Composable Flexible Real-time Packet Scheduling for Networks on-Chip

Dai Bui
Edward A. Lee

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2012-114
http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-114.html

May 16, 2012

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), #1035672 (CPS: PTIDES), and #0931843 (ActionWebs)), the U. S. Army Research Laboratory (ARL #W911NF-11-2-0038), the Air Force Research Lab (AFRL), the Multiscale Systems Center (MuSyC), one of six research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation program, and the following companies: Bosch, National Instruments, Thales, and Toyota.

Composable Flexible Real-time Packet Scheduling for Networks on-Chip

Dai Bui, Edward A. Lee
Department of Electrical Engineering & Computer Sciences
University of California, Berkeley
{daib,eal}@eecs.berkeley.edu

***Abstract*—Network on chip is an emerging interconnection paradigm to address the scalability of traditional bus architecture. Time-critical systems need the capability to control packet delays in a network on-chip. The inflexibility and/or non-composability reduce the scalability of several proposed real-time service approaches for hard real-time networks on-chip.**

In the era of "dark silicon" when large portions of multicore chips are turned off to save energy and control temperature, the incremental deployment capability of applications is crucial. In incremental deployment, application components are turned on and off. For time-critical applications, these components need to be composable in the sense that new incoming applications should not affect the behaviors of existing applications.

In this paper, we propose a composable and flexible work-conserving packet scheduling discipline for hard real-time networks on chip. Our scheduling discipline employs an earliest deadline first (EDF) scheduler, which reduces average packet delays by 80% in comparison with a previous non-work-conserving EDF scheduling discipline running on the same 8×8 network with various popular traffic patterns. Our proposed scheduling discipline provides guaranteed service without sacrificing high consistent average performance. We also derive sufficient buffer sizes for our scheduling discipline. However, our scheduling discipline incurs a reasonable communication overhead.

I. Introduction

The advent of multicore architectures poses a question to the real-time community about how to exploit the power of multicore systems in modern critical, real-time systems such as aircraft control, medical devices. Deploying real-time applications on uniprocessor systems is hard; it is even much harder on multicore systems due to the complicated interference between applications running in parallel. Analyzing temporal behaviors of real-time applications on multicore machines is not an easy task, because, besides the computational timing uncertainty due to the cache and memory systems, multicore machines also have communication interference between cores.

A. Temporal Isolation

One approach is to design temporally analyzable real-time systems, in the sense that if we have a set of applications, we can analyze their worst-case execution times (WCETs) [20], [3], [4]. Dark silicon [7] presents a challenge to such an approach because future multicore systems could be composed of thousands of cores, where applications come and go, and cores are turned on and off to save power. Estimating the WCETs of all applications at a time is no longer sufficient. Real-time applications need incremental deployment capabilities.

Multicore interconnection infrastructures play an important role in answering the above question. The communication infrastructures need the capability to temporally *isolate* real-time flows such that new incoming real-time flows of new real-time applications do not affect the packets' worst-case end-to-end delays of existing real-time applications. In other words, real-time flows need to be *composable*. We set this as the design goal for our packet scheduling discipline developed in this paper.

B. Motivating Example

Networks on chip are an emerging scalable interconnection paradigm. However, devising a scalable guaranteed service discipline for this interconnection paradigm is a challenging problem. We identify the two following major factors influencing the scalability of a guaranteed service discipline: *composability* and *flexibility*.

1) Composability: To demonstrate the notion of composability as proposed in [11], let us take the scenario in Figure 1. In the figure, the dark region represents cores that are turned off. Now, suppose that a new real time application arrives residing in PE5 and PE14. The new real-time application needs a real-time flow, say the red one. This incoming flow might affect the worst-case end-to-end delays of the two existing flows in the network, orange and blue ones because it directly interferes with the orange flow through the shared link between node 6 and node 10; as a consequence, it indirectly interferes with the traffic of the blue flow via the orange flow. This behavior is not desirable because if adding a new flow would change the worst case packet end-to-end delays of other existing real-time flows in a network, we would need to recertify running real-time applications. Recertifying running applications at runtime is difficult and possibly unsafe. In this context, the static priority packet scheduling discipline proposed by Shi [23] is not composable. This is because, similarly to the above general case, adding the red real-time flow will affect the end-to-end delay of the orange one *directly* through their shared link between node 6 and node 10 if the priority of the red flow is higher than that of the orange one. In addition, it would also affect the end-to-end packet delay of the blue flow *indirectly* through the orange flow.

Non-composable guaranteed service disciplines make it difficult to incrementally deploy real-time applications because they require global arrangements of all applications in parallel real-time systems at a time. As a consequence, the scalability of parallel systems suffers.

2) Flexibility: The designers of the Æthereal guaranteed service network on-chip architecture are aware of the composability issue and tackle it by using a TDMA packet scheduling scheme [9]. However, Æthereal's composability comes at a price of flexibility because Æthereal requires global slot scheduling schemes to avoid packet collisions at intermediate links. The inflexibility caused by global slot scheduling schemes would greatly reduce the scalability of a parallel system, as it is not easy to find suitable slots to avoid packets collisions at intermediate links. It is also questionable whether such parallel systems could be deployed incrementally in the presence of global slot scheduling schemes.

With the composable and flexible design criteria in mind, in this paper, we propose a packet scheduling discipline that is more composable and flexible than the existing ones that we discussed. By tackling both the composability and flexibility issues altogether, our guaranteed service packet scheduling discipline is more scalable.

In this paper, we make three main contributions:

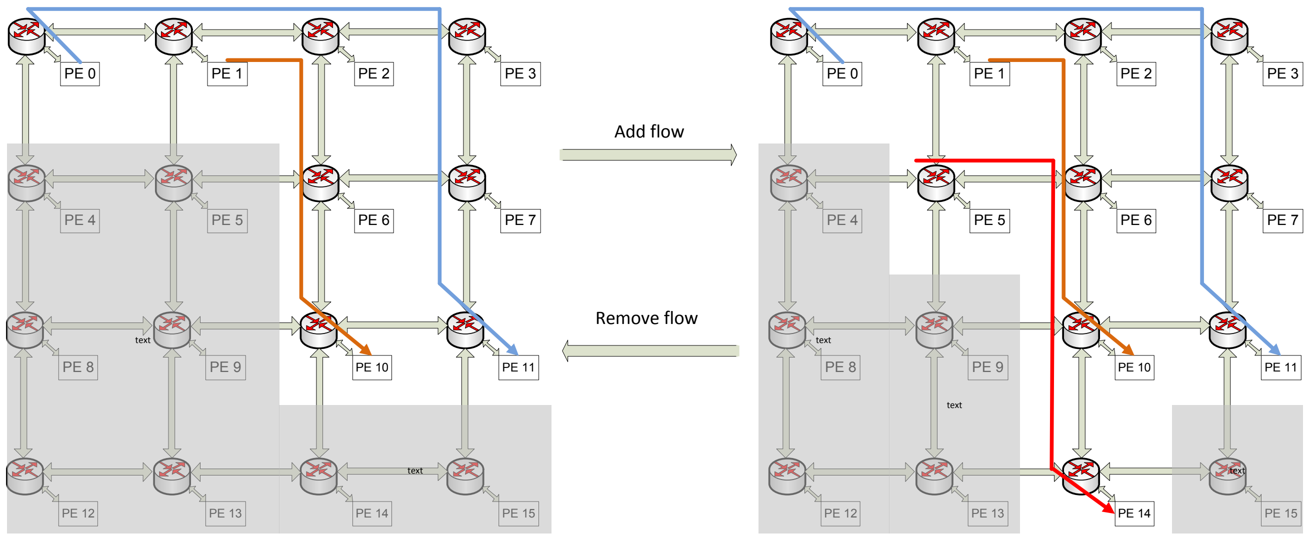

Fig. 1. Problems with adding/removing real-time flows

- We advocate for composable and flexible real-time packet scheduling policies as important design criteria for future large-scale parallel real-time systems.
- We propose a work-conserving EDF packet scheduling scheme that reduces average packet delays by 80% in comparison with previous the non-work-conserving EDF packet scheduling scheme without substantial new hardware requirements. The application of average packet delay reduction is that, although the main goal of designing real-time systems is to make their WCETs analyzable, it would still be beneficial that real-time systems run as fast as possible, so that if they finish their work early, they could be put to sleep to save energy if their future invocations are far enough in the future.
- We derive sufficient buffer sizes for our packet scheduling discipline, something that has not been done in the previous work [23], [28], [16].

II. Background

In this section, we will briefly review network on-chip basics, real-time traffic models and packet timing schemes.

A. Networks on Chip

Networks on chip are packet-switch networks. Packets in networks on-chip are segmented into smaller units called *flits*, standing for flow control units. This feature allows sending packets gradually flit by flit. As a consequence, packet scheduling is *flit-preemptable*. In this paper, we assume a similar router arbitration model to that in [23] shown in Figure 2.

In the router arbitration model, flits of packets of each flow are put into one single separate buffer, called a virtual channel (VC), when they arrive at an input port of a router. Each packet is assigned a deadline at each router. The arbitration unit of routers employs a preemptive Earliest Deadline First (EDF) scheduling. For each output link, at any instant of time, a flit of the packet with closest deadline is chosen to forward to the next router.

B. Traffic Model

We assume a traffic model for real-time flows similar to the one used in [23]. Each real-time flow f is characterized by a tuple (s^f, d^f, T^f, S^f), where s^f and d^f are the addresses of the source

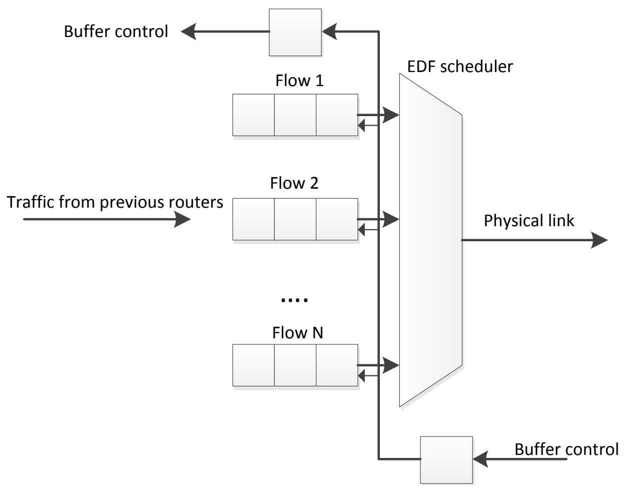

Fig. 2. Router arbitration model

and the destination of flow f respectively. T^f is the minimum packet interval between two successive packets of the flow at its source. S^f is the maximum packet size in terms of flits for all packets of flow f. We then denote $C_l^f = \texttt{transmit}_l(S^f)$, where $\texttt{transmit}_l(s)$ is the function determining the amount of time used to transmit a packet of size s through link l. Finally, similarly to [23], we also assume that each real-time flow f has its own VC at each router it traverses.

C. Packet Timing

A packet is said to *arrive* at a hop when all of its flits arrive at that hop. We call a_h^p the arrival time of packet p at hop h. The maturation time of a packet p at hop h is denoted as m_h^p is the *latest* time for the packet p to arrive at hop h, in other words $m_h^p = \sup a_h^p$. Finally, a packet p *departs* a hop h when all of its flits are forwarded. We call d_h^p the departure time of packet p at hop h.

As we are using an EDF scheduler, let D_h^p be the deadline to completely forward packet p at hop h. As a packet could depart before its deadline, we denote jitter j_h^p as the amount of time packet

Parameter	Description
T^f	Minimum interval between two successive packets of flow f at its source
S^f	Maximum packet size in terms of flits for all packets of flow f
$\texttt{transmit}_l(s)$	Function determining the amount of time used to transmit a packet of size s through link l
C_l^f	$\texttt{transmit}_l(S^f)$
a_h^p	Arrival time of packet p at hop h
m_h^p	Maturation time of packet p at hop h
e_h^p	Time packet p is eligible for forwarding at hop h
j_h^p	Jitter of packet p at hop h
D_h^p	Deadline of packet p at hop h
d_h^p	Departure time of packet p at hop h
b_h^f	Delay bound of packets of flow f at hop h
$P_{h_1 \to h_2}$	Propagation time from hop h_1 to hop h_2
$\lceil x \rceil$	Ceiling value of x
x^+	is equal to x if $x \geq 0$; 0 otherwise

TABLE I
PARAMETERS AND SYMBOLS

p departs before its deadline at hop h:

$$j_h^p = D_h^p - d_h^p \tag{1}$$

III. FLEXIBLE AND COMPOSABLE REAL-TIME PACKET SCHEDULING DISCIPLINES

Our goal is to come up with a new packet scheduling discipline that is more flexible than Æthereal and more composable than Shi's packet scheduling discipline. The main idea is that at each hop in a network, delays of packets of a flow are bounded by a value, and that value cannot be affected by packets of later incoming real-time flows as is the case in Shi's scheduling mechanism. We will show that this could be done by employing a preemptive EDF scheduler, described below.

A. EDF Non-Work-Conserving Scheduling Discipline

Our final goal is to come up with a work-conserving packet scheduling discipline, where available packets are forwarded even if they are not yet mature whenever outgoing links are idle. As a result, work-conserving disciplines have lower average packet delays in comparison with its non-work-conserving discipline counterpart, where packets might be held at sending hops even when outgoing links are idle. However, we will first begin with a non-work-conserving discipline and later, we will derive a work-conserving discipline from the results of the non-work-conserving discipline. We employ the following delay jitter control non-work-conserving discipline.

1) Delay Jitter Control: We employ the delay jitter control mechanism proposed in [25]. We will then prove that our scheduling discipline is still valid without the delay jitter control mechanism.

The delay jitter control mechanism is illustrated in Figure 3. A packet p will be kept at a waiting queue of a router when it arrives earlier than its arrival deadline at the router. In delay jitter control, only mature packets are eligible for scheduling to be forwarded to their next routers:

$$e_h^p = m_h^p \tag{2}$$

where e_h^p is the time packet p is eligible for forwarding at hop h and m_h^p is the maturation time of packet p at hop h. The following equation shows how the maturation time of a packet is computed at each hop h:

$$m_h^p = a_h^p + j_{h-1}^p \tag{3}$$

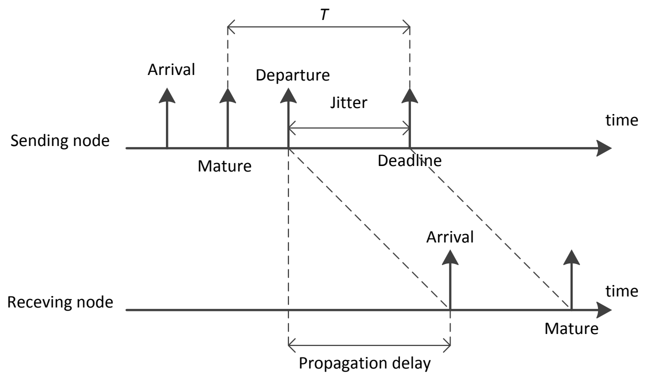

Fig. 3. Delay jitter control mechanism

where $(h-1)$ represents the previous hop of hop h. a_h^p is the arrival time of packet p at hop h. j_{h-1}^p is the amount of time packet p departs hop $(h-1)$ before its deadline at the hop.

The above delay jitter control scheme is used to keep intervals between successive packets of a flow at any intermediate hops unchanged from the intervals of the packets at the flow's source.

2) Preemptive EDF Scheduling: As wormhole flow control allows sending packets sent flit-by-flit, we could employ a preemptive packet scheduling scheme similarly to the one used in [23]. However, we use an EDF scheduler instead of a static priority one.

In our EDF scheduling scheme, the deadline D_h^p of a packet p of a flow f at a hop h is equal to the maturation time m_h^p of the packet at hop h plus the delay bound b_h^f of all packets of flow f at that hop:

$$D_h^p = m_h^p + b_h^f, \forall p \in f \tag{4}$$

Further, suppose that the propagation time for flits from a previous hop $h-1$ to a next hop h remains the same for all flits, and is denoted by $P_{h-1\to h}$; then:

$$a_h^p = d_{h-1}^p + P_{h-1\to h} \tag{5}$$

where a_h^p is the arrival time of packet p at hop h and d_{h-1}^p is the departure time of packet p at hop $(h-1)$.

From equations (1)(3)(4) and (5), we have:

$$\begin{aligned} m_h^p &= a_h^p + j_{h-1}^p \\ &= d_{h-1}^p + P_{h-1\to h} + j_{h-1}^p \\ &= D_{h-1}^p + P_{h-1\to h} \\ &= m_{h-1}^p + b_h^f + P_{h-1\to h} \end{aligned} \tag{6}$$

Substitute equation (6) recursively to get:

$$m_h^p = m_0^p + \sum_{i=0}^{h-1} b_i^f + P_{0\to h} \tag{7}$$

Equation (7) implies that the intervals between maturation times of successive packets remains unchanged at intermediate hops if no packet misses its deadline at its previous hops.

Further, from (4) and (7), the deadline becomes:

$$D_h^p = m_0^p + \sum_{i=0}^{h} b_i^f + P_{0\to h} \tag{8}$$

3) Practical Packet Deadline Assignment: Note that in both equations (4) and (8), packet deadlines are computed through other parameters that might not be available at intermediate hops. It is beneficial to have a packet deadline assignment mechanism where deadlines are computed based on local parameters available at each hop. This is done as follows.

Suppose that whenever a packet departs a hop, its jitter at the hop is included in the packet. Now, from equations (3) and (4), we can compute deadline D_h^p of packet p directly from local parameters such as its arrival time at hop h and its jitter j_{h-1}^p is included in p when it departs the previous hop $(h-1)$[1]. The delay bound b_h^f is:

$$D_h^p = a_h^p + j_{h-1}^p + b_h^f \tag{9}$$

Please note that equation (9) also holds for the work-conserving packet scheduling discipline in Section III-B below.

4) General Delay Bound Scheduling: The above framework allows us to start with the baseline scheduling algorithm by Zheng and Shin [28].

