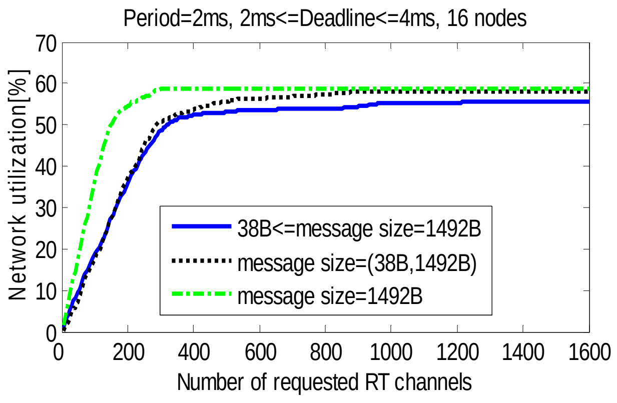


(b)

Figure 8. Estimation of introduced pessimism as a function of number of accepted real-time channels expressed as: (a) PD and SD and (b) DOR



(a)



(b)

Figure 9. Network utilization as a function of total number of requested real-time channels (a) varying message sizes longer than 1492 bytes (b) short messages with length up to 1492 bytes.

This evaluation shows us the detailed pessimism at the packet level. In the following studies, we will see how performance is affected on the overall system level.

Effect of Message Sizes on Network Utilization

Our analysis is capable of analyzing varying types of network traffic, for example applications having short messages, applications having large messages or a mix thereof. To that end, we will study how message sizes (pure data, excluding protocol headers) affect the network utilization. Since we study switched Ethernet protocols, we will assume traffic containing short messages (minimum Ethernet frame including 38 bytes of data up to full-sized Ethernet frames including 1492 bytes of data) as well as large messages (up to 10 Ethernet frames).

Figure 9(a) reports the results of simulations that were conducted with 16 nodes, periods randomly chosen from the set {5 ms, 10 ms} and deadlines randomly chosen from the interval [1 ms, 10 ms]. We observe the results obtained when the message sizes are chosen from sets A={7500 bytes} (uniform message size), B={x | 1000 bytes ≤ x ≤ 14 000 bytes} (large variance) and C={x | 5000 bytes ≤ x ≤ 10 000 bytes} (medium variance). The maximum network utilization is slightly higher when message sizes are chosen from set B than for the other cases because the existence of small message sizes in Set B increases the chance for the channels to be feasibly scheduled. Figure 9(a) also reveals that the variance of the message size of the real-time channels does not have a significant impact on the maximum achievable network utilization.

In many real-time systems, such as in the automation industry, a major fraction of the traffic consists of short messages [21]. We have thus carried out a separate simulation study for short-message traffic, the results of

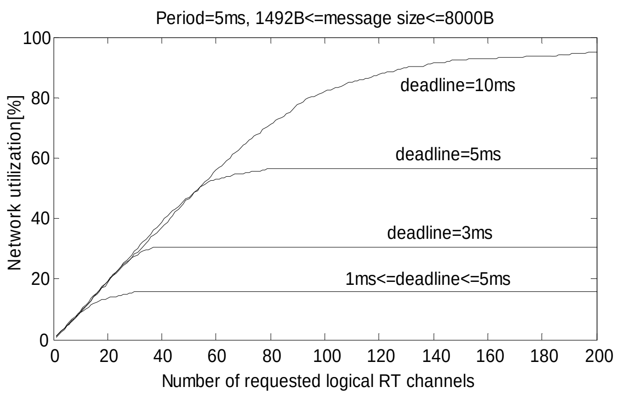


Figure 10. Network utilization vs total number of requested real-time channels (varying deadlines)

which are reported in Figure 9(b). The simulations were conducted with 32 nodes, all the channels having a period of 2 ms, and the end-to-end deadlines being randomly chosen from the interval [2 ms, 4 ms]. We observe results when the message sizes are chosen from sets A={x | 38 bytes ≤ x ≤ 1492 bytes} (variable-sized frames), B={38 bytes, 1492 bytes} (50% very short messages) and *C*={1492 bytes} (full-sized Ethernet frames). Figure 9(b) shows that the maximum achievable network utilization can be as high as 60% in the case in which there is a large amount of short messages in the system. When the number of real-time channels is low, the network utilization is higher when message sizes are chosen from set *C* than the other two cases, because the real-time channels in Set *C* have higher utilization than in the other two sets. Still, the plots show that the maximum achievable network utilization is not significantly affected by the message size.

