

Load-Balanced Routing and Scheduling for Real-Time Traffic in Packet-Switch Networks

Sangman Bak¹, Albert M. K. Cheng², Jorge A. Cobb³ and Ernst L. Leiss²

Abstract

Future computer networks are expected to carry bursty real-time traffic with stringent time-delay requirements. Popular shortest-path routing protocols have the disadvantage of causing bottlenecks due to their single-path routing. We propose a real-time routing and scheduling scheme that randomly distributes the traffic load over all available paths to the destination for load balancing and transmits the packet with the most urgency ahead of the other packets at a switch in a packet-switched network with two objectives – minimizing packet loss in the network and maximizing network throughput. Our scheme consists of two components, a routing algorithm and a scheduling algorithm. The routing algorithm randomly distributes data packets over the whole network to remove bottlenecks caused by the single-path routing of shortest-path routing protocols. The end-to-end delay is bounded by the scheduler, which uses a least-laxity scheduling algorithm. Our simulation results indicate that the proposed scheme decreases the number of packets dropped due to deadline miss and due to buffer overflow in transit in a network and increases network throughput. A desirable by-product is that the traffic load on the network tends to get evenly distributed.

1. Introduction

Rapid advances in Internet technology have made possible the integration of a diverse set of services such as voice, video and other applications into a single Broadband-Integrated Services Digital Network (B-ISDN). Switching systems based on asynchronous transfer mode (ATM) technology are expected to be a main technology for implementing B-ISDN. ATM networks are being developed in an effort to support multimedia applications like video conferencing, remote medical imaging, video-on-demand, and multimedia mail [18]. Many of these multimedia applications will have stringent performance

requirements such as delay, throughput and packet-loss rate. Among these performance requirements, delay requirement is essential to real-time applications. Also, real-time applications consider not the average delay, but the delay of each packet as the quality of service (QoS) requirement. Each packet in these applications will need to be delivered to the destination node within a specific time frame from the time when they were introduced into the network at the source node. A scheduling algorithm can be effectively used to achieve this objective. When a packet has missed its delivery deadline, the scheduling technique must drop the packet at the earliest instance so as to free up network resources for other packets in the network. In general, the scheduling strategy is responsible for the delay experienced by a packet based on the requested QoS of its associated application. We apply the least-laxity heuristic [14] from real-time process scheduling to packet scheduling at a switch in a packet-switched network. We use the term real-time routing and scheduling algorithm for any routing and scheduling method that considers the delay bounds. Our recent paper [18] proposes a real-time scheduling and routing protocol without load-balancing.

Shortest routing protocols suffer performance degradation because all data packets are routed from the source via the same shortest-distance path to the destination as long as the routing tables are unchanged. The problem with these routing protocols is that there are no mechanisms for altering the routing other than updating the routing tables. The shortest path may be highly congested, even when many other paths to the destination have low link utilization. This congestion may trigger the loss of valuable data packets. Using a single path to the destination limits the maximum throughput possible between the source and the destination to be at most the minimum capacity of the links along the shortest path from the source to the destination.

If the network uses shortest routing protocols to carry real-time traffic (data packets with delivery deadlines), then many of these data packets would be dropped due to their missed deadlines or due to the limited buffer space of each node when these shortest paths are congested. In this

¹ Korea Telecom Telecommunications Network Laboratory, Jonmin-Dong, Yusung-Gu, Daejeon, Korea, smbak@kt.co.kr.

² Department of Computer Science, University of Houston, Houston, TX 77204, {cheng, coscel}@cs.uh.edu.

³ Department of Computer Science, University of Texas at Dallas, Dallas, TX 75083-0688, jcobb@utdallas.edu.

paper, we consider the case of a network for soft real-time applications, where the objective is to minimize the packet loss both due to deadline misses and due to the buffer overflow at each node. We also want to maximize the network throughput. Our approach increases the effective bandwidth between the source and the destination so that more data packets can be delivered on time. A result in network flow theory, known as the max-flow min-cut theorem [7], shows that distributing the traffic load over all available paths between a source and a destination in the network, instead of using only one path of minimum cost, increases the effective bandwidth up to the capacity of the minimum cut separating these two nodes.

