Revision of a Non-Preemptive EDF Packet Scheduling Algorithm

Dai Bui

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2009-50
http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-50.html

April 24, 2009

Acknowledgement

This work was supported in part by the Center for Hybrid and Embedded Software Systems (CHESS) at UC Berkeley, which receives support from the National Science Foundation (NSF awards #0720882 (CSR-EHS: PRET) and #0720841 (CSR-CPS)), the U. S. Army Research Office (ARO #W911NF-07-2-0019), the U. S. Air Force Office of Scientific Research (MURI #FA9550-06-0312), the Air Force Research Lab (AFRL), the State of California Micro Program, and the following companies: Agilent, Bosch, Lockheed-Martin, National Instruments, and Toyota.

This work is also supported by Vietnam Education Foundation (VEF).

Revision of a Non-Preemptive EDF Packet Scheduling Algorithm

Dai Bui
Department of Electrical Engineering & Computer Science
University of California, Berkeley

Abstract— **One issue with today's Internet is that it only supports best-effort service; hence Internet users often experience unpredicted delay. Deployment of new real-time, highly reliable applications that require fixed delay bound on packets, such as remote surgery, become very difficult. Communication environment that can assert a strict delay bound will help estimate the worst-case execution time of distributed real-time applications.**

This paper revises a flaw in estimation of delay bound and an incorrect proof for of a non-preemptive Earliest Deadline First (EDF) packet scheduling algorithm used in packet scheduling for implementing real-time communication services in packet-switching network described in [1,2,3], which is one of few seminal works on guaranteed service in packet-switching network. We discuss the found flaw then correct the error and provide a substitute proof for this new correction.

I. Introduction

Real-time communication and quality of service of networking have been an active research topic for decades [4,5,6,7,1,2,3, 8, 9]. Real-time communication plays an important role in supporting several real-time applications such as remote surgery, online conference, Internet game and multimedia applications. However, today's Internet has difficulty supporting those applications due to its unpredictable delay.

Packets in packet-switching network from different sources can be routed to go through shared communication links. This can easily result in congested condition that makes packets dropped or experience unpredicted delay. Thus, a scheme for establishing real-time communication channels and scheduling packets to meet their end-to-end delay requirements becomes essential [4, 5, 6, 7].

Hard real-time systems often require very strict delay bound, if one task misses a deadline, a severe accident might happen. Estimating wrong worst-case execution time, thus, can cause severe effects. In 1990s, Ferrari et al. published a series of papers [1, 2, 3] proposing a model for guaranteeing delays in packet-switching network using EDF [10]. Those papers use the term Earliest Due Date (EDD) for EDF, however, we use EDF for it is more popular. Those papers also show a method for estimating of delay bounds for packets at each node when using EDF scheduling algorithm, from the delay bounds at each node, the end-to-end delays of real-time packets can be computed. However, those papers contain a severe flaw in incorrect delay bound estimation and the proof for the estimation [1] is not satisfactory.

In this paper, we show the flaw, discuss the reason leading to the flaw and propose a solution to correct the flaw. This non-preemptive EDF scheduling algorithm is not only useful for packet scheduling, but also applicable to several other applications like I/O scheduling, multitasking scheduling since it incurs lower scheduling overhead than preemptive EDF[1]. This paper is organized as follows: First, we summarize the main concepts of the papers [1, 2, 3] in Section II. Then we will show a counter example and discuss the source of error of the original proof by Ferrari et al. in Section III-A. Finally, we correct the flaw with new delay bound estimation with a substitute proof in Section III-B.

[1]High scheduling overhead renders preemptive EDF less implemented in practice

II. Problem Summary

This section is devoted to summarizing the main ideas and algorithms of the work on real-time communication services in packet-switching network in [1, 2, 3].

A. Packet-Switching Network

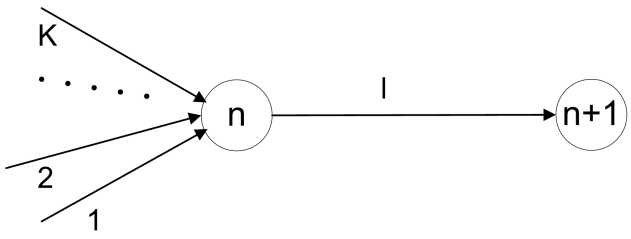

Fig. 1
Model of network.

Figure 1 shows the model of a packet-switching network, in which K channels from 1 to K all going through node n to link l to node $n+1$. The packets of those channels are multiplexed to going through link l. Packets will interact with each other resulting in delay of packets at node n since some packets have to wait for other packets to finish.

This is a *store-forward* network, in which a node only forwards a packet when it receives the entire packet. A packet is said to *depart* a node when it is *completely* forwarded to next node. A packet is said to arrive at a node when it is completely received by the node.