This study shows that network resources can be used efficiently in many situations with our feasibility analysis, regardless of message size, since the message size does not affect the performance significantly. We will study how other parameters influence the performance.

Effect of Channel Deadline Tightness on Network Utilization

Intuitively, the choice of end-to-end deadlines of real-time channels should have a significant impact on the network utilization. Since our analysis is capable of analyzing traffic with arbitrary deadlines, we will study how the choice of end-to-end deadlines affects network utilization.

Figure 10 reports the results of a simulation study that was conducted with 8 nodes, all channels having a period of 5 ms and the message sizes randomly chosen from the interval [1492 bytes, 8000 bytes]. The deadlines are chosen from sets *A*={*x* | 1 ms ≤ x ≤ 5 ms} (large variance), *B*={3 ms} (short deadlines), *C*={5 ms} (deadline being equal to period) and *D*={10 ms} (long deadlines). We observe that

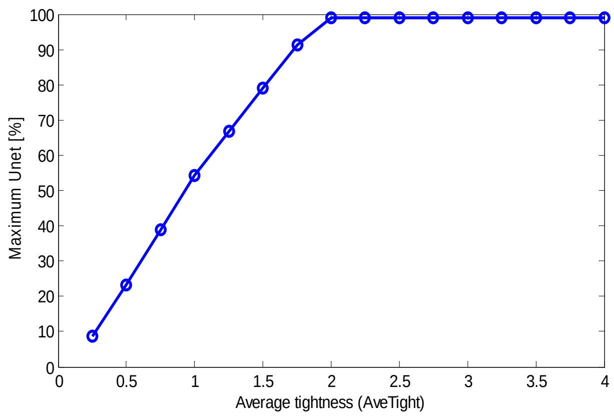


Figure 11. The maximum value of Unet achieved in the simulations as a function of AveTight

the lowest network utilization is obtained when deadlines are chosen from set *A* while the highest utilization occurs when end-to-end deadlines are chosen from set *D*. This is not surprising since these sets include the shortest and longest end-to-end deadlines, respectively.

One import aspect of the choice of deadlines is how it allows for pipelining of messages, that is, different messages belonging to the same channel can be in the network at the same time. In the case of pipelining, we could expect a higher maximum achievable network utilization. We will now study how the *deadline tightness* (defined below) of the real-time channels affects the network utilization.

Definition 8 The *deadline tightness*, $Tight_i$, for a real-time channel τ_i is the ratio of the end-to-end deadline to the period of τ_i, that is,

$$Tight_i = \frac{Tdl_i}{Tperiod_i} .$$

Definition 9 The *average deadline tightness*, *AveTight*, for a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ is

$$AveTight = \sum_{i=1}^{n} Tight_i / n .$$

Intuitively, the network will be less utilized for smaller values of *AveTight* because the high demand on meeting the deadlines makes it difficult to accept real-time channels. This behaviour is corroborated by the plot in Figure 11, which shows the maximum *Unet* (maximum value achieved during the simulations) as a function of *AveTight*. The plot shows that *Unet* increases as the value of *AveTight* increases, and then stabilizes at a value above 98% as the value of *AveTight* increases up to 2. The reason is that when the deadlines are long enough, the acceptance of real-time traffic is more constrained by the physical limitations of the

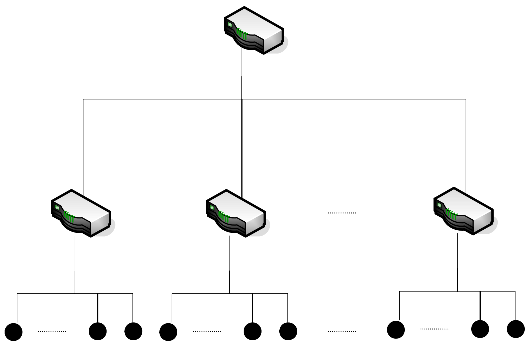


Figure 12. Tree topology.