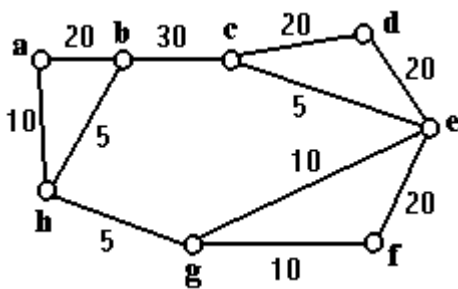


Figure 1. Network topology

For example, let's consider Figure 1. The number b each link represents its capacity. Suppose that node a wants to send data packets to node f . Suppose that we use the hop count in order to calculate the cost of a path in the network. Then the effective bandwidth between node s a and f is 30, while the effective bandwidth of the shortest path (a - h - g - f) from node a to node f is 5.

Several non-real-time multiple-path routing techniques to increase the effective bandwidth between each pair of nodes and to attempt thereby to improve performance have been proposed ([3], [8], [19], [23], [24]). These routing protocols improve performance by routing data packets via multiple paths to the destination. They provide alternate paths to distribute data traffic when the selected shortest path to the destination becomes congested. The disadvantages of these techniques are that they require considerable processing overhead, need significant storage space, or increase the complexity of the routing algorithms, making these approaches unsuitable for real-time applications. Several non-real-time randomized multiple-path routing schemes ([6], [16]) have been proposed for regular network topologies, such as mesh, torus, and butterfly, but these schemes are not suitable for the Internet, which has an irregular network topology. In [11], Kwan and Ramanathan proposed a real-time multiple-path routing scheme called *multiple route real-time channels in packet-switched networks*. This scheme increases the fraction of accepted requests by exploiting the existence of

multiple routes between two nodes of a distributed system, but it has significant processing overheads.

In a recent paper [4], we proposed a non-real-time randomized routing scheme that improves network performance by randomly distributing the traffic load over all available paths to the destination. The routing protocol was formulated for an IP (Internet Protocol) network with an irregular topology.

We propose a real-time load-balanced routing and scheduling scheme that distributes the traffic load over all available paths to the destination and transmits the packet with the most urgency ahead of the other packets in the queue in order to maximize the number of packets that reach their destination before their deadline has passed and to minimize the number of packets that are dropped both due to buffer overflow and due to missed deadlines at each network node.

The rest of this paper is organized as follows. Section 2 introduces the real-time communication. In Section 3, we survey past approaches to transmission scheduling. Section 4 sketches the least laxity scheduling algorithm, which we use for packet scheduling. Section 5 sketches the load-balanced routing protocol. In Section 6, we describe an implementation of our real-time routing and scheduling scheme in a network. In Sections 7 and 8, we present the simulation model and our results. In Section 9, we draw conclusion.

2. Real-Time Communication

Real-time and non-real-time applications are expected to coexist in future integrated service networks. Conventional network communication applications, such as file transfer, electronic mail and remote login, are examples of non-real-time applications. Multimedia conferencing, remote medical diagnosis, process monitoring systems, aircraft control systems and interactive games are examples of real-time applications. The most important characteristic of all real-time traffic is the fact that the value of each packet depends on the time of delivery of the packet at the destination node. Typically, the desired delivery time for each packet is bounded by a specific maximum delay or latency, resulting in a *deadline* being associated with each packet. This delay bound is an application-layer end-to-end timing constraint. If a packet arrives at the destination after its deadline has passed, its usefulness to the end application is assumed to be nil. Such a packet is considered lost. Thus, the packet delay must be bounded and predictable for real-time applications. An early arrival of a packet may also even be considered harmful since it requires buffering at the receiver to achieve constant end-to-end delay. Hence, delivery at the destination node as close as possible to the deadline is sufficient and often desired.

Real-time communication applications are commonly classified as either *hard* or *soft* real-time. Hard real-time applications require a guaranteed maximum delay and are intolerant to packet loss. Being able to predict network delays accurately and act upon their detection becomes more important than network utilization considerations in networks serving such applications. A late or lost packet could have disastrous consequences. Soft real-time applications, on the other hand, can sustain a certain amount of packet loss without significantly affecting the overall quality of the communication. To a certain extent, packet loss can occur due to buffer overflow, