Available online at www.sciencedirect.com
SCIENCE DIRECT®
Performance Evaluation 60 (2005) 275–302
PERFORMANCE EVALUATION
An International Journal
www.elsevier.com/locate/peva

Performance analysis of a QoS capable cluster interconnect☆

Eun Jung Kim[a,*], Ki Hwan Yum[b], Chita R. Das[c]

[a] *Department of Computer Science, Texas A&M University, College Station, TX 77843, USA*
[b] *Department of Computer Science, University of Texas, San Antonio, TX 78249, USA*
[c] *Department of Computer Science and Engineering, the Pennsylvania State University, University Park, PA 16802, USA*

Available online 8 December 2004

Abstract

The growing use of clusters in diverse applications, many of which have real-time constraints, requires quality-of-service (QoS) support from the underlying cluster interconnect. All prior studies on QoS-aware cluster routers/networks have used simulation for performance evaluation. In this paper, we present an analytical model for a wormhole-switched router with QoS provisioning. In particular, the model captures message blocking due to wormhole switching in a pipelined router, and bandwidth sharing due to a rate-based scheduling mechanism, called VirtualClock. Then we extend the model to a hypercube-style cluster network. Average message latency for different traffic classes and deadline missing probability for real-time applications are computed using the model.

We evaluate a 16-port router and hypercubes of different dimensions with a mixed workload of real-time and best-effort (BE) traffic. Comparison with the simulation results shows that the single router and the network models are quite accurate in providing the performance estimates, and thus can be used as efficient design tools.

Keywords: Analytical model; Cluster network; Pipelined router architecture; Quality-of-service; VirtualClock; Wormhole switching

☆ A preliminary version of this paper was presented at the 11th GI/ITG Conference on Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB 2001), September 2001.

* Corresponding author.
E-mail addresses: ejkim@cs.tamu.edu (E.J. Kim), yum@cs.utsa.edu (K.H. Yum), das@cse.psu.edu (C.R. Das).

0166-5316/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.peva.2004.10.008

1. Introduction

Quality-of-service (QoS) provisioning in clusters is becoming a critical issue with the widespread use of clusters in diverse commercial applications. The traditional best-effort (BE) service model that has been used for scientific computing is not adequate to support many cluster applications with varying consumer expectations. As an example, many web servers and database servers make efficient use of clustering technology from cost, scalability, and availability standpoints. However, the tremendous surge in dynamic web contents, multimedia objects, e-commerce, and other web-enabled applications requires QoS guarantees in different connotations. This in turn mandates that the cluster system, and hence the the cluster interconnect, should be able to handle user specified service demands instead of adopting the *same-service-to-all* model. High performance cluster networks, also known as system area networks (SANs), usually use switch-based architectures. Most commercial routers (switches) such as SGI SPIDER, Cray T3D/E, Tandem Servernet-II, Intel Cavallino, IBM SP2, and Myricom Myrinet [1–6] use wormhole switching to provide high performance. However, they have not been designed for QoS assurance except for the Servernet-II [3], which provides a link arbitration policy (called ALU-biasing) for implementing limited bandwidth and delay control. Hence, design and analysis of QoS capable routers[1] and cluster networks has become a current research focus [7,8].

In view of this, a few router architectures with QoS provisioning have been proposed recently [9–11,7,12,13]. Most of these designs have used a hybrid approach with two different types of switching mechanisms within the same router—one for best-effort traffic and the other for real-time traffic. They have refrained from using wormhole switching because of the potential unbounded delay for real-time traffic.

On the contrary, since wormhole switching has been adopted in most commercial routers, it would really be advantageous if we could leverage off of the large amount of effort that has gone into the design and development of such routers, and make them QoS capable with minimal design changes. Some recent modifications to wormhole routers have been considered for handling traffic priority [14–20]. The options vary from providing hardware support in the router for bandwidth assurance [14,17–20] to software solutions on existing routers [16]. In the hardware approach, the most logical solution is to assign separate virtual channels (VCs) (the VC concept was introduced by Dally [21].) to different traffic classes and use a rate-based scheduling mechanism such as Fair Queueing (FQ) [22] or VirtualClock [23] to share the link bandwidth proportionately [14,18,20]. Techniques such as preemption of a lower priority traffic in favor of a higher priority traffic [17,19] is likely to provide better performance, but at added complexity. Software solution like the self-synchronizing scheduling [16] does not need any hardware modification, but the solution may not be scalable.

A limitation of all prior studies on routers/networks that support integrated traffic is that they use simulation to evaluate the performance of various design trade-offs. In addition, the evaluations are confined to a single router in many cases. For example, in [9,18,20] where the router designs are evaluated with multimedia video streams, the study is limited to a single (4-port/8-port) router or a small network. Detailed flit-level simulation is quite expensive and prohibits full-blown analyses of various design trade-offs, especially in large networks. On the other hand, an accurate analytical model can provide quick performance estimates with various design parameters and will be a valuable design tool. For example,

[1] QoS capable routers for Internet have been designed with various flavors. ATM networks can provide QoS guarantee, but incur high latency. We are specially interested in low-latency cluster networks based on cut-through switching.

we can analyze the impact of different buffer organizations, traffic types and link widths on overall system performance much faster by avoiding tedious and time-consuming simulation.

We present in this paper a mathematical model for analyzing QoS capable cluster networks. We use a bottom-up approach first by developing the model for a single router and then extending it to a network. Here, we use a hypercube-style cluster network primarily to keep the analysis tractable due to the symmetric nature of the network. Such a topology has been used in the SGI Origin architecture [24]. However, our QoS-capable router model can be extended to any regular topology such as *k*-ary *n*-cubes and meshes as long as the topology and routing algorithm can be captured mathematically. In fact, it should be possible to integrate our router model with the prior network models [25–29] to analyze different QoS-aware cluster networks.

Like many commercial designs, we use a pipelined wormhole router architecture. The model considers an integrated workload consisting of C different classes of traffic. $(C-1)$ classes represent real-time applications with distinct service requirements. The last class is used for best-effort traffic. Each class is statically assigned at least one VC, and the VCs are scheduled with a rate-based scheduling algorithm, VirtualClock [23], to regulate the bandwidth requirements. The model computes average message latency (includes router/network latency and source queueing delay) for different classes, and the deadline missing probability for real-time workloads. While the first performance metric is an important criterion to evaluate the effectiveness of a network design with different workloads, the deadline missing probability is a QoS parameter for time-constrained applications.

Two main contributions of this analytic model are that it captures the chained blocking possible in pipelined wormhole-switched networks, and the bandwidth sharing mechanism of the VirtualClock algorithm in finding the average latency. Unlike the prior lumped delay models [25–27], here we analyze the contention at different stages of a pipelined router. Moreover, the average behavior analysis of the VirtualClock algorithm is applicable to other work conserving techniques such as Fair Queueing [22] and Weighted Round Robin (WRR).

We validate the single router model (16-port) and the cluster network model (up to 7-cubes) through extensive simulation. We use a mixed workload of three traffic classes ($C = 3$, two real-time and one best-effort) in this study. It is shown that the models are quite accurate in predicting the average delay and deadline missing probability. Using the model, it is not only possible to predict the per class QoS behavior, but also the impact of application mix. In addition, the model can quantify different components of a message latency (queueing time, transfer time, blocking time), effect of buffer length, and other design trade-offs in an effective manner. Thus, it can be used as an efficient design tool to analyze network and application centric performance parameters.

The rest of the paper is organized as follows. In Section 2, the router architecture and the VirtualClock algorithm are discussed. In Section 3, we present the analytic models. The performance results are analyzed in Section 4, followed by the concluding remarks in Section 5.

2. A QoS-aware router architecture

Most routers now use a pipelined design to minimize the network cycle time [30,31]. Accordingly, we use a pipelined, wormhole-switched router in this paper. Fig. 1 shows the pipelined router consisting of five stages. Stage 1 of the pipeline represents the functional units, which synchronize the incoming flits, demultiplex a flit so that it can go to the appropriate input virtual channel buffer to be subsequently

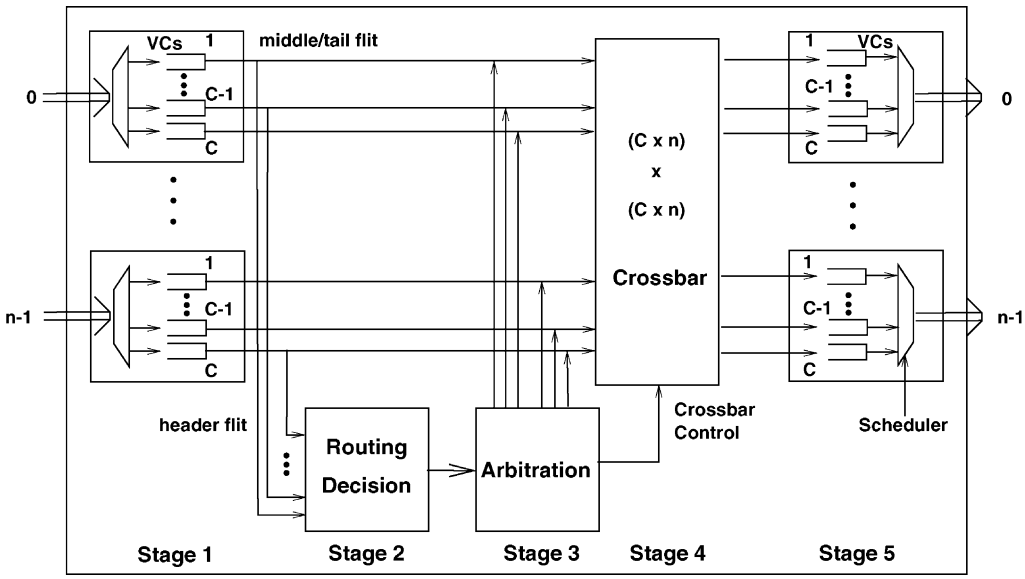

Fig. 1. The pipelined router architecture with a full crossbar.

decoded. If the flit is a header flit, routing decision and arbitration for the correct crossbar output is performed in the next two stages (Stage 2 and Stage 3). On the other hand, middle flits and the tail flit of a message directly move to Stage 4. Flits get routed to the correct crossbar output port in Stage 4. Finally, the last stage of the router performs buffering for flits flowing out of the crossbar, multiplexes the physical channel bandwidth amongst multiple VCs, and transmits one flit at a time to the neighboring router or to the network interface of the node attached to this router.

In this n-port router architecture, we provide one VC for each of the C traffic classes (thus C input and output VCs). More VCs per class should improve the performance. Note that the crossbar used in our router is called a *full crossbar* since it has $n \times C$ inputs and $n \times C$ outputs. The model can be modified for a multiplexed crossbar, where the VC multiplexing will be done before the crossbar stage.

Unlike the lumped router models analyzed before [32,26,27,29], a message entering the above pipelined router can experience delay at Stages 1, 3 and 5 of the router. In Stage 1, if the corresponding input buffer is full, the message must wait outside the router until adequate space is available. In Stage 3, the message again may be delayed because its destination crossbar output port could be busy. Crossbar output port arbitration is performed at a message level granularity. So the message has to wait until the output port is released by the message currently using it. Finally in Stage 5, multiple VCs compete for the physical channel bandwidth. Traditionally, a round Robin (RR) or FIFO scheduler is used to schedule the output channel in a time-division manner.

The above router design is modified to support QoS provisioning by simply incorporating a rate-based scheduling algorithm in Stage 5 to share the physical channel bandwidth. Similar techniques have been proposed for the Internet router line cards. We use the VirtualClock algorithm [23] in this paper although all the three rate-based scheduling algorithms (VirtualClock, Fair Queueing and Weighted Round Robin) are shown to provide similar performance.

In the VirtualClock algorithm, there are two variables, called *auxVC* and *Vtick* for each connection. The values of these two variables are determined when a connection is set up. The *auxVC* indicates the virtual clock value of the connection, while the *Vtick* is the amount of time that should be incremented whenever a flit arrives at that connection. The *Vtick* value specifies the interarrival time of packets from the connection. Therefore, a smaller *Vtick* value implies higher bandwidth. Once these two values are set, the VirtualClock algorithm works as follows. For each connection i, when a flit arrives at the scheduler, the following computation is done.

$auxVC_i \leftarrow \max(\text{real time}, auxVC_i)$,
$auxVC_i \leftarrow auxVC_i + Vtick_i$,
timestamp the packets with the $auxVC_i$.

In this study, we are interested in a connectionless paradigm without any explicit connection setup since this provides more efficient use of the network resources. Thus, we assign the *Vtick* value for each class not per a connection. The $Vtick_c$ value for real-time traffic belonging to class c is given by $1/(\lambda_c^g M)$ where λ_c^g is the arrival rate of class c and M the message length. This *Vtick* value specifies the interarrival time of flits from the same class.

The packets are queued and serviced in increasing timestamp order. For the best-effort traffic, the *Vtick* value is set as ∞. So, the best-effort flits are processed only if there are no other flits with lower timestamp values.

3. An analytical model

In this section, we develop a mathematical model for a single router and then extend it to a hypercube network. The motivation for developing a single router model is two fold. First, the model can be used for evaluating small, single node clusters. Second, it can be extended to any other topology by capturing the impact of the network and routing algorithm. The hypercube topology is used as an example in this paper to demonstrate the applicability of the model.

As described in the previous section, the router model assumes a pipelined architecture with $P = 5$ stages. The model is derived for C classes of traffic with different service requirements. Here, we assume that there are $(C - 1)$ real-time traffic classes and one class of best-effort traffic. Each class is assigned a dedicated VC. (This assumption can be relaxed to assign multiple VCs to a class.) In addition, the model is based on the following assumptions typically used in analytical models:

- The arrival pattern of each class c follows the Poisson processes with an average arrival rate of λ_c^g.
- Messages are M-flit long.
- Message destinations are uniformly distributed.
- The input and output buffers (VCs) in Stages 1 and 5 can hold b_s flits. Each class is assigned a dedicated injection/ejection queue outside the router, and these queues have infinite capacity.

To capture the burstiness of real-time traffic, we also use an ON/OFF source for real-time traffic. The ON/OFF traffic is generated as a stream of messages between a pair of source and destination nodes. During the OFF period, the source does not generate any messages, while during the ON period, an

exponentially distributed random number of messages, with an average N, are generated at a fixed rate p. The average generation rate a_i of stream i with the average OFF time, I, is given by $1/a_i = I/N + 1/p$. The average arrival rate of class c is $\lambda_c^g = \sum a_i$, where a_i is the average rate of stream i which belongs to class c. The ON/OFF model with exponentially distributed ON and OFF times is commonly used [33,34].

The average message latency of class c ($1 \le c \le C$) is composed of the average network latency, $\overline{L_c}$, which is the time to traverse the router (network), and the average waiting time, $\overline{W_c}$, at the injection channel. Thus,

$$\overline{Latency_c} = \overline{L_c} + \overline{W_c}. \tag{1}$$

3.1. Single router model

The average network latency ($\overline{L_c}$) of a message of class c consists of two parts. The first part is the actual message transfer time, T. The second part is due to blocking caused by the wormhole switching scheme, and due to sharing of the physical channel bandwidth by multiple VCs at Stage 5 of Fig. 1. The actual transmission time with P pipeline stages in a single router is $(P - 1 + M)$ cycles for an M-flit message.

In order to compute the second part of the network latency, let us define $\overline{B_c}$ as the average blocking length (in number of flits) seen by the header flit at the input, output, and arbitration stage in the router. $\overline{B_c}$ captures the message blocking in a pipelined wormhole router. Then the effective length of the message becomes $(M + \overline{B_c})$ flits. Let $\overline{S_c}$ be the average number of cycles required to transfer a flit of a class c message. $\overline{S_c}$ represents the effect of bandwidth sharing mechanism of the VirtualClock algorithm. Thus, the average network latency ($\overline{L_c}$) for $1 \le c \le C$ is

$$\overline{L_c} = P - 1 + (M + \overline{B_c})\overline{S_c}. \tag{2}$$

Since blocking can occur in Stages 1, 3, and 5 of the router as discussed in Section 2, the average blocking length ($\overline{B_c}$) can be separated into three parts as

$$I = P[\text{input buffer is not empty}] \cdot \left\{ \frac{\max(b_s, M)}{2} \right\},$$
$$O = P[\text{output buffer is not empty}] \cdot \left\{ \frac{\max(b_s, M)}{2} \right\},$$
$$A = P[\text{arbiter is busy}] \cdot \frac{M}{2},$$

where I, O, and A represent the corresponding blocking lengths. In each of the above expressions, the first term represents the probability that the corresponding buffer is not empty, and the second term is the average message length that will be affected due to blocking. For example, if the input buffer is not empty, the header flit will face an average delay of $\max(b_s, M)/2$ flits. In the steady state, since the input and output rates of the router are the same, it is intuitively clear that the above three probabilities are equal, and are denoted as $P_{b,c}$.

The technique to calculate the average blocking length is similar to the previous wormhole router models [25–29] except that we consider a pipeline router here. Since the pipelined worm-

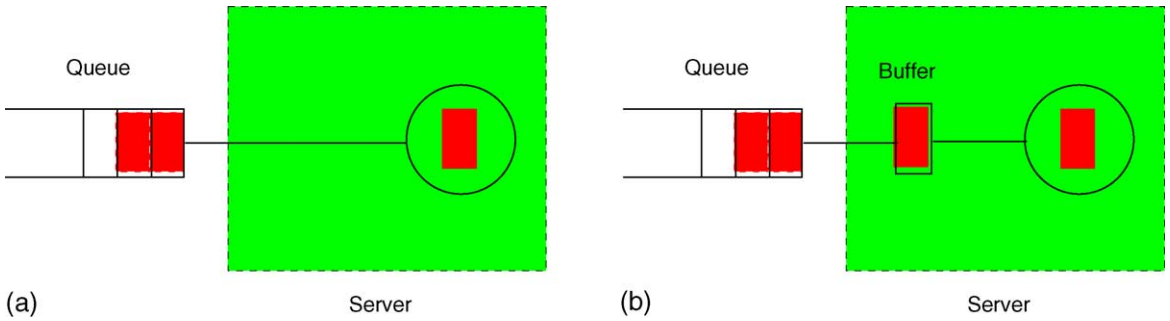

Fig. 2. Two server organizations. (a) Server without internal buffer, (b) Server with internal buffer.

hole router has input/output buffers, the blocking probability[2] should be obtained differently. While the blocking probability of a router without any buffer is the system utilization ($= \rho$) of an $M/G/1$ queue (shown in Fig. 2(a)), that of a router with one message buffer (shown in Fig. 2(b)) is $\sum_{n=2}^{\infty} P[\text{the number of customers in the system} = n] = \rho^2$ since blocking occurs when the buffer in the router is full and the router is busy. For a router with b buffers, the blocking probability can be generalized as $\sum_{n=1+b}^{\infty} P[X = n] = \rho^{(1+b)}$ where X is the number of messages in the router.

Since the input/output buffer sizes are b_s flits, the blocking probability, $P_{b,c}$ for class c ($1 \le c \le C$) can be expressed as

$$P_{b,c} = (\overline{L_c}\lambda_c')^{1+2(\max(b_s,M)/M)}, \tag{3}$$

where λ_c' is the steady state message arrival rate of class c traffic, and $(\overline{L_c}\lambda_c')$ the router utilization (ρ_c) for class c. Since λ_c' and $\overline{L_c}$ are considered at the message-level granularity, the total buffer size ($2b_s$ flits) of input and output queues becomes $2b_s/M$ when converted to message length. (Note that we are considering the worst case scenario here by using the entire buffer length $2b_s$.) Including the currently serviced message, the total number of messages becomes $2b_s/M + 1$. Hence, the channel utilization (or blocking probability of class c) is given by Eq. (3). The $\max(b_s, M)$ term is used to capture the buffer length $b_s < M$, since a new message must wait until the service for the previous message is complete. The steady state arrival rate λ_c' in Eq. (3) is given by

$$\lambda_c' = (1 - P_{b,c})\lambda_c^g. \tag{4}$$

Combining the three blocking lengths (I, O and A), $\overline{B_c}$ becomes

$$\overline{B_c} = P_{b,c}\left(\max(b_s, M) + \frac{M}{2}\right). \tag{5}$$

The next unknown term in Eq. (2) is $\overline{S_c}$, for $1 \le c \le C$. Since the scheduler treats a real-time message and a best-effort message differently, we compute $\overline{S_c}$ separately for the two broad classes of traffic. First we compute $\overline{S_c}$, $1 \le c \le C - 1$, for all real-time traffic based on a Markov model, and then use a different technique to compute $\overline{S_C}$ for best-effort traffic.

[2] Here, *blocking probability* implies that a flit cannot enter the corresponding stage of the router because of unavailable buffer.

When a real-time message of class c arrives at the output buffer, if other buffers are empty, it will take only one cycle to transfer a flit from that message. Otherwise, the output channel bandwidth is shared among the VCs according to the corresponding *Vtick* values. For example, when two output buffers are occupied by class i and j messages whose *Vtick* values are $Vtick_i$ and $Vtick_j$ respectively, the number of cycles to transfer a flit of class i (S_i) is $(1/Vtick_i + 1/Vtick_j)/(1/Vtick_i)$, and the number of cycles required for a flit of class j (S_j) is $(1/Vtick_i + 1/Vtick_j)/(1/Vtick_j)$.

With $(C-1)$ classes of real-time traffic, for any tagged class i, there are 2^{C-2} combinations of other real-time traffic that denote whether they occupy the corresponding VCs or not. All these combinations will affect the output channel bandwidth sharing. To model this effect, we number the combinations serially so that for each combination k we can compute the effective cycle time. We can express the average number of cycles per flit for real-time traffic c as

$$\overline{S_c} = \sum_{k=0}^{2^{(C-2)}-1} S_c(k) P_c(k) \tag{6}$$

where $S_c(k)$ is the number of cycles required for class c in the kth combination, and $P_c(k)$ the probability of kth combination for traffic c.

Let Z be the buffer state of the $(C-1)$ output VCs assigned to real-time traffic. Then it can be expressed with a bit string $Z = (d_1, d_2, \ldots, d_{C-1})$, where $d_i = 1$ if the VC i is occupied, or $d_i = 0$ otherwise. Let us define $|Z|_j = d_j$, for $1 \le j \le (C-1)$, where the state $Z = (d_1, d_2, \ldots, d_{C-1})$. Then $|Z|_j$ indicates the state of the jth output VC.

Definition 1. *Let Z be a given state where $|Z|_c = 1$. Thus, $Z = (d_1, d_2, \ldots, d_{c-1}, 1, d_{c+1}, \ldots, d_{C-1})$. The numbering function ν, for a given Z and c, returns the serial number for each output buffer combination as*

$$\nu(Z, c) = \sum_{j=1}^{C-2} d'_j 2^{j-1} \quad \text{where} \quad d'_j = \begin{cases} d_j & 1 \le j < c \\ d_{j+1} & c \le j \le C-2. \end{cases}$$

For a given real-time of class c that occupies VC c, we can find all combinations of the other $(C-2)$ VCs from the above expression. Given $(C-1)$ types of real-time traffic, and a state $Z = (d_1, d_2, \ldots, d_{C-1})$, where $|Z|_c = 1$ and $\nu(Z, c) = k$,

$$S_c(k) = \frac{(\sum_{\forall j, d_j = 1} 1/Vtick_j)}{(1/Vtick_c)}. \tag{7}$$

This generalization is obtained from the two class example discussed earlier. Next, the probability of kth combination for class c, $P_c(k)$, can be determined using a Markov model. For example, let us consider the state transitions for two classes of real-time traffic as shown in Fig. 3. The two-tuple notation denotes the presence or absence of the two traffic classes at the output VCs. Thus, state (1,0) denotes that the output buffer for class 1 is occupied and that of class 2 is empty. The four states in Fig. 3 are numbered Z_0 to Z_3. The transition rate between two adjacent states can be formalized as follows.

