Technical Report **IDE0701**, January 2007

# Guaranteed Real-Time Communication in Packet-Switched Networks with FCFS queuing - Analysis and Simulations

Xing Fan

School of Information Science, Computer and Electrical Engineering,
Halmstad University, Box 823, S-30118 Halmstad, Sweden

*IDE report No. 0701*

# Guaranteed Real-Time Communication in Packet-Switched Networks with FCFS queuing — Analysis and simulations

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

*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}@ide.hh.se, http://www.hh.se/ide*

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

# Contents

# Chapter 1 Introduction

In this report, we present a real-time analytical framework and the performance evaluation on our analysis. 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. We choose switched Ethernet as an example to present the idea of our analysis and our experimental evaluations in this report.

The remainder of the report is organized as follows. In Chapter 2, we define the network models, important concepts and terminology for real-time analysis. Chapter 3 presents our real-time analysis for isolated network elements. Chapter 4 gives end-to-end real-time analysis. Chapter 5 presents the performance evaluation of our results by simulation and comparison study and summarizes this report.

# Chapter 2 Terminology, Assumptions and Notations

Real-time communication over switched Ethernet network can be quite complex with different network architectures, different types of traffics, different time requirements and metrics. This chapter introduces network architecture, basic terminology and models, notations, assumptions and relaxations necessary to fully understand the remaining chapters in this report.

## 2.1 Network Architecture and Notations

The network architecture is now described in this section. We consider a network with a number of end nodes and multiple switches, which enables the structuring of the different network topologies and different configurations, thereby supporting different types of applications.

### Network elements

Our communication network is represented as the interconnection of fundamental building blocks, called network elements, as shown in Figure 2.1. We define the following four types of network elements.

A *physical link* is a unidirectional transmission link which accepts network traffic from one network element and transmits network traffic to another network element at a constant bit rate. According to the switched Ethernet standard, a physical link can carry traffic in both directions simultaneously. However, for easy understanding of the subsequent real-time analysis, one duplex physical link is decomposed into a pair of unidirectional links. In our topology figures, we put an arrow on each edge to represent a unidirectional link, and the corresponding unidirectional link in the opposite direction is not shown in the figures.

A *switch* is a network device which is able to receive network traffic from several input ports and is able to decompose input traffic to several output ports. The number of switches in the network is denoted as *Nswi*, the number of input/output ports in switch $j$ is denoted as $Nport_j$ 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).

An *end node* is a network device which is able to transmit network traffic to a single input port of a switch and is able to receive network traffic from a single output port of a switch. The number of end nodes in the network is denoted as *Nnode* and the bit rate of the physical link originating from end node $k$ is $Rnode_k$ (bits/s).

An *output queue* is a buffer for an outgoing physical link which stores the traffic being ready to be delivered to the outgoing physical link.

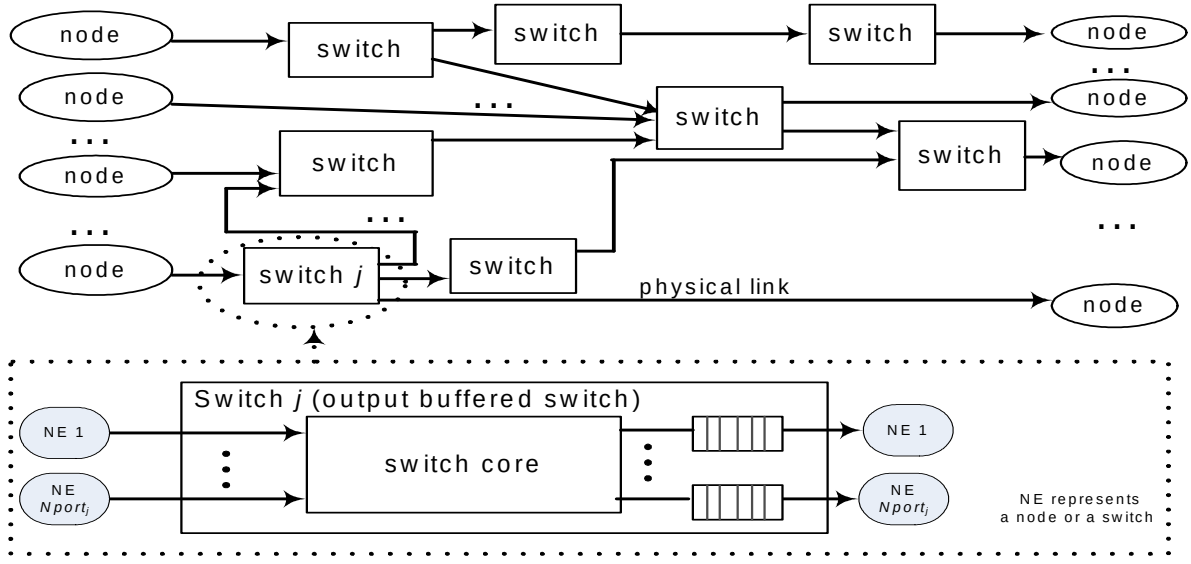



**Figure 2.1. Interconnection of network elements.**

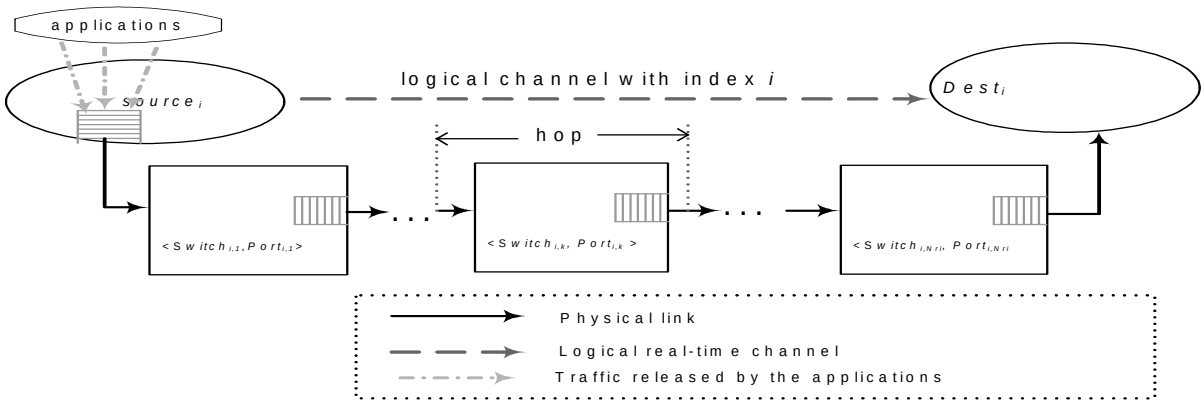



**Figure 2.2. The relation between physical connection and logical channel.**

## Network routing

We assume the end nodes and the switches are connected via point-to-point links. The network operates in a packet-switched mode, which means a transmitted data unit being an Ethernet frame. Frames from any given user traverse a predetermined fixed route through the network in order to reach their destination.

We refer to the frame flow transmitted from a source node to a destination node as a logical channel (the strict definition will be given in Section 2.2). The network maintains multiple simultaneous logical channels and *Nch* is the total number of logical channels in the network. For the logical channel with index *i* (denoted by $\tau_i$), $Source_i$ is used to indicate the source node and $Dest_i$ is used to indicate the destination node.

As illustrated in Figure 2.2, under a fixed routing strategy, once a logical channel $\tau_i$ from $Source_i$ to $Dest_i$ 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:

$$Route_i = \left(\left\langle Switch_{i,k}, Port_{i,k} \right\rangle\right) \quad k=1,...,Nr_i, \tag{2.1}$$

where $Nr_i$ indicates the total number of switches on the route, $Switch_{i,k}$ indicates *k*th switch on the route and $Port_{i,k}$ indicates which 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. The reason will be explained in Chapter 4.

In this report, we will use the term *hop* to indicate the intermediate transmission for a logical real-time channel. For example, the transmission from the $Source_i$ to $\left\langle Switch_{i,1}, Port_{i,1} \right\rangle$ is called the first hop, while the transmission from $\left\langle Switch_{i,k-1}, Port_{i,k-1} \right\rangle$ to $\left\langle Switch_{i,k}, Port_{i,k} \right\rangle$ is called the *k*th hop. More specifically, for each hop that is traversed, a frame is transferred from the queue of an incoming link, through the controller at a source node or at a switch and to the queue of an outgoing link.

## Traffic handling

In this section, we describe the traffic handling in switched Ethernet networks.

Figure 2.3(a) illustrates the traffic handling at a source node. Once a real-time message is released by an application, it is immediately put in the output queue in the form of a sequence of Ethernet frames. Several applications may release real-time traffic simultaneously, which leads to a burstiness in the output queue. It should be noted that the frames belonging to one message might be interrupted by frames belonging to

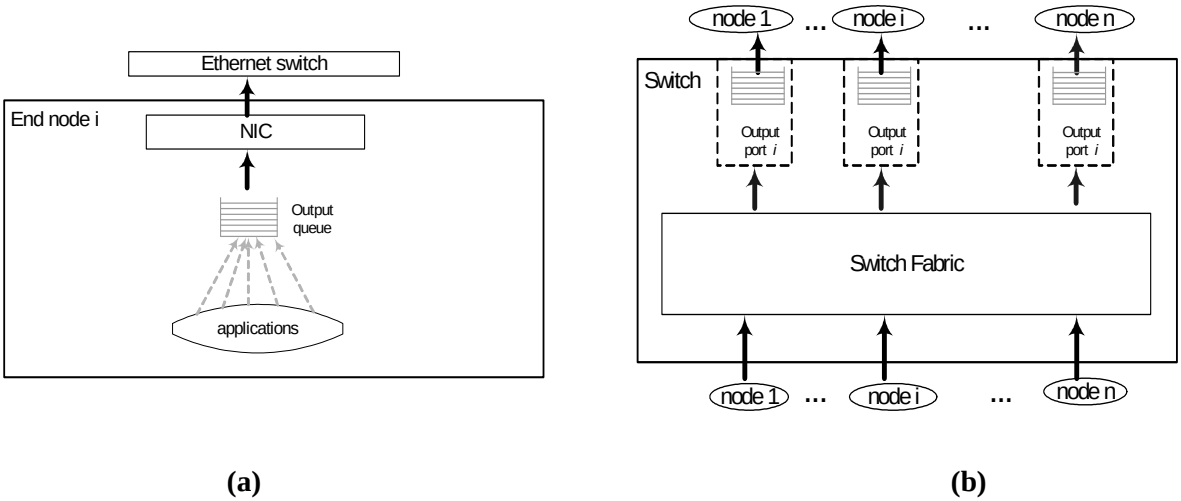



**Figure 2.3. Output queues and traffic handling. (a) in an end node. (b) in a switch.**

other real-time channels, therefore they are not always stored continuously in the output queue. Before entering the Network Interface Card (NIC), the frames are stored in the output queue, which is a FCFS queuing according to standard Ethernet configuration. If the outgoing link is available, the frames stored in the NIC will be transmitted.

The switch is of the store-and-forward type, which can be decomposed into three main components: the queuing model, the control logic and the switch fabric. The queuing model refers to the buffering and the congestion mechanisms located in the switch, the control logic refers to the decision making process within the switch and the switch fabric is the path that data takes to move from one port to another.

As shown in Figure 2.3(b), when a frame arrives at the switch, the control logic determines the transmit port and tries to transmit the frame immediately to the output port. If the port is busy because another frame is already under transmission, the frame is stored in the transmit port's queue, which is a FCFS queue according to the Ethernet standard.

Although our goal is to support hard real-time traffic, we still allow other traffic classes in the network, e.g., best effort traffic and non real-time traffic. To prioritize different traffic classes and minimize the interference with other traffic when transmitting periodic time-critical messages, the traffic differentiation mechanism introduced by the IEEE 802.1D/Q standards is used in our proposal.

The IEEE 802.1D queuing feature [IEEE 1998] [IEEE 2003] enables Layer 2 to set priorities to traffic. The content of an IEEE 802.1D tagged Ethernet frame is shown in Figure 2.4(a). The IEEE 802.1D prioritization works at the Medium Access Control (MAC) framing layer. If the value of the Tag Protocol Identifier (TPID) field in an Ethernet frame is equal to 8100, this frame carries the tag IEEE 802.1D/Q. The Tag Control Information (TCI) field is the 802.1D header, including a three-bit field for setting priorities, allowing packets to be grouped into various traffic classes, a one-bit field for the Canonical Format Indicator (CFI), and a 12-bit Virtual Local Area Network (VLAN) ID. The latter two fields are not used in our work. By adding the 802.1D header to the frames, traffic is simply classified and put into queues with different priorities. There could be up to eight priority queues according to the standard, while we assume a minimum of three.

In the example shown in Figure 2.4(b), there are three priority queues for each output port in a switch. The hard real-time (HRT) frames are put into the hard real-time traffic queue, which has the highest priority among all traffic classes, while soft real-time (SRT) frames are put into the soft real-time queue with a lower priority than the hard real-time queue. Outgoing non-real-time (NRT) traffic from the end node is treated as lowest priority. In the same way, there are three priority queues for three traffic classes in each source node.

Since our real-time analysis in this report aims for providing guarantees for hard real-time traffic, we only illustrate HRT queues in most of the topology figures.

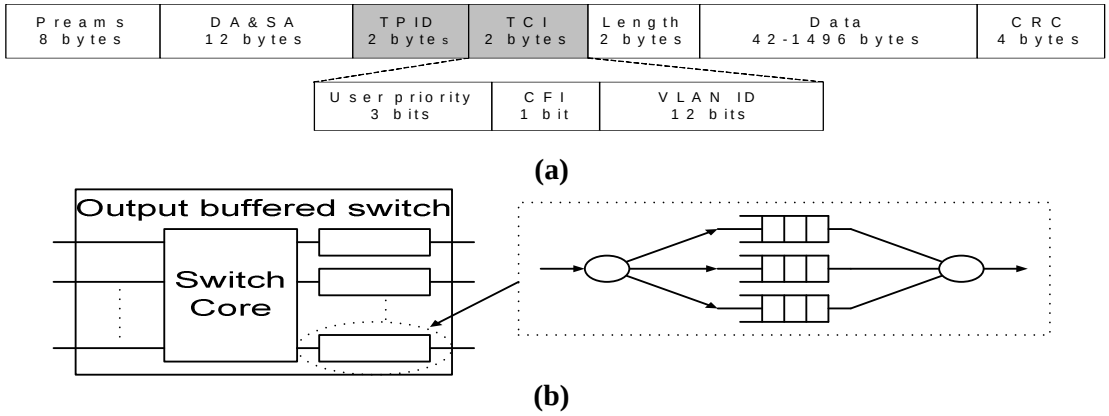



**Figure 2.4 Traffic differentiation. (a) IEEE 802.1D/Q extended Ethernet frame. (b) Priority queuing.**

The network related parameters are summarized in Table 2.1.