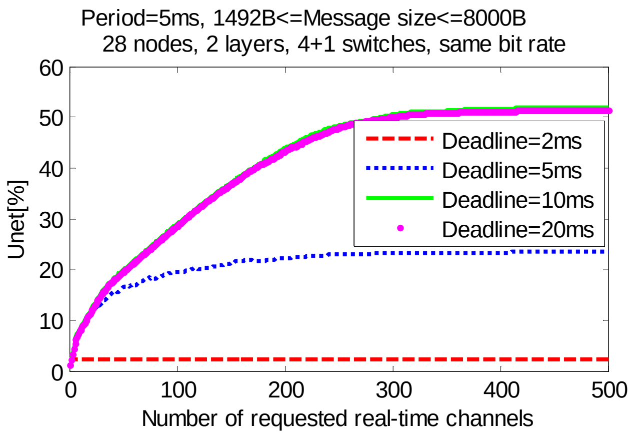
network than by the deadlines. Figure 11 also shows that our system can achieve a utilization close to the theoretical (average) upper bound, 100%.

Effect of number of hops on Network Utilization

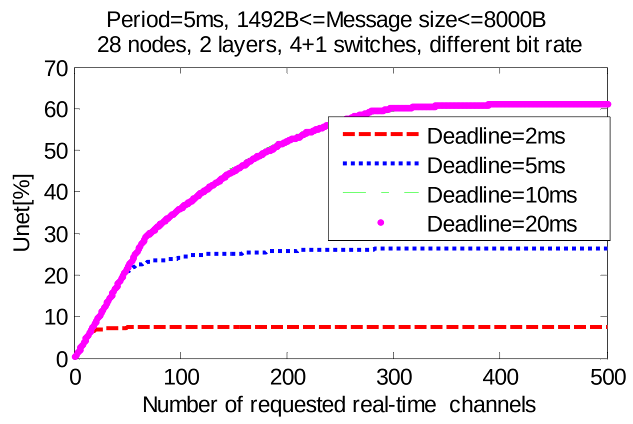
Considering the fact that our analysis is capable of analyzing networks with multiple switches, we will study how the number of hops affects the network performance. We consider a network with a tree topology where a number of end nodes are connected with slave switches and the slave switches are connected to upper-layer switches or a root switch, shown in Figure 12.

Figure 13(a) reports the results of a simulation study that was conducted for a network where 28 end nodes are connected to four switches and then these four switches are connected to one master switch. Figure 13(a) shows the results when the period is set to 5 ms for all real-time channels, the message sizes are randomly chosen from the interval [1492 bytes, 8000 bytes]. The deadlines are chosen from sets *A*={2 ms}, *B*={5 ms}, *C*={10 ms} and *D*={20 ms}. We observe that the network utilization is fairly low (2%) when deadlines are chosen from set *A* because the tight deadlines make it extremely difficult to accept real-time channels. The network utilization is increased while the end-to-end deadlines are increased. However, the network utilization is same for cases when deadline are chosen from set C and from set D. The reason is that, the links between the slave switches and the root switch experience a bottleneck effect under a high real-time traffic load.

To avoid such a bottleneck effect caused by the identical bit rate on all links, we increase the bit rate of the root switch's incoming and outgoing links to 1 Gbits/s (the bit rate of other links is still 100 Mbits/s). The simulation results for such a setup are reported in Figure 13(b). It can be observed that the maximum value of *Unet* is slightly increased because of the use of higher bit-rate links to and from the root switch. For example, the maximum value of *Unet* can reach 75% when the deadlines are chosen from set *D* under



(a)



(b)

Figure 13. Network utilization vs total number of requested real-time channels (two-layer tree network), (a) same bit rate links (b) different bit rate links.

such setup, while it can only reach 50% under the identical bit rate link setup. However, the network utilization still can not reach 100% for cases when deadline are chosen from set *C* and from set *D*, because in these cases the acceptance of real-time channels is utilization-constrained by the links between the source nodes and the slave switches.

Moreover, Figure 14 reports the results of a simulation when adding one more layer in the network. The number of nodes is still 28 nodes and they are connected to four slave switches via bit rate of 100 Mbits/s links rate. These four switches are connected to two upper-layer switches via links with 1 Gbits/s rate. The links to and from the root switch has bit rate of 10 Gbits/s. The period is set to 5 ms for all real-time channels, the message sizes are randomly chosen from the interval [1492 bytes, 8000 bytes]. The deadlines are chosen from sets *A*={2 ms}, *B*={5 ms} and *C*={10 ms}. It can be seen that the network utilization is not dramatically decreased because of the increasement of the