Let Z_1 be a state such that the cth output buffer is empty ($|Z_1|_c = 0$) and Z_2 be the state such that the cth output buffer is not empty($|Z_2|_c = 1$). The status of the rest $(C-2)$ buffers are all identical in the two states to make Z_1 and Z_2 adjacent. Using the definition of ν, we can find $\nu(Z_2, c) = k$. Now, the transition

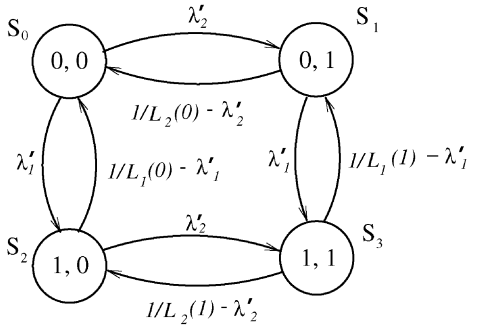

Fig. 3. State transition diagram with 2 classes of real-time traffic.

rate from state Z_1 to Z_2 is λ'_c, where λ'_c is the traffic rate of the cth class (Eq. (4)), while the rate from Z_2 to Z_1 is $(1/L_c(k) - \lambda'_c)$, where $L_c(k) = P - 1 + (\overline{B_c} + M)Z_c(k)$ from Eq. (2). The transition rate from Z_2 is reduced by λ'_c to account for the arrival of a message while channel c is busy. From the Markov model, we get all the state probabilities, Π_{Z_i}, $0 \le i \le 2^{C-1} - 1$. Then,

$$P_c(k) = \frac{\Pi_{Z_u}}{\sum_{\forall j, |Z_j|_c = 1} \Pi_{Z_j}}, \quad \text{where} \quad \nu(Z_u, c) = k. \tag{8}$$

Let us use the prior two class example to show the computation of $\overline{S_c}$. There are only two cases for each class to denote the bandwidth sharing (0: no sharing, 1: sharing). Let $S_2(0)$ be the number of cycles to transfer a flit of class 2 in the nosharing case. Then $S_2(0) = 1$. Let $S_2(1)$ be the number of cycles to transfer a flit of class 2 in the sharing case. Then $S_2(1) = (1/Vtick_1 + 1/Vtick_2)/(1/Vtick_2)$. To find the corresponding bandwidth sharing probabilities, we use the Markov model of Fig. 3. Considering a specific example in the figure, the rate from Z_2 to Z_3 is λ'_2 while the rate from Z_3 to Z_2 is $(1/L_2(1) - \lambda'_2)$. $L_2(1)$ using Eq. (2) becomes $(P - 1 + (\overline{B_2} + M)S_2(1))$. Similarly, $L_2(0)$ becomes $(P - 1 + (\overline{B_2} + M))$. Using these transition rates, we can find the state probabilities. The sharing probability of class 2, $P_2(1)$, is $\Pi_{Z_3}/(\Pi_{Z_1} + \Pi_{Z_3})$, and the nosharing probability of class 2, $P_2(0)$, is $\Pi_{Z_1}/(\Pi_{Z_1} + \Pi_{Z_3})$.

Average number of cycles to transfer a flit of best-effort traffic ($\overline{S_C}$): Note that the above computation for S_c is only for the $(C - 1)$ classes of real-time traffic. Since the *Vtick* value for the best-effort traffic is set to infinite, the best-effort message only uses the empty cycles when there is no real-time traffic. Moreover, transfer of best-effort flits can be interrupted if a real-time message arrives at the output buffer, and should be resumed after the real-time traffic transfer is complete. This can be modeled as a *preemptive resume priority queue* [35].

We model this phenomenon by computing the overall time to transfer a best-effort message of M flits long. This time, denoted as S_m, consists of three parts. The first part is the actual service time, which is M cycles. The second part is the average waiting time of the best-effort message or the residual service time of all real-time messages already in the output buffers. Note that the situation here is slightly different from the original *preemptive resume priority queue discipline* model in that, in our case we have already included the waiting time in the input and output buffers when we calculated the blocking length in Eq. (5). Hence, we only need the waiting time when the best-effort message is at the head of output buffer. This is written as $R_r/(1 - (\rho_1 + \rho_2 \ldots + \rho_{C-1}))$, where R_r is the residual time of all real-time messages

in the output buffers, and is given by $\sum_{c=1}^{C-1} \lambda'_c M^2/2$ [35]. The last part of the delay is due to preemption of the best-effort traffic to yield to any of the $(C-1)$ classes of real-time traffic. This inflates the overall transfer time by the channel utilization of $(C-1)$ classes. S_m now becomes

$$S_m = M + \frac{R_r}{1-(\rho_1+\rho_2+\cdots+\rho_{C-1})} + \sum_{i=1}^{C-1} \rho_i S_m.$$

The average number of cycles to transfer a best-effort flit after simplification becomes

$$\overline{S_C} = \frac{S_m}{M} = \frac{(2-\rho_r)}{2(1-\rho_r)^2}. \tag{9}$$

With Π_0 as the probability that there is no real-time traffic, we can find $\rho_r = 1-\Pi_0$, where $\rho_r = \sum_{i=1}^{C-1} \rho_i$. We can get Π_0 from the previous Markov model, where state 0 is $(d_1, d_2, \ldots, d_{C-1})$, $\forall j, d_j = 0$. All the terms in Eq. (2) are now quantified to compute $\overline{L_c}$. Note that due to the inter-dependency between $P_{b,c}$ and λ'_c, the solution becomes iterative.

3.2. Modeling of a cluster interconnect

The single router model can be extended to most of the regular networks as long as the topology and routing algorithm can be captured analytically. Models for such topologies like hypercubes, meshes, and k-ary n-cubes have been developed to predict the performance of best-effort traffic [25–29]. Here, we consider integrated traffic in the network, and use a hypercube topology to demonstrate this idea. We use the deadlock-free e-cube routing algorithm for message transfer.

The wormhole-switched hypercube (n-cube) model proposed in [26] is combined with our pipelined router model to compute the average message latency. The motivation for using the model proposed in [26] is that it is not only quite accurate over the entire workload, but also computes the message latency per link, which is required for accurate performance estimates per connection.

In an n-cube network, each node has n input and n output links in addition to an injection and an ejection channel for the local host. Messages generated by a node could travel h-hops, where $1 \le h \le n$. Thus, each physical channel is likely to experience a different load, and therefore, the single router model of the previous section should be modified to express traffic analysis for each link s, where $0 \le s \le n-1$.

3.2.1. Average network latency ($\overline{L_c}$)

The actual transmission time with a P-stage router in an n-cube ($\bar{T}$) is $(P-1+P\bar{h}+M)$ cycles, where $\bar{h}$ is the average number of hops a message travels in the hypercube. The average number of hops is given by $\bar{h} = \sum_{k=1}^{n} kP_k$, where $P_k = {}_nC_k/(N-1)$, and ${}_nC_k = \binom{n}{k}$. $N\ (=2^n)$ is the number of nodes in an n-cube. The average network latency $\overline{L_c}$ of Eq. (2) needs to be modified to consider the latency for each class c when it starts with a specific physical channel. Let $L_{c,s}$ be the latency of a class c message when it uses the physical channel s as the first path to traverse towards its destination. Then $h_s = \sum_{k=0}^{n-s-1}(k+1)\cdot {}_{n-s-1}C_k/(2^{n-s-1})$ denotes the average number of hops a message travels starting with the physical channel s as the first path.

The network contention in the hypercube network is divided into three separate parts—blocking at the input stage of the first router, blocking at the ejection channel of the last router, and blocking in the middle routers. Let $I_{c,s}$ be the blocking length of a class c real-time message at Stage 1 of the first router that uses channel s as the first route, and let $O_{c,n}$ be the blocking length at Stages 3 and 5 in the ejection channel of the last router. Also, let $B_{\text{middle}}(c, s)$ be the blocking length between the source and the destination (*i.e.* middle nodes) excluding the blocking length at Stage 1 of the source and the blocking length at Stages 3 and 5 of the destination. Note that $B_{\text{middle}}(c, s)$ captures the effect of chained blocking possible along the path while $I_{c,s}$ and $O_{c,n}$ capture delay due to input and output stages being busy.

To compute the average number of cycles required to transfer a flit due to sharing of real-time messages at the output VCs, we need to consider two separate cases again—sharing at the ejection channel and in the rest of the channels. Consequently, let $\overline{S_{c,s}}$ be the average number of cycles required to transfer a flit of class c message that uses channel s for its first path, and $\overline{S_{c,n}}$ be the average number of cycles required per flit in the ejection channel. With these definitions, $L_{c,s}$ can be expressed as

$$L_{c,s} = \{P - 1 + Ph_s\} + \{(O_{c,n} + M) \cdot \overline{S_{c,n}}\} + \{(I_{c,s} + B_{\text{middle}}(c, s)) \cdot \overline{S_{c,s}}\}. \tag{10}$$

The first term in Eq. (10) indicates the number of cycles the header will take without contention. The second term represents the transfer time at the ejection channel (n). The total message length that includes the blocking length ($O_{c,n}$) at the ejection channel and the message length (M) is multiplied by the average number of cycles required to transfer a flit ($\overline{S_{c,n}}$) at the ejection channel. Similarly, the total message length at the input buffer ($I_{c,s}$) and in the middle nodes is multiplied by the inflated cycle time ($\overline{S_{c,s}}$) to find the last term of Eq. (10). The average network latency ($\overline{L_c}$) becomes

$$\overline{L_c} = \sum_{s=0}^{n-1} L_{c,s} \cdot \frac{\lambda'_{c,s}}{\lambda'_c},$$

where λ'_c is the steady state message generation rate of class c, while $\lambda'_{c,s}$ is the steady state message generation rate of class c for channel s. We need three types of traffic rates to complete the delay analysis. The first one is $\lambda^g_{c,s}$ which represents the message generation rate of class c for channel s, the second is $\lambda'_{c,s}$, and the third is $\lambda_{c,s}$, the total steady state rate for channel c (including transit message). We have used the traffic analysis equations given in [26]. For better readability, we defer the traffic rate equations to the Appendix.

Now from Eq. (3), the probability of blocking for class c traffic in channel s can be written as

$$P^s_{b,c} = (L_{c,s}\lambda_{c,s})^{1+2(\max(b_s,M)/M)}. \tag{11}$$

Note that $\lambda_{c,s}$ is the total rate here. Similarly Eq. (4) is modified as

$$\lambda'_{c,s} = (1 - P^s_{b,c})\lambda^g_{c,s}. \tag{12}$$

The $I_{c,s}$ and $O_{c,n}$ terms in Eq. (10) are similar to the I term and $\overline{B_c}$ in the single router and are given by $\overline{I_{c,s}} = (P^s_{b,c} \cdot \max(b_s, M)/2)$, and $\overline{O_{c,n}} = (P^n_{b,c} \cdot (\max(b_s, M)/2 + M/2))$. Finally the average number of cycles for transferring a flit at the ejection channel ($\overline{S_{c,n}}$) and at the other output buffers ($\overline{S_{c,s}}$) can be obtained from Eqs. (6)–(8) after modifying the terms for representing the starting channel s for class c traffic. The only unknown term in Eq. (10) is $B_{\text{middle}}(c, s)$, which again for better readability, is deferred to the Appendix A.

Average number of cycles for transferring a flit of Best Effort traffic ($\overline{S_{c,s}}$): Similar to the single router model, now we need to compute the inflated number of cycles to transfer best-effort traffic. However, we cannot use the *preemptive resume priority queue model* alone here due to the fact that the idle period between best-effort flits becomes completely random while traversing through the network. Therefore, we use the *Busy and Idle period* concepts from the $M/G/1$ queue to find $\overline{S_{c,s}}$ and $\overline{S_{c,n}}$.

Let *Busy* be the average length of busy period and *Idle* be the average length of idle period for real-time traffic. Given the total rate of real-time traffic $\lambda_r^g = \sum_{i=1}^{C-1} \lambda_i^g$, the *Idle* and *Busy* periods become $Idle = 1/\lambda_r^g$ and $Busy = Idle(1 - \Pi_0)/\Pi_0$, where Π_0 is the probability that the server is idle ($Idle/(Busy + Idle) = \Pi_0$).