## 2.2 Terminology, Models and Notations

The basic terminology and model for real-time analysis is presented in this section. The following definitions are grouped into three classes: channel level, link level and schedulability level. Other terminology will be introduced in later chapters when it applies to a particular algorithm or system configuration.

### Channel Level

The logical channel concept, used for traffic modeling, has been introduced in Section 2.1. The strict definitions of the traffic model and properties are given here.

**Definition 2.1** A *logical real-time channel* (with index *i*), $\tau_i$, is a virtual unidirectional connection from the source node, $Source_i$, to the destination node, $Dest_i$. The channel is characterized by the maximum pure data traffic volume ($Cap_i$) given by the application, the maximum practical traffic volume given by network implementation ($C_i$) including data and Ethernet header, and a set of time properties. Both $Cap_i$ and $C_i$ are expressed in bits.

The derivation from $Cap_i$ to $C_i$ is given by Equation 2.2:

| Name | Definition |
|---|---|
| *Nnode* | the number of end nodes in the network. |
| *Nswi* | the number of switches in the network. |
| *Nch* | the number of logical channels in the network. |
| $Nport_j$ | the number of ports in switch *j*. |
| $Rnode_k$ | the rate of the physical link from source node *k* (bits/s). |
| $Rswi_{s,p}$ | the rate of the physical link at output port *p* in switch *s* (bits/s). |
| $Source_i$ | the source node of logical real-time channel *i*. |
| $Dest_i$ | the destination node of logical real-time channel *i*. |
| $Nr_i$ | the number of switch ports for the packets belonging to channel *i* |
| $Route_i$ | the route from $Source_i$ to $Dest_i$, $Route_i = \left(\left\langle Switch_{i,k}, Port_{i,k} \right\rangle\right)$ $k$=1,..., $Nr_i$. |
| $Switch_{i,k}$ | *k*th switch that the messages belonging to real-time channel *i* go through. |
| $Port_{i,k}$ | Output port in $Switch_{i,k}$ is used for the logical real-time channel *i*. |

**Table 2.1. Notations and definitions for the network configuration.**

| Name | Definition | Value (without IP or UDP) | Value (incl. IP and UDP) |
|---|---|---|---|
| $T_{ef}$ | the length of a full-sized Ethernet frame including inter-frame gap | 1538 bits | 1538 bits |
| $T_h$ | the length of the header in an Ethernet frame | 46 bits | 74 bits |
| $T_{mind}$ | the minimum length of the data field in the Ethernet frame without pad field | 38 bits | 10 bits |
| $T_{maxd}$ | the length of the data field in a full-sized Ethernet frame | 1492 bits | 1464 bits |

**Table 2.2. Notations for traffic volume calculation.**

$$C_i = \begin{cases} \left\lfloor \frac{Cap_i}{T_{\text{maxd}}} \right\rfloor T_{ef} + ((Cap_i \bmod T_{\text{maxd}}) + T_h), & \text{if } (Cap_i \bmod T_{\text{maxd}}) \geq T_{\text{mind}}; \\ \left\lfloor \frac{Cap_i}{T_{\text{maxd}}} \right\rfloor T_{ef} + 72, & \text{if } 0 < (Cap_i \bmod T_{\text{maxd}}) < T_{\text{mind}}; \\ \left\lfloor \frac{Cap_i}{T_{\text{maxd}}} \right\rfloor T_{ef}, & \text{if } (Cap_i \bmod T_{\text{maxd}}) = 0; \end{cases} \quad (2.2)$$

where $T_{ef}$ is the length of a full-sized Ethernet frame including the inter-frame gap, $T_h$ is the length of the Ethernet frame header including the inter-frame gap, $T_{mind}$ is the minimum length of the data field in an Ethernet frame without pad field and $T_{maxd}$ is the length of the data field in a full-sized Ethernet frame. These notations for traffic volume calculation are all expressed in bits and explained in Table 2.2.

**Definition 2.2** A *message* is a collection of data being communicated over a real-time channel. The maximum size of the message equals to $C_i$.

As explained in Section 2.1, the data entered into the network by the applications is divided into frames; therefore, a message can be viewed as a sequence of frames. In other words, the minimum unit for data transmission over switched Ethernet is an Ethernet frame, while the basic unit used in the real-time analysis is a message.

**Definition 2.3** The *message release time*, $r_i$, for real-time channel $\tau_i$, is the time instant that $\tau_i$ releases its first messages.

**Definition 2.4** A *periodic logical real-time channel* $\tau_i$ is one which releases messages regularly with a constant interval called the period, $T_{period,i}$ (in s). A periodic logical real-time channel will be called *real-time channel* in the rest of the report.

**Definition 2.5** The *end-to-end relative deadline*, $T_{dl,i}$ (in s), for real-time channel $\tau_i$, is the maximum allowed time interval between the release of a message of $\tau_i$ at the source node and the arrival of the message at the destination node for that channel.

**Definition 2.6** The *end-to-end absolute deadline*, $T_{absdl,i,j}$, for the $j$th message of $\tau_i$, is the time instant by which the message must arrive at its destination node. In other words, $T_{absdl,i,j} = r_i + T_{period,i} \cdot (j - 1) + T_{dl,i}$.

**Definition 2.7** The *relative deadline*, $T_{srcdl,i}$, (in s) for real-time channel $\tau_i$ at the source node, is the maximum allowed time interval between the release of a message at the source node and the arrival of the message at the next hop.

**Definition 2.8** The *absolute deadline*, $T_{abssrcdl,i,j}$, (in s) for the $j$th message of $\tau_i$ at the source node, is a time instant by which the message must arrive at the next hop. In other words, $T_{abssrcdl,i,j} = r_i + T_{period,i} \cdot (j - 1) + T_{srcdl,i}$.

**Definition 2.9** A *synchronous pattern* is a scenario in which a set of real-time channels release their first messages at the same time (usually considered time zero).

The notations related to real-time channels are listed in Table 2.3.

| Name | Definition |
|---|---|
| $\tau_i$ | Logical real-time channel with index *i*. |
| $Source_i$ | The source node of $\tau_i$. |
| $Dest_i$ | The destination node of $\tau_i$. |
| $T_{period,i}$ | The period of data generation belonging to $\tau_i$ (s). |
| $Cap_i$ | The amount of pure data generated by the application per period belonging to $\tau_i$ (bits). |
| $C_i$ | The amount of traffic, including data and header, per period for $\tau_i$ (bits). |
| $r_i$ | The time instant when $\tau_i$ releases its first message. |
| $T_{dl,i}$ | The end-to-end relative deadline of $\tau_i$ specified by the application (s). |
| $T_{absdl,i,j}$, | The end-to-end absolute deadline for the $j$th message of $\tau_i$. |
| $T_{srcdl,i}$ | The relative deadline of $\tau_i$ at the source node. |
| $T_{abssrcdl,i,j}$ | The absolute deadline for the $j$th message of $\tau_i$ at the source node. |

**Table 2.3. Notations and definitions for the real-time channels.**

## Link level

In this report, we target for hard real-time communication, which means to guarantee that the messages are delivered within their deadlines. In a packet-switched network, a message that starts from a source node passes through a series of switch/port pairs on the way and ends its journey in a destination node. As the message travels from one node to another, it experiences different types of delays along the path, which are therefore introduced as follows.

**Definition 2.10** The *worst-case delay*, $T_{sdelay,i}$ (in s) at the source node of real-time channel $\tau_i$, is the longest time that passes from the message release time at the source node to the last bit of the message leaving the outgoing physical link.

**Definition 2.11** The *worst-case delay*, $T_{i,k}$ (in s), at the switch's output port $\langle Switch_{i,k}, Port_{i,k} \rangle$ of real-time channel $\tau_i$, is the longest time that passes from the time instance when the message being put in the output port queue to the last bit of the message leaving the outgoing physical link from that output port.

**Definition 2.12** The *worst-case delay*, $Dnode_k$ (in s) at source node *k*, is the worst-case delay for any channel originating from node *k*.

**Definition 2.13** The *worst-case delay*, $Dport_{s,p}$ (in s) at output port *p* of switch *s*, is the worst-case delay for any channel traversing the outgoing link from that port.

**Definition 2.14** The *buffer size*, $BN_k$ (in bits) at source node *k*, is the maximum buffer population for hard real-time traffic at source node *k*.

**Definition 2.15** The *buffer size*, $BS_{s,p}$ (in bits) at output port *p* of switch *s*, is the maximum buffer population for hard real-time traffic at that port.

Definitions 2.12-2.15 will be explained further in Chapter 4.

**Definition 2.16** The *end-to-end worst-case delay*, $T_{e2edelay,i}$ (in s) of real-time channel $\tau_i$, is the longest possible time between the time instance when the message being released by the source node and the last bit of the message arriving at the destination node.

**Definition 2.17** A *tight worst-case delay* is the accurately predicted worst-case delay without any overestimation.

The end-to-end delay includes queuing delays, transmission delays and propagation delays on the physical link. The transmission delay is the amount of time required to transmit all the bits of a message onto the link. Once a bit is pushed onto the link, it needs to propagate to the next hop. The propagation delay, $T_{prop}$ (in s) is the amount of time required to propagate over a physical link, which can be easily calculated and added as a constant in the delay analysis.

To determine whether real-time channels satisfy their timing requirements, it is necessary to find out whether the time constraints are met even in the worst-case situation. The following definitions are formed for the worst-case delay analysis.

**Definition 2.18** The *critical instant* for real-time channel $\tau_i$, is defined as the message release pattern of all the real-time channels that leads to the worst-case delay of $\tau_i$.

**Definition 2.19** The *source link utilization*, $U_k$, for a set of real-time channels $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$ originating from source node *k* is the average fraction of time that the outgoing link is busy, that is, $U_k = \sum_{i=1}^{n} \frac{C_i}{Rnode_k T_{period,i}}$.

**Definition 2.20** The *switch link utilization*, $U_{s,p}$, for a set of real-time channels $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$ traversing the switch/port <*s*, *p*> is the average fraction of time that the outgoing link is busy, that is, $U_{s,p} = \sum_{i=1}^{n} \frac{C_i}{Rswi_{s,p} T_{period,i}}$.

**Definition 2.21** 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 $W_k(t_1, t_2) = \sum_{i=1}^{n} \max\left(\left(\left\lfloor \frac{t_2 - r_i}{T_{period,i}} \right\rfloor + 1\right) C_i, 0\right)$.

**Definition 2.22** The *cumulative traffic volume*, $Traffic_k(t_1, t_2)$ (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 delivered from node *k* to the second hop during time interval $[t_1, t_2)$.

**Definition 2.23** *Busy-period* is an interval of continuous link utilization time.

**Definition 2.24** *Idle-period* is an interval of link idle time. Note that a link idle period can be of zero length, if the last pending message completes at the same time a new message is released.

**Definition 2.25** The *synchronous busy-period* is the first busy period in the schedule of a synchronous periodic channel set.

**Definition 2.26** The *length of the synchronous busy-period*, $BP(\Gamma)$ (in s), 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 $\begin{cases} BP^0(\Gamma) = W(0,0) = \sum_{i=1}^{i} C_i; \\ BP^i(\Gamma) = W(0, \frac{BP^{i-1}(\Gamma)}{R}),\ i \ge 1; \end{cases}$

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

| *Name* | Definition |
|---|---|
| $T_{sdelay,i}$ | The worst-case delay at the source node of logical real-time channel i (s). |
| $Dnode_k$ | The worst-case delay for any channel originating from source node *k* ( s) |
| $T_{i,k}$ | The worst-case delay of logical real-time channel i at the output port in its *k*th switch $\langle Switch_{i,k}, Port_{i,k} \rangle$ (s). |
| $Dport_{s,p}$ | The worst-case delay at output port *p* of switch *s* $\langle s, p \rangle$ (s). |
| $BN_k$ | The maximum buffer population for hard real-time traffic at source node *k* (bits) |
| $BS_{s,p}$ | The maximum buffer population for hard real-time traffic at output port *p* of switch *s* (bits) |
| $T_{e2edelay,i}$ | The end-to-end worst-case delay of real-time channel $\tau_i$. |
| $T_{prop}$ | The maximum propagation delay over a link between two network elements (end-node or the switch) (s). |
| $HP(\Gamma)$ | $HP = lcm\{T_{period,i} \mid i = 1,...,n\}$(s). |
| $U_k$ | Utilization of the link from source node. |
| $U_{s,p}$ | Utilization of the link from switch/port <*s*, *p*>. |
| $W_k(t_1,t_2)$ | The sum of the traffic volumes of messages released by the real-time channels originating from source node k during time interval $[t_1,t_2)$. |
| $Traffic_k(t_1,t_2)$ | The sum of the traffic volumes of delivered messages from source node *k* to the second hop during time interval $[t_1,t_2)$. |
| $BP(\Gamma)$ | The length of the synchronous busy period (s). |

**Table 2.4. Notations and definitions for delay and buffer bound analysis**

The notations for delay and buffer bound analysis are summarized in Table 2.4.

### Schedulability level

The time constraints are guaranteed by addition of real-time channels. To establish real-time channels, schedulability analysis is essential. The relevant concepts are defined as follows.

**Definition 2.26** A real-time channel $\tau_i$ is said to be *schedulable* if its end-to-end worst-case delay does not exceed its deadline, that is, $T_{e2edelay,i} \le T_{dl,i}$.

**Definition 2.27** A *feasible link* is one for which the set of real-time channels allocated to it are schedulable.

**Definition 2.28** A *feasible network system* is one for which every link is feasible.

## 2.3 Assumptions and relaxations

A key issue in real-time analysis involves the underlying assumptions made. In the subsequent analysis, the following assumptions are made:

- A1: The real-time channels are independent in that there is neither shared resources other than the transmission link between them, nor relative dependencies or constraints on release times or completion times.
- A2: There are no overhead costs for performing switching functions, e.g., destination address look-up, fabric set-up time, and queuing operations (sorting, inserting and removing). Such delays are assumed as zero in our analysis.
- A3. We assume deadlock free routing, meaning that the network ensures not only that the route for each logical real-time channel is individually loop free, but also that the routes for all logical real-time channels do not interact in a way that would create deadlocks. Classical deadlock-free routing algorithms impose an artificial order for visiting the network nodes, for example, forming a spanning tree.

One goal of this report is to achieve realistic real-time analysis, with as adequate models of the network and its traffic as possible. For this reason, many assumptions used in other related work will be relaxed in this report. The relaxations made in this report are:

- NA1: The deadline for a real-time channel is not related to its period. This means that deadlines may be shorter than periods or that pipelining of messages may occur. Note that many real-time analysis results in the literature are only developed for the case that the deadline is equal to the period. In fact, in actual systems it may be useful to specify deadlines shorter than periods, in order to improve the responsiveness of a given real-time channel, or to enforce a minimum time gap between two consecutive messages of the same real-time channel.
- NA2: Real-time channels do not necessarily release their first messages according to the synchronous pattern (the scenario in which messages are released at the same time). In fact, any release pattern can be assumed.
- NA3: One message is a sequence of Ethernet frames possibly varying in size, which is a strong motivation to release fix-sized frames assumption. However, previous research results are limited by assuming fix-sized frames.
- NA4: Our analysis supports both switches with homogeneous bit-rate ports and switches with different bit-rate ports. Using links with different bit rates is a promising alternative to reduce bottlenecks, e.g., between the switch and the master node in a master-slave automation system. However, most related work only investigates the homogeneous case.