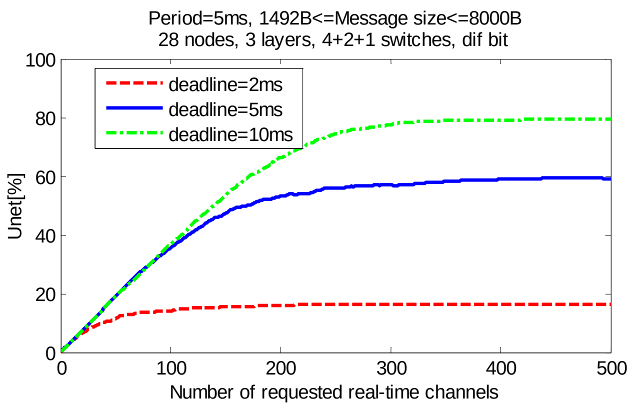


Figure 14. Network utilization vs total number of requested real-time channels (three-layer tree network and different bit rate)

number of hops. For example, the maximum achievable network utilization can be as high as 60% when the deadlines are chosen from set *C*.

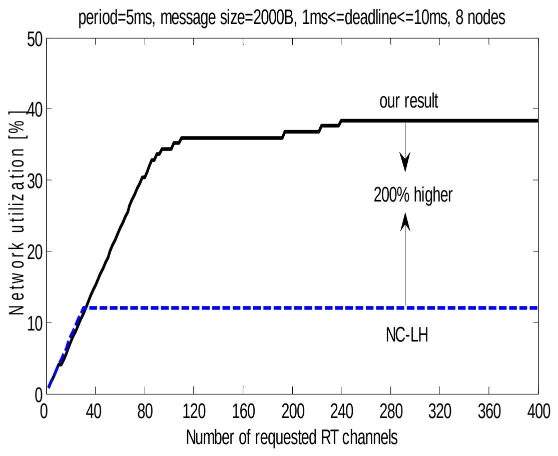
Two conclusions can be drawn from the above simulations. First, resources can still be effectively utilized for networks with multiple switches with our real-time analysis. Second, in a network with multiple switches, the network performance is not only affected by the pessimism in the real-time analysis, but also the choice of link bit rates.

Effect of Analysis Method on Network Utilization

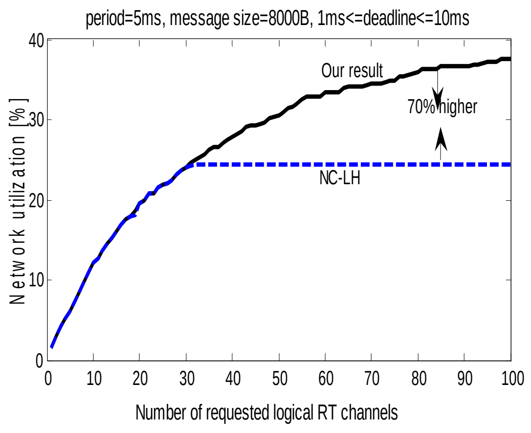
Our conceptual comparison in Section 5 revealed that the worst-case delay for the individual components in a single-switch network derived by our analysis is tight. However, for the end-to-end delay, we showed that neither our analysis nor the NC analysis produces tight estimates. For this reason, we have conducted simulations to compare the efficiency of our analysis with the NC analysis for an Ethernet network with a single switch (NC-LH).

Recall that, in order to compare two real-time analysis schemes, we choose to measure the allowed amount of HRT traffic in the network under the condition of guaranteeing the worst-case delays for all real-time channels. A higher degree of allowed real-time traffic would then indicate less overestimation in the worst-case delay analysis. Hence, in our experimental comparison study, we observe the network utilization as a function of total number of requested real-time channels.

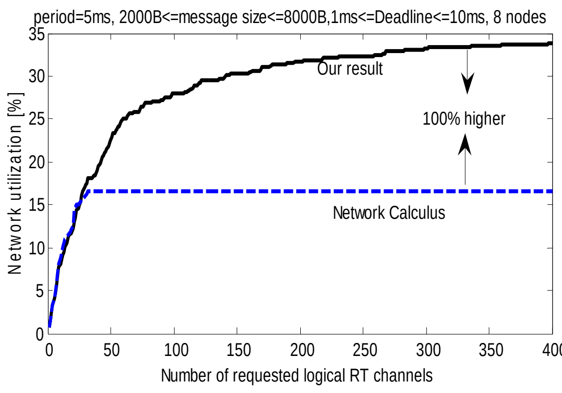
Figure 15(a) reports the results of a simulation study that was conducted for a system with 8 nodes, with all channels having a period of 5 ms, a message size of 2000 bytes (small message sizes) and end-to-end deadlines that are randomly chosen from the interval [1 ms, 10 ms]. It can be observed that, for light traffic loads, the network utilization steadily increases with the traffic load. However, at a certain load, the network utilization saturates because it becomes increasingly difficult to accept new channels. The



(a)



(b)



(c)

Figure 15. Network utilization as a function of number of requested real-time channels (comparing different analytical schemes) (a) small message sizes (b) large message sizes and (c) varying message sizes

plots indicate that NC-LH saturates with a network utilization of approximately 12%. In contrast, our analysis method results in a significantly higher utilization. For this particular simulation set-up, our analysis can achieve a network utilization of 37% (three times as much as that achieved using NC-LH).

If we increase the message sizes for all the logical real-time channels to 8000 bytes, we obtain the results plotted in Figure 15(b). For this simulation set-up, we notice an interesting phenomenon: while our analysis maintains its performance, the achievable network utilization for NC-LH increases to 25% (twice as high than in the previous study).

In each of the first two simulations, all channels have the same message sizes. For the third simulation set-up, the message sizes are randomly chosen from the interval [2000 bytes, 8000 bytes]. The results of this simulation are plotted in Figure 15(c). Here, it is shown that our feasibility analysis still maintains its performance, while NC-LH achieves a network utilization of 17%.

It can be observed from the plots in Figure 15 that our analysis consistently achieves higher utilization than the NC-LH analysis for the case of network saturation; for example, our analysis can achieve a network utilization that is three times higher as that achieved for NC-LH. This behavior can be explained by the way the aggregated outgoing traffic from the source node is modeled by the two different analytical methods. By considering the fact that the output traffic is shaped by the physical link, our analysis is able to produce less pessimistic delay estimates. In contrast, NC-LH uses a model in which the burstiness of the resulting outgoing traffic is the sum of the burstiness of individual channels. This in turn leads to overly pessimistic worst-case delay estimations in the NC-LH analysis, and the pessimism is dramatically high when the network traffic load reaches the saturation point. Hence, our analysis produces tighter estimates for the worst-case delay.

Furthermore, we observe that the performance gap between the two methods is significantly narrowed for large message sizes. However, the plots reveal that the performance of our analysis does not significantly change, but maintains a maximum achievable network utilization of 35%-40% in all studies. Instead, the performance of NC-LH is significantly affected by the message sizes. This behaviour is implied by Equation 32 and Equation 33 in the following sense: short messages allow for a higher number of real-time channels to be allocated to the outgoing links, and thus there are more evenly distributed real-time channels on all physical links. An even traffic distribution in turn leads to a small value of g_{max} (Equation 33), and a small value of g_{max} gives a high worst-case delay (Equation 32).

In this section, we evaluated the performance of our real-time analysis by conducting simulations. To that end, we measured the pessimism of our analysis and how traffic characteristics, network characteristics and analytical schemes affect the performance. The following conclusions can be drawn from our study. First, our simulations indicate that our end-to-end real-time analysis introduces pessimism in the order of 10% over-estimation of the delay. Second, for our analysis, deadline tightness has a significant impact on the network utilization, while variances of message sizes does not affects network utilization very much. Third, it is shown that our analysis offers higher network utilization than NC when the network is highly loaded, which indicates that our analysis gives a tighter end-to-end delay estimation. Fourth, our analysis is effective for networks with multiple-switches.

7. Conclusion

In this paper, we contribute with the real-time analysis for periodic hard real-time traffic in packet-switched networks only utilizing FCFS queuing. Our results can be used in many real-time applications, such as radar applications and control systems. The correctness of our analysis is given by strict proofs. For components in single-switch networks, we derive the tight worst-case delay, which has not been achieved by the other related works. Moreover, our real-time analysis is flexible and practicable, since it supports networks with multiple switches, variable-sized frames, and switches with different bit-rate ports.

Although the worst-case delay is accurately estimated in our analysis locally at a source node and at the first switch port, the global worst-case delay aggregation can lead to pessimism, even for networks with a single switch. The reason is that the tightness of the end-to-end worst-case delay estimation relies on the co-appearance of the worst-case scenario at the source node and the switch port. Unfortunately, the channel sets at a source node and at a switch port are usually not the same. Hence, the worst-case scenario at the source node and that at the switch may not necessarily co-appear. For networks with multiple switches, it is difficult to avoid overestimation due to the accumulated jitter.

Being aware of this fact, we have conducted simulations to measure the efficiency of our real-time analysis and then quantify the pessimism that has been introduced.

Moreover, we demonstrated the conceptual similarities and differences between our analyses and Network Calculus (NC). We then carried out a set of experiments to evaluate our analysis and compared the performance with methods based on NC. The conclusions from our simulation and comparison study show that our analysis gives tighter estimations than NC in the majority of the cases.

References

[1] D. Ferrari, D. C. Verma, A scheme for real-time channel establishment in wide area network, in: *IEEE Journal on. Selected Areas in Communications*, 8(3) (1990), 368-379.

[2] Q. Zhang, K. G. Shin, On the ability of establishing real-time channels in point-to-point packet-switched networks, in: *IEEE Trans. on Communications*, 42(2/3/4) (1994).

[3] H. Zhang, Service disciplines for guaranteed performance in packet-switching networks, in: *Proc. of IEEE*, 83(10) (1995), 1374-1396.

[4] D. D. Kandlur, K. G. Shin, D. Ferrari, Real-time Communication in multi-hop networks, in: *IEEE Trans. on Parallel and Distributed Systems*, 5(10) (1994), 1044-1056.

[5] K. C. Lee, S. Lee, M. H. Lee, Worst case communication delay of real-time industrial switched Ethernet with multiple levels, in: *IEEE Trans. on Industrial Electronics*, 53(5) (2006).

[6] R. L. Cruz, A calculus for network delay, I: network elements in isolation, in: *IEEE Trans. on Information Theory*, 37(1) (1991).

[7] R. L. Cruz, "A calculus for network delay, II: network analysis," in: *IEEE Trans. on Information Theory*, 37(1) (1991).

[8] J. Y. Le Boudec, P. Thiran, Network Calculus, in: *Springer Verlag Lecture Notes in Computer Science*, 2050(2001).

[9] V. Cholvi, J. Echague, J. Y. Le Boudec, On the feasible scenarios at the output of a FIFO server, in: *IEEE Communication Letters*, 9(5) (2005).

[10] L. Lenzini, E. Mingozzi, G. Stea, Delay bounds for FIFO aggregates: a case study, in: *Computer Communications*, 28(3), (2005).

[11] J. P. Georges, E. Rondeau, T. Divoux, Evaluation of switched Ethernet in an industrial context by using the network calculus, in: *Proc. of 4th International Workshop on Factory Communication Systems,* 2002.

[12] J. Loeser, H. Haertig, Low-latency hard real-time communication over switched Ethernet, in: *the 16th Euromicro International Conference on Real-time Systems (ECRTS'2004),* Catania, Italy, June 30−July 2, 2004.

[13] J. Loeser, H. Haertig, Using switched Ethernet for hard real-time communication, in: *proc. of the International Conference on Parallel Computing in Electronic Engineering (PARELEC'04),* Dresden, Germany, 2004, pp. 349-353.

[14] J. Jasperneite, P. Neumann, M. Theis, K. Watson, Deterministic real-time communication with switched Ethernet, in: *Proc. of 4th International Workshop on Factory Communication Systems*, Västerås, Sweden, Aug. 2002.

[15] Y. Song, A. Koubaa, F. Simonot, Switched Ethernet for real-time industrial communication modelling and message buffering delay evaluation, in: *Proc. of 4th International Workshop on Factory Communication Systems*, Aug. 2002.

[16] L. Lenzini, L. Martorini, E. Mingozzi and G. Stea, Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks, in: *Performance Evaluation*, 63, (2006) 956-987.

[17] X. Fan, M. Jonsson, Guaranteed real-time services over standard switched Ethernet, in: *Proc. of the 30th Annual IEEE Conference on Local Computer Networks*, Sydney, Australia, 2005.

[18] X. Fan, Real-time Services in Packet-switched Networks for embedded applications, PhD thesis, Chalmers University of Technology, *Sweden*, 2007.

[19] S. K. Baruah, L. E. Rosier, and Q. R. Howell, Algorithms and complexity concerning the preemptive scheduling of periodic real-time tasks on one processor, in: *Real-time systems*, 2(1990).

[20] M. Spuri. Analysis of deadline scheduling in real-time systems, *Technical Report,* No. 2772, INRIA, France, 1996.

[21] R. Weber, U. Legedza and J. G. L. Amorim, A survey of messaging software issues and systems for Myrinet-based cluster, *Parallel and Distributed Computing Practices, special issxe on High-Performance Computing on Clusters,* 2(2), 1996.