A best-effort message could arrive during the idle period or busy period of real-time traffic.[3] If the message arrives during the idle period, then the message completion time is M cycles. If it arrives during the busy period, the completion time is $(M + Busy)$. In addition, a real-time traffic could arrive during the transmission of a best-effort traffic. This increases the message completion time by $(M \cdot \rho_r)$ or $(M + Busy) \cdot \rho_r$ corresponding to the above two cases, where ρ_r is the channel utilization due to all real-time traffic as explained in Eq. (9). Hence, the total transfer time is given by

$$S_m(s) = (M + M \cdot \rho_r)\Pi_0 + (M + Busy + (M + Busy) \cdot \rho_r)(1 - \Pi_0). \tag{13}$$

Then $\overline{S_{c,s}}$ can be obtained by dividing $S_m(s)$ by the message length M.

To compute $\overline{S_{c,n}}$ for best-effort traffic in Eq. (10), we need to capture the delay between successive flits of a best-effort message. Unlike most real-time transmissions, after the header flit of a best-effort message arrives at the destination, the remaining flits will arrive in random intervals due to delay at different hops of the network. Let the effective length of a best-effort message (M_s flits) be the difference between the arrival time of the header flit and that of the tail flit when a message uses channel s as the first path. Each one of the $(M - 1)$ flits will need $\overline{S(C, s)}$ cycles to transfer. However, the overall time can be reduced due to blocking of the header flit in the middle routers, which is given by $B_{\text{middle}}(c, s)$. Thus, the effective length of a best-effort message becomes $M_s = \max(M, (M - 1)\overline{S(C, s)} + 1 - B_{\text{middle}}(C, s))$. The extra one cycle in this expression represents the header flit transfer time at the ejection channel.

Now using this effective message length, we compute the transfer time at the ejection channel by considering the two cases used in deriving Eq. (13). While the best-effort message yields to real-time traffic during a busy period, flits of the best-effort message gets accumulated in the buffer. So the effective length of best-effort message decreases. To quantify this phenomenon, the $(M + Busy)$ term in Eq. (13) is replaced by $(M_s/2 + Busy + M/2)$. Thus, the ejection channel transfer time is

$$S_m(n) = (M_s + M_s \cdot \rho_r)\Pi_0 + \left(\frac{M_s}{2} + Busy + \frac{M}{2} + \left(\frac{M_s}{2} + Busy + \frac{M}{2}\right) \cdot \rho_r\right)(1 - \Pi_0).$$

Then, $\overline{S_{c,n}}$ can be obtained by dividing $S_m(n)$ by the message length M.

[3] Actually there are three types of arrival epochs. In the third case, the arrival could span over the busy and idle period. For simplicity, we handle only two cases here.

3.3. Average waiting time at the source node

Finally, we need the average waiting time at the source node ($\overline{W_c}$) to find the average message latency in Eq. (1). Since the average waiting time accounts for the time consumed outside the router, it can be seen as the delay in the Network Interface (NI). Typically the delay is due to two data transfers involved in the NI. One is from the host to the NI and the other is from the NI to the router or network. In this model we only consider the transfer time from the NI to the network. To facilitate real-time scheduling in the NI, we assume that the NI buffer is divided into C classes and the VirtualClock algorithm is implemented among the C buffers [19].

The average waiting time consists of two parts. The first part is the time spent in the injection channel before arriving at the head of the respective queue. Since each class has a dedicated injection buffer, messages are transfered in FIFO manner within each class, Thus, the waiting time can be obtained by the queueing time of an $M/G/1$ queue as $(\lambda_c^g \overline{L_c}^2(1+\delta^2/\overline{L_c}^2))/(2(1-\lambda_c^g\overline{L_c}))$ with an arrival rate λ_c^g, mean service time $\overline{L_c}$, and variance $\delta^2 \approx (\overline{L_c}-\overline{T})^2$. The second part is the delay due to the VirtualClock algorithm, and is given by the average number of cycles required to transfer the header flit of a message to the network. This is simply $\overline{S_c}$ in Eq. (6) for the single router model. For modeling of a cluster interconnect, we can get $\overline{S_c}$ by

$$\overline{S_c} = \sum_{s=0}^{n-1} S_{c,s} \cdot \frac{\lambda'_{c,s}}{\lambda'_c} \tag{14}$$

where $S_{c,s}$ is the average number of cycles required to transfer a flit of class c message which uses physical channel s for its first path. $\lambda'_{c,s}$ and λ'_c are defined before while deriving $\overline{L_c}$.

Summation of two parts yields the average waiting time as

$$\overline{W_c} = \frac{\lambda_c^g \overline{L_c}^2(1+\delta^2/\overline{L_c}^2)}{2(1-\lambda_c^g\overline{L_c})} + \overline{S_c}. \tag{15}$$

3.4. Deadline missing probability

In the previous sections, we derive the average message latency for all traffic classes. In addition to message latency, deadline missing probability (DMP) of time-constrained applications has been used as a performance parameter [14,36]. Since a wormhole-switched network cannot provide a hard guarantee due to chained blocking, the system can provide a soft guarantee in terms of the probability of missing a deadline. For a given source and destination pair, the probability of missing the deadline is the probability that a message cannot be delivered within a specified time (D). The above definition can be generalized for the average case. Here, we consider the deadline to traverse a network. Source queueing is not included to keep the discussion simple. Using the network latency of class c, we can find the deadline missing probability as follows.

3.4.1. Deadline missing probability in a single router

We compute the DMP for a class c traffic in a single router. The network latency of class c, L_c, is a discrete random variable, L_c. Like Eq. (2), the network latency (L_c) for $1 \le c \le C$ can be written as

$$L_c = (B_c + M)S_c + P - 1, \tag{16}$$

where B_c is the message blocking in a pipelined wormhole router and S_c the effect of bandwidth sharing mechanism of the VirtualClock algorithm.

While blocking happens among the same class of messages, the sharing depends on the traffic of other classes. Thus, these two random variables (B_c and S_c) are independent. We can combine them to a random parameter, $\beta_c = (B_c + M)S_c$. We know that $\beta_c = L_c - (P-1)$ and $\beta_c \geq M$.

Let $P_{m,c}(D)$ be the probability of missing the deadline D of class c. If we can find the c.d.f. of L_c, $P\{L_c \leq D\}$, then $P_{m,c}(D)$ is $1 - P\{L_c \leq D\}(= 1 - P\{\beta_c \leq D'\})$, where $D' = D - (P-1)$.

For an accurate estimation of β_c, first we consider the two random variables(B_c and S_c) separately and then combine them. To compute the blocking length B_c, note that blocking is possible at the input buffer stage, output buffer stage and arbitration stage. The worst case of blocking occurs when all these places are occupied by other messages. Thus the worst blocking length will be $(2\max(b_s, M) + M)$ where b_s is the input/output buffer size and M the message length. Let us assume that we know the probability mass function $P_{m,c}(B)$ of B_c ($P_{m,c}(B) = P\{B_c = B\}$), which will be described later.

With a given blocking delay(B), the effective message length will be $(M + B)$. When each flit of $(M + B)$ arrives at the head of the output VC, there are $2^{(C-2)}$ combinations of other real-time traffic that denote whether they occupy the corresponding output VCs or not. All these combinations will determine how to share the bandwidth.

Let $X_c(i)$ be the number of flits, which needs $S_c(i)$ cycles at the output VC such that $\sum_{i=0}^{2^{(C-2)}-1} X_c(i) = B + M$, given that the blocking length is B. Then β_c, the actual delay for a blocking length B can be denoted as

$$\beta_c = \sum_{i=0}^{2^{(C-2)}-1} X_c(i)S_c(i).$$

Let's define the c.d.f. of β_c, $P\{\beta_c \leq D'\}$, as

$$P\{\beta_c \leq D'\} = \sum_{B=0}^{B^u} \sum_{X_0=0}^{X_0^u} \cdots \sum_{X_{2^{(C-2)}-1}=0}^{X^u_{2^{(C-2)}-1}} P_{m,c}(B)P_{x,c}(X_0, 0|B) \ldots P_{x,c}(X_{2^{(C-2)}-1}, 2^{(C-2)} - 1|B). \quad (17)$$

There are $(2^{(C-2)} + 1)$ summation notations in Eq. (17). The first notation is for B and the remaining $2^{(C-2)}$ notations correspond to the total number of combinations of the output VC status. In Eq. (17), $P_{x,c}(X, i|B)$ is the probability that $X_c(i) = X$ given the blocking length is B. B^u, the upper bound of B, is $2\max(b_s, M) + M$ which represents the worst case of blocking, $X_0^u = \min(B + M, D'/S_c(0))$, and $X_i^u = D' - \sum_{j=0}^{i-1} S_c(j)X_j/S_c(i)$, for $1 \leq i \leq 2^{(C-2)} - 1$.

We need the solution of $P_{m,c}(B)$ and $P_{x,c}(X, i|B)$ to find the deadline missing probability. Since the exact estimation of the terms is extremely hard, we approximate these probabilities from the operational behavior of the router/network. Using $P_{b,c}$ in Eq. (3), we can get $P_{m,c}(B)$ as follows. Since the blocking length(B_c) varies between 0 and $2\max(b_s, M) + M$, under the uniform distribution assumption, $P_{m,c}(B)$ can be written as

$$P_{m,c}(B) \approx \begin{cases} 1 - P_{b,c}, & B = 0 \\ P_{b,c}/(2\max(b_s, M) + M), & 1 \leq B \leq 2\max(b_s, M) + M \\ 0, & \text{otherwise.} \end{cases} \quad (18)$$

Similarly we can get $P_{x,c}(X, k|B)$ using $P_c(k)$, which is derived in Eq. (8). Since $\overline{X_c(i)} = P_c(i) \cdot (B + M)$ for a given B, we assume that $X_c(i)$ varies between 0 and $(B + M)$ under the uniform distribution assumption. Hence,

$$P_{x,c}(X, k|B) \approx \begin{cases} 1 - P_c(k), & X = 0 \\ P_c(k)/(B + M), & 1 \le X \le (B + M) \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

3.4.2. Deadline missing probability in a hypercube interconnect

The DMP for the single router model is extended to compute the DMP for a class c traffic that traverses h hops in an n-cube network. We start with the network latency expression, L_c. The network latency for a given *Path*, which is a set of h channels traversed by a message using e-cube routing, can be modified as

$$L_c = \sum_{s \in \text{Path}} (B_{c,s} S_{c,s} + P) + ((B_{c,n} + M) S_{c,n} + P - 1), \tag{20}$$

where $B_{c,s}$ $(0 \le s \le n - 1)$ is the blocking length seen by a header flit of class c in channel s, and $S_{c,s}$ the number of cycles required for transferring a flit of class c in channel s. The first term in Eq. (20) represents the time spent at each hop, and the last term denotes the time at the ejection channel. Note that this does not include the queueing delay outside the router. As in the single router case, we combine the two random variables to $\beta_{c,s}$. Thus, we write $\sum_{s \in \text{Path}_e} (\beta_{c,s}) = L_c - (P - 1 + Ph)$, where Path_e includes the ejection channel ($\text{Path}_e = \text{Path} \cup \{n\}$). $\beta_{c,s}$ is given by $\beta_{c,s} = B_{c,s} S_{c,s}$, $0 \le s \le n - 1$ and $\beta_{c,n} = (B_{c,n} + M) S_{c,n}$. Let $D' = D - (P - 1 + Ph)$ and $P_{h,c,s}(D_s)$ be the p.m.f of $\beta_{c,s}(P_{h,c,s}(D_s) = P\{\beta_{c,s} = D_s\})$.

Let $P_{m,c}(D)$ be the probability of missing the deadline D for a given $Path = \{s_1, s_2, \ldots s_h\}$. If we can find the c.d.f. of L_c, $P\{L_c \le D\}$, then $P_{m,c}(D)$ is $1 - P\{L_c \le D\} (= 1 - P\{\sum_{s \in \text{Path}_e} \beta_{c,s} \le D'\})$. Considering the delay of each hop$(\beta_{c,s_1}, \beta_{c,s_2} \ldots)$ independent to each other, we can write

$$P\left\{ \sum_{s \in \text{Path}_e} (\beta_{c,s}) \le D' \right\} = \sum_{D_{s_{h+1}}=M}^{D^u_{s_{h+1}}} \cdots \sum_{D_{s_1}=0}^{D^u_{s_1}} P_{h,c,s_1}(D_{s_1}) \ldots P_{h,c,s_{h+1}}(D_{s_{h+1}}). \tag{21}$$

The above equation has $(h + 1)$ terms corresponding to the $(h + 1)$ hops a message travels(including the ejection channel). The lower bound of each hop except the $(h + 1)$th hop is zero. From $\sum_{s \in \text{Path}_e} (\beta_{c,s}) \le D'$, we can get the upper bounds as $D^u_{s_i} = D' - \sum_{j=i+1}^{h+1} D_{s_j}$ and $D^u_{s_{h+1}} = D'$.