## 2.4 Summary of Notations

For ease of reference later in this report, the notations defined in this chapter are summarized in Table 2.5.

*Network parameters*

| Name | Definition |
|---|---|
| *Nnode* | the number of end nodes in the network. |
| *Nswi* | the number of switches in the network. |
| $Nport_j$ | the number of ports in switch j. |
| $R$ | the rate of the physical link (bits/s). |
| $Rnode_k$ | the rate of the physical link from source node *k* (bits/s). |
| $Rswi_{s,p}$ | the rate of the physical link at output port *p* in switch *s* (bits/s). |

*Logical real-time channel parameters*

| Name | Definition |
|---|---|
| *Nchs* | The number of logical real-time channels in the network. |
| $\tau_i$ | Logical real-time channel with index *i.* |
| $Source_i$ | the source node of $\tau_i$ . |
| $Dest_i$ | the destination node of $\tau_i$ . |
| $T_{period,i}$ | the period of data generation belonging to $\tau_i$ (s). |
| $Cap_i$ | the amount of pure data generated by the application per period belonging to $\tau_i$ (bits). |
| $C_i$ | The amount of traffic, including data and header, per period for $\tau_i$ (bits). |
| $T_{dl,i}$ | the end-to-end relative deadline of $\tau_i$ (s) |
| $T_{absdl,i,j}$, | The end-to-end absolute deadline for the *j*th message of $\tau_i$ |
| $T_{srcdl,i}$ | The relative deadline of $\tau_i$ at the source node |
| $T_{abssrcdl,i,j}$ | The absolute deadline for the *j*th message of $\tau_i$ at the source node |
| $Route_i$ | the route from $Source_i$ to $Dest_i$ , $Route_i = \left(\left\langle Switch_{i,k}, Port_{i,k} \right\rangle\right)$ $k$=1,..., $Nr_i$ |
| $Switch_{i,k}$ | *k*th switch that the messages belonging to real-time channel *i* go through |
| $Port_{i,k}$ | Output port in $Switch_{i,k}$ is used for the logical real-time channel *i*. |

*Delay parameters*

| Name | Definition |
|---|---|
| $T_{sdelay,i}$ | the worst-case delay at the source node of logical real-time channel *i* (s). |
| $Dnode_k$ | The worst-case delay for any channel originating from source node k ( s) |
| $T_{i,k}$ | The worst-case delay of logical real-time channel i at the output port in its *k*th switch $\left\langle Switch_{i,k}, Port_{i,k} \right\rangle$ (s). |
| $T_{prop}$ | The maximum propagation delay over a link between two network elements (end-node or the switch) (s). |
| $Dport_{s,p}$ | The worst-case delay at output port p of switch s $\left\langle s, p \right\rangle$ (s). |
| $BN_k$ | The maximum buffer population for hard real-time traffic at source node k (bits) |
| $BS_{s,p}$ | The maximum buffer population for hard real-time traffic at output port p of switch s (bits) |
| $T_{e2edelay,i}$ | The end-to-end worst-case delay of real-time channel $\tau_i$ . |
| $HP(\Gamma)$ | $HP(\tau) = lcm\left\{T_{period,i} \mid i = 1,...,n\right\}$(s). |
| $U_k$ | Utilization of the link from source node. |
| $U_{s,p}$ | Utilization of the link from switch/port <s, p>. |
| $W_k(t_1,t_2)$ | The sum of the traffic volume of messages released by the real-time channels originating from source node k during time interval $[t_1,t_2)$. |
| $Traffic_k(t_1,t$ | The sum of the traffic volume of delivered messages from source node *k* to the second hop during time interval $[t_1,t_2)$. |
| $BP(\Gamma)$ | The length of the synchronous busy period (s). |

*Notations and values for traffic volume calculation*

| Name | Definition | Value (without IP or UDP) | Value (incl. IP and UDP) |
|---|---|---|---|
| $T_{ef}$ | the length of a full-sized Ethernet frame including inter-frame gap | 1538 bits | 1538 bits |
| $T_h$ | the length of the header in an Ethernet frame | 46 bits | 74 bits |
| $T_{mind}$ | the minimum length of the data field in the Ethernet frame without pad field | 38 bits | 10 bits |
| $T_{maxd}$ | the length of the data field in a full-sized Ethernet frame | 1492 bits | 1464 bits |

**Table 2.5. Notations for real-time analysis**

# Chapter 3 Real-time Analysis for Isolated Network Elements

Estimation of worst-case delay at every hop is of critical importance to estimate the end-to-end worst-case delay. This chapter deals with all the common real-time analysis for the isolated fundamental building blocks, called network elements. The results obtained in this chapter will be useful for the subsequent real-time analysis for the whole communication network, as given in the following chapters.

## 3.1 Introduction

As described in Chapter 2, we represent a communication network as the interconnection of different network elements. The network operates in packet-switched mode and messages released by a real-time channel traverse a predefined sequence of hops. Performing real-time analysis on multi-hop packet-switched networks is complicated because of the existence of burstiness and jitter.

*Burstiness*, which is the variance in traffic rate, is caused by the difference between the bit rate of the physical link and the injection rate from a traffic source (e.g., an application or the previous hop). The physical links transmit frames one by one at a constant rate, while the applications release frames in bursts. This mismatch results in burstiness and queuing.

*Jitter*, a natural result of queuing, is the variation of a time metric with respect to some reference metric. For example, the variation in periodicity is called release jitter and the delay variation is called delay jitter.

We will first use several examples to illustrate our observations. For easy understanding, in the following examples, we only give values to the parameters without specifying the units for them. All the parameters can be viewed as being expressed in the number of full-sized Ethernet frames including the inter-frame gap.

Figure 3.1 illustrates the traffic characteristics from the applications to a source node. Consider two applications at a source node, each requesting one real-time channel. Let $\tau_1$ with $\langle T_{period,1} = 10, C_1 = 3 \rangle$, and $\tau_2$ with $\langle T_{period,2} = 5, C_2 = 2 \rangle$. The following observations can be seen from this example.

- Burstiness. Once a message is released by the application, it is split into a number of frames and put in the output queue immediately, which leads to a burstiness in the output queue at the message release time. In this example, both $\tau_1$ and $\tau_2$ release their first messages at time instant 0, so five Ethernet frames are injected into the output queue at time 0, as shown in Figure 3.1. Similarly, another burstiness occurs at time instant 5, when $\tau_2$ release its second message.

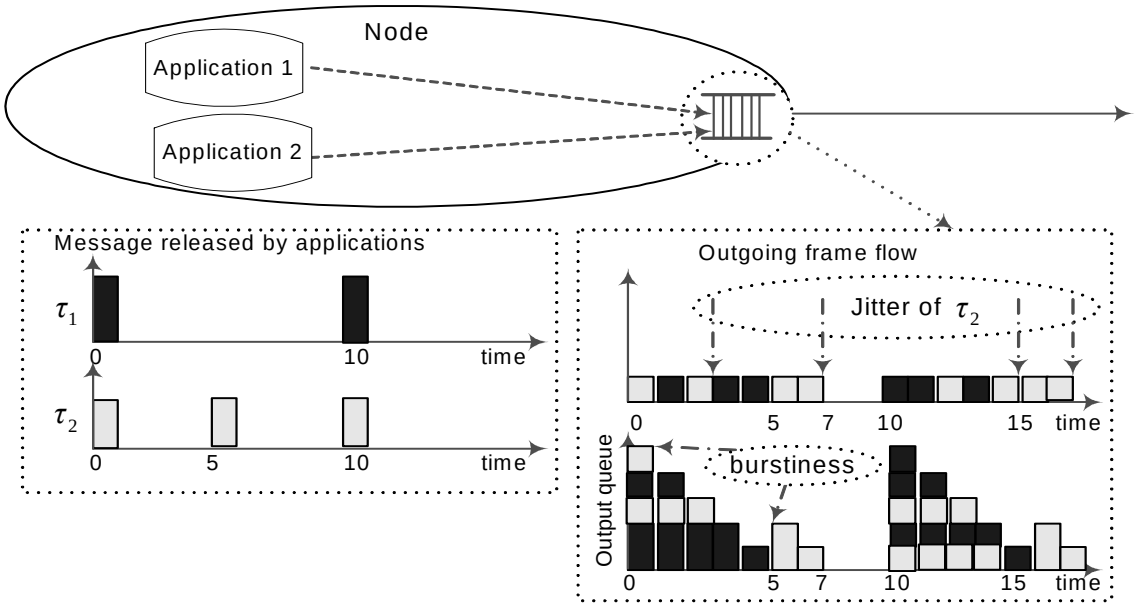



**Figure 3.1. Traffic injection from applications to the source node output queue.**

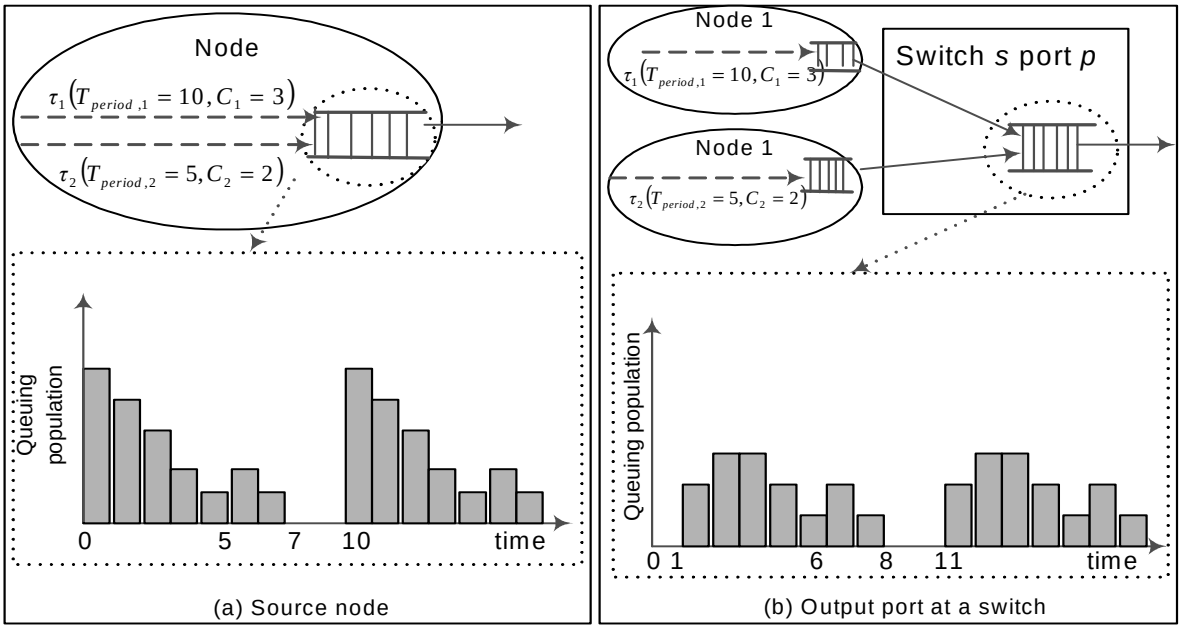



**Figure 3.2. Transmission characteristics comparison between a source node and a switch**

- Jitter. According to the standard Ethernet configuration, frames are sorted in an FCFS-queue. In some situation, there might be multiple simultaneous arrivals of multiple frames from different applications. This could mean that the frames belonging to one message might not be stored continuously in the queue due to the interference with another channel. Consequently, although the messages belonging to the same real-time channel are released with a fixed interval of time by the application, the messages might not leave the source node with a constant inter-arrival time. For instance, $\tau_2$ releases its message at the application level with a fixed interval, of 5. However, due to the interference of $\tau_1$, the first message of $\tau_2$ leaves the source node at time instant 3, the second message leaves at time instance 7, the third message leaves at time instance 15 and the fourth leaves at time instance 17. Consequently, the output interval can go from 2 up to 8.

Next, we will investigate the transmission characteristics at a switch. First, we study the simple case in which a switch only receives real-time traffic from source nodes. This case exists mainly in a network with a star topology, where each node is connected to other nodes via the switch. Figure 3.2 illustrates a comparison between the transmission characteristics at a source node and at such a switch/port.

In Figure 3.2(a), we use the same example as Figure 3.1 to show how two channels at a source node interfere and lead to burstiness and jitter. Figure 3.2(b) illustrates a switch's output port queue. We again use the same channels, but from different source nodes. It should be noted that the propagation delays are not included here. We will add them afterwards when we analyze end-to-end worst case delay in Chapter 4. Our observations are:

- Incoming traffic is less bursty in the switch. At time instant 0, five frames arrive at the source node's output queue, while only two frames arrive at the output queue in the switch after one time slot because of the limited bit rate of the incoming physical links. In contrast, the output queue at a source node receives frame flows from multiple simultaneous applications without being limited by the bit rate. In other words, each frame flow to the switch is shaped by the physical link, while there is no such transmission smoothing functionality from the applications to the source node queue. It is worth noting that this observation is crucial to avoid over-estimation of the worst-case delay at the intermediate switches.
- Jitter still exists due to the interference among the real-time channels and the FCFS sorting policy. This is shown in Figure 3.3. Due to the interference of $\tau_1$, the first message of $\tau_2$ leaves the switch output

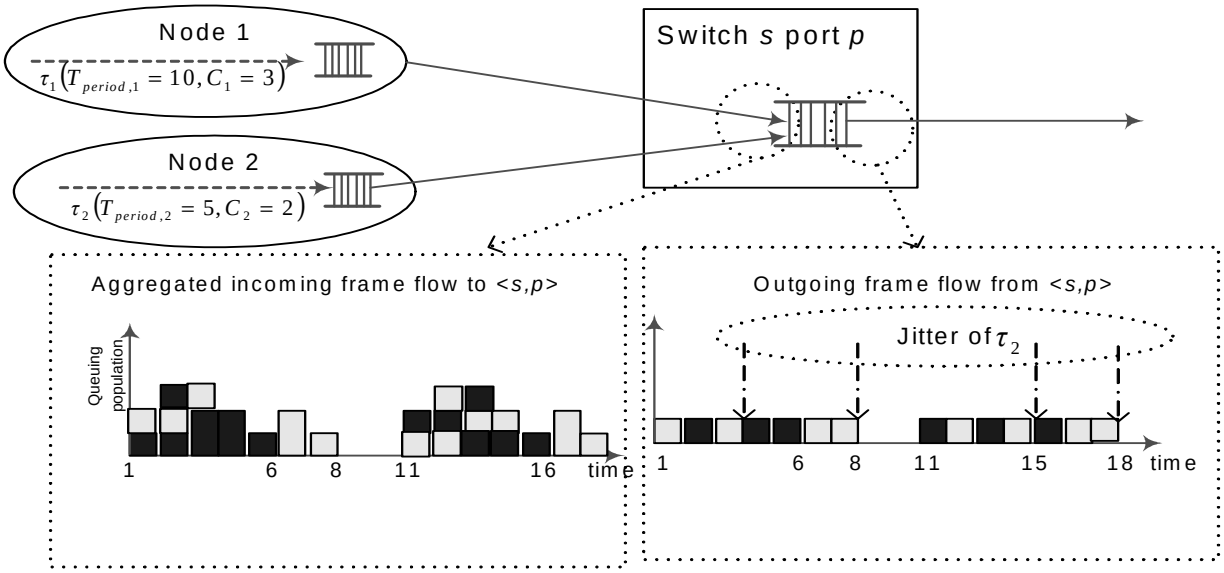



**Figure 3.3. Traffic injection and introduced jitter from the source nodes to the second hop.**

queue at time instant 3, the second message leaves at time instance 8, the third message leaves at time instance 15 and the fourth leaves at time instance 18. Consequently, the output interval can go from 3 up to 7.

In fact, with the knowledge of the traffic model at the source nodes, it is possible to model the traffic at the second hop. However, 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.

Based on the above observations, we can now conclude that the periodicity are only respected when the applications release messages at the source node, while the period of the messages varies at the intermediate hops, depending upon the arrival pattern of messages from other channels. Recall that we have made the decision of not using regulators, since this is not implemented in standard Ethernet switches. Thus, *we face the challenge of predicting the interference from other real-time channels and re-characterizing the traffic arrival pattern in the intermediate network elements.*

This challenge motivates us to develop three separate analytical schemes for the two types of network elements:

- Case 1: Source node receiving real-time traffic from the applications.
- Case 2: Switch only receiving real-time traffic from source nodes.
- Case 3: Switch receiving traffic from source nodes as well as other switches.

The remainder of this chapter is organized as follows. Section 3.2 presents the worst-case delay analysis for Case 1. Section 3.3 gives the worst-case delay analysis for Case 2. Section 3.4 describes the worst-case delay analysis for Case 3. Section 3.5 summarizes the chapter.

## 3.2 Case 1: Source node receiving traffic from applications

In this section, we derive the worst-case delay at a source node.

To analyze the schedulability for a set of real-time channels, it is important to find the critical instant (the message release pattern maximizing the delay of a channel). Hence, 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 proof strategy is as follows: first to find the critical instant and then proceed to analyze the worst-case delay.

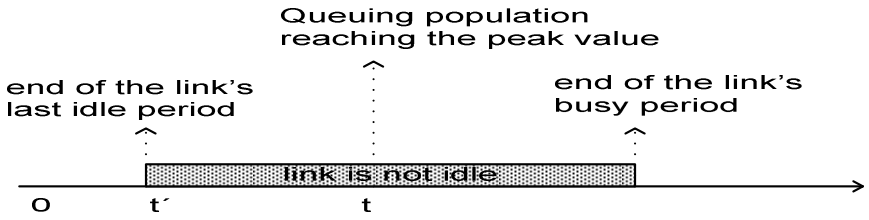



**Figure 3.4. Timing figures used in the proof of Lemma 3.1.**

In Lemma 3.1, we will prove that this fact also holds for FCFS scheduling. The proof of Lemma 3.1 is inspired by similar arguments as used for deadline-driven scheduling [Baruah et al. 1990] [Spuri 1996]. However, regarding FCFS, a new arriving frame has to wait for the completion of the transmission of all the remaining frames in queue. Therefore, the delay of it corresponds to the amount of traffic in the output queue at the arrival time, called queuing population and expressed in bits. Obviously, the worst-case delay corresponds to the maximum queuing population.

*Our proof idea of Lemma 3.1 is as follows. We will prove that given any message release pattern, the peak value of the queuing population is not higher than that of the synchronous pattern.*

**Lemma 3.1**. 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 end node *k*. Then the critical instant for any channel $\tau_i$ is the synchronous pattern of $\Gamma$**.**

**Proof.** Given any message release pattern, assume the peak value of the queuing population 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), as illustrated in Figure 3.4.

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. If we shift the messages released by all the other real-time channels after *t´* up to *t´* keeping their periodicity after the shifting, 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 shift 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. □

---

*Lemma 3.1 suggests studying the synchronous pattern of an FCFS-scheduled channel set in the first busy-period.*

Therefore, in Theorem 3.1, we will analyze the worst-case delay and prove that the achieved worst-case delay is tight (the minimum bound without any overestimation).

*Our proof idea of Theorem 3.1 is as follows. First, we calculate the queuing population, viewed as a function of time, QP(t), which is the cumulative workload arriving before t, excluding 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(t), t ≥ 0}. Finally, we show that the obtained worst-case delay is also tight.*

**Theorem 3.1** Assume FCFS-queuing is used for a set of real-time channels $\Gamma=\{\tau_1, \tau_2, .., \tau_n\}$ on the same source node $k$. If $U_k \le 100\%$, then $T_{sdelay,i} = \sum_{j=1}^{n} C_j / Rnode_k$ holds for $\forall i \in [1, n]$. In addition, the worst-case delay occurs at the beginning of the busy-period.

**Proof.** According to Lemma 3.1, the critical instant is the synchronous pattern. Without loss of generality, assume all real-time channels start at time 0.

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

$$W_k(0,t) = \sum_{j=1}^{n} \left( 1 + \left\lfloor \frac{t}{T_{period,j}} \right\rfloor \right) C_j . \tag{3.1}$$

Since $U_k \le 100\%$ holds, we have:

$$\sum_{j=1}^{n} \frac{C_j}{T_{period,j} Rnode_k} \le 100\% . \tag{3.2}$$

Therefore,

$$\sum_{j=1}^{n} \frac{C_j}{T_{period,j}} \le Rnode_k . \tag{3.3}$$

Hence,

$$\sum_{j=1}^{n} \frac{C_j}{T_{period,j}} - Rnode_k \le 0 . \tag{3.4}$$

Since $t$ is in the first busy period, $Rnode_k \cdot t$ bits are transmitted (removed from the queue). Consequently, according to Equation 3.1 and Equation 3.4, the amount of bits remaining to be transmitted at time $t$, $QP(t)$, is:

$$\begin{aligned} QP(t) &= W_k(0,t) - t \cdot Rnode_k \\ &= \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} \left( \left\lfloor \frac{t}{T_{period,j}} \right\rfloor \right) C_j - t \cdot Rnode_k \\ &\le \sum_{j=1}^{n} C_j + \sum_{j=1}^{n} \left( \frac{t}{T_{period,j}} \right) C_j - t \cdot Rnode_k \\ &= \sum_{j=1}^{n} C_j + t \cdot \left( \sum_{j=1}^{n} \frac{C_j}{T_{period,j}} - Rnode_k \right) \\ &\le \sum_{j=1}^{n} C_j \end{aligned} \tag{3.5}$$

This shows that the maximum queuing population is:

$$\max\{QP(t), t \ge 0\} = \sum_{j=1}^{n} C_i \tag{3.6}$$

Recall that the frame at the end of the queue has to wait for the transmission of all the remaining frames in the queue. With the maximum queuing population, the worst-case delay, $T_{sdelay,i}$, is calculated as:

$$T_{sdelay,i} = \sum_{j=1}^{n} C_j / Rnode_k \quad (\forall i \in [1,n]) \tag{3.7}$$

In order to show that $T_{sdelay,i}$ is the tight worst-case bound without any overestimation, according to Equation 3.5, we calculate the queuing population at time 0, $QP(0)$:

$$QP(0) = W_k(0,0) - 0 = \sum_{j=1}^{n} \left( 1 + \left\lfloor \frac{0}{T_{period,j}} \right\rfloor \right) C_j - 0 = \sum_{j=1}^{n} C_j = \max\{QP(t),\, t \geq 0\}. \tag{3.8}$$

Equation 3.8 indicates:

$$\max\{QP(t),\, t \geq 0\} = QP(0) = \sup\{QP(t),\, t \geq 0\}, \tag{3.9}$$

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 $\sum_{j=1}^{n} C_j / Rnode_i$.

This concludes the proof. □

---

It is worth noting that the worst-case delay will be the same for all real-time channels originating from the same source node due to FCFS. That is:

$$T_{sdelay,i} = T_{sdelay,j} = Dnode_k \quad if \;\; Source_i = Source_j = k, \tag{3.10}$$

where $Dnode_k$ is the worst-case delay at source node $k$. According to Theorem 3.1, we have:

$$Dnode_k = \sum_{Source_j = k} C_j / Rnode_k \quad . \tag{3.11}$$

By taking this into account, instead of calculating the worst-case delay for each real-time channel, the worst-case delay calculation for the source nodes can be improved by simply calculating $Dnode_k$ for each source node.

Recall the correspondence between the worst-case delay and the maximum queuing population. Using Theorem 3.1, we are now able to calculate the maximum required buffer size in the source node.

**Corollary 3.1** 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 $UNode_k = \sum_{j=1}^{n} \frac{C_j}{T_{period,j} Rnode_k} \leq 100\%$, then the maximum required buffer size at source node k is $BN_k = \sum_{Source_j = k} C_j$.

**Proof.** See Equation 3.6 in Theorem 3.1. □

---

## 3.3 Case 2: Switch only receiving traffic from source nodes

In this section, we investigate the worst-case delay for the second hop, in the case where a switch port only receives traffic from source nodes.

Recall that the real-time analysis of a switch is different from that of a source node due to the traffic shaping nature of the incoming physical link and the introduced jitter caused by the interference of other channels (as illustrated in Figure 3.2 and 3.3). Nevertheless, the idea of worst-case delay analysis is still to first find the critical instant and then to calculate the worst-case delay. This time we use Lemma 3.2, Theorem 3.2 and Algorithm 3.1.


**Figure 3.5. Timing figures used in the proofs of Lemma 3.2.**

It should be noted that the propagation delays are not included in the worst-case delay analysis for Case 2 and Case 3. We will add them afterwards when we analyze end-to-end worst-case delays in Chapter 4.

In contrast to the source node case, the worst-case delay at the switch output port 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, as illustrated in Figure 3.2. In Lemma 3.2, we will prove that the cumulative traffic from source node $k$ to the second hop under the synchronous pattern at that source node is not less than that under any other release pattern. Then, in Theorem 3.2, we will prove that the critical instant at the second hop 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 Lemma 3.2 is as follows. We analyze the cumulative traffic volume from a source node to the second hop during a certain time interval. It should be noted that the cumulative traffic volume is the amount of traffic that has been delivered from a source node to the second hop during a certain time interval, and it affects the workload at the second hop. It is proven that the cumulative outgoing traffic volume under the synchronous pattern is not less than any other message release pattern by considering two cases.*

**Lemma 3.2** Assume that FCFS queuing is used to schedule a set of real-time channels $\Gamma_k=\{\tau_i: 1\leq i \leq n_k\}$ originating from source node $k$. Also, assume that the outgoing link from the second hop is busy at time instant $t$ and $t'$ is the end of the last link idle period before $t$ ($0 \leq t' \leq t$). Then, the volume of cumulative traffic from source node $k$ to the second hop 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 ($0 \leq t' \leq t$).

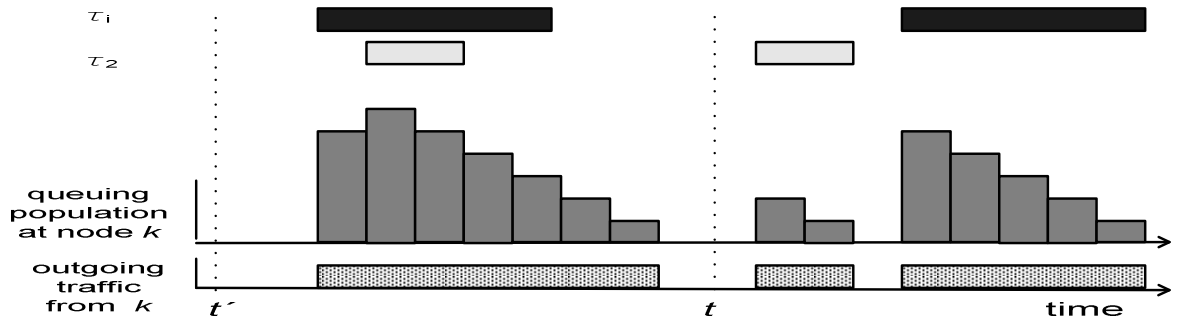
**Proof.** Given any message release pattern, without loss of generality, assume that $\tau_i$ ($1\leq i \leq n_k$) release messages at $t' + \Delta r_i$, ($\Delta r_i \geq 0$), as illustrated in Figure 3.5.

According to Definition 2.21, the cumulative workload for the incoming link from node $k$ during $[t', t)$, $W_k(t', t)$, is derived as:

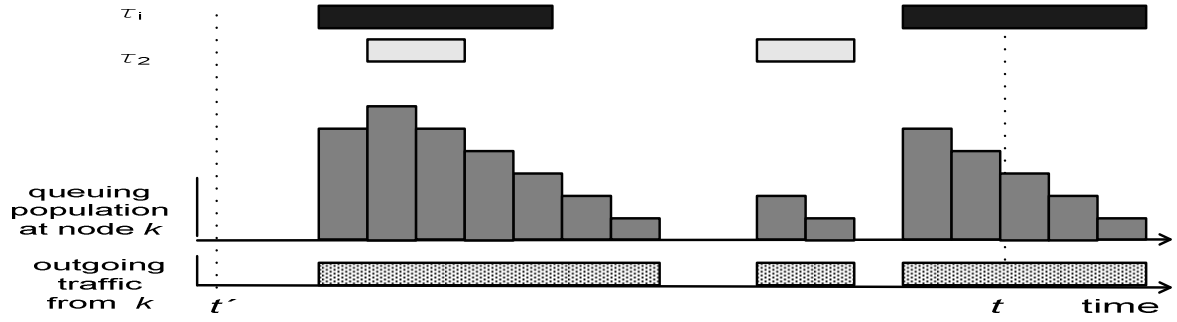
$$W_k(t',t)=\sum_{i=1}^{n_k}\left(1+\left\lfloor \frac{t-t'-\Delta r_i}{T_{period,i}} \right\rfloor\right)C_i . \tag{3.12}$$

Let $<s, p>$ represent the second hop, the output port $p$ at switch $s$, and let $Traffic_k(t', t)$ represent the volume of cumulative traffic from source node $k$ to $<s, p>$ during $[t', t)$. We will now observe $Traffic_k(t', t)$ by considering two cases: These two cases are illustrated in Figure 3.5(a) and Figure 3.5 (b), respectively. The fist one represents the case in which the link from $k$ is idle at time instant $t$. The second one represents the case in which the link from $k$ is busy at time instant $t$. It should be noted that the incoming 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. As illustrated in Figure 3.6 (a), in this case, 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,




**(a)**




**(b)**

**Figure 3.6. Illustration of workload and outgoing traffic from node k. (a) Case I. (b) Case II.**

$$Traffic_k(t',t) = Rnode_k(t-t'-t_{idle}) \le Rnode_k(t-t') . \quad (3.14)$$

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:

$$Traffic_k(t',t) = W_k(t',t). \quad (3.15)$$

Case II. As illustrated in Figure 3.6 (b), in this case, 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:

$$Traffic_k(t',t) = Rnode_k(t-t'-t_{idle}) \le Rnode_k(t-t') , \quad (3.16)$$

And the remaining traffic at node $k$ indicates:

$$Traffic_k(t',t) \le W_k(t',t). \quad (3.17)$$

If we shift earlier the messages released after $t'$ by all the real-time channels $\Gamma_k=\{\tau_i: 1\le i \le n_k\}$ to $t'$ keeping their periodicity, the workload of the incoming link originating from node $k$ after the shifting, $W_k(t', t)$, will not be decreased, that is:

$$W'_k(t',t) \ge W_k(t',t). \quad (3.18)$$

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 observe $Traffic_k(t', t)$ 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 3.14 and Equation 3.17, 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:

$$Traffic'_k(t', t) = Rnode_k(t - t' - t'_{idle}) \le Rnode_k(t - t') . \quad (3.19)$$

According to Equation 3.17, the non-decreased workload after the shifting indicates a non-increased length of idle time, that is:

$$t'_{idle} \le t_{idle} . \quad (3.20)$$

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 3.16 and Equation 3.17, we have $Traffic'_k(t', t) \ge Traffic_k(t', t)$.

b), Otherwise, if 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:

$$Traffic_k'(t', t) \ge Traffic_k(t', t) . \quad (3.21)$$

This concludes Lemma 3.2 □

---

*The importance of Lemma 3.2 is summarized as follows:*

- *It gives a hint on how the critical instant at the second hop can be found by studying the synchronous pattern at the source node.*
- *The need for multiple cases and sub-cases in the proof indicates the difficulty of deriving a simple analytical expression of the incoming traffic volume to the second hop, which leads to the difficulty of deriving the worst-case delay at the second hop. Therefore, we consider developing an algorithm to solve this problem.*
- *It also gives a hint on how to calculate the worst-case delay at the second hop efficiently. This will be discussed further later in this section.*

With the knowledge of each incoming flow, we are now able to study the aggregated traffic flow to the switch/port and to find the critical instant (in Theorem 3.2).

*Our proof idea of Theorem 3.2 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 the switch/port. Second, we calculate the queuing population, viewed as a function of time, QP(t), which is the aggregated traffic from all the incoming links arriving before t, excluding 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 3.2** Assume that FCFS queuing is used to schedule a set of real-time channels $\Gamma_k=\{\tau_i: 1\le i \le n_k\}$ all of which traverse the same output port $p$ of the switch $s$. Then, the critical instant for any channel $\tau_i$ at the output port is a direct consequence of the synchronous pattern of $\Gamma$ at the source nodes.

**Proof.** Without loss of generality, assume that channel set $\Gamma$ can be decomposed into several subsets:

$$\Gamma = \bigcup_{k=1}^{Nnode} \Gamma_k , \Gamma_k = \{\tau_i : Source_i = k\} , \quad (3.22)$$

where $\Gamma_k$ includes the real-time channels originating from the same source node $k$.

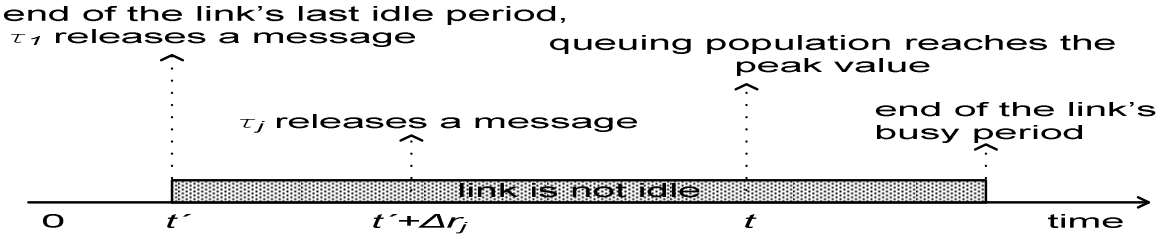



**Figure 3.7. Timing figures used in the proofs of Theorem 3.2.**

Given any message release pattern, assume that the peak value of the queuing population in the queue for port *p* in switch *s* 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 $\tau_1$ releases a message at *t'*, and each of the other channels $\tau_j (2 \leq j \leq n)$ release messages at $t' + \Delta r_i$, ($\Delta r_i \geq 0$), as illustrated in Figure 3.7.

Let <*s*, *p*> represent output port *p* at switch *s* and let $Traffic_k(t', t)$ represent the volume of cumulative traffic from source node *k* to <*s*, *p*> during [*t'*, *t*).

We know that 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:

$$W_{s,p}(t',t) = \sum_{k=1}^{Nnode} Traffic_k(t',t) \tag{3.23}$$

Hence, the queuing population (the amount of bits that remain to be transmitted from <*s*, *p*>) at time *t*, $QP_{s,p}(t)$ is:

$$QP_{s,p}(t) = W_{s,p}(t',t) - (t-t')Rswi_{s,p}\,. \tag{3.24}$$

If we shift earlier the messages released after *t'* by all the other real-time channels $\{\tau_2, \ldots, \tau_n\}$ to *t'* keeping their periodicity, then $Traffic'_k(t', t)$, the amount of cumulative outgoing traffic from source node *k* to <*s*, *p*> during [*t'*, *t*) after the shifting will not be decreased, according to Lemma 3.2. That is,

$$Traffic_k'(t',t) \geq Traffic_k(t',t) \tag{3.25}$$

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

$$W'_{s,p}(t',t) = \sum_{k=1}^{Nnode} Traffic'_k(t',t) \tag{3.26}$$

Hence, according to Equation 3.24 and Equation 3.25, we have:

$$W'_{s,p}(t',t) \geq W_{s,p}(t',t). \tag{3.27}$$

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

$$QP'_{s,p}(t) = W'_{s,p}(t',t) - (t-t')Rswi_{s,p} \geq W_{s,p}(t',t) - (t-t')Rswi_{s,p} = QP_{s,p}(t). \tag{3.28}$$

Since the outgoing link from <*s*, *p*> is not idle during the time interval [*t'*, *t*) before the shift, there will be no link idle time during the time interval [*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 3.2. □

---

*Theorem 3.2 suggests studying the schedule of the synchronous pattern in the first busy-period. Therefore, in our worst-case delay calculation algorithm, we check the delay within the first busy-period under the synchronous pattern.*

Similar to the improvement suggested in Equation 3.10, we do not need to calculate the worst-case delay at the second hop for each real-time channel, because the following equation holds:

$$T_{i,1} = T_{j,1} = Dport_{s,p} \quad if \ \langle switch_{i,1}, port_{i,1} \rangle = \langle switch_{j,1}, port_{j,1} \rangle = \langle s, p \rangle, \tag{3.29}$$

where $Dport_{s,p}$ is the worst-case delay for the frames through output port $p$ in switch $s$. Therefore, the worst-case delay analysis at the switch/port pair can be improved by simply calculating $Dport_{s,p}$ for each port.

According to the observation in Figure 3.2, the worst-case delay at the second hop does not always occur at the beginning of the first busy period. However, Lemma 3.2 shows the difficulty of finding a simple analytical expression to calculate the maximum queuing population and the worst-case delay. That means that the calculation of $Dport_{s,p}$ will differ from the calculation of $Dnode_k$ in the sense that we must check the queuing population at each time instant during the first busy period in order to find the maximum value.

The calculation of $Dport_{s,p}$ is an utility function that checks the queuing delay (queuing population) at different time instants, and it is presented in Algorithm 3.1. The important issues addressed in Algorithm 3.1 are described below.

- First, according to Lemma 3.2, the examined interval can be limited to the first busy-period, which reduces the time and memory complexity of the algorithm. Without loss of generality, the start of the busy-period is assumed to be at time instance 0.
- Second, to calculate $Dport_{s,p}$, we check the amount of queued traffic (in bits) at certain time instants. However, making checks at bit-level resolution suffers from high computational complexity. In our analysis, checking the queuing population will therefore be event driven. To that end, the queuing population will be checked only at such time instants when

  1) a new period of any related logical real-time channel starts.

  2) the output queue from any source node becomes empty.

  Our rationale for this checking strategy is as follows.

  Recall that the queuing population is the aggregated traffic volume from all the incoming links (Equation 3.22). Hence, 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 following two cases (illustrated in Figure 3.8 (a), and Figure 3.8 (b), respectively). The first one represents the case in which the link from $k$ is idle at time instant $t$. The second one represents the case in which the link from $k$ is busy at time instant $t$.

  Case 1. As illustrated in Figure 3.8 (a), the link from node $k$ is idle. Hence, the cumulative workload during $[t', t)$ has been delivered to the next hop, that is:

  $$Traffic_k(t', t) = W_k(t', t). \tag{3.30}$$

  As long as $t$ remains in the same idle period, 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.

**Algo_3_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. Find out 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 Find out how many bits on the way for each incoming physical link queued in each source node.
   **for** *i = 1..Nchs*
   **if** $\langle s, p \rangle = \langle Switch_{i,1}, Port_{i,1} \rangle$ && *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.
   f**or** *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 Find out 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 next time instant.
   **if** ( *t* < *BP(Γ)* ) **then** repeat step 2; **end if**
4. **Return** ( $Dport_{s,p}$ =$Dport_{s,p}$ / $Rswi_{s,p}$ , $BS_{s,p}$ =$Dport_{s,p}$).

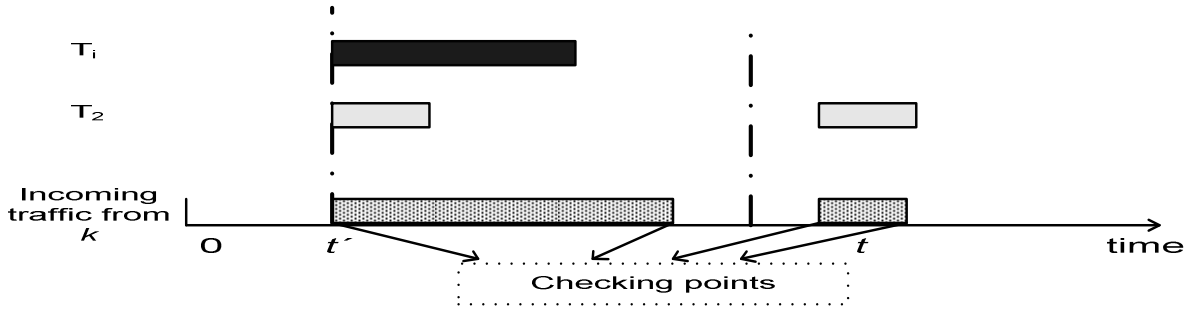
**Algorithm 3.1. Dports,p and BSs,p calculation algorithm (FCFS queuing and <s, p> only receiving real-time traffic from source nodes.**

Case 2. As illustrated in Figure 3.8 (b), the link from *k* is busy at time instant *t*. Note that link from node *k* may have one or several idle periods during [*t´*, *t*). Let $t_{idle}$ be accumulated length of all link idle periods during [*t´*, *t*). Thus, we have:

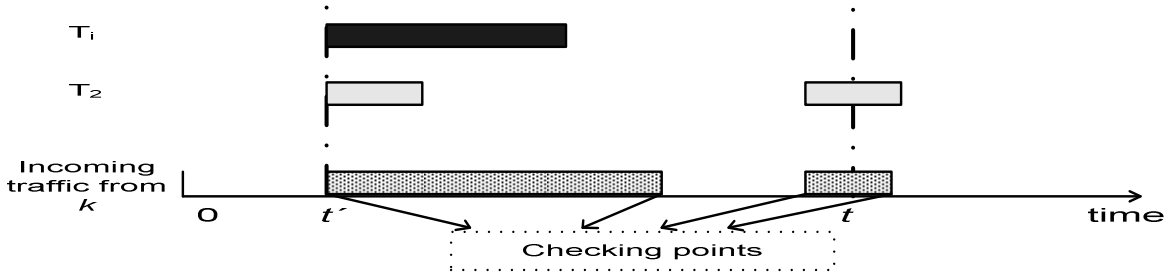
$$Traffic_k(t', t) = Rnode_k(t - t' - t_{idle}), \tag{3.31}$$

As long as *t* remains in the same busy period, the value of $Traffic_k$(*t´*, *t*) does not change due to the unchanged value of $t_{idle}$. Therefore, we only need to keep track of its value at the beginning 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




**(a)**




**(b)**

**Figure 3.8. Illustration of event-driven queuing population checking. (a) Case I: $[t', t)$ is in a busy-period. (b) there is idle time during $[t', t)$.**

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.

- Third, we simulate the transmission over the physical links in Algorithm 3.1 by keeping track of the exact amount of incoming traffic at each time interval. The simulator works as follows. After a given time interval, 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.

To conclude, the output of Algorithm 3.1 is the tight worst-case delay because we do not miss the peak value and we do not overestimate according to the previous discussion.

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

In this section, we will show how to calculate the worst-case delay for a switch/port that can receive traffic both from source nodes and other switches. Similar to the analysis in Section 3.3, 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 the case of the second hop (Lemma 3.2), 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-

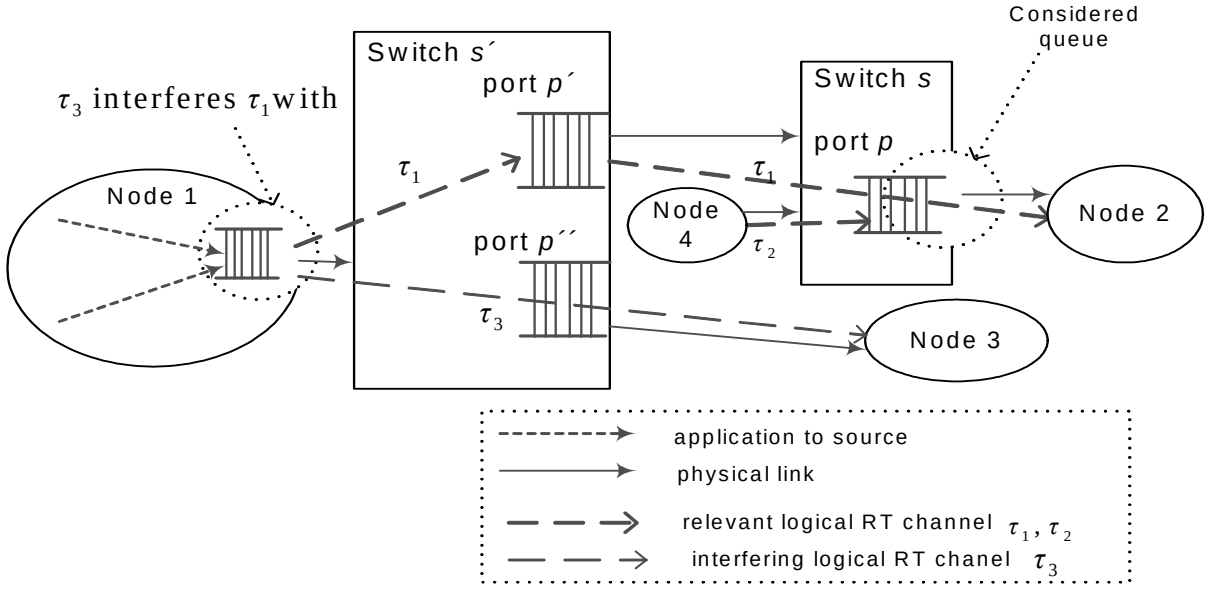



**Figure 3.9. Traffic injection and interference after the second hop.**

case delay analysis in a similar way as we did for the second hop: we first derive the worst-case delay in the first busy-period and then use a utility function to calculate the queuing population.