Theorem 1: (Zheng and Shin) A set of flows $f = (T^f, S^f, b_h^f), f = 1, 2, \ldots, n$ at hop h are schedulable over a link l under the preemptive deadline policy if and only if both of the following conditions hold:

- $\sum_{f=1}^{n} \frac{C_l^f}{T^f} \leq 1$.
- $\forall t \in S, \sum_{f=1}^{n} \lceil (t - b_h^f)/T^f \rceil^+ C_l^f \leq t$ where $S = \bigcup_{f=1}^{n} S^f, S^f = \{b_h^f + nT^f : n = 0, 1, \ldots, \lfloor (t_{max} - b_h^f)/T^f \rfloor\}$ and $t_{max} = \max\{b_h^1, b_h^2, \ldots, b_h^n, (\sum_{f=1}^{n}(1 - b_h^f/T^f)C_l^f)/(1 - \sum_{f=1}^{n} C_l^f/T^f)\}$.

where $\lceil x \rceil^+ = n$ if $n-1 \leq x < n, n = 1, 2, 3, \ldots$ and $\lceil x \rceil^+ = 0$ for $x < 0$. T^f is the minimum interval between two successive packets of flow f at its source. S^f is the maximum packet size in terms of flits for all packets of flow f. $\texttt{transmit}_l(s)$ is the function determining the amount of time used to transmit a packet of size s through link l. $C_l^f = \texttt{transmit}_l(S^f)$.

Theorem 1 might not be as complicated as it looks. The key idea of the theorem is that it makes sure the amount of arrived traffic of all flows going through a link is not larger than service capability of the link by checking the condition $\sum_{f=1}^{n} \lceil (t - b_h^f)/T^f \rceil^+ C_l^f \leq t$ at a set of representative time points S. It is proved in [28] that those points are sufficient and necessary. We will use an example from [28] to illustrate the main ideas of the theory.

Example 1: Consider three flows at hop h being scheduled on link l: $f_1 = (T_h^{f_1}, C_l^{f_1}, b_h^{f_1}) = (10, 2, 5), f_2 = (T_h^{f_2}, C_l^{f_2}, b_h^{f_2}) = (8, 4, 8), f_3 = (T_h^{f_3}, C_l^{f_3}, b_h^{f_3}) = (13, 3, b_h^{f_3})$. Now, we will use Theorem 1 to check the schedulability of two cases $b_h^{f_3} = 9$ and $b_h^{f_3} = 8$.

Solution: The first condition:

$$\sum_{i=1}^{3} \frac{C_l^{f_i}}{T^{f_i}} = 0.95 < 1$$

For $b_h^{f_3} = 9$, we compute $t_{max} = 35$. Then $S_1 = \{5, 15, 25, 35\}, S_2 = \{8, 16, 24, 32\}, S_3 = \{9, 21, 33\}$ and $S = S_1 \cup S_2 \cup S_3$. Now we check the condition $\sum_{i=1}^{3} \lceil (t - b_h^{f_i})/T^{f_i} \rceil^+ C_l^{f_i} \leq t$ for all points $t \in S$ and see that the condition is satisfied. Therefore, the three flows are schedulable on link l with $b_h^{f_3} = 9$.

[1]This jitter information is the communication overhead of this delay jitter control mechanism. As jitters are bounded by the worst-case end-to-end delays of real-time flows and the worst-case end-to-end delays are supposed to be small, we postulate that 8 to 10 bits in tail flits of packets are enough to forward jitter information.

When $b_h^{f_3} = 8 \Rightarrow t_{max} = 40$. Then $S_1 = \{5, 15, 25, 40\}, S_2 = \{8, 16, 24, 32, 40\}, S_3 = \{8, 20, 32\}$ and $S = S_1 \cup S_2 \cup S_3$. At $t = 8 \in S$, $\sum_{i=1}^{3} \lceil (t - b_h^{f_i})/T^{f_i} \rceil^+ C_l^{f_i} = 9 > t = 8$, therefore, three links cannot be scheduled over the link with $b_h^{f_3} = 8$.

5) Simplified Delay Bound Scheduling: The general delay bound scheduling might be expensive to check; for example, when the utilization of a link is close to 1, the number of representative points could become very large. Accordingly, routing a flow across multiple highly-utilized links could become highly expensive, as a routing procedure would require at least a polynomial number of times to test the conditions in Theorem 1. Notice that in Theorem 1, if we are able to set $b_h^f = T^f$ under the permission of some applications, the above theorem becomes the EDF schedulability theorem in the well-known work by Liu and Layland [17]. The following theorem is directly adapted from Theorem 7 in [17], where deadlines consist of run-ability constraints only; i.e. each task must be completed before the next request for it occurs.

Theorem 2: (Liu and Layland) For a given set of m flows, the deadline driven scheduling algorithm is feasible if and only if:

$$\sum_{f=1}^{m} \frac{C_l^f}{T^f} \leq 1 \tag{10}$$

6) Multihop Delay Bound: The scheduling scheme in the previous section is for single hop, however, a path of a flow is composed of several nodes. We will prove that our scheduling mechanism can maintain multihop delay bounds.

Theorem 3: If all packets arrive on time for scheduling at a router to be forwarded to the next router on a link l, and the delay jitter control mechanism is applied, then no packet will miss its deadline when Theorem 1 is satisfied.

Proof: Because a packet has to completely depart a router when the next packet of the same flow becomes mature, each flow has at most one packet eligible for scheduling at any time. By applying the Theorem 1, no packet will miss its deadline at a router. □

Theorem 4: If for all links l in a network the conditions of Theorem 1 are satisfied, and all flows honor their declared traffic models (T^f, S^f), then no packet will miss its deadline at any router.

Proof: No packet arrives early at its initial router because of its flow traffic model. By induction using Theorem 3, no packet will miss its deadline at its next router, therefor no packet will miss its deadline at its final destination. □

B. EDF Work-Conserving Scheduling Discipline

The scheduling discipline presented in the previous sections is not work-conserving. This section is used to present a work-conserving scheduling discipline based on the previous non-work-conserving discipline. In this work-conserving discipline, packets are eligible for forwarding even if they are not yet *mature*.

The motivation for coming up with a work-conserving discipline is because implementing a circuit for keeping non-mature packets in the non-work-conserving discipline could be expensive; e.g. it requires more counters to know when packets become mature. The non-work-conserving discipline also results in larger average packet delays that can hinder possible optimization as discussed in Section I-B.