We can compute the p.m.f $(P_{h,c,s}(D_s))$ in Eq. (21) from the c.d.f. $(P\{\beta_{c,s} \le D_s\})$ by $P_{h,c,s}(D_s) = P\{\beta_{c,s} \le D_s\} - P\{\beta_{c,s} \le D_s - 1\}$. Rewriting Eq. (17), we obtain $P\{\beta_{c,s} \le D_s\}$ as

$$P\{\beta_{c,s} \le D_s\} = \sum_{B=0}^{B^u} \sum_{X_0=0}^{X^u_0} \cdots \sum_{X_{2^{(C-2)}-1}=0}^{X^u_{2^{(C-2)}-1}} P_{m,c,s}(B) P_{x,c,s}(X_0, 0|B) \ldots P_{x,c,s}(X_{2^{(C-2)}-1}, 2^{(C-2)} - 1|B). \tag{22}$$

As explained in Eq. (17), there are $2^{(C-2)} + 1$ summation notations in Eq. (22). $P_{m,c,s}(B)$ is the p.m.f of $B_{c,s}$ and $P_{x,c,s}(X, i|B)$ the probability that $X_{c,s}(i) = X$ for a given B $(\sum_{i=0}^{2^{(C-2)}-1} X_{c,s}(i) = B)$.

$\beta_{c,s} = \sum_{i=0}^{2^{(C-2)}-1} X_{c,s}(i) S_{c,s}(i)$. B^u is defined in the single router model, while $X_0^u = \min(B, D_s/S_{c,s}(0))$, $X_i^u = D_s - \sum_{j=0}^{i-1} S_{c,s}(j) X_j / S_{c,s}(i)$. For the ejection channel ($s = n$), the c.d.f. is slightly different, and should include the message length M with blocking length B. So, $P_{x,c,s}(X, i|B)$, $0 \le i \le 2^{(C-2)} - 1$, will be replaced by $P_{x,c,n}(X, i|B + M)$, $0 \le i \le 2^{(C-2)} - 1$. Also, the upper bound of X_0 changes to $X_0^u = \min(B + M, D_s/S_{c,n}(0))$.

From Eq. (18), $P_{m,c,s}(B)$ can be written as

$$P_{m,c,s}(B) \approx \begin{cases} 1 - P_{b,c}^s, & B = 0 \\ P_{b,c}^s/(2\max(b_s, M) + M), & 1 \le B \le 2\max(b_s, M) + M \\ 0, & \text{otherwise,} \end{cases} \tag{23}$$

where $P_{b,c}^s$ is the blocking probability of class c in channel s as per Eq. (11).

Similarly from Eq. (19), we can get $P_{x,c,s}(X, k|B)$ from $P_{c,s}(k)$ as

$$P_{x,c,s}(X, k|B) \approx \begin{cases} 1 - P_{c,s}(k), & X = 0 \\ P_{c,s}(k)/B, & 1 \le X \le B \\ 0, & \text{otherwise.} \end{cases}$$

Note that all these equations can be derived from the single router model for a given number of hops(h) and for a physical channel s by setting the proper boundary values.

4. Performance results

In this section, we analyze the performance results for a 16-port router and n-cubes of various sizes. The performance parameters are average message/network latency (in cycles) and deadline missing probability(DMP). To validate the analytical models, we have developed a flit-level simulator using CSIM. The default parameters used in this study are given in Table 1. In the following subsections, we discuss only a selected set of results.

The results are reported for a mixed workload of two real-time (R1, R2) and one best-effort (BE) traffic types. For a given real-time load in messages/cycle, we generate two types of real-time traffic such that the intergeneration time of the second type is twice that of the first type. For example, if the input load is 0.01, then the intergeneration time for the first real-time traffic (R1) is 100 cycles, and for the second real-time traffic (R2) is 200 cycles. If there are 3 types of real-time traffic, the intergeneration time of R2 is given by 1.5× (intergeneration time of R1) and that of R3 is given by 2× (intergeneration time of R1). So in the following figures, real-time load implies the message generation rate of R1 only. The actual link load should include R1, R2, and BE. Note that these intergeneration times are used to

Table 1
Simulation parameters

Switch size	Single router : 16×16 n-cube : $(n + 1) \times (n + 1)$
Message size (M)	32 flits
Input/output buffer size	32 flits (or 4, 16, 64)
Number of VCs	3 (or 4)

represent the *Vtick* values ($Vtick_c = 1/(\lambda_c^g M)$. Best-effort traffic load is generated independent of the real-time traffic. After determining the intergeneration time for each class, messages are generated using exponential distribution.

4.1. Single router results

We first validate the analytical model with simulation results for the 16-port router. Fig. 4(a) shows the variation of average network latency ($\overline{L_c}$) for the three classes of traffic. The analytical and simulation results differ at most by 5%. The graphs exhibit the QoS ability of the router in that both classes of real-time traffic incur smaller latency compared to the best-effort traffic, and also the R1 traffic with a smaller *Vtick* value has better performance than the R2 class with a higher *Vtick* value. (Although not shown in the figure, it was observed that by replacing the VirtualClock algorithm with Round Robin, the performance differences were lost. Rather R1 latency was higher than R2 latency since its input load was higher.)

Fig. 4(b) depicts the effect of the VirtualClock scheduler. Since the best-effort messages are serviced only when there are no real-time messages, and a separate VC is provided for each type of traffic, the best-effort load variation does not affect the real-time traffic latencies.

Fig. 5 plots individual components of the average message latency ($Latency_c$) and how they change with the real-time load. In the figure, the queueing time represents the average waiting time ($\overline{W_c}$), the transfer time is given by (T), and ($\overline{L_c} - T$) represents delay due to blocking and sharing. By comparing Figs. 4(a) and 5, it is evident that with the inclusion of the average waiting time ($\overline{W_c}$), the latency difference between the real-time traffic and best-effort traffic increases. Also, the graphs indicate that with increasing real-time load, the waiting time and the blocking time of best-effort traffic increases faster compared to those of real-time traffic.

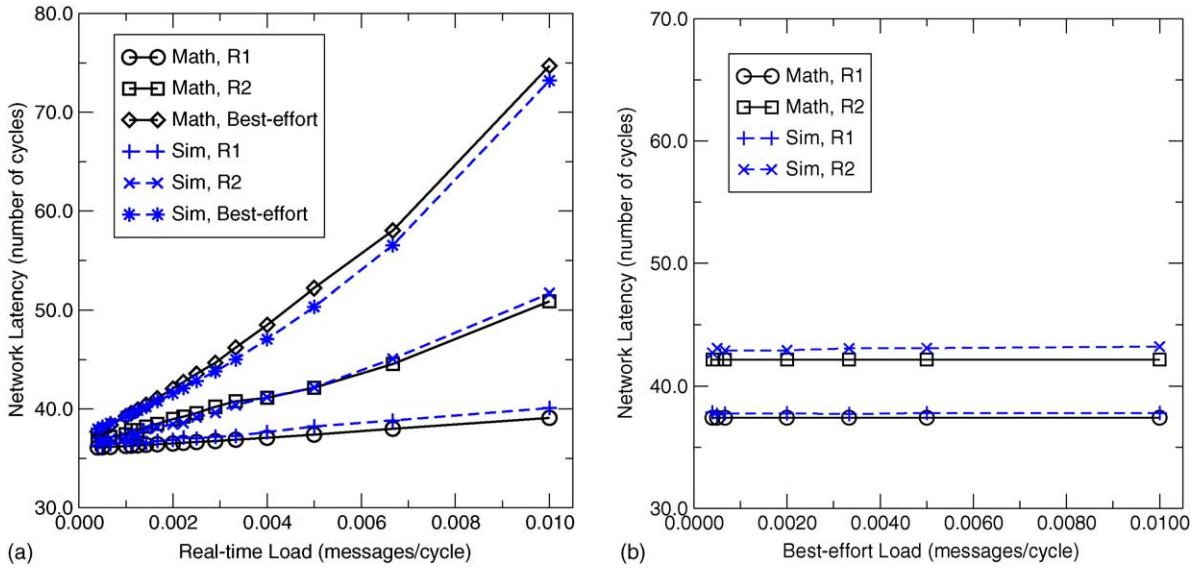

Fig. 4. Network latency comparison of analytical model and simulation model in a 16-port router with (a) varying real-time load and fixed best-effort load (0.01 msgs/cycle), and (b) varying best-effort load and fixed real-time load (R1: 0.005 msgs/cycle).

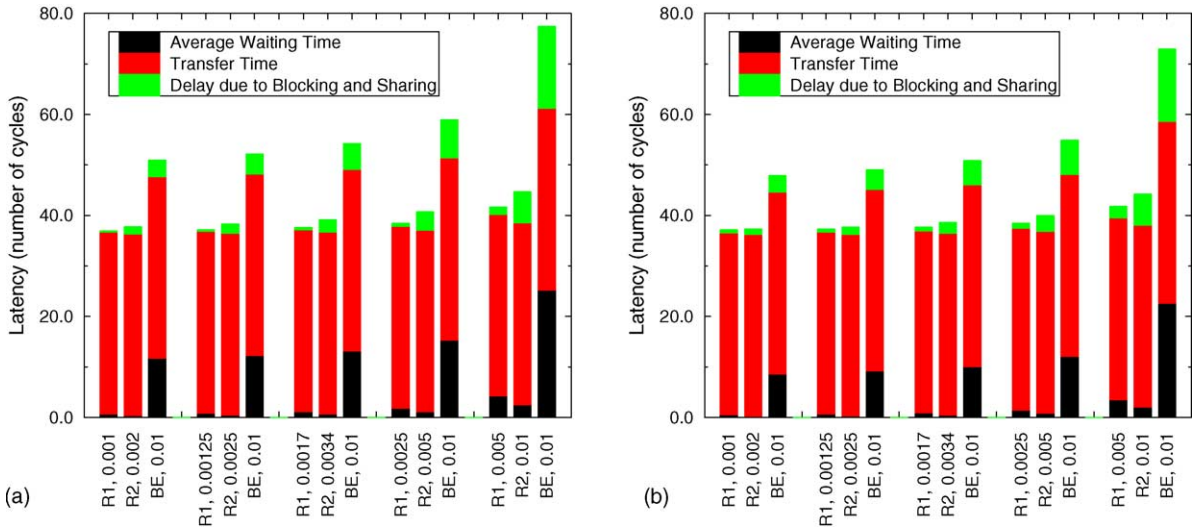

Fig. 5. Components of message latency in a 16-port router with varying real-time load and fixed best-effort load (0.01 msgs/cycle). (a) Analytical model, (b) simulation model.

Fig. 6 shows the simulation results for Fair Queueing, VirtualClock and Weighted Round Robin scheduling algorithms in order to reconfirm that these three algorithms have the same/similar performance [37]. The average network latency curves for each traffic type (R1, R2, BE) match over the entire workload. We also get similar performance from our analytical model for the VirtualClock algorithm, as

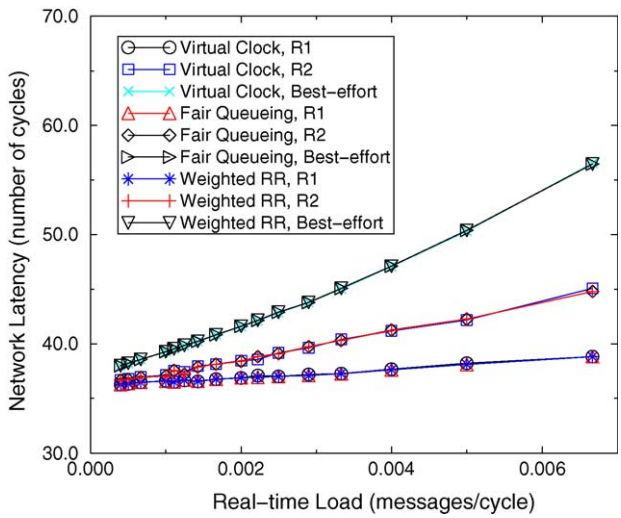

Fig. 6. Comparison of VirtualClock, Fair Queueing, and Weighted Round Robin in the 16-port router with varying real-time load and fixed best-effort load (0.01 msgs/cycle).

depicted in Fig. 4 (a). Thus, we can restrict our discussion to only the VirtualClock algorithm, although the results are applicable to the other two scheduling schemes. Moreover, unlike the bounding analysis reported in [38,39,37], here we can predict the average behavior of the work conserving scheduling mechanisms.