When a real-time channel is set up in a network, it has to declare its end-to-end delay requirement. The end-to-end delay requirement is then broken into local delays at each node on the path from the source node to the destination node.

We define the local delay of a channel i at a node n as $d_{i,n}$. Then the end-to-end delay $D_i \geq \sum_{j=1}^{N} d_{i,j}$ where the path is from source node 1 to destination node N.

B. Real-time Channels

In [1], each real-time channel has a declared traffic characteristics and performance requirements at the time of channel establishment. Hence, each real-time channel will be associated with the following parameters:

- $x_{min,i}$ is the minimum packet interval time between two successive packets of a real-time channel i that the channel commits.
- $s_{max,i}$ is the maximum packet size of channel i.
- $t_{i,n}$ is the maximum service time for a packet of channel i at node n.

We focus on *deterministic* channels that have packets of each real-time channel can never have delay greater than the delay end-to-end bound D. Packets of deterministic channels require strict a delay bound.

From those parameters, the following algorithms will verify if it is possible to establish a new real-time channel on an existing packet-switching network.

C. Original Algorithms

C.1 Link Utilization:

Suppose that the link l is from node n to node $n+1$. The maximum utilization of each real-time channel i over any link is computed as:

$$U_i = \frac{t_{i,n}}{x_{min,i}} \tag{1}$$

The total utilization of a link can never exceed 100% so the total utilization of all channels over a link l cannot exceed 1:

$$\sum_{i \in l} U_i = \sum_{f \in l} \frac{t_{i,n}}{x_{min,i}} \leq 1 \tag{2}$$

Whenever a link wants to admit a new channel through it, it has to test if its total utilization including the utilization of the new channel exceeds 1 or not. If the link's total utilization will exceed 1 when the link admits a new channel, it rejects the new channel.

C.2 Packet Scheduling:

The EDF algorithm is used for packet scheduling at each node. Suppose that at each intermediate node j on the path from source to destination, the EDF scheduling algorithm can guarantee a bounded delay $d_{i,j}$ assigned to each packet of channel i. The deadline for a packet p of a channel i at each node is computed as:

$$dl_{p,n} = g_p + \sum_{j=1}^{n} d_{i,j} \tag{3}$$

Where g_p is the time the packet p departs the source node. The above deadline is used for the deadline-based algorithm, in which the packet with closest deadline will be selected to forward first.

C.3 Delay Bound Test

Suppose that there are K channels going through node n, they are then divided into two sets:

- U: the set of all channels i whose the assigned delay bound $d_{i,n} < \sum_{j=1}^{K} t_{j,n}$.
- V: the set of all channels i whose the assigned delay bound $d_{i,n} \geq \sum_{j=1}^{K} t_{j,n}$.

$$U = \left\{ i \mid i = 1, \ldots, u; d_{i,n} < \sum_{j=1}^{K} t_{j,n} \right\} \tag{4}$$

$$V = \left\{ k \mid k = u+1, \ldots, K; d_{k,n} \geq \sum_{j=1}^{K} t_{j,n} \right\} \tag{5}$$

The following assumption is used to derive the delay bound for packets of each channel.

Scheduling assumption 1: Without loss of generality, we assume that U has u channels numbered from 1 through u in the order in which their packets would be scheduled by the EDF algorithm if they arrive *at the same instant*: channel 1 will be the one whose packet would be shipped first, channel u the one whose packet would be shipped last.

We quote the proof of Theorem 1 by Ferrari et al. described and proved in [1] here so that readers can take a close look at it. However, readers can skip the proof temporarily and move to the next section on discussion of the algorithm flaw.

If the condition (6) holds, then the Theorem 1 shows the scheduling feasibility to guarantee the delay bound in (7).

$$x_{min,i} \geq \sum_{j=1}^{K} t_{j,n}, (i = 1, \ldots, K) \tag{6}$$

The condition (6) basically means that when estimating the delay bound for a packet at a node, each channel has at most one packet.

Theorem 1: . [Ferrari et al] Scheduler saturation is impossible if and only if:

$$d_{i,n} \geq \sum_{j=1}^{i} t_{j,n} + T \tag{7}$$

Where T is the largest of the service times of packets of any channels other than channels in U.

Proof: Since we have *excluded the possibility of node saturation* even in the worst case (i.e., when all the channels we are considering are carrying packets at their maximum rates $1/x_{min,j}$), *there cannot be any buildup of queues in time; we can therefore assume that the node is empty when we start examining arrival patterns, and call time 0 the instant at which the first packet arrives at the node*. Arrival times can be assumed to be arbitrary, since channels are supposed to be independent of each other; the dependencies that may result from the sharing of an input link by two or more channels can only improve the situation, as this sharing serializes arrivals on those channels at the node. Packets arriving on a given channel after the first we consider are not independent of that first: the second packet on channel j will in the worst case arrive $x_{min,j}$ time units after the

first packet on the same channel. Because of assumptions (6), no deadline for a subsequent packet will fall within the interval between time 0 and time $\sum t = \sum_{j=1}^{K} t_{j,n}$. Subsequent packets can therefore be ignored in this proof.

- *Only if* part. We prove that, if condition (7) is not satisfied for some i , there is at least one arrival pattern that causes scheduler saturation. Let a packet with a service time T arrive at time 0, and packets from all other channels numbered 1 through u arrive at time $0+$. Then, the packet on channel i will be completely shipped only at time $\sum t + T$, which is *greater* than its deadline $d_{i,n}$.
- *if* part. We prove that, if conditions (7) are all satisfied, there cannot be scheduler saturation. Let the packet on channel i arrive at time 0; in this case, its latest possible departure time is $\sum_{j=1}^{i} t_{j,n} < d_{i,n}$. If it arrives at $0+$, the latest departure time is $\sum_{j=1}^{i} t_{j,n} + T \leq d_{j,n}$. With an arrival time between $0+$ and T, the maximum time spent by the packet in the node is less than $d_{i,n}$. Arriving at $T+$, this maximum time is $\sum_{j=1}^{i} t_{j,n} < d_{i,n}$, and later arrivals yield even smaller maximum times. □

The above proof by Ferrari et al. is unsatisfactory and the delay bound estimated in (7) is incorrect. We will discuss in the next section.

III. Problem Revision

In this section, we show the flaw of the Theorem 1 by a counter example. We then discuss the source of the flaw and correct the flaw.

A. Flaw Discussion

The Theorem 1 not only estimates packet delay bound incorrectly, but also only shows that there is some scheduling order that can satisfy the delay bound and does *not* show that the EDF scheduling algorithm with the above deadline assignments can satisfy the delay in (7). There is no clear connection between the proof and EDF. Note that EDF is optimal if it is preemptive, however, packet scheduling, in this case, is non-preemptive so schedulability does not imply that EDF can achieve that.

A.1 Counter Example

Without difficulty, we can come up with a counter example for the original Theorem 1 in [1] by showing that it cannot guarantee the delay bound computed by (7).

In equation (7), T is defined to be "the largest of the service times of packets that may traverse the node but *do not travel on channels in the set U*". This definition makes the delay bound estimation (7) wrong because, suppose that a packet p_k of channel $k > i$ in the set of U that has service time $t_{k,n} = d_{i,n} + 10$. Note that we can set $t_{k,n} = d_{i,n} + 10$ because $k > i$ and channel k is in the set U, in case $k < i$, we cannot do so since $d_{i,n} \geq \sum_{j=1}^{i} t_{j,n} + T \geq t_{k,n}$. Let π be the start of sending packet p_k, and p_i packet of channel i arrives at time $\pi + 1$. Then the completion of sending the packet p_k is $\pi + t_{k,n}$ so the packet p_i of channel i has to wait at node n for at least $(\pi + t_{k,n}) - (\pi + 1)$. However, $(\pi + t_{k,n}) - (\pi + 1) = t_{k,n} - 1 = (d_{i,n} + 10) - 1 = d_{i,n} + 9 > d_{i,n}$, then the packet p_i of channel i has to wait more than its delay bound, therefore it will miss its deadline.

A.2 Proof Discussion

We can correct the above issue by setting T to the largest possible service time of packets of any channel $j > i$, which means that $T = \sup_{j>i}\{t_{j,n}\}$. Here we define $\sup\{\emptyset\} = 0$ then (7) becomes:

$$d_{i,n} \geq \sum_{j=1}^{i} t_{j,n} + \sup_{j>i}\{t_{j,n}\} \tag{8}$$

Note that now we cannot set $t_{k,n} = d_{i,n} + 10$ any more because $d_{i,n} \geq t_{k,n}\ \forall 1 \leq i, k \leq K$.

However, even though after correcting T like that, the equation (8) is not automatically true, and we cannot apply the proof by Ferrari et al. as it does not reflect all possible situations.

We are not convinced by the proof of the Theorem 1 in [1] that was quoted in Section II-C.3 because, in the proof, Ferrari et al. have made an assumption that is not really true: "Since we have excluded the possibility of node saturation even in the worst case (i.e., when all the channels we are considering are carrying packets at their maximum rates $1/x_{min,j}$), there cannot be any buildup of queues in time; we can therefore assume that the node is empty when we start examining arrival patterns, and call time 0 the instant at which the first packet arrives at the node". There cannot not be any buildup of queues in time, however, queues are still *temporarily* built up since packets still have to wait to be serviced, therefore we cannot "assume that the node is empty when we start examining arrival patterns".

This wrong assumption leads to a wrong proof since it does not reflect all possible situations of packets. As we can see from the proof, worst case delay $\sum_{j=1}^{i} t_{j,n} + T$ of a packet p_i of channel i, only accounts for service time of packets of channels from 1 to i, as a packet of any channel $j < i$ arrives at the same instant with p_i, it will be forwarded before p_i, and *only one* packet of *another channel* that is being sent [2]. This estimation is not automatically true, since it is possible there exist two or more packets of two or more channels other than channels from 1 to i, those packets are queueing in the node and have deadlines closer than the deadline of p_i so they are should be forwarded earlier.

In proof of Theorem 1, Ferrari et al. have implicitly assumed or only considered that packets of channels in the set of U all arrive at the same instant. With this implicit assumption, the authors have neglected the effect of some packets of channels in the set U that arrive early can affect the delays of packets of other channels in the set U. This wrong implicit assumption leads to wrong estimation of bounded delay as we saw above.

Furthermore, the proof of the Theorem 1 only shows that *scheduler saturation is impossible*. This means that there exists a scheduling order of packets that the delay bound at each node can be satisfied, however, the proof does not show that the EDF scheduling algorithm can do that. EDF scheduling algorithm is optimal if it is *preemptive* [11], but packet scheduling in [1,2,3] is non-preemptive. The proof of the Theorem 1 does not show any relationship between the scheduling feasibility and the EDF scheduler.

[2] Note that $t_{i,n}$ is added up to the service time of p_i since the delay is from one packet's arrival time till the completion of sending the packet

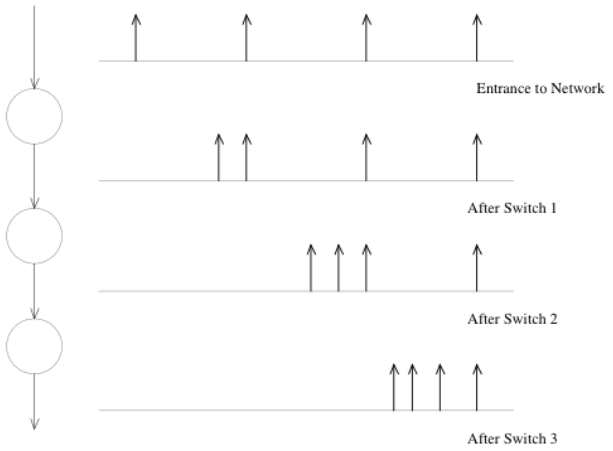

Fig. 2
TRAFFIC PATTERN DISTORTION

B. Flaw Correction

B.1 Delay Bound Estimation

We use the condition (10) to easy the process of estimating the delay bound. However, we do not use the Scheduling assumption 1 that makes assumptions for only channels in the set of U. We realize that there is no clear reason to split the K channels going through node n into two sets U and V as in Section II-C.3. Instead, we make a scheduling assumption for all K channels.

Scheduling assumption 2: Without loss of generality, we assume that K channels numbered from 1 through K in the order in which their packets would be scheduled by the EDF algorithm if they arrive *at the same instant*: channel 1 will be the one whose packet would be shipped first, channel K the one whose packet would be shipped last.

With the above scheduling assumption, the delay bound of packets of a channel i at node n is as follows:

$$\hat{d}_{i,n} = \sum_{j=1}^{i} t_{j,n} + \sup_{j>i}\{t_{j,n}\} \tag{9}$$

This delay bound means that a packet of channel i has to wait at most $\hat{d}_{i,n}$ at node n. This delay bound is proved in the Theorem 2.

B.2 Deadline Assignment and Scheduling

We modify the condition (6) a little bit into the following condition:

$$x_{min,i} > \sum_{j=1}^{K} t_{j,n} = \tau, (i = 1, \ldots, K) \tag{10}$$

The difference between (6) and (10) is that $x_{min.i}$ is *strictly* greater than $\sum_{j=1}^{K} t_{j,n} = \tau$.

Real arrival times of packets are distorted by packet scheduling algorithm at previous nodes as in Figure 2. This distortion complicates the process of scheduling packets at downstream nodes since the real interval between two successive packets at an intermediate node like node n is different from the original committed interval of each channel at source node.

To solve this issue, and we use a similar jitter control technique as in [2], in which, if a packet p arrives early at a node, it will be held in a waiting queue until a logical arrival time before it is eligible for scheduling. We define the *logical* arrival time of each packet p_i of channel i at node n as:

$$a_{p_i,n} = g_{p_i} + \sum_{j=1}^{n-1} \hat{d}_{i,j} \tag{11}$$

We can see that now the traffic pattern of logical arrival times of packets in one channel is the same as at the source node of the channel because the constant $\sum_{j=1}^{n-1} d_{i,j}$ is filtered out when we subtract the logical arrival times of two successive packets. We call packet p *logically* arrive at node n at time $a_{p,n}$. From now on, if we say that a packet *arrives* at a node, it means that the packet *logically arrives* at a node.

We redefine the deadline of a packet p of channel i at a node n as follows:

$$dl_{p\in i,n} = a_{p\in i,n} + \hat{d}_{i,n} \tag{12}$$

$$\Rightarrow dl_{p\in i,n} = a_{p\in i,n} + \sum_{j=1}^{i} t_{j,n} + \sup_{j>i}\{t_{j,n}\} \tag{13}$$

Now the EDF scheduler uses the above deadline assignment for each packet to choose the packet with closest deadline from all logically arrived packets.

Lemma 1: With condition (10) and arrival times for packets are computed as in (11), each channel from 1 to K has at most one packet that has logical arrival time within any interval $[t, t+\tau]$.

Proof: Suppose that p'_k, p_k are two successive packets of channel k going through the node n, and packet p_k has arrival time $a_{p_k,n}$ within $[t, t+\tau]$:

$$t \le a_{p_k,n} \le t + \tau \tag{14}$$

Moreover, from (11) and $x_{min,k}$ is the minimum interval between the departures of two successive packets at source $g_{p_k} - g_{p'_k} \ge x_{min,k}$, we have:

$$a_{p_k} - a_{p'_k} = g_{p_k} - g_{p'_k} \ge x_{min,k} > \tau \tag{15}$$

From(14) and (15), we have $a_{p'_k} < a_{p_k} - \tau \le (t+\tau) - \tau = t$, so the logical arrival time $a_{p'_k}$ of packet p'_k is not within the interval $[t, t+\tau]$. □

Lemma 2: With condition (10), the node n is empty at some point, i.e. there is no queueing packet, within any interval $[t, t+\tau]$.

Proof: We prove this by contradiction. The node n is empty at the beginning. Suppose that $[t, t+\tau]$ is the first interval that the node n is not empty at any time within the interval, so we know that node n is empty just before t. Due to Lemma 1, during $[t, t+\tau]$, each channel has at most one packet logically arrives, therefore the total service time of packets logically arriving during $[t, t+\tau]$ is at most $\sum_{j=1}^{K} t_{j,n} = \tau$. We know that node n

is not empty during $[t, t+\tau]$ so during $[t, t+\tau]$, the node n always busy sending packets. However, the interval $[t, t+\tau)$ has total service time τ, which is greater than or equal to the total service time of all packets that logically arrive during $[t, t+\tau]$. Moreover, the node n is empty before t so no traffic remains at t, therefore node n must be empty at $t+\tau$ because it has sent all its traffic during $[t, t+\tau)$. This results in contradiction. So node n must be empty at some point within any interval $[t, t+\tau]$. □

This lemma means that if the utilization of a link is smaller than 1, then during some interval, the arrival rate is smaller than the departure rate then the node should be empty at some point of time within the interval.

Lemma 3: If $y > x$ if and only if $\hat{d}_{y,n} > \hat{d}_{x,n}$. In other words, $\hat{d}_{j,n}$ is a strictly increasing function of j.

Proof: If part: since $x < y$ then $\sum_{j=x+1}^{y} t_{j,n} + \sup_{j>y}\{t_{j,n}\}$ includes $\sup_{j>x}\{t_{j,n}\}$ then $\sum_{j=x+1}^{y} t_{j,n} + \sup_{j>y}\{t_{j,n}\} > \sup_{j>x}\{t_{j,n}\}$. Therefore:

$$\Rightarrow \sum_{j=1}^{x} t_{j,n} + \Big(\sum_{j=x+1}^{y} t_{j,n} + \sup_{j>y}\{t_{j,n}\}\Big) > \sum_{j=1}^{x} t_{j,n} + \sup_{j>x}\{t_{j,n}\}$$

$$\Rightarrow \sum_{j=1}^{y} t_{j,n} + \sup_{j>y}\{t_{j,n}\} > \sum_{j=1}^{x} t_{j,n} + \sup_{j>x}\{t_{j,n}\}$$

$$\Rightarrow \hat{d}_{y,n} > \hat{d}_{x,n}$$

Only if part: in case $\hat{d}_{y,n} > \hat{d}_{x,n}$ then $x \geq y$ is impossible since
- If $x = y \Rightarrow \hat{d}_{y,n} = \hat{d}_{x,n}$, which is different from $\hat{d}_{y,n} > \hat{d}_{x,n}$.
- If $x > y$ then from the *If* part, we have $\hat{d}_{y,n} < \hat{d}_{x,n}$ which is different from $\hat{d}_{y,n} > \hat{d}_{x,n}$.

So $\hat{d}_{y,n} > \hat{d}_{x,n} \Leftrightarrow y > x$. □

B.3 New Theorem

In this section, we provide a substitute theorem for the original incorrect theorem by Ferrari et al. [1]. This theorem can be generalized to task scheduling in single processor.

Theorem 2: With the Scheduling assumption 2, if the condition (10) holds and the deadlines for packets of a channel are assigned as in (13), then the non-preemptive EDF scheduling algorithm will guarantee the delay bound $\hat{d}_{i,n}$ for each packet p_i of channel i at node n computed in (9).

Proof: Please note that, (10) implies that the utilization condition (2) holds.

Suppose that p_i of channel i is the first packet that can miss its deadline at node n. p_i arrives at node n on time at time $a_{p_i,n}$ computed from (11). Then the deadline for finish sending this packet is $dl_{p_i,n} = a_{p_i,n} + \hat{d}_{i,n}$.

We set $\pi_{p_i} = dl_{p_i,n} - \tau$. Let us call π'_{p_i} the most recent time that the node n is empty i.e. there is no outstanding packet of any channel. From Lemma 2, we know that the node n is empty at some point within the interval $[\pi_{p_i}, dl_{p_i,n}]$, then $\pi_{p_i} \leq \pi'_{p_i}$.

We only consider the interval $(\pi'_{p_i}, dl_{p_i,n}]$. We do not need to care about any packets before π'_{p_i} since the node n is empty at π'_{p_i}. The length of the interval $(\pi'_{p_i}, dl_{p_i,n}]$ is:

$$dl_{p_i,n} - \pi'_{p_i} \leq dl_{p_i,n} - \pi_{p_i} = \tau \tag{16}$$

Figure 3 shows the time-line that would be beneficial for readers to understand the following parts. Let ω be the last time within $(\pi'_{p_i}, dl_{p_i,n}]$ that a packet p_l of some channel l departs the node n such that the packet's logical arrival time is within $(\pi_{p_i}, dl_{p_i,n}]$ and its deadline:

$$dl_{p_l,n} > dl_{p_i,n} \tag{17}$$

We now have two cases:

a) If such a packet p_l does not exist. Let k be the channel with largest value k that has packet with deadline within $(\pi'_{p_i}, dl_{p_i,n}]$ and still remains at the node in that interval, and let the packet of channel k be p_k. So we have:

$$dl_{p_k,n} \leq dl_{p_i,n} \tag{18}$$

We see that $k \geq i$ because channel i has a packet p_i with deadline within $(\pi'_{p_i}, dl_{p_i,n}]$ and still remains at the node during the interval, since otherwise, the packet is forwarded before its deadline. Node n is empty at π'_{p_i} so we know that the packet p_k must arrive after π'_{p_i} because it remains at the node in the interval $(\pi'_{p_i}, dl_{p_i,n}]$:

$$\pi'_{p_i} < a_{p_k,n} \tag{19}$$

From (18)(19)

$$dl_{p_i,n} - \pi'_{p_i} > dl_{p_k,n} - a_{p_k,n} = \sum_{j=1}^{k} t_{j,n} + \sup_{j>k}\{t_{j,n}\} \tag{20}$$

Now we have:
- During $(\pi'_{p_i}, dl_{p_i,n}]$, no packet of channels other than channels from 1 to k is forwarded since k is the channel that k is largest that has a packet with deadline within $(\pi'_{p_i}, dl_{p_i,n}]$ and no packet with deadline after $dl_{p_i,n}$ is forwarded during $(\pi'_{p_i}, dl_{p_i,n}]$.
- No channel from 1 to k has more than one packet during $(\pi'_{p_i}, dl_{p_i,n}]$ since from Lemma 1, no channel has more than one packet during any interval with length τ and the length interval $(\pi'_{p_i}, dl_{p_i,n}]$ is smaller than or equal to τ, and the node n is empty at π'_{p_i}.
- Node n is not empty during $(\pi'_{p_i}, dl_{p_i,n}]$, so the node n must be busy sending during $(\pi'_{p_i}, dl_{p_i,n}]$.
- Equation (19) shows that the length of $(\pi'_{p_i}, dl_{p_i,n}]$ is enough to completely forward one packet for each channel from 1 to k.

The above reasons lead to a conclusion that: each channel from 1 to k has one packet completely forwarded during $(\pi'_{p_i}, dl_{p_i,n}]$. Furthermore, channel i has at most one packet during $(\pi'_{p_i}, dl_{p_i,n}]$ due to Lemma 1[3], so the packet of channel i completely forwarded during $(\pi'_{p_i}, dl_{p_i,n}]$ must be p_i. Thus p_i does not miss its deadline $dl_{p_i,n}$. The equality of equation (9) happens when $i = k$.

Intuitively, node n has served an amount q of traffic within an interval, and a set S of packets that has at most an amount q of traffic is the only set of packets that n can serve, therefore n must have served all packets in S. In addition, set S contains packet p_i, thus the packet p_i must be forwarded within the interval.

[3] We only consider packets that logically arrives

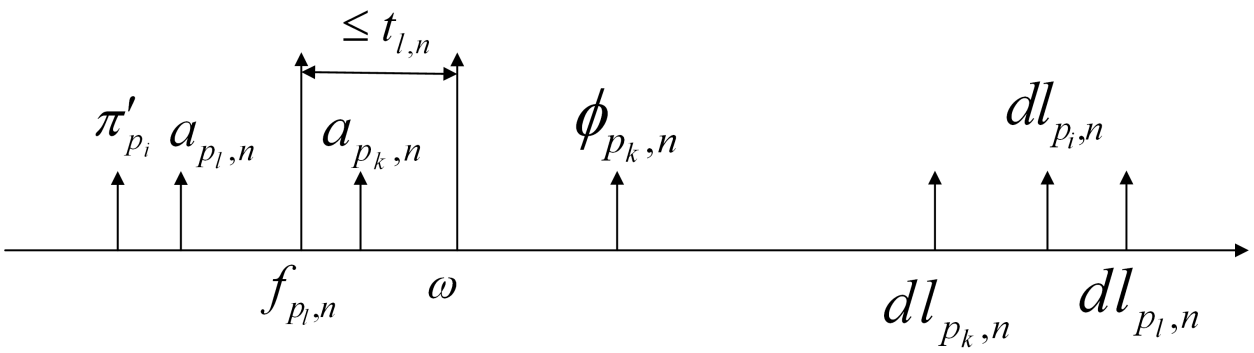

Fig. 3
TIME-LINE FOR PACKETS.

b) If that packet p_l exists. Let k be the channel with the *largest* value k that still has a packet not forwarded within the interval $(\omega, dl_{p_i,n}]$ and such that the packet's deadline is smaller than or equal to $dl_{p_i,n}$. Let the packet of channel k be p_k then:

$$dl_{p_k,n} \leq dl_{p_i,n} \tag{21}$$

We can see that

$$k \geq i \tag{22}$$

since otherwise if $k < i$, we know k is the channel that k is largest and that channel has a packet that still remains at node n within the interval $(\omega, dl_{p_i,n}]$ or such channel k does not exist then channel i does not have any packet within $(\omega, dl_{p_i,n}]$ so the packet p_i must have been forwarded before deadline $dl_{p_i,n}$ since it is only packet of channel i due to Lemma 1.

p_l departs node n at ω and p_k still remains at node n at some point within the interval $(\omega, dl_{p_i,n}]$, so we know that p_l is forwarded before p_k. However, from (17) and (21), we have:

$$dl_{p_k,n} > dl_{p_l,n} \tag{23}$$

This means that p_k must arrive at node n after the *start* of forwarding p_l, otherwise, EDF scheduler would select p_k to forward instead of p_l because p_k has earlier deadline. Let $f_{p_l,n}$ be the time of start of forwarding the packet p_l at node n, then:

$$f_{p_l,n} < a_{p,k} \tag{24}$$

It is obvious that the packet p_l must be started to forward after it has arrived, then:

$$a_{p_l,n} \leq f_{p_l,n} \tag{25}$$

From (24)(25), we have:

$$a_{p_l,n} < a_{p_k,n} \tag{26}$$

Combining (26)(23) and (12):

$$dl_{p_k,n} - a_{p_k,n} < dl_{p_l,n} - a_{p_l,n} \Leftrightarrow \hat{d}_{k,n} < \hat{d}_{l,n} \tag{27}$$

Applying Lemma 3, $\hat{d}_{k,n} < \hat{d}_{l,n} \Rightarrow k < l$ then $t_{l,n}$ is a member of the set $\{t_{j,n} : j > k\} \Rightarrow \sup_{j>k}\{t_{j,n}\} \geq t_{l,n}$. Let us define $\phi_{p_k,n}$ as follows:

$$\phi_{p_k,n} = a_{p_k} + \sup_{j>k}\{t_{j,n}\} \geq a_{p_k} + t_{l,n} \tag{28}$$

Furthermore, the completion of forwarding p_l is at ω and the maximum service time of a packet of channel l is at node n is $t_{l,n}$

$$\Rightarrow \omega - f_{p_l,n} \leq t_{l,n} \Rightarrow \omega \leq t_{l,n} + f_{p_l,n} \tag{29}$$

From (29)(24)(28):

$$\Rightarrow \omega \leq t_{l,n} + f_{p_l,n} \leq t_{l,n} + a_{p_l,k} \leq \phi_{p_k,n} \tag{30}$$