However, we could relax the scheduling policy to make it a work-conserving discipline that can reduce average packet delays. While traditional work-conserving service disciplines would require that intermediate routers maintain counters for each flow [27], our new packet scheduling scheme could dispense with flow counters as long as routers maintain a minimal buffer requirement for each of its flows.

The main modification of this work-conserving discipline from the non-work-conserving discipline in previous sections is that a packet p becomes eligible for forwarding immediately when it arrives even though it is not mature yet, $e_h^p = a_h^p$. Readers can compare with equation (2) to see the difference.

The following theorem proves that, the modification will not change the end-to-end packet delay bound of a flow.

Theorem 5: If a work-conserving EDF scheduler is used, the deadline of a packet at a hop is set using equation (8), and all real-time flows honor their declared traffic models then no packet will miss its deadline.

Proof: We prove this by contradiction. Suppose p is the first packet to miss its deadline at a hop h because of packet congestion over its outgoing link l. Let $[\hat{t}, \hat{t}+L)$ be the busy period[2] of length L during which p misses its deadline. Further, let $\{p_i\}$ be the set of packets waiting for forwarding over link l during the period $[\hat{t}, \hat{t}+L)$. We can see that:

$$m_h^{p_i} \geq a_h^{p_i} \geq \hat{t} \text{ and } m_h^p \geq a_h^p \geq \hat{t} \tag{11}$$

Note that these are the maturation and arrival times at hop h when the work-conserving discipline is use. We also have:

$$D_h^{p_i} \leq D_h^p = \hat{t} + L \quad \forall p_i \tag{12}$$

$$\texttt{transmit}_l(\texttt{size}(p)) + \sum_i \texttt{transmit}_l(\texttt{size}(p_i)) > L \tag{13}$$

where $\texttt{transmit}_l(s)$ is the function determining the amount of time used to transmit a packet of size s through link l.

Let us consider the same packet trace sent through the network, where, now instead *packets are scheduled using the non-work-conserving discipline* that we describe in the previous sections. Note that now packets are scheduled using the non-work-conserving discipline, packet p will not miss its deadline at hop h anymore. As a result, the same set of packets $\{p_i\}$ with the non-work-conserving discipline cannot cause p to miss its deadline at hop h.

Note that the maturation time of a packet x at hop h is the same for both work-conserving and non-work-conserving disciplines because packet x's deadline at hop h is the same, and the maturation time can be inferred from the deadline based on equation (4). As for the non-work-conserving discipline, $e_h^x = m_h^x$ for a packet x, combining with (11), we see that the eligible times for packets $\{p_i\}$ and p are at least $\hat{t}$:

$$e_h^{p_i} \geq \hat{t} \text{ and } e_h^p \geq \hat{t} \tag{14}$$

Furthermore, note that all packets $\{p_i\}$ and p still have the same deadlines as in the case using the work-conserving discipline, and as a result, from (12) (14), we see that packets $\{p_i\}$ and p have to be sent during the period $[\hat{t}, \hat{t}+L)$. However equation (13) shows that the total traffic of $\{p_i\}$ and p exceeds the capacity of the link during the period $[\hat{t}, \hat{t}+L)$. As a result, at least one of the packets $\{p_i\}$ or p has to miss its deadline. This contradicts the fact that no packet can miss its deadline at hop h when the non-work-conserving packet scheduling discipline is used. $\square$

C. Augmented EDF Work-Conserving Scheduling Discipline

The work-conserving discipline in the previous section requires that only packets arriving entirely are eligible forwarding although this requirement incurs additional waiting delays on packets. This requirement is because the deadline D_h^p of packet p is computed from jitter j_{h-1}^p as in equation (9) and jitter j_{h-1}^p is not available when packet p has not entirely arrived at hop h. When deadline D_h^p cannot be computed, packet p cannot be eligible for scheduling.

[2]The link is idle before $\hat{t}$ and after $\hat{t}+L$.

However, we still can relax the requirement using an augmented EDF work-conserving scheduling algorithm to further reduce average packet delays. The scheduling algorithm has two steps in each clock cycle: 1) If there are entirely arrived packets, forward the top flit of the arrived packet with the closest deadline; 2) If no packet has fully arrived, choose any available flit of an arbitrary partially arrived packet to forward.

Note that this augmented scheduling algorithm does not allow non-fully-arrived packets to interfere with fully arrived packets, therefore Theorem 5 still holds. In Section VI-C, this augmented scheduling algorithm could result in significant improvements.

IV. Sufficient Buffer Size Estimation

Our previous scheduling policies assume that VCs have enough buffer space for each flow at each router so that packets are forwarded immediately without the need for waiting for buffer space. This assumption is generally not true in networks on-chip, because routers in networks on-chip are designed to be as small as possible to reduce cost and power consumption. It is beneficial to derive the buffer space sufficient for both work-conserving and non-work-conserving disciplines.

A. Buffer Size for Non-Work-Conserving Discipline

In this discipline, as mature real-time packets are forwarded whenever possible, specially when there are no other mature packets with closer deadlines, destination VCs are required to have enough space to store received packets, mature packets being forwarded and non-mature waiting packets. The following theorem proves that we can derive sufficient virtual channel buffer sizes for flows at each hop.

Theorem 6: Buffer size $\hat{B}_h^f$ for each flow f at each hop h computed as in equation (15) is sufficient to store all received, mature and non-mature packets of the flow at anytime.

$$\hat{B}_h^f = \lceil \frac{b_{h-1}^f + b_h^f}{T^f} \rceil \times S^f \tag{15}$$

where S^f is the maximum packet size of flow f.

Proof: Suppose that k is the number of queueing packets of flow f at some arbitrary point in time at hop h, and $p_1, \ldots, p_k$ form such a set of packets. We will prove that k is upper bounded, and derive that bound thereby proving the sufficient buffer size in equation (15). As packet p_k has to depart hop $h-1$ *after* it has become mature, therefore $d_{h-1}^{p_k} \geq m_{h-1}^{p_k}$. Combining with (6),

$$d_{h-1}^{p_k} \geq m_0^{p_k} + \sum_{i=0}^{h-2} b_i^f + P_{0\rightarrow h-1} \tag{16}$$

From (5) and (16), we have:

$$a_h^{p_k} \geq m_0^{p_k} + \sum_{i=0}^{h-2} b_i^f + P_{0\rightarrow h} \tag{17}$$

From the assumption, p_k arrives and p_1 has not departed, therefore:

$$D_h^{p_1} \geq a_h^{p_k} \tag{18}$$

From (8), (17) and (18):

$$\begin{aligned} m_0^{p_1} + \sum_{i=0}^{h} b_i^f + P_{0\rightarrow h} &\geq m_0^{p_k} + \sum_{i=0}^{h-2} b_i^f + P_{0\rightarrow h} \\ \Leftrightarrow b_{h-1}^f + b_h^f &\geq m_0^{p_k} - m_0^{p_1} \end{aligned} \tag{19}$$

As the minimum interval between packets at the source node of flow f is T^f and $m_0^{p_1}, m_0^{p_2}, \ldots, m_0^{p_k}$ are available times of the k consecutive packets at the source of f, therefore $m_0^{p_k} - m_0^{p_1} \geq (k-1) \times T^f$. Combining with (19), we have: $\frac{b^f_{h-1}+b^f_h}{T^f} \geq k-1 \Rightarrow \lceil \frac{b^f_{h-1}+b^f_h}{T^f} \rceil \geq k$.

Since f cannot have more than $\lceil \frac{b^f_{h-1}+b^f_h}{T^f} \rceil$ packets at a hop h, the VC of flow f at hop h of buffer size $\hat{B}^f_h$, calculated using the equation (15), will provide sufficient buffer space for the non-conserving discipline to execute. □

B. Buffer Size for Work-Conserving Discipline

While the above buffer size is derived for a non-work-conserving packet scheduling discipline, a work-conserving packet scheduling would require more buffering mechanisms as non-mature packets could be forwarded. We will prove that the same buffer size could be used for the work-conserving discipline. However, it requires a buffer credit mechanism often found in networks on-chip [5].

The buffer credit mechanism operates as follows. Routers maintain *counters* of the numbers of free slots of VCs at their destination routers. Whenever a sending router sends a flit, it decreases the respective counter of the respective receiving VC. Whenever, a flit is removed from the receiving VC, a credit is sent back to the sending router and the router will increase the respective counter of the receiving VC. The sending router will stop forwarding flits to the receiving VC when the respective counter of that VC reaches 0, indicating that the VC is full.

The following theorem will prove that the satisfiable buffer sizes of the non-work-conserving packet scheduling combining with the above buffer credit mechanism is sufficient for the work-conserving discipline to operate.

Theorem 7: If the minimum buffer sizes for the work-conserving EDF packet scheduling discipline are the same as in the non-work-conserving case, the worst-case delays of packets of each flow at each hop stay the same.

Proof: As the minimum VC buffer sizes are the same as in the non-work-conserving case, only non-mature packets could be stalled by the buffer credit mechanism, therefore the packet worst-case delays stay the same. □

V. ROUTING COMPOSABILITY

Our packet scheduling discipline is well-suited for the application-aware deadlock-free oblivious routing scheme proposed in [10]. However, our packet scheduling discipline increases routing freedom because it does not require turn-restrictions to avoid deadlock as is the case of best effort packet scheduling scheme presented in [10]. Improving routing freedom enhances the routing *composability* for our packet scheduling scheme in the sense that real-time flows in a system could be deployed gradually without causing routing problems to later real-time flows due to turn-restrictions as is the best-effort routing scheme in [10].

A. Deadlock-Free Routing

Theorem 8: Routing for real-time flows scheduled using either the work-conserving or non-work-conserving discipline is deadlock free.

Proof: Theorem 4 proves that packets of real-time flows have bounded latency, therefore, packets of real-time flows cannot participate in any deadlocked cycle. □

Traffic pattern	**# VCs**	**Flow utilization**	**Routing time** (ms)
Transpose	3	1/3	8.610
Shuffle	3	1/3	9.379
Bit reversal	3	1/3	8.692
Bit complement	4	1/4	4.210
Symmetric	4	1/4	4.152

TABLE II
ROUTING RESULTS

B. Heuristic Routing Scheme

We employ the heuristic routing scheme in [10] based on Dijkstra's weighted shortest path algorithm. We create a directed graph $G = (V, E)$ where V is the set of nodes composed of routers, and E is the set of edges composed of links between routers. The weights of edges in E are derived from the residual capacities of respective links. Let $\hat{c}(e)$ and $c(e)$ be the current residual capacity and the initial capacity of the link. The utilization u^f_e of a flow f at link e is computed as $u^f_e = \frac{C^f_e}{T^f}$. The weighting function $w(e)$ is computed as $w(e) = \frac{1}{\hat{c}(e) - u^f_e}$.

Real-time flows are gradually routed through the network by running the Dijkstra's algorithm on the graph G to select the path $\mathcal{P}$ such that $\sum_{e \in \mathcal{P}} w(e)$ is smallest. $\hat{c}(e) \forall e \in \mathcal{P}$ is updated after each time a flow is routed.

VI. EXPERIMENTS

A. Simulation Setup

We use a cycle-accurate network on-chip simulator to measure packet delays of both work-conserving and non-work-conserving disciplines and compare their average end-to-end packet delays. The simulator is configured so that each flit of a packet takes one cycle to reach its next hop. We use an 8×8 network where the buffer size in number of flits for each VC of a flow at a hop is set as the sufficient buffer size estimated in Section IV.

B. Routing Evaluation

We employ the simplified delay bound scheduling mechanism in Section III-A5 to evaluate our routing strategy. For each traffic pattern, we find the smallest number of VCs required at each input of a router as well as maximum flow utilizations. We assume that all real-time flows have the same utilization, which is the fraction of time to send a packet of the maximum size of each flow and the minimum interval between packets of the flow. Table II shows the result of our routing evaluation. For the popular traffic patterns, the requirement for the number of VCs at each input port for real-time flows remains reasonable. The table also shows the running time for the routing procedure for different traffic patterns on a 3.00GHz Intel Xeon CPU. The small routing times exhibit high degree of routing composability.

C. Work-Conserving and Non-Work-Conserving Disciplines Average Delay Comparison

In this section, we make a comparison of the average packet delays between the work-conserving and non-work-conserving scheduling disciplines. To do the comparison, we use the optimal configurations of flows and routers in each traffic pattern found in Table II. We examine four scheduling disciplines:

- The non-work-conserving discipline in Section III-A.
- The standard work-conserving discipline in Section III-B.
- The augmented work-conserving discipline in Section III-C.
- A best-effort round-robin discipline.

Figure 4 shows the average delay reduction of the other disciplines over the baseline non-work-conserving discipline by Zheng and Shin [28]. From the figure, the standard work-conserving discipline significantly reduces the average packet delays by more than 55% in comparison with baseline non-work-conserving discipline. However, the standard work-conserving discipline only results in better average delays than the best-effort round-robin discipline in two out of five traffic patterns. The performance of the best-effort round-robin discipline is rather arbitrary. The augmented work-conserving discipline performs best in all the cases. It drastically reduces the average packet delays by more than 80%. It also has rather persistent performance in all the five cases.

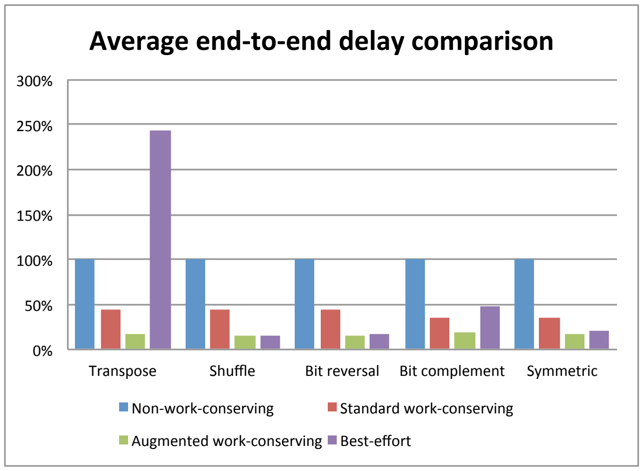

Fig. 4. Average end-to-end packet delay reduction for different traffic patterns

D. Packet Delay Sample

Figure 5 shows samples of packet delays between two processing elements PE58 and PE23 in an 8×8 transpose traffic pattern. We examine the same four service disciplines as in the previous section. In the figure, the non-work-conserving packet scheduling exhibits a constant packet delay due to the jitter control mechanism. Whereas standard work-conserving packet scheduling exhibits non-constant packet delays, however, they are bounded by the packet delay in the non-work-conserving scheduling scheme. The augmented work-conserving packet scheduling also exhibits non-constant packet delays but the variation is small and the packet delays are considerably reduces. Best-effort scheme exhibits widely varied packet delays in the same traffic conditions and routing scheme.

VII. Related Work

Our work avoids the global scheduling problem of Æthereal [9] by scheduling packets with local information, making it more flexible than Æthereal. This comes at the price of more buffer space requirements and a reasonable communication overhead for jitter information propagation. However, our scheduling scheme still guarantees to have bounded buffer size, which could be difficult to estimate in the work by Shi [23]. Our scheme is also more composable than the scheme by Shi.

In this paper, we propose a work-conserving service discipline extending existing non-work-conserving disciplines [16], [28]. This extension reduces average packet delays by 80% in comparison with the existing non-work-conserving discipline in popular traffic patterns in an 8×8 network while retaining the same worst case delays of packets. The main motivation of average packet delay reduction is

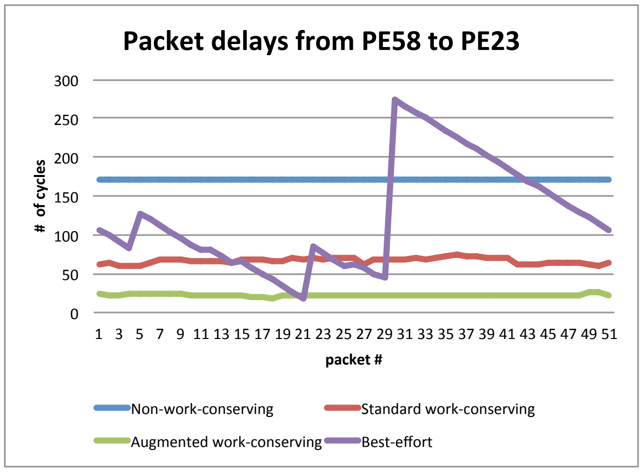

Fig. 5. Packet delays of between PE58 and PE23 for work-conserving, non-work-conserving, and best-effort packet scheduling policies

that real-time systems not only need to maintain their worst-case delays, but also should run as fast as possible and go to sleep to save energy if the next invocation is far enough in the future to save energy. In addition, [16], [28] do not contain sufficient buffer size estimation.

Several soft real-time service disciplines for networks on-chip have been recently proposed [8], [19], [13]. QNoC [6] uses the same static priority scheme as in Shi's work, so it suffers from the composability problem. SoCBUS [26] employs a circuit switch scheme where a dedicated physical path is reserved for a flow. This approach is restrictive as links could be under utilized, and it also presents potential routing problems.

Zhang [27] summarizes service disciplines for internet packet switch networks. However, they need to be adapted for networks on chip due to restricted buffer space in networks on chip. There are also several notable other approaches for quality of service in packet-switched networks [22], [24], [14], [1], [15].

Qian et al. in [21] use network calculus to derive worst-case delay bounds of best-effort packets, however, the delay bounds could be conservative when several flows intersect each other. [18], [2], [12] use queueing models to estimate packet delays. However, queueing models often do not account for blocking effects in networks on chip.

VIII. Conclusion

In this paper, we have discussed composability and flexibility as important design criteria for communication infrastructure in future “dark silicon” real-time parallel computing systems. We also propose an extension to the existing non-work-conserving EDF discipline that results in a considerable average packet delay reduction. The augmented EDF work-conserving discipline provides guaranteed service while maintaining high consistent average performance. This work could be extended in several directions. First, packets need to arrive fully before being eligible for scheduling. This requirement would result in larger packet delay bounds. Second, although our sufficient buffer size estimations are close to the real lower buffer size bounds, it is not tight. We could tighten the equation (18) by carefully taking into account the case where packet p_k is arriving, packet p_1 is departing. In this case, p_1 has been transmitted partially while a portion of p_k has been sent.

References

[1] Shobana Balakrishnan and Füsun Özgüner. A priority-driven flow control mechanism for real-time traffic in multiprocessor networks. *IEEE Transactions on Parallel and Distributed Systems*, 9(7):664–678, 1998.

[2] Yaniv Ben-Itzhak, Israel Cidon, and Avinoam Kolodny. Delay analysis of wormhole based heterogeneous noc. In *Proceedings of the 2011 Fifth ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '11, pages 161–168, New York, NY, USA, 2011. ACM.

[3] Dai Bui, Edward Lee, Isaac Liu, Hiren Patel, and Jan Reineke. Temporal isolation on multiprocessing architectures. In *Proceedings of the 48th Design Automation Conference*, DAC '11, pages 274–279, New York, NY, USA, 2011. ACM.

[4] Dai N. Bui, Hiren D. Patel, and Edward A. Lee. Deploying hard real-time control software on chip-multiprocessors. In *Proceedings of the 2010 IEEE 16th International Conference on Embedded and Real-Time Computing Systems and Applications*, RTCSA '10, pages 283–292, Washington, DC, USA, 2010. IEEE Computer Society.

[5] William Dally and Brian Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[6] Rostislav (Reuven) Dobkin, Ran Ginosar, and Israel Cidon. QNoC asynchronous router with dynamic virtual channel allocation. In *Proceedings of the 2007 First International Symposium on Networks-on-Chip*, NOCS '07, page 218, Washington, DC, USA, 2007. IEEE Computer Society.

[7] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proceeding of the 38th Annual International Symposium on Computer Architecture*, ISCA '11, pages 365–376, New York, NY, USA, 2011. ACM.

[8] Boris Grot, Stephen W. Keckler, and Onur Mutlu. Preemptive virtual clock: a flexible, efficient, and cost-effective QoS scheme for networks-on-chip. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 268–279, New York, NY, USA, 2009. ACM.

[9] Andreas Hansson. *A Composable and Predictable On-Chip Interconnect*. PhD thesis, Department of Electrical Engineering, Eindhoven University of Technology, 2009.

[10] Michel A. Kinsy, Myong Hyon Cho, Tina Wen, Edward Suh, Marten van Dijk, and Srinivas Devadas. Application-aware deadlock-free oblivious routing. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 208–219, New York, NY, USA, 2009. ACM.

[11] Hermann Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1st edition, 1997.

[12] Mingche Lai, Lei Gao, Nong Xiao, and Zhiying Wang. An accurate and efficient performance analysis approach based on queuing model for network on chip. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 563–570, New York, NY, USA, 2009. ACM.

[13] Jae W. Lee, Man Cheuk Ng, and Krste Asanovic. Globally-synchronized frames for guaranteed quality-of-service in on-chip networks. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ISCA '08, pages 89–100, Washington, DC, USA, 2008. IEEE Computer Society.

[14] Sunggu Lee. Real-time wormhole channels. *Journal of Parallel and Distributed Computing*, 63(3):299–311, 2003.

[15] Jong-Pyng Li and Matt W. Mutka. Priority based real-time communication for large scale wormhole networks. In *Proceedings of the 8th International Symposium on Parallel Processing*, pages 433–438, Washington, DC, USA, 1994. IEEE Computer Society.

[16] Jörg Liebeherr, Dallas E. Wrege, and Domenico Ferrari. Exact admission control for networks with a bounded delay service. *IEEE/ACM Transactions Networking*, 4:885–901, December 1996.

[17] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20:46–61, January 1973.

[18] Nikita Nikitin and Jordi Cortadella. A performance analytical model for network-on-chip with constant service time routers. In *Proceedings of the 2009 International Conference on Computer-Aided Design*, ICCAD '09, pages 571–578, New York, NY, USA, 2009. ACM.

[19] Jin Ouyang and Yuan Xie. Loft: A high performance network-on-chip providing quality-of-service support. In *Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '43, pages 409–420, Washington, DC, USA, 2010. IEEE Computer Society.

[20] Marco Paolieri, Eduardo Quiñones, Francisco J. Cazorla, Guillem Bernat, and Mateo Valero. Hardware support for WCET analysis of hard real-time multicore systems. In *Proceedings of the 36th Annual International Symposium on Computer Architecture*, ISCA '09, pages 57–68, New York, NY, USA, 2009. ACM.

[21] Yue Qian, Zhonghai Lu, and Wenhua Dou. Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip. In *Proceedings of the 2009 Third ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '09, pages 44–53, Washington, DC, USA, 2009. IEEE Computer Society.

[22] Jennifer Rexford, John Hall, and Kang G. Shin. A router architecture for real-time communication in multicomputer networks. *IEEE Transactions on Computers*, 47:1088–1101, 1998.

[23] Zheng Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proceedings of the 2008 Second ACM/IEEE International Symposium on Networks-on-Chip*, NOCS '08, pages 161 –170, april 2008.

[24] Hyojeong Song, Boseob Kwon, and Hyunsoo Yoon. Throttle and preempt: A new flow control for real-time communications in wormhole networks. In *Proceedings of the International Conference on Parallel Processing*, ICPP '97, pages 198–202, Washington, DC, USA, 1997. IEEE Computer Society.

[25] D.C. Verma, H. Zhang, and D. Ferrari. Delay jitter control for real-time communication in a packet switching network. In *Communications Software, 1991, 'Communications for Distributed Applications and Systems', Proceedings of TRICOMM '91., IEEE Conference on*, pages 35–43, apr 1991.

[26] Daniel Wiklund and Dake Liu. SoCBUS: Switched network on chip for hard real time embedded systems. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, page 78.1, Washington, DC, USA, 2003. IEEE Computer Society.

[27] Hui Zhang. Service disciplines for guaranteed performance service in packet-switching networks. *Proceedings of the IEEE*, 1995.

[28] Qin Zheng and Kang G Shin. On the ability of establishing real-time channels in point-to-point packet-switched networks. *IEEE Transactions on Communications*, 42(234):1096–1105, 1994.