4.2. n-cube results

Fig. 7(a)–(c) depict the average network latency ($\overline{L_c}$) results form the analytical and simulation models in a 5-, 6- and a 7-cube, respectively. The figures reveal that the analytic results closely match with the simulation results. For the 7-cube, the error in the best-effort results is relatively large for higher workload. This is due to the fact that with the same total load, the best-effort traffic enters the saturation region faster.

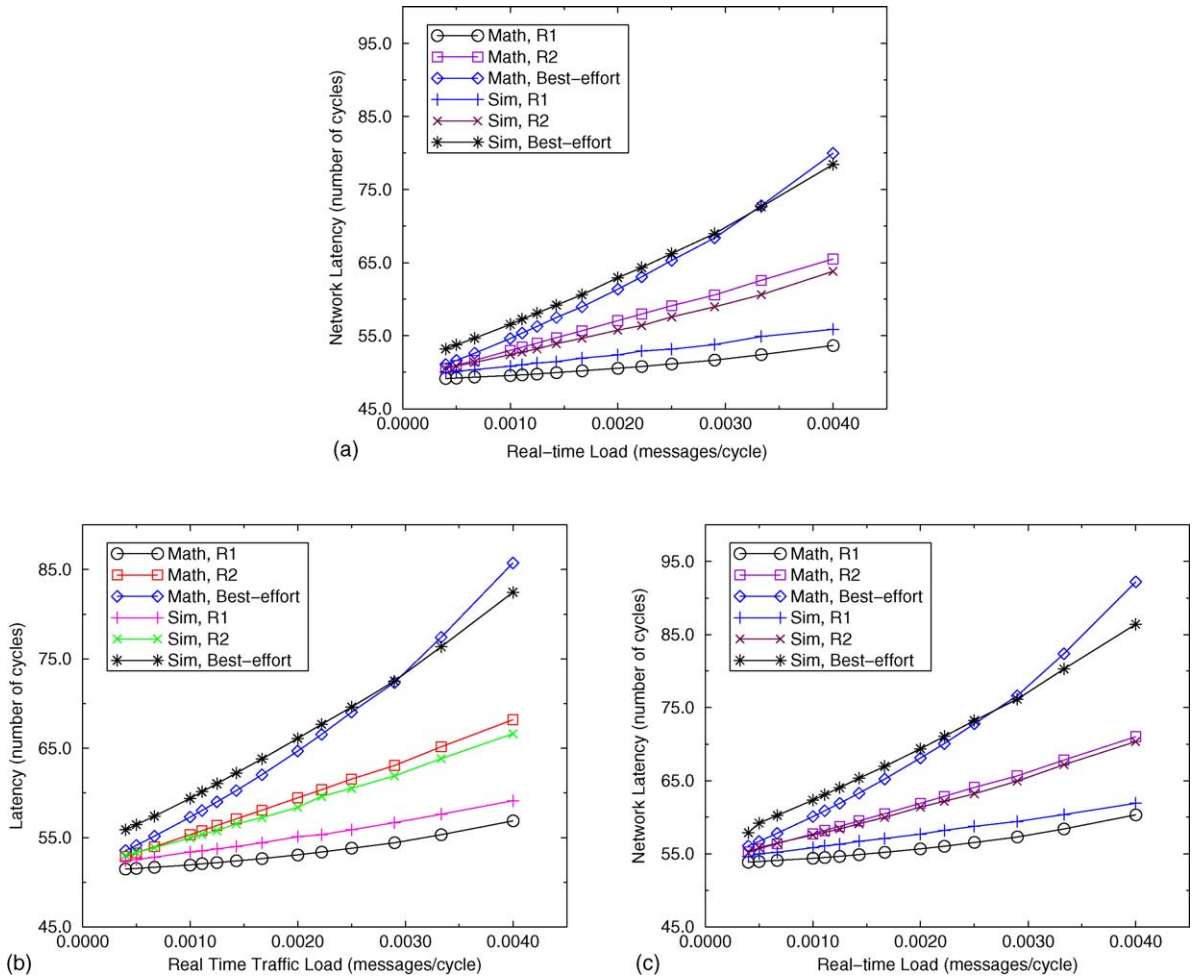

Fig. 7. Network latency comparison of analytical and simulation models in (a) a 5-cube, (b) a 6-cube and (c) a 7-cube with varying real-time load and fixed best-effort load (0.002 msgs/cycle).

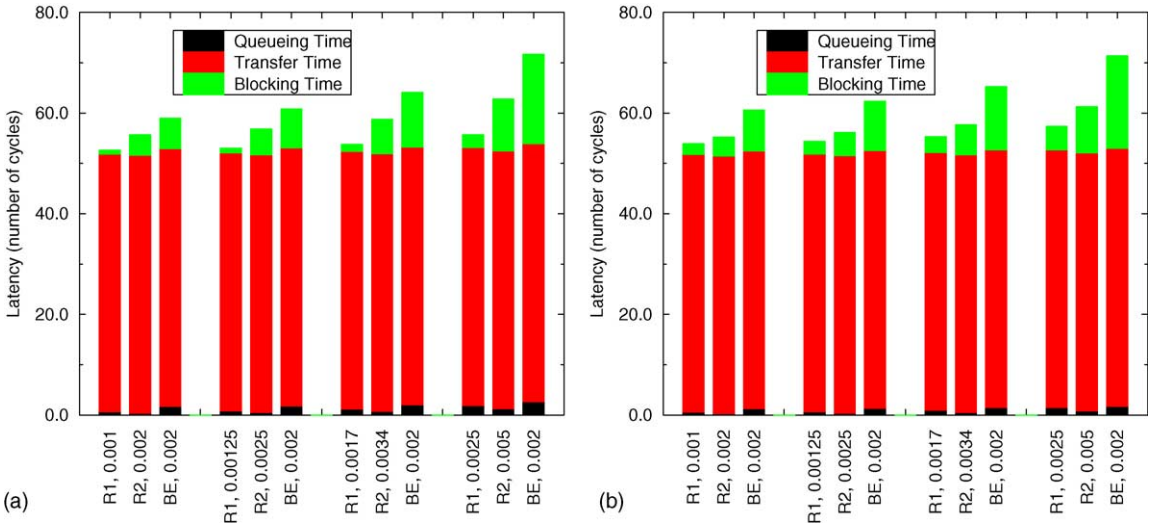

Fig. 8. Latency components of analytical model and simulation model in a 6-cube with varying real-time load and fixed best-effort load (0.002 msgs/cycle). (a) Analytical model, (b) simulation model.

Fig. 8 shows individual components of the average message latency($Latency_c$) and how they change with the real-time load. The results reconfirm the validity of the analytic model.

Fig. 9 plots the variation in best-effort traffic latency ($\overline{L_c}$) as a function of both real-time and best-effort load in a 6-cube. The results support our intuition that for a fixed best-effort load and increasing real-time load, the network latency of best-effort traffic should increase faster than for a fixed real-time load and increasing best-effort load. As mentioned earlier, the network latency is affected by two different parameters: one is due to blocking and the other is due to sharing. Generally the blocking delay of a particular class is proportional to its input load, while the delay due to sharing increases as the workload of other types increases. For the best-effort traffic, the delay due to sharing seems to be the main contributing factor. Hence, with a fixed real-time traffic, the latency of best-effort traffic ($\overline{L_C}$) is not affected significantly by the best-effort load as much as by the real-time load.

Fig. 10 shows the network latency in a 6-cube with different best-effort message length ($M = 64$). The message length of best-effort is two times that of the real-time message. By comparing Fig. 7, which has same lengths for all messages, it is evident that the latency of real-time traffic is not affected by the length of best-effort message. Using the model as a design tool, in Fig. 11(a), we examined the effect of input/output the buffer size. The results concur with prior studies in that the network latency is marginally affected by the buffer size. Finally, Fig. 11(b) depicts the network latency results for a 6-cube with 3 classes of real-time traffic and one best-effort traffic. With more classes of real-time traffic, the best-traffic latency enters into saturation region even faster compare to the results of Fig. 7. The graphs indicate that the rate-based scheduler favors higher priority traffic and thus, lower priority traffic suffers.

Next, we examine the accuracy of our analytical model with respect to bursty traffic using an ON/OFF source. We again have two real-time and one best-effort traffic classes. For each real-time class, we use 14 ON/OFF sources each with a generation rate of $\lambda_c^g/14$ messages. The destinations are uniformly

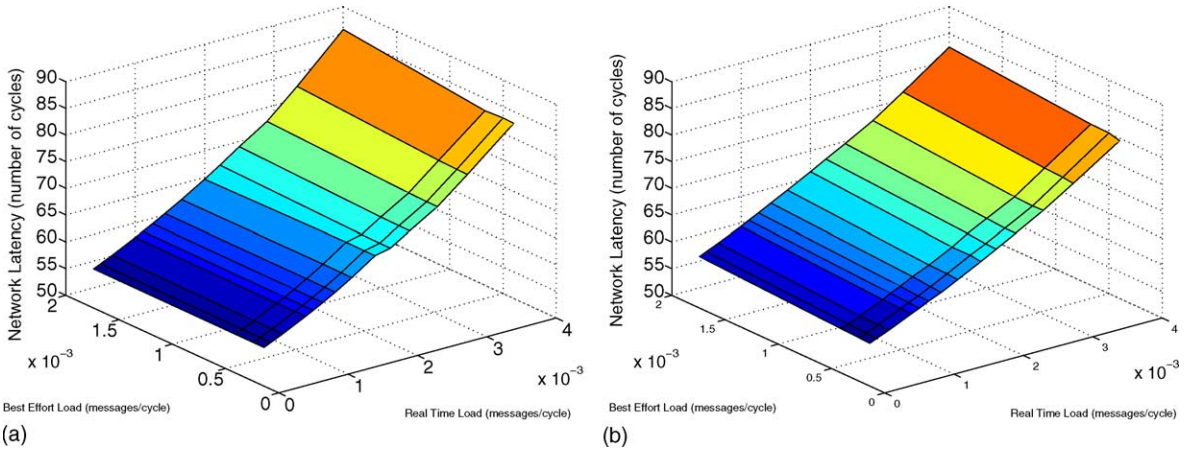

Fig. 9. Best-effort network latency in a 6-cube as a function of both real-time and best-effort load. (a) Analytical model, (b) simulation model.

distributed. Fig. 12 shows the network latencies of analytical and simulation models. Since the generation is evenly scattered to avoid traffic burst by a NI, our analytical model can predict the network latency without any modification, while we need another model to get the average waiting time at the source for ON/OFF source.

4.3. Deadline missing probability results

Using the equations derived in Section 3.4, we compute the deadline missing probabilities in a single router and in a 6-cube. Note that we need a deadline parameter D to estimate the DMP. In our pipelined router model, the minimum transfer time for a 32-flit message is 36 cycles ($P + M - 1$). Hence, we set

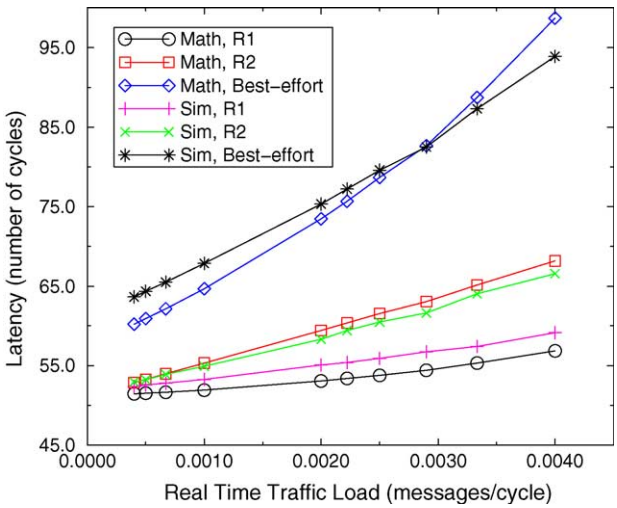

Fig. 10. Network Latency in a 6-cube with variable best-effort message length ($M = 64$) (best-effort traffic load: 0.002 msgs/cycle).

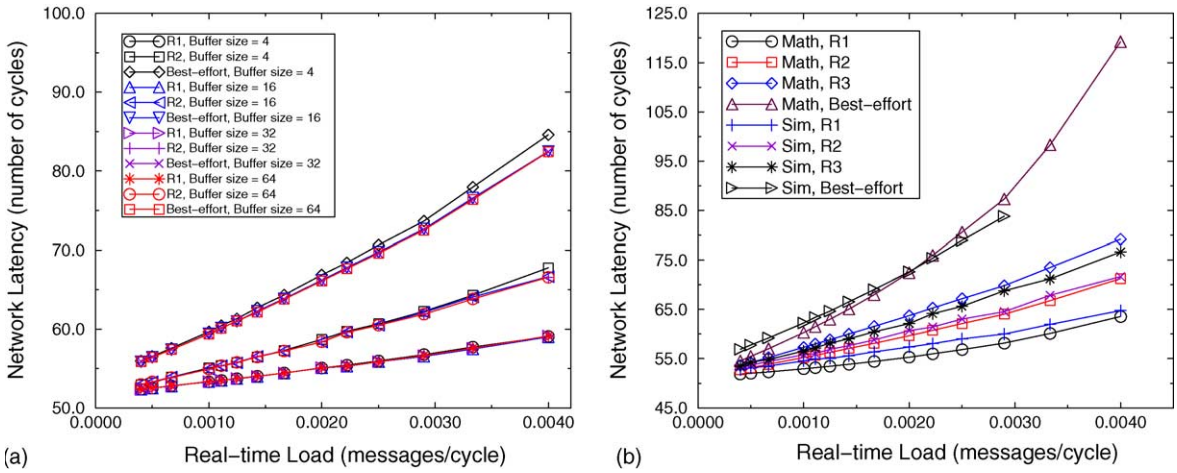

Fig. 11. Network latency in a 6-cube with (a) different input/output buffer size (b_s =4, 16, 32 and 64 flits) and (b) 3 real-time classes under fixed best-effort load (0.002 msgs/cycle). (a) Various buffer sizes, (b) 3 real-time classes.

$D = 42$ or 47 cycles for the single router. Similarly for 2-hop messages in a 6-cube, the minimum transfer time is 46 cycles ($M + Ph + P - 1$). We set $D = 55$ or 60 cycles for 2-hop messages, and $D = 70$ or 75 cycles for 5-hop messages.

In Figs. 13 and 14, we plot the DMP results. In a 6-cube, the deadline missing probabilities of 2-hop and 5-hop messages are shown for different D values. The single router results are more accurate compared to the 6-cube results. This is because we approximate the upper bound of blocking length in each hop to $(2\max(b_s, M) + M)$ without accounting for the chained blocking. There is no chained blocking in a

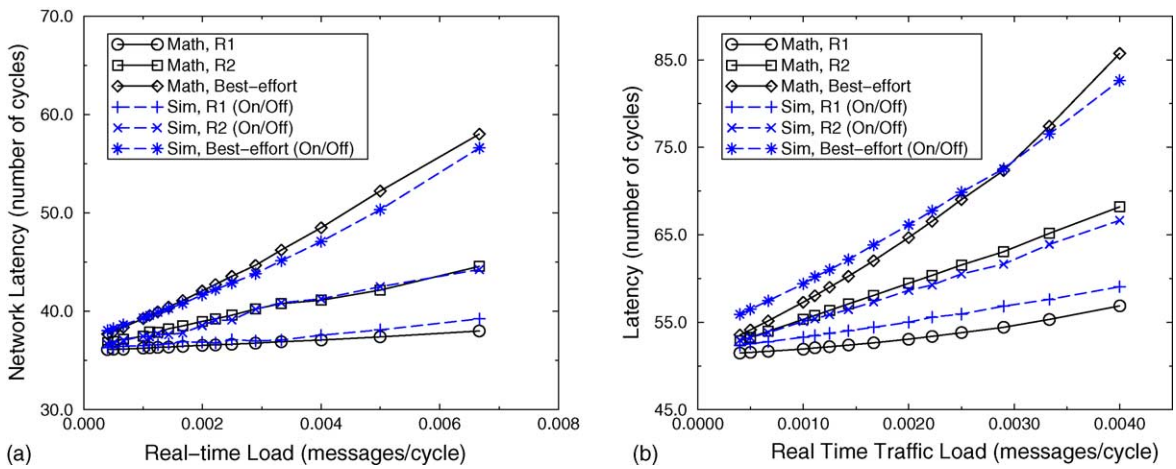

Fig. 12. Network latency comparison of analytical model and simulation model with ON/OFF real-time traffic (a) in a 16-port router with varying real-time load and fixed best-effort load (0.01 msgs/cycle), and (b) in a 6-cube with varying real-time load and fixed best-effort load (0.002 msgs/cycle). (a) Single router, (b) 6-cube.

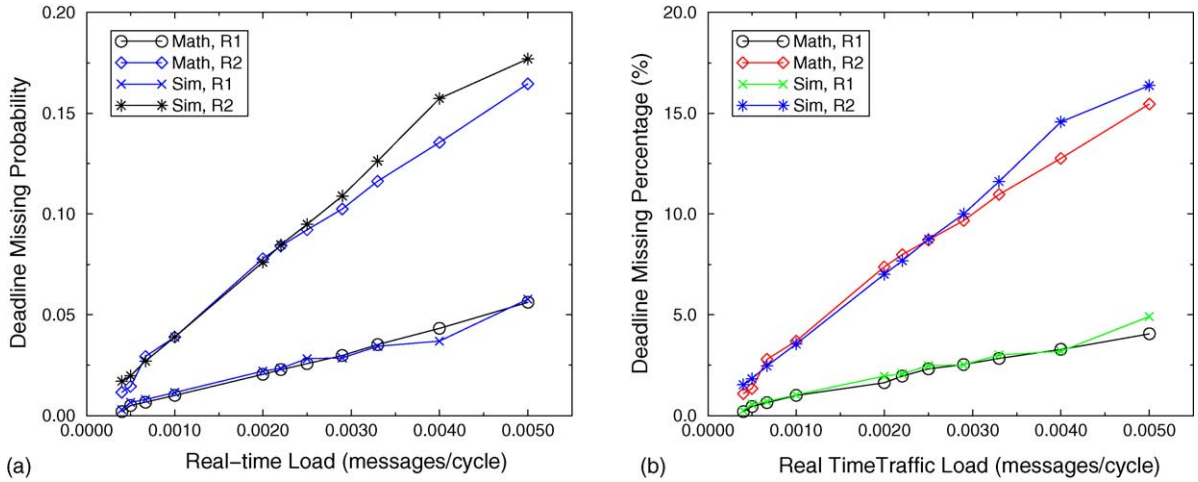

Fig. 13. DMP comparison of analytical and simulation models in a single router with fixed best-effort load (0.01 msgs/cycle). (a) Single router with deadline 42 cycles, (b) single router with deadline 47 cycles.

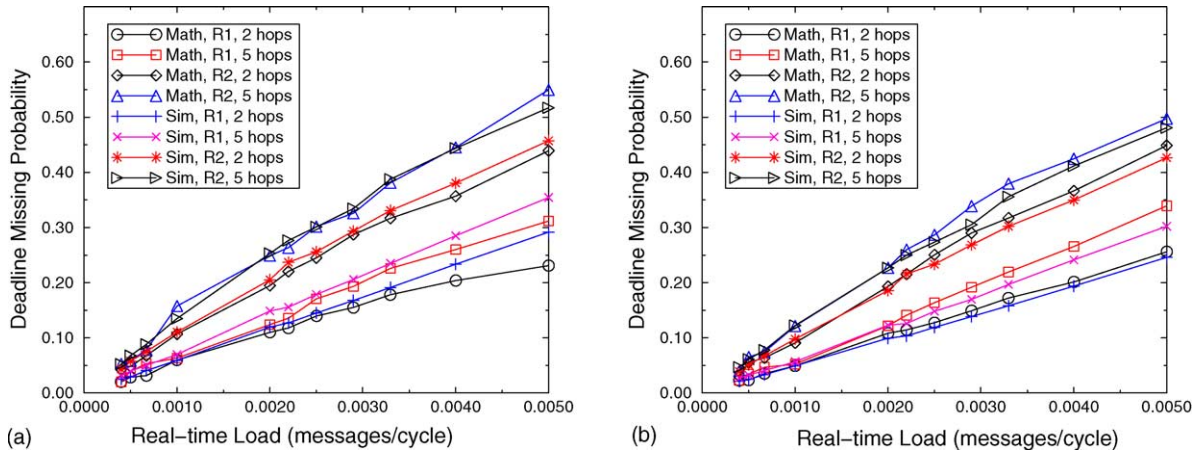

Fig. 14. DMP comparison of analytical and simulation models in a 6-cube with varying real-time load and fixed best-effort load (0.002 msgs/cycle). (a) Deadline : 55 cycles (2 hops), 70 cycles (5 hops), (b) deadline : 60 cycles (2 hops), 75 cycles (5 hops).

single router and hence, the upper bound approximation is more accurate. Even with this approximation, the deadline missing probability results from the analytical model of a 6-cube differ with the simulation results by 10%.

5. Concluding remarks

Provisioning for QoS in cluster networks is becoming a pressing issue with the increasing use of clusters in many commercial applications that need more sophisticated service than the traditional best-effort service model. While a few design alternatives have been proposed to support QoS in clusters, to our

knowledge, there is no efficient mathematical technique to evaluate the design trade-offs. The simulation or limited implementation approach used in prior studies is expensive and inflexible in providing fast-hand estimates to the wealth of questions that arise in making QoS design decisions. This paper introduces an analytic model for evaluating QoS-aware wormhole routers and hypercube-style cluster networks, designed using such routers. The model captures the pipelined design, and analyzes the blocking delay at different stages of the pipe. In addition, the effect of VirtualClock scheduling to provide prioritized service is reflected in the model. Comparison with the simulation results indicates that the router as well as the hypercube models are quite accurate in predicting average message latency and deadline missing probability.

The models can be used as an efficient design tool in studying various design trade-offs. For example, the impact of message length, buffer length, scheduling mechanism, relative performance difference between two traffic types, and many other design questions can be answered quickly using the model either for a single cluster or for a multi-switch cluster. Such performance estimates and quick design overviews are difficult to obtain via a simulation study.

The present model can be improved in a variety of ways and some of them are currently pursued in our group. First, the exponential arrival distribution and ON/OFF model for real-time traffic may not be quite practical. We need to develop a model with a CBR/VBR source to capture inputs like media streams. Second, QoS comes with different connotations, and extension of the model to predict other performance parameters such as bandwidth assurance should be useful. Next, the model can be extended to other topologies. Finally, co-evaluation of a cluster network with network interfaces should answer many questions regarding the QoS ability of an entire communication system.

Acknowledgment

This research was supported in part by NSF grants CCR-9900701, CCR-0098149, CCR-0208734 and EIA-0202007.

Appendix A. The appendix summarizes the principal traffic rates in an *n*-cube

A.1. Traffic rates in an n-cube router

Two types of messages arrive at a router using the input channels. One is called a terminating message, and the other is a transit message that passes through the router using one output channel. Let λ_c^t be the total transit message rate of traffic c at a router. The generation rate of traffic c in the steady state is λ_c'. Therefore, the total message rate at the output of a router (over all the n output channels) is $\lambda_c = \lambda_c^t + \lambda_c'$. Let $\lambda_c^{t,s}$ be the transit message arrival rate of traffic c from other nodes at physical channel s of a router. Similarly, $\lambda_{c,s}'$ represents traffic generated by the source node for a physical channel s and virtual channel c. We give here the expressions derived in [26] for completeness.

The transient message arrival rate at physical channel s and virtual channel c of a router is given by

$$\lambda_c^{t,s} = \sum_{k=2}^{n} P_k \lambda_c' \left[\sum_{j=m}^{M} \frac{{}_sC_{j-1} \cdot {}_{n-s-1}C_{k-j}}{{}_nC_k} \right]$$

where $0 \le s \le (n-1)$, $m = \max(2, k-n+s+1)$, and $M = \min(s+1, k)$. The traffic generation rates have the following relations.

$$\lambda_c^t = \sum_{k=2}^{n} P_k (k-1)\lambda_c', \quad \lambda_c = \lambda_c^t + \lambda_c' = \sum_{k=1}^{n} P_k k \lambda_c'$$

New messages for physical channel s and VC c are generated at a rate $\lambda_{c,s}^g$ by the local host, and is given by

$$\lambda_{c,s}^g = \sum_{k=1}^{n-s} P_k \lambda_c^g \cdot {}_{n-s-1}\frac{C_{k-1}}{{}_nC_k}.$$

The message rate for each virtual channel c in an n-cube is the same regardless of its position, and is given as

$$\lambda_{c,s} = \lambda_c' \cdot \frac{\bar{h}}{n} = \frac{\lambda_c}{n}.$$

A.2. Computation of $B_{\text{middle}}(c, s)$ for Eq. (10)

To compute $B_{\text{middle}}(c, s)$, we use the delay model from [26], except that we include the input and output queueing delays, while [26] captures only blocking delay.

$$B_{\text{middle}}(c,s) = \left(1 - \frac{P_1 \cdot \lambda_c'}{n \cdot \lambda_{c,s}'}\right) \times \sum_{j=s+1}^{n-1} P_{b,c}^j \cdot (\max(b_s, M) + d_{c,j}) \frac{\sum_{m=0}^{n-j-1} P_{m+2}(m+1)\, {}_{n-j-1}C_m / {}_nC_{m+2}}{\sum_{m=0}^{n-s-1} P_{m+2}\, {}_{n-s-1}C_{m+1} / {}_nC_{m+2}},$$

$$d_{c,s} = \frac{1}{2}\left[\max(M, b_s) + M + (1 - P_{t,s}) \sum_{j=s+1}^{n-1} P_{b,c}^j \cdot (d_{c,j} + \max(M, b_s)) \cdot H\{j|s\} \right],$$

$$P_{t,s} = \sum_{k=0}^{s} P_{k+1} \cdot \frac{{}_sC_k}{{}_nC_{k+1}} \cdot \frac{1}{S(0)},$$

$$H\{j|s\} = \sum_{m=0}^{n-j-1} \sum_{k=0}^{s} P_{m+k+2} \cdot \frac{{}_{n-j-1}C_m \cdot {}_sC_k}{{}_nC_{m+k+2}} \cdot \frac{(m+1)}{S(1)},$$

and

$$S(j) = \sum_{m=j}^{n-s-1} {}_{n-s-1}C_m \cdot \sum_{k=0}^{s} P_{m+k+1} \cdot \frac{{}_sC_k}{{}_nC_{m+k+1}}.$$

References

[1] M. Galles, Scalable pipelined interconnect for distributed endpoint routing: the SGI SPIDER chip, Proceedings of Symposium on High Performance Interconnects (Hot Interconnects), August 1996, pp. 141–146.

[2] S.L. Scott, G.M. Thorson, The Cray T3E network: adaptive routing in a high performance 3D torus, Proceedings of Symposium on High Performance Interconnects (Hot Interconnects), August 1996, pp. 147–156.

[3] D. Garcia, W. Watson, Servernet II, Proceedings of the 1997 Parallel Computing, Routing, and Communication Workshop (PCRCW'97), June 1997.

[4] J. Carbonaro, F. Verhoorn, Cavallino: the teraflops router and NIC, Proceedings of the Symposium on High Performance Interconnects (Hot Interconnects 4), August 1996, pp. 157–160.

[5] C.B. Stunkel, D.G. Shea, B. Abali, M.G. Atkins, C.A. Bender, D.G. Grice, P. Hochschild, D.J. Joseph, B.J. Nathanson, R.A. Swetz, R.F. Stucke, M. Tsao, P.R. Varker, The SP2 high-performance switch, IBM Syst. J. 34 (2) (1995) 185–204.

[6] N.J. Boden, D. Cohen, R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic, W.-K. Su, Myrinet: a gigabit-per-second local area network, IEEE Micro 15 (February 1995) 29–36.

[7] J.H. Kim, Bandwidth and latency guarantees in low-cost, high-performance networks, Ph.D. thesis, Department of Electrical Engineering, University of Illinois at Urbana-Champaign, 1997.

[8] A.A. Chien, J.H. Kim, Approaches to quality of service in high-performance networks, Proceedings of the Parallel Computing, Routing and Communication Workshop, Lecture Notes in Computer Science, Springer-Verlag, July 1997.

[9] J. Duato, S. Yalamanchili, M.B. Caminero, D. Love, F.J. Quiles, MMR: a high-performance multimedia router-architecture and design-tradeoffs, Proceedings of the IEEE International Symposium on High-Performance Computer Architecture, January 1999, pp. 300–309.

[10] M.B. Caminero, F.J. Quiles, J. Duato, D.S. Love, S. Yalamanchili, Performance evaluation of the multimedia router with MPEG-2 video traffic, Proceedings of the Third International Workshop on Communication, Architecture and Applications on Network Based Parallel Computing (CANPC'99), January 1999, pp. 62–76.

[11] J.H. Kim, A.A. Chien, Rotating combined queueing (RCQ): bandwidth and latency gurantees in low-cost, high-performance networks, Proceedings of the International Symposium on Computer Architecture, May 1996, pp. 226–236.

[12] J. Rexford, J. Hall, K.G. Shin, A router architecture for real-time point-to-point networks, Proceedings of the International Symposium on Computer Architecture, May 1996, pp. 237–246.

[13] H. Eberle, E. Oertli, Switcherland: a QoS communication architecture for workstation clusters, Proceedings of the International Symposium on Computer Architecture, June 1998, pp. 98–108.

[14] J.-P. Li, M. Mutka, Priority based real-time communication for large scale wormhole networks, Proceedings of International Parallel Processing Symposium, May 1994, pp. 433–438.

[15] M. Gerla, B. Kannan, B. Kwan, P. Palnati, S. Walton, E. Leonardi, F. Neri, Quality of service support in high-speed wormhole routing networks, Proceedings of 1996 International Conference on Network Protocols, October 1996, pp. 40–47.

[16] K. Connelly, A.A. Chien, FM-QoS: real-time communication using self-synchronizing schedules, Proceedings of Supercomputing Conference, November 1997.

[17] H. Song, B. Kwon, H. Yoon, Throttle and preempt: a new flow control for real-time communications in wormhole networks, Proceedings of International Conference on Paralle Processing, August 1997, pp. 198–202.

[18] K.H. Yum, A.S. Vaidya, C.R. Das, A. Sivasubramaniam, Investigating QoS support for traffic mixes with the mediaWorm router, Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), January 2000, pp. 97–106.
[19] K.H. Yum, E.J. Kim, C.R. Das, QoS provisioning in clusters: an investigation of router and NIC design, Proceedings of the International Symposium on Computer Architecture, June 2001, pp. 120–129.
[20] K.H. Yum, E.J. Kim, C.R. Das, A.S. Vaidya, MediaWorm: a QoS capable router architecture for clusters, IEEE Trans. Parallel Distributed Syst. 13 (December 2002) 1261–1274.
[21] W.J. Dally, Virtual-channel flow control, IEEE Trans. Parallel Distributed Syst. 3 (May 1992) 194–205.
[22] A. Demars, S. Shenker, Analysis and simulation of a fair queueing algorithm, Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, 1989, pp. 1–12.
[23] L. Zhang, VirtualClock: a new traffic control algorithm for packet-switched networks, ACM Trans. Comput. Syst. 9 (May 1991) 101–124.
[24] J. Laudon, D. Lenoski, The SGI origin 2000: a CC-NUMA highly scalable server, Proceedings of the International Symposium on Computer Architecture, June 1997, pp. 241–251.
[25] V.S. Adve, M.K. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, IEEE Trans. Parallel Distributed Syst. 5 (March 1994) 225–246.
[26] J. Kim, C.R. Das, Hypercube communication delay with wormhole routing, IEEE Trans. Comput. 43 (July 1994) 806–814.
[27] W.J. Dally, Performance analysis of *k*-ary *n*-cube interconnection networks, IEEE Trans. Comput. 39 (June 1990) 775–785.
[28] J.T. Draper, J. Ghosh, A comprehensive analysis model for wormhole routing in multicomputer systems, J. Parallel Distributed Comput. 32 (1994) 202–214.
[29] P.T. Gaughan, S. Yalamanchili, A performance model of pipelined *k*-ary *n*-cubes, IEEE Trans. Comput. 44 (Auguest 1995) 1059–1063.
[30] L.-S. Peh, W.J. Dally, A delay model for router micro-architectures, Proceedings of Symposium on High Performance Interconnects (Hot Interconnects), August 2000.
[31] L.-S. Peh, W.J. Dally, A delay model for router micro-architectures, IEEE Micro 21 (January/February 2001) 26–34.
[32] V.S. Adve, M.K. Vernon, Performance analysis of mesh interconnection networks with deterministic routing, IEEE Trans. Parallel Distributed Syst. 5 (March 1994) 225–246.
[33] S. Jamin, P.B. Danzig, S.J. Shenker, L. Zhang, A measurement-based admission control algorithm for integrated services packet networks, IEEE/ACM Trans. Network. 5 (February 1997) 56–70.
[34] J. Qiu, E. Knightly, QoS control via robust envelop-based MBAC, Proceedings of the Sixth International Workshop on Quality of Service (IWQoS 98), May 1998, pp. 62–64.
[35] N.K. Jaiswal, Priority queues, Academic Press, 1968.
[36] B. Kim, J. Kim, S. Hong, S. Lee, A real-time communication method for wormhole switching networks, Proceedings of International Conference on Paralle Processing, August 1998, pp. 527–534.
[37] N. Pekergin, Stochastic bounds on delays of fair queueing algorithms, Proceedings of Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'99), March 1999, pp. 1212–1219.
[38] A.K. Parekh, R.G. Gallager, A generalized processor sharing approach to flow control in integrated services networks: the single-node case, IEEE/ACM Trans. Network. 1 (June 1993) 344–357.
[39] D. Stiliadis, A. Varma, Rate-proportional servers: a design methodology for fair queueing algorithms, IEEE/ACM Trans. Network. 6 (April 1998) 164–174.

Eun Jung Kim received the BS degree in computer science from Korea Advanced Institute of Science and Technology, Korea, in 1989, the MS degree in computer science from Pohang University of Science and Technology, Korea, in 1994, and the PhD degree in computer science and engineering from the Pennsylvania State University in 2003. From 1994 to 1997, she worked as a member of technical staff in Korea Telecom Research and Development Group. Dr Kim is currently an assistant professor in the Department of Computer Science in Texas A&M University. Her research interests include computer architecture, parallel/distributed systems, computer networks, cluster computing, QoS support in cluster networks and internet, performance evaluation, and fault-tolerant computing.

Ki Hwan Yum received the BS degree in mathematics from Seoul National University, Korea, in 1989, the MS degree in computer science from Pohang University of Science and Technology, Korea, in 1994, and the PhD degree in computer science and engineering from the Pennsylvania State University in 2002. From 1994 to 1997, he was a member of technical staff in Korea Telecom Research and Development Group. Dr Yum is currently an assistant professor in the Department of Computer Science in the University of Texas at San Antonio. His research interests include computer architecture, parallel/distributed systems, cluster computing, and performance evaluation.

Chita R. Das received the MSc degree in electrical engineering from the Regional Engineering College, Rourkela, India, in 1981, and the Ph.D. degree in computer science from the Center for Advanced Computer Studies, University of Louisiana, Lafayette, in 1986. Since 1986, he has been with the Pennsylvania State University, where he is currently a Professor in the Department of Computer Science and Engineering. His main areas of interest are parallel and distributed computing, cluster computing, Internet QoS, multimedia systems, performance evaluation, and fault-tolerant computing. He is currently an editor of the IEEE Transactions on Computers and has served on the editorial board of the *IEEE Transactions on Parallel and Distributed Systems*. Dr Das is a fellow of the IEEE and a member of the ACM.