Furthermore:

$$\begin{aligned} dl_{p_k,n} - \phi_{p_k,n} &= (a_{p_k} + \sup_{j>k}\{t_{j,n}\} + \sum_{j=1}^{k} t_{j,n}) - \\ &\quad (a_{p_k} + \sup_{j>k}\{t_{j,n}\}) \\ &= \sum_{j=1}^{k} t_{j,n} \end{aligned} \tag{31}$$

Combining (30)(21)(31):

$$dl_{p_i,n} - \omega \geq dl_{p_k,n} - \phi_{p_k,n} = \sum_{j=1}^{k} t_{j,n} \tag{32}$$

Summing up:

- Equation (32) shows that the length of $(\omega, dl_{p_i,n}]$ is enough to forward one packet of each channel from 1 to channel k.
- Only packets of channels from 1 to k are forwarded during interval $(\omega, dl_{p_i,n}]$ since during $(\omega, dl_{p_i,n}]$ no packet with deadline after $dl_{p_i,n}$ is forwarded and no channel $c > k$ with packet that has deadline within $(\omega, dl_{p_i,n}]$ forwarded during that interval because we choose k to be the channel with largest k that has a packet with deadline within $(\omega, dl_{p_i,n}]$ and the packet does not departs node n before ω.
- No channel from 1 to k has more than one packet during $(\omega, dl_{p_i,n}]$ since node n is empty at ω and from Lemma 1, no channel has more than one packet arriving during any interval

with length τ and the length interval $(\omega, dl_{p_i,n}]$ is smaller than or equal to τ.

• Node n is not empty during $(\omega, dl_{p_i,n}]$, so the node n must be busy sending during $(\omega, dl_{p_i,n}]$ and service an amount $\sum_{j=1}^{k} t_{k,n}$ of traffic

From above reasons, we conclude that each channel from 1 to k has one packet completely forwarded during $(\omega, dl_{p_i,n}]$. Furthermore, from (22), $k \geq i$, this means that channel i has one packet forwarded during $(\omega, dl_{p_i,n}]$, and it does not have any other packet during $(\omega, dl_{p_i,n}]$ thanks to Lemma 1, so the packet must be p_i. Therefore, p_i does not miss its deadline $dl_{p_i,n}$.

Again, the intuition of the above reasoning is that node n has served an amount q of traffic within an interval, and a set S of packets that has at most an amount q of traffic is the only set of packets that n can serve, therefore n must have served all packets in S. In addition, set S contains packet p_i, thus the packet p_i must be forwarded within the interval.

The equality in equation (9) happens when $k = i$ and one packet p_l of channel l such that $t_{l,n} = \sup_{j>i}\{t_{j,n}\}$ arrives just before p_i at time $a^{-}_{p_i,n}$ and packets of other channels from 1 to i arrive just after that at time $a_{p_i,n}$ then, the packet p_i has to wait for p_l to finish and other packets of channels 1 to $i-1$, then the node will send packet p_i and complete sending it at the delay bound $\hat{d}_{i,n}$. □

IV. Conclusion

We have revised a packet scheduling algorithm from its original proof. The importance of this algorithm lies in the fact that we can use it to deploy real-time packet switching network that supports real-time distributed applications. In our new proof, different from the original algorithm, the EDF algorithm is considered to reason about packet orders.

The non-preemptive EDF scheduling algorithm is also applicable to several other real-time scheduling problems. The study of the scheduling algorithm in this paper can help readers gain more insight knowledge about general non-preemptive scheduling algorithms.

References

[1] D. Ferrari and D. C. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, 1990.

[2] D. C. Verma, H. Zhang, and D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *In Proceedings of TriComm '91*, 1991.

[3] D. Ferrari and D. C. Verma, "Real-time communication in a packet-switching network," in *In second International Workshop on Protocols for High Speed Networks, IFIP WG6.1/WG6.4, Palo Alto (CA*. NorthHolland, 1990, pp. 207–18.

[4] S. Shenker, "Fundamental design issues for the future internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1176–1188, 1995.

[5] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," in *SIGCOMM '92: Conference proceedings on Communications architectures & protocols*. New York, NY, USA: ACM, 1992, pp. 14–26.

[6] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *SIGCOMM '89: Symposium proceedings on Communications architectures & protocols*. New York, NY, USA: ACM, 1989, pp. 1–12.

[7] H. Zhang, "Service disciplines for guaranteed performance service in packet-switching networks," vol. 10, 1995, pp. 1374–1396.

[8] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: The single-node case," *IEEE/ACM Transactions on Networking*, vol. 1, pp. 344–357, 1993.

[9] J. C. R. Bennett and H. Zhang, "Wf 2 q: Worst-case fair weighted fair queueing," in *INFOCOM '96: Proceedings of IEEE Conference on Computer Communications*, 1996.

[10] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[11] G. C. Buttazzo, *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*. Norwell, MA, USA: Kluwer Academic Publishers, 1997.