Unfortunately, another fact complicates the delay analysis after the second hop. Figure 3.9 illustrates this issue regarding the delay at <*s*, *p*>. Here, $\tau_1$ and $\tau_2$ are the relevant channels, while $\tau_3$ is an interfering channel. $\tau_1$ originates from node 1, traverses port <*s'*, *p'*>, port <*s*, *p*> and finally arrives at node 2. $\tau_2$ is relevant because it also traverses port <*s*, *p*>. $\tau_3$ originates from node 1, traverses port <*s'*, *p´´*> and finally arrives at node 3. Although $\tau_3$ does not traverse port <*s*, *p*>, it does interfere with $\tau_1$ 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 $\tau_1$ and $\tau_2$ leave at their previous hops at the same time, not-yet-transmitted frames belonging to $\tau_1$ might remain in the output queue of the previous port <*s'*, *p'*>, due to interference with frames of $\tau_3$ 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 port, we also need to find the upper-bound of the additional delay introduced by such remaining frames in the previous port. Since we choose not to use traffic regulators, we cannot predict for how long time the messages have remained in the buffer. 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.

Before we present our algorithm, we use the example shown in Figure 3.9 to explain our idea. For example, to analyze the delay at port <*s*, *p*>, we need to calculate the maximum populations 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). We derive that the maximum populations of the remaining frames at <*s'*, *p'*> equals to $BS_{s',p'}$, by concerning the facts, a) the upper bound of the buffer population at <*s'*, *p'*> can be calculated by using Algorithm 3.1; b) in the worst case, all these queued frames traverse <*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 3.2. To calculate the worst-case delay for channel *i* at its *j*th switch/port, $T_{i,j}$, we begin with the calculation at the first switch/port, then proceed to the next hop by taking the remaining frames into account, finally stop when having the delay for the *j*th switch/port. The worst-case delay and buffer size calculation for each switch/port is calculated by Algorithm 3.3, which is an extension of Algorithm 3.1, 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.

**Algo_3_2** ($i$, $j$)
**Input** ($i$, $j$)
**Output** ( $T_{i,j}$ )

1 **for** ($k=1$; $k<=j$; $k++$)
Call Algo_4_3 to calculate the worst-case delays at port <$switch_{i,k}$, $port_{i,k}$>;
**end for**
2 $s=switch_{i,k}$. $p=port_{i,k}$
3 $T_{i,j=}$ $Dport_{s,p}$
4 **return** ( $T_{i,j}$ )

**Algorithm 3.2. Iterative method to calculate $T_{i,j}$, the worst-case delay** for $\tau_i$ **at its $j$th hop (j ≥1) .**

The iterative idea behind Algorithm 3.2 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 would create deadlocks.

Algorithm 3.3 is extension of Algorithm 3.1, 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.

Note that as more hops are considered in the Algorithm, more pessimism will be introduced by the recursive approach. Therefore, we will evaluate the amount of pessimism by simulations. This is what we do in Chapter 5.

Similar to the improvement suggested in Equation 3.20, we do not need to calculate the worst-case delay at hop <$s$, $p$> for each real-time channel, because the following equation holds:

$$T_{i,k} = T_{j,k} = Dport_{s,p} \quad if \ \langle switch_{i,k}, port_{i,k} \rangle = \langle switch_{j,k}, port_{j,k} \rangle = \langle s, p \rangle, \tag{3.32}$$

where $Dport_{s,p}$ is the worst-case delay for the frames through output port $p$ in switch $s$. Therefore, the worst-case delay analysis <$s$, $p$> can be improved by simply calculating $Dport_{s,p}$ for each port.

## 3.5 Summary

In this chapter, we have presented the worst-case delay analysis for different types of isolated network elements: a source node receiving real-time traffic from applications, a switch only receiving real-time traffic from source nodes and a switch receiving real-time traffic from both source nodes and other switches.

We have developed methods to obtain accurate worst-case delays and buffer bounds for the first two cases. For the third case, we have also developed a recursive algorithm to calculate the worst-case delay and buffer bound, which might give pessimistic results, noting the fact that it is difficult to avoid overestimation for that case. We are aware that the obtained worst-case delay may be not tight. In order to access the quality of that, we must do simulation study. Our simulation results will be presented in Chapter 5.

With the worst-case delays at isolated network elements, we are able to obtain the end-to-end worst-case delay and carry out real-time analysis for the whole communication network in the subsequent chapters.

**Algo_3_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_3 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*.
2.1 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 Find out 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 next check time instant.
**if** ( *t < BP(I)* ) **then** repeat step 3; **end if**
5. **Return** ( $Dport_{s,p} = Dport_{s,p}$ / $Rswi_{s,p}$ , $B_{s,p}=Dport_{s,p}$).

**Algorithm 3.3. $Dport_{s,p}$ and $BS_{s,p}$ calculation algorithm for FCFS queuing at the switch port communicating with source nodes and other switches.**

# Chapter 4 Real-time Analysis for Switched Ethernet Networks

Based on our worst-case delay estimation for isolated network elements in Chapter 3, we will now propose deterministic approaches towards end-to-end delay analysis of real-time traffic under two switched Ethernet network configurations: standard switched Ethernet with FCFS in all elements and switched Ethernet with source node EDF. Our analysis is developed for networks with multiple switches, which can of course be used for single-switch networks.

Recall that if only schedule the traffic according to FCFS without any further analysis, we can only provide best-effort service. In this section, we will use the results achieved in Chapter 3 to find the schedulability condition to guarantee that all channels meet their deadlines.

In accordance with the definitions and discussion in Chapter 2, the time constraints are guaranteed by 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, 100%:

$$UNode_k = \sum_{i:Source_i=k} \frac{C_i}{T_{period,i} Rnode_k} \leq 100\% . \quad (4.1)$$

Likewise, and for the physical link from any switch/port pair $<s, p>$, $Uswi_{s,p}$, must be less than or equal to the maximum value, 100%:

$$Uswi_{s,p} = \sum_{i:\langle s,p\rangle \in Route_i} \frac{C_i}{T_{period,i} Rswi_{s,p}} \leq 100\% . \quad (4.2)$$

The utilization check is not enough. 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,

$$T_{e2edelay,i} \leq T_{dl,i} . \quad (4.3)$$

The end-to-end worst case delay is illustrated in Figure 4.1, which includes worst-case delay at each hop and propagation delays on the traversed physical links.

Taking the delays from Theorem 3.1 and Algorithm 3.2, we have

$$T_{e2edelay,i} = T_{sdelay,i} + \sum_{j=1}^{Nr_i} T_{i,j} + (Nr_i + 1)T_{prop} + T_{node} + Nr_i \cdot T_{switch} , \quad (4.4)$$

where $T_{switch}$ and $T_{node}$ are the worst case process latency for an Ethernet frame at the top of the hard real-time queue to leave the source node and to leave the switch, respectively, since we cannot interrupt the transmission of frames that have been stored in the NIC (Network Interface Card) or the transmission of frames on a physical link. We assume (the constant can however, easily be changed in the equation):

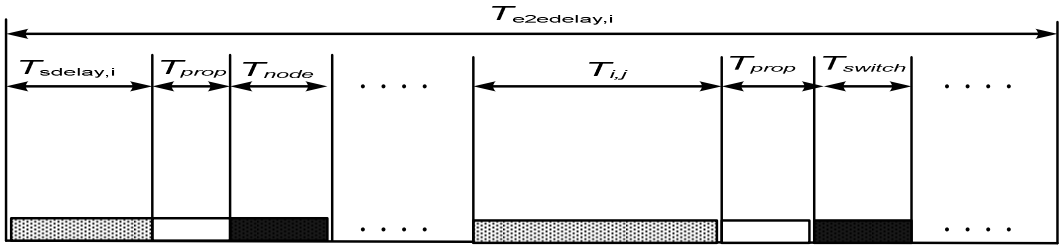



**Figure 4.1. Timing diagram of the worst-case end-to-end delay over standard switched Ethernet only utilizing FCFS.**

$$T_{node} = 2T_{frame}, \quad (4.5)$$
$$T_{switch} = T_{frame}, \quad (4.6)$$

where $T_{frame}$ is the transmission duration of a maximum-sized Ethernet frame (1526 bits) over the Ethernet medium, which is 121 μs for Fast Ethernet and 12 μs for Gigabit Ethernet. The operating system delay is not treated in this paper, but can also be added as a constant in the worst-case delay calculation [Skeie et al. 2002] [Bello and Mirabella, 2004].

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 the real-time channels meet their deadlines.

In the previous sections, we have developed schedulability conditions under two different switched Ethernet network configurations. The schedulability test is done by the admission controller, which can be implemented in an end node or the switch. Furthermore, our analysis provides the flexibility of running the admission control either offline or online.

In Chapter 3, we have discussed that the delay analysis for Case 3 is not accurate. Hence, it is obvious that the extended end-to-end analysis for networks with multiple switches is not accurate.

However, in Chapter 3, we have proved that our delay analysis at the source node and at the switch in single-switch network is accurate. Furthermore, is extended end-to-end analysis for single-switch network is still accurate? The answer to this question is no. Even though the worst-case delay is accurately estimated locally, but the global aggregation can lead to pessimism. The accuracy of our end-to-end analysis 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 one. At a source node, we analyze the channels originating from this node but may traversing different switch ports, while at the switch, we consider the channels may originating from different source nodes but traversing the same port in the switch. Obviously, the worst-case scenario at the source node and that at the switch may not co-appear. Therefore, the end-to-end analysis is not accurate.

In Chapter 5, we will use simulation analysis and comparison analysis to verify our feasibility analysis.

# Chapter 5 Performance Evaluation

In this section, we will evaluate our real-time analysis by simulation and comparison study, concerning the previous discussions on the fact that our estimation on end-to-end worst-case delay may not be tight. We will do two types of study. The first one is a simulation evaluation of our analysis, while the second one is to compare with NC both conceptually and experimentally.

## 5.1 Simulation Evaluation on our analysis

In order to evaluate the performance of our approaches, we conducted several sets of simulations. In the first set of simulations, we evaluate the performance of the admission control algorithm for different traffic loads and network characteristics. Packet level simulations are conducted in the second set of simulations for observation of, e.g., the average end-to-end packet delay.

### Experimental Setup

In this section, we present our simulation setup.

We have simulated a network with a single full-duplex Ethernet switch and a number of end nodes. The Fast Ethernet switch with the same bit rate on all ports, 100 Mbits/s, is chosen in most of the following simulations except one simulation shown in Figure 5.1(c).

We have developed a simulator including three main function blocks: traffic generation, admission control and packet transmission. The first step is the traffic generation. In each simulation, the real-time channels are randomly generated with uniformly distributed source and destination nodes. Then the logical real-time channels are checked one by one by the admission controller as to whether or not they have been accepted. The periodic hard real-time traffic intensity is increased by increasing the number of logical channels traversing the system. The last step is to deliver the packets following the defined traffic handling policy, the priority FCFS-queuing.

To evaluate our real-time analysis, we have conducted both channel level simulations and packet level simulations. In the channel level simulations, only the first two steps, traffic generation and admission control, are executed, while in the packet-level simulations, all three steps are executed.

Such simulations are run 100 times to get the average performance at different traffic and network characteristics.

Certain assumptions were made:

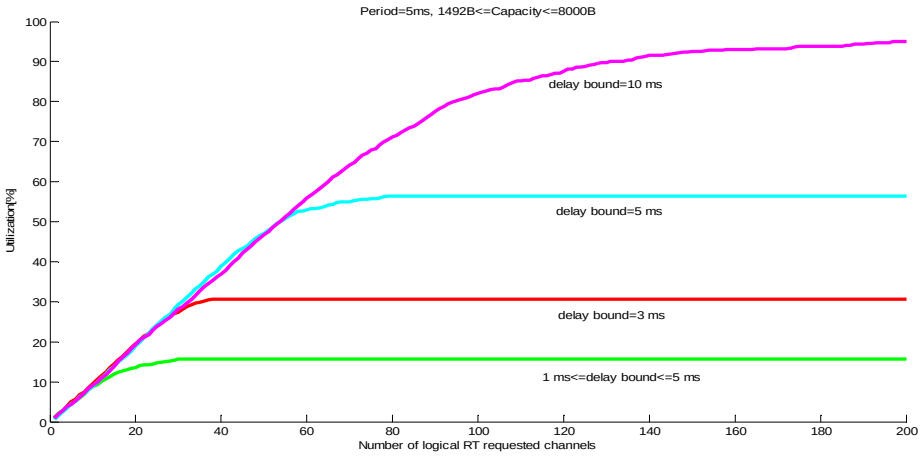
- Hard real-time traffic is periodic, and the packets are generated at the beginning of the period.
- All buffers are large enough. Therefore no packet loss occurs because of buffer overflow.
- The link propagation delay is set to 500 ns, corresponding to a 100 meter link.

### Network Utilization

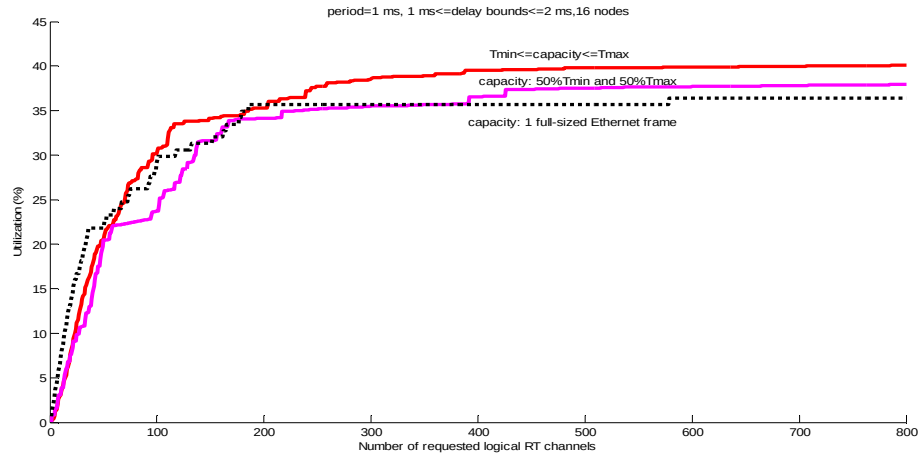
The aim of the first set of simulations is to observe the performance of our feasibility analysis. For this purpose, it is important to see how many real-time channels that can be accepted in the system, under the condition of guaranteeing the worst-case delays for all real-time channels. More accepted channels means lower worst-case delays in the estimation made by the admission controller. Therefore, we measure the network utilization, *UNet*, which is defined as below.

**Definition 5.1** 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 one link is busy, that is,

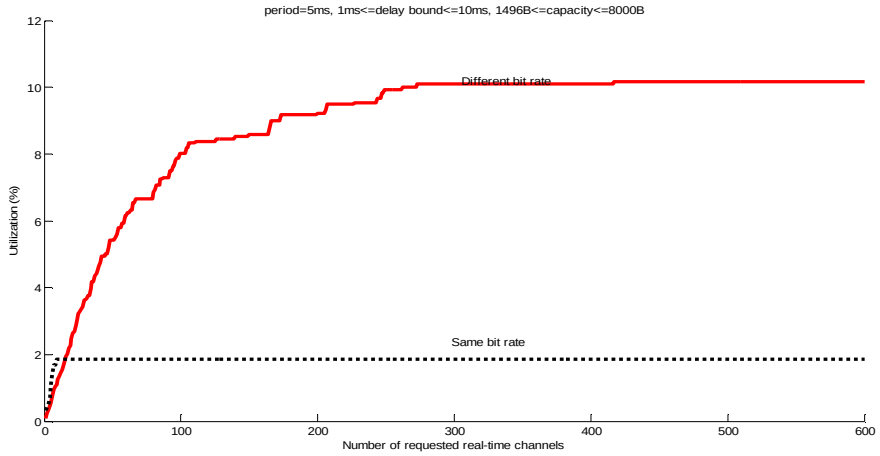
$$UNet = \frac{\sum_{k=1}^{Nnode} U_k + \sum_{s=1}^{Nswi} \sum_{p=1}^{Nport_s} U_{s,p}}{Nnode + \sum_{s=1}^{Nswi} Nport_s}.$$




**(a)**




**(b)**




**(c)**

**Figure 5.1: Network utilization vs total number of requested real-time channels. (a) Different deadlines. (b) Short messages. (c) Asymmetric traffic distribution (1 master node, 31 slave nodes).**

Figure 5.1 shows the relation between the network utilization and the total number of requested real-time channels.

How end-to-end deadlines affect utilization is presented in Figure 5.1(a). The number of nodes is 8. The period is set to 5 ms for all the logical real-time channels, the capacities are randomly generated from 1492 bits to 8000 bytes, and the deadlines are randomly selected from a certain set. We observe different results when the worst-case delays set is changed among the values $A$={$x$ | 1 ms ≤ x ≤ 5 ms}, $B$={3 ms}, $C$={5 ms} and $D$={10 ms}. In other words, sets $B$, $C$ and $D$ have the same deadlines for all channels. Note that the lowest utilization is obtained when all the end-to-end deadlines are chosen from set $A$, which includes very short end-to-end deadlines, and the highest utilization occurs when end-to-end deadlines are chosen from set $D$, which includes long end-to-end deadlines. The reason is that a longer end-to-end deadline makes it easier to meet the constraint and thus higher utilization can be achieved, which has already been proven by the scheduling theory. Figure 5.1(a) also reveals that our system can reach rather high utilization, for example, it reaches 90% when the worst-case delay is twice the length of the period.

In many real-time systems, for example, automation industry, a major fraction of the traffic consists of short messages [Weber et al. 1999]. Therefore we have carried out a simulation study for real-time short-message traffic, shown in Figure 5.1(b). There are 32 nodes in the network. The period is set to 1 ms for all the logical real-time channels, the end-to-end deadlines are randomly generated from 1 ms to 2 ms, and the capacities are randomly selected from a certain set. We observe different results when the capacity set is changed among the values $A$={$x$ | $T_{mind} \leq x \leq T_{maxd}$} including variable-sized frames, $B$={$T_{mind}$, $T_{maxd}$} including 50% very short messages, and $C$={$T_{maxd}$}, which only has fix-sized frames. Figure 5.1(b) verifies that resources can be efficiently used in many situations with our feasibility analysis, e.g. in the case where there is a large amount of short messages in the system.

Considering the fact that one of the key requirements in the fieldbus domain is to transfer data from a master node to a number of slave nodes, we have simulated a network with asymmetric traffic distribution, shown in Figure 5.1(c). The system has one master node that periodically distributes hard real-time traffic to the other 31 slave nodes. The period is set to 5 ms for all the real-time channels, capacities are randomly generated from 1492 bits to 8000 bits, and the end-to-end deadlines are randomly generated between 1 ms to 10 ms. It can clearly be seen that, when the traffic is unevenly distributed in the system (slave nodes only send packets to the master node), a dramatic performance gain is achieved by the different bit-rate network where the master node is connected to the switch via a link with a bit rate of 1 Gbits/s and each slave node is connected to the switch via a 100 Mbits/s link. The network with the same bit rate on all ports (100 Mbits/s) instead experiences the link between the master node and the switch as a bottleneck. It can thus be concluded that resources are more effectively utilized by using a switch with different bit rates than by using a homogeneous bit-rate switch for this master-slave traffic pattern. As mentioned, our feasibility analysis supports such configurations.

### Throughput measured by packet-level simulation

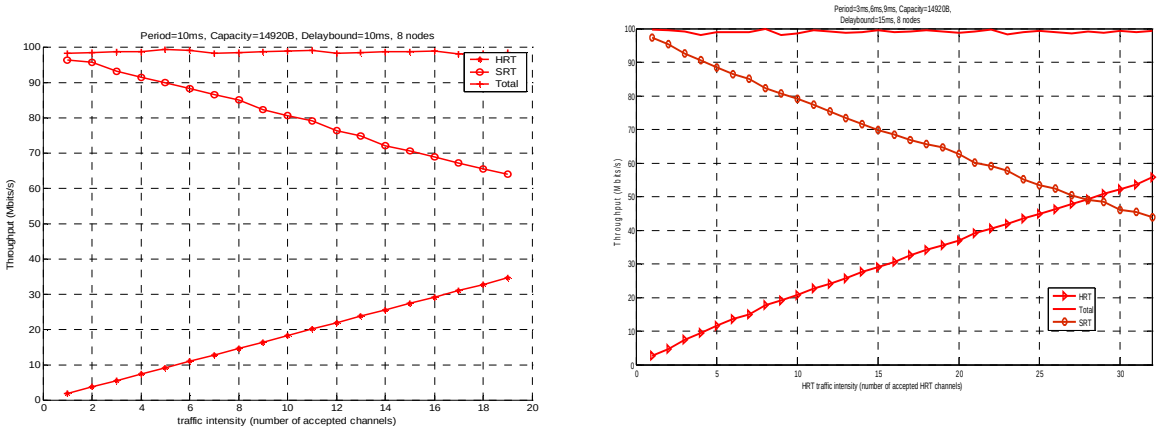
In addition to network utilization, throughput has been used as the performance measure for communication systems.

**Definition 5.2** *Throughput*, *TP* (Mbits/s), is the amount of traffic transmitted per time unit in the network.

We conducted packet-level simulations to measure the throughput. In our simulation, the HRT traffic is generated and checked in the same way as in the previous simulations. Meanwhile, we keep the periodic SRT traffic intensity at a constant rate of 100%. If the HRT queue is empty, the bandwidth can be used to carry SRT traffic. Such a simulation set-up is effective to observe how different traffic classes affect each other and to show the maximum possible throughput for both HRT and SRT.

Figure 5.2 provides us with the information of how the HRT traffic intensity affects the throughput. There are 8 nodes in the network.

Figure 5.2(a) shows the results when every logical real-time channel has a period of 10 ms, a capacity of 14920 bits and a end-to-end deadline of 10 ms. It can be seen that an increasing HRT traffic load leads to a steady increase in HRT throughput, while the SRT traffic experiences a steady decrease due to its low priority. The total throughput reaches up to a value of about 100 Mbits/s.




**(a)** **(b)**

**Figure 5.2. Throughput vs total number of accepted hard real-time channels. (a) Same parameters for all real-time channels. (b) Different parameters are selected for real-time channels.**

Figure 5.2(b) illustrates the results when having traffic with different traffic parameters. The period is randomly chosen from the set {3 ms, 6 ms and 9 ms}, the capacity is 14920 bits, while all end-to-end deadlines are set to 15 ms. The trends of the curves are similar to the curves in Figure 5.2(a), but where the maximum possible HRT traffic load is higher, because the parameters of the real-time channels make it possible to accept more HRT traffic.

## End-to-end Delay

In the admission algorithm, in order to achieve the guaranteed real-time services, we predict the worst-case end-to-end packet delay. However, it is also interesting to see the average performance. For example, what is the average packet delay? How much is the difference between the average delay and the worst-case delay? Is our delay prediction pessimistic or not? In this section, we conduct simulations to answer these questions.

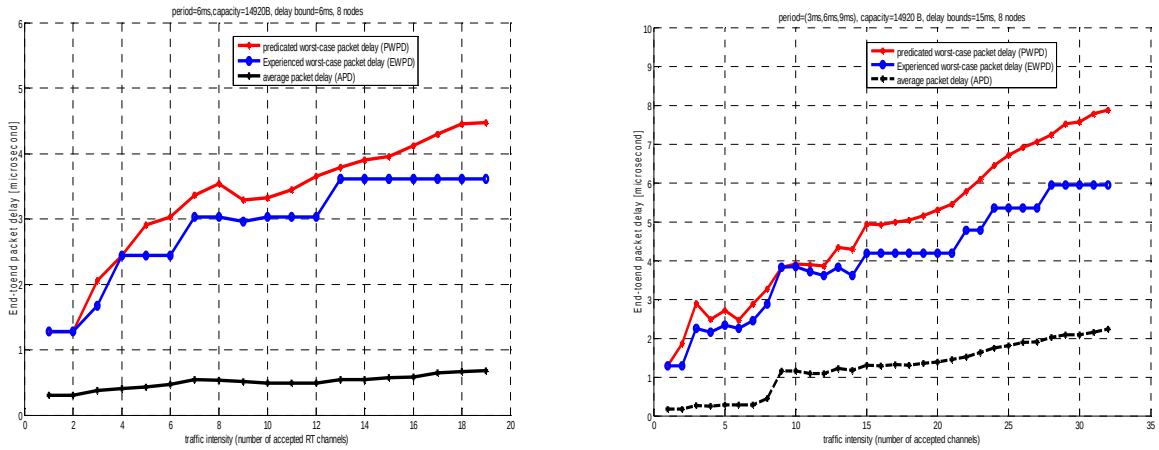
The end-to-end delay is defined as the time from the moment a packet is ready to be transmitted to the moment the packet has successfully arrived at the destination, which includes the transmission delay, queuing delay and link propagation delay.

**Definition 5.3** The *predicted worst-case end-to-end packet delay*, *PWPD* (in s), for the set of all the accepted real-time channels $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$ in the network is the longest end-to-end packet delay predicated in the real-time analysis, that is, $PWPD = max\ (T_{sdelay,i})$.

**Definition 5.4** The *average end-to-end packet delay*, *APD* (in s), for the set of all the accepted real-time channels $\Gamma = \{\tau_1, \tau_2, ...\tau_n\}$ in the network is the average end-to-end packet delay obtained by running packet-level simulation for some time, obtaining the end-to-end delay for each packet and then calculating the average .

**Definition 5.5** The *experienced longest end-to-end packet delay*, *EWPD* (in s), for the set of all the 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.

The different end-to-end delays are plotted in Figure. 5.3. The results reported in Figure 5.3(a) were conducted with every logical real-time channel having a period of 6 ms, a capacity 14920 bits and an end-to-end deadline of 6 ms. Figure 5.3(b) considers different parameters for real-time channels. The period is randomly chosen from the set {3 ms, 6 ms and 9 ms}, the capacity is 14920 bits and all the worst-case




**(a)** **(b)**

**Figure 5.3. End-to-end packet delay vs hard real-time traffic load. (a) Same parameters for all real-time channels. (b) Different parameters are selected for real-time channels.**

delays are 15 ms. Both graphs verify our feasibility analysis. The value of PWPD is always higher than the values of APD and EWPD, as we expected. The reason is that the worst-case does not happen all the time, although it must be considered in the admission control.

## 5.2 Comparision

This section aims a comparison study of our method with another switched Ethernet solution based on NC [Loeser and Haertig, 2004A] [Loeser and Haertig, 2004B]. The motivation for making the comparison is that the two approaches are based on the same network architecture (star topology) and similar traffic handling (FCFS queuing) but use different traffic models (periodic or so called (*r*, *b*)-model) and different analytical schemes (schedulability analysis and NC) to calculate the worst-case end-to-end delays. We will first present the model transformation from the periodic traffic model used in our analysis into the model used in NC, and then present the conceptual and experimental comparison.

### Model transformation

The traffic model used in NC analytical scheme is called (*r*, *b*)-model [Cruz 1991 A] [Cruz 1991 B], which satisfies certain regularity constraints and is defined as shown below.

**Definition 5.6** The arrival curve from node *k*, $\alpha_k$ has the form of (*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.

However, the (*r*, *b*)-model can not be used directly for periodic traffic. In order to compare, we need to transform our periodic model into the T-SPECs model. The model transformation is presented and proved in Theorem 5.1.

Our proof idea of Theorem 5.1 is as follows. First we will derive $r_k$ and $b_k$ according to their definitions, then we will prove that the traffic flow satisfies the (*r*, *b*)-model.

---

**Theorem 5.1**. Given 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. Than the traffic flow from source node *k* to the switch/port satisfies the (*r*, *b*)-model and the parameters of the corresponding (*r*, *b*)-model are: $r_k = \sum_{i=1}^{n} \frac{C_i}{T_{period,i}}, b_k = \sum_{i=1}^{n} C_i.$

**Proof.** According to the definition, $r_k$, the long term average rate, can be calculated as:

$$r_k = \sum_{i=1}^{n} \frac{C_i}{T_{period,i}} \tag{5.1}$$

Obviously, the maximum burstiness of the released traffic by channel $\tau_i$ is $C_i$, according to the definition of periodic real-time channel. Therefore, the aggregated flow has the maximum burstiness $\sum_{i=1}^{n} C_i$ , that is,

$$b_k = \sum_{i=1}^{n} C_i \,, \tag{5.2}$$

It is known that the cumulative incoming traffic from source node $k$ to the port, $a_k(t)$ is less than the cumulative workload of $\Gamma$ during time interval [0, $t$), that is,

$$a_k(t) \le \sum_{i=1}^{n} \left( \left\lfloor \frac{t}{T_{period,i}} \right\rfloor + 1 \right) C_i \,. \tag{5.3}$$

Consequently, we have:

$$a_k(t) \le \sum_{i=1}^{n} \left\lfloor \frac{t}{T_{period,i}} \right\rfloor C_i + \sum_{i=1}^{n} C_i \le \sum_{i=1}^{n} \frac{C_i}{T_{period,i}} t + \sum_{i=1}^{n} C_i = r_k t + \sum_{i=1}^{n} C_i = r_k t + b_k \,. \tag{5.4}$$

Equation 4.4 shows that the cumulative incoming traffic from source node $k$ to the port satisfies the ($r$, $b$)-model.

This concludes the proof of Theorem 5.1. □

With Theorem 5.1, the traffic released by a given periodic channel set can be transformed into the ($r$, $b$)-model that is used in NC.

### Conceptual comparison

In this section, we will compare our real-time analysis with another related work [Loeser and Haertig, 2004A] [Loeser and Haertig, 2004B]. For readers' convenience, we use the notations defined in Chapter 2 to explain their analysis. In that work, they use Network Calculus introduced by [Boudec and Thiran, 2001] to derive maximum queuing length and maximum queuing delay for switched Ethernet.

In that analysis, the delay and buffer bounds of a switch/port depend on the traffic arriving at the switch/port, the *arrival curve*, $\alpha$ and the availability of the switch/port to send that data, described by the *service curve* $\beta$.

The arrival curve $\alpha$ is the sum of the arrival curves of the traffic from each source node $k$, $\alpha_k$ . All $\alpha_k$ has the form satisfying

$$\alpha_k(t) = \min(Rnode_k t + T_{ef}, r_k t + b_k) \,. \tag{5.5}$$

The traffic flow arriving at a switch/port is described by its *arrival curve* $\alpha$, which is the sum of the arrival curves of the traffic from at the receive ports $\alpha_k$ , with $k$ denoting the traffic source.

The service curve of an Ethernet switch/port <$s$, $p$> is described by the following rate-latency function:

$$\beta(t) = \begin{cases} Rswi_{s,p}(t - t_{switch}) & (t \ge t_{switch}); \\ 0 & (t < t_{switch}) \end{cases}, \tag{5.6}$$

where $t_{switch}$ 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.

According to Boudec's result [Boudec and Thiran, 2001], $Dport_{s,p}$, the worst-case delay of a frame for the output port $p$ at the switch $s$ is

$$Dport_{s,p} = \sum_{k=1}^{Nnode} \frac{b_k}{Rswi_{s,p}} - g_{\max}(1 - \sum_{k=1}^{N} \frac{r_k}{Rswi_{s,p}}) + t_{switch}\ , \tag{5.7}$$

and $g_{max}$ is:

$$g_{\max} = \max_{k=1}^{Nnode}(\frac{b_k - T_{ef}}{Rswi_{s,p} - r_k})\ . \tag{5.8}$$

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

$$BS_{s,p} = Rswi_{s,p} Dport_{s,p} = \sum_{k=1}^{N} b_k - g_{\max}(Rswi_{s,p} - \sum_{k=1}^{N} r_k) + Rswi_{s,p} t_{switch}\ . \tag{5.9}$$

Our theoretical proofs presented in Chapter 4 show the tightness of our worst-case delay analysis for network components in single-switch network, while the maximum delay and buffer size derived by Equation 5.7 and 5.9 may not be tight, as stated in [Loeser and Haertig, 2004A]. The reasons are discussed as follows.

First, we consider the fact that the output traffic of an FCFS queue is in general less bursty than the input traffic, because the outgoing flow is shaped by the physical link [Boudec and Thiran, 2001]. This observation is crucial to avoid over-estimation of the worst-case delay at the switch ports and we take the advantage in our anlysis by concerning this fact. In contrast, NC analysis involves inefficiency in finding performance bounds by iteratively applying output burst bounds hop-by hop.

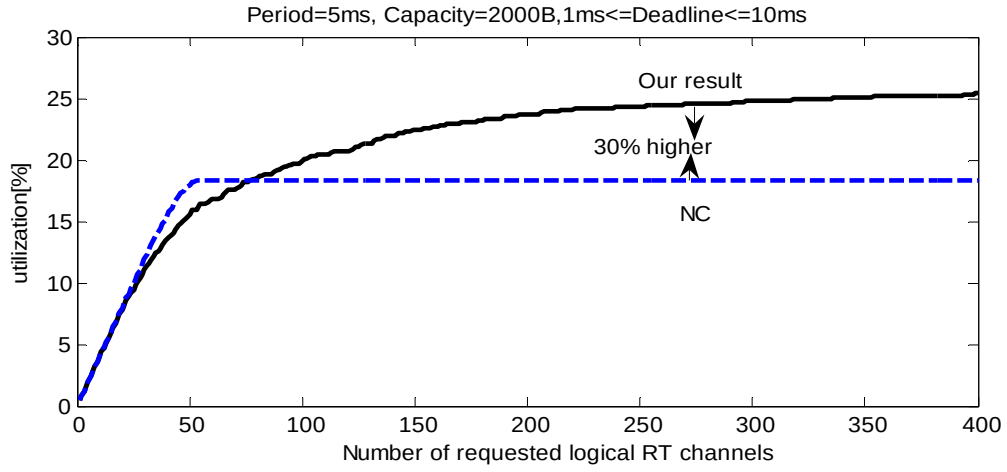
Moreover, in our proofs in Chapter 3, we know that $\alpha_k(t) \le \min(R_k t + T_{ef}, r_k t + b_k)$ holds for a synchronous periodic real-time channel set, because the incoming link to a switch/port may not always be busy. However, to transform the periodic model to the ($r$, $b$)-model, we will lose some information of the traffic model. Specifically, the traffic assumption in NC, $\alpha_k(t) = \min(R_k t + T_{ef}, r_k t + b_k)$ brings pessimism for analyzing a synchronous periodic channel set. Consequently, the maximum delay and buffer size derived by Equation 5.7 and 5.9 may not be tight. In contrast, we are able to use an algorithm (Algorithm 3.1) to keep track of the exact amount of incoming traffic volume to a switch/port, without any overestimation in our analysis.

### Simulation comparison

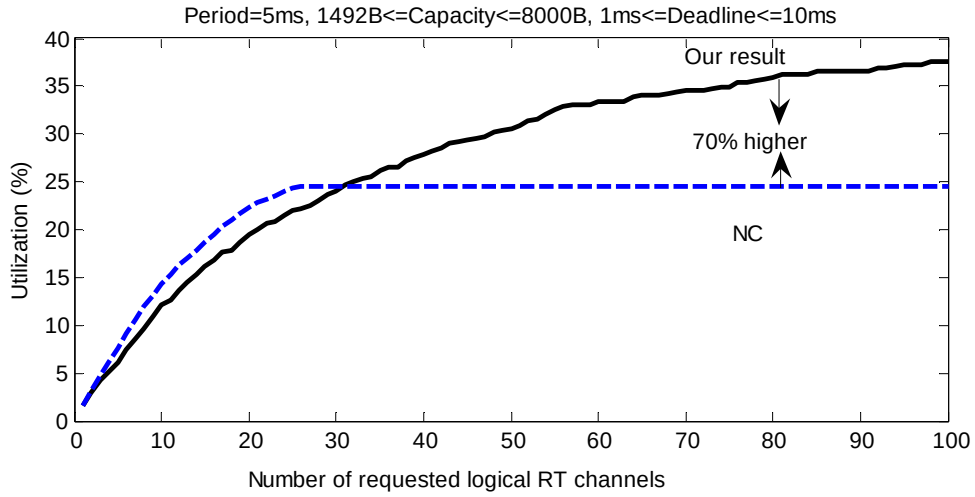
Although the conceptual comparison shows that the worst-case delay for the components in a single-switch network derived by our analysis is tight, we are aware of the fact that neither our end-to-end worst-case delay estimation or the NC estimation is always tight. Therefore, we have conducted simulations to compare our analysis with NC for Ethernet with single switch.

To compare two real-time analysis schemes, it is important to see the allowed amount of HRT traffic in the network, under the condition of guaranteeing the worst-case delays for all real-time channels. More allowed real-time traffic means less overestimation in the worst-case delay analysis. Hence, in our experimental comparison, we observe the relation between the utilization of the physical channels and the total number of requested real-time channels, as illustrated in Figure 5.4. There are 8 nodes in the network.

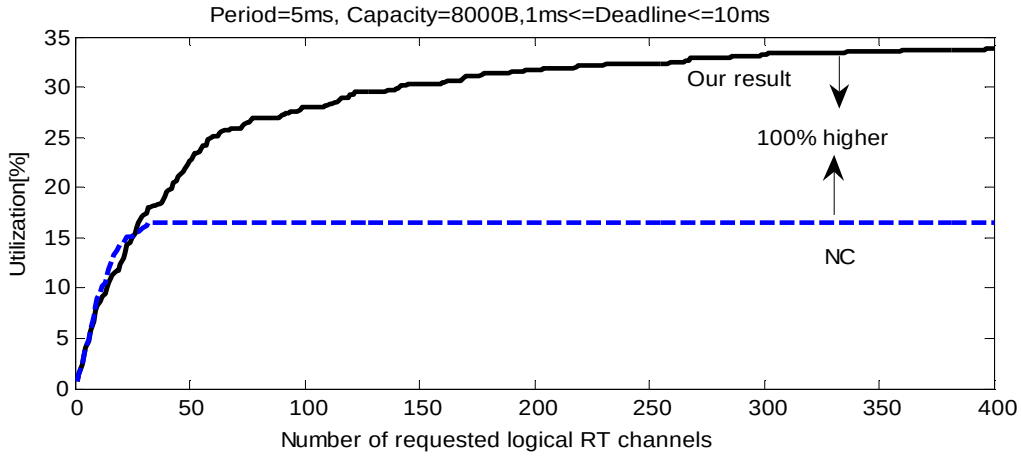
In Figure 5.4(a), the period is set to 5 ms and the capacity is set to 2000 bits for all the logical real-time channels, while the end-to-end deadlines are randomly generated between 1 ms and 10 ms. It can be observed that the utilization steadily increases when the traffic load is increased. Later, it keeps the same




**(a)**




**(b)**




**(c)**

**Figure 5.4. Utilization comparison. (a) Small capacities. (b) Large capacities. (c) Variant capacities.**

value because it is difficult to accept a new logical real-time channel under the high workload in the system. Note that a higher utilization (about 30% higher) can be gained with our solution.

If we increase the capacities for all the logical real-time channels to 8000 bytes, we obtain the results shown in Figure 5.4(b). Note that a quite higher utilization (more than 100% higher) can be gained with our solution.

In the first two simulations, all channels have the same capacity. For the third simulation set-up shown in Figure 5.4(c), the capacities are randomly selected between 1492 bytes (one full-sized Ethernet frame) and 8000 bytes. Figure 6(c) shows, under this traffic generation, that our feasibility analysis still gains higher utilization (70% higher) than NC-based analysis.

It can be observed from Figure 5.4 that our analysis achieves higher utilization than the NC analysis in many cases, although our analysis and the NC analysis are effective in the sense of guaranteeing the traffic meeting their deadlines. It is seen that the larger the capacities are chosen, the higher utilization is gained by our analysis. The reason for this is, i.e., the traffic smoothing transmission characteristics are considered in our analysis, thus allowing a less pessimistic delay. In contrast, in the Network Calculus analysis, the large capacities mean large burst values and large burst values lead to longer worst-case delays.

The following conclusions can be drawn from the above simulations and comparison studies. i) Our analysis is effective. ii) Our analysis gives tighter estimation than NC in majority of the cases.

## 5.3 Conclusion

In this report, we contribute with the real-time analysis for periodic hard real-time traffic in switched Ethernet networks only utilizing FCFS-queuing. 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. Although the end-to-end analysis for single-switch network may not be tight, our simulation and comparison study show that our analysis is less pessimistic than NC in the majority of the cases. For networks with multiple switches, we have developed an iterative algorithm to calculate the worst-case delay and buffer bound, which might give pessimistic results. 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.

One of our future works is to evaluate our analysis for networks with multiple switches by simulation study.

# References


[Baruah et al. 1990] 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," *Real-time systems*, 2, 1990.

[Bello and Mirabella, 2004] L. L. Bello and O. Mirabella, "An approach to reduce application-to-application latency in a real-time distributed architecture," *Proc. of the 3rd International Workshop on Real-time Networks (RTN'04) in conjuction with 16th Euromicro International Conference on Real-time Systems (ECRTS'04)*, Catania, Italy, June 29, 2004, pp. 61−64.

[Boudec and Thiran, 2001] J. Y. Le Boudec and P. Thiran, Network Calculus, *Springer Verlag Lecture Notes in Computer Science*, vol. 2050, July 2001.

[Cruz 1991 A] R. L. Cruz, "A calculus for network delay, I: network elements in isolation," *IEEE Transaction on Information Theory*, vol. 37, no. 1, pp. 114−131, Jan. 1991.

[Cruz 1991 B] R. L. Cruz, "A calculus for network delay, II: network analysis," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 132−141, Jan. 1991.

[IEEE 1998] IEEE 802.1D, *Information Technology – Telecommunications and Information Exchange between Systems-Local and Metropolitan Area Networks – Common Specification – Media Access Control (MAC) Bridges*, Std, Piscataway, NJ, USA, 1998.

[IEEE 2003] IEEE Std 802.1Q, *Virtual Bridged Local Area Networks*, Std, Piscataway, NJ, USA, 2003.

[Loeser and Haertig, 2004A] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched Ethernet," *the 16th Euromicro International Conference on Real-time Systems (ECRTS'2004)*, Catania, Italy, June 30−July 2, 2004, pp. 13−22.

[Loeser and Haertig, 2004B] J. Loeser and H. Haertig, "Using switched Ethernet for hard real-time communication," *proc. of the International Conference on Parallel Computing in Electronic Engineering (PARELEC'04)*, Dresden, Germany, Sept 7−10, 2004, pp. 349−353.

[Skeie et al. 2002] T. Skeie, S. Johannessen, and O. Holmeide, "The road to an end-to-end deterministic Ethernet,", *Proc. of 4th International Workshop on Factory Communication Systems (WFCS'2002)*, Västerås, Sweden, Aug. 28−30, 2002, pp. 3−9.

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

[Weber et al. 1999] R. Weber, D. Santos, R. Bianchini, and C. L. Amorim, "A survey of messaging software issues and systems for Myrinet-based cluster", *Parallel and Distributed Computing Practives, special issue on High-Performance Computing on Clusters*,vol. 2, no. 2, June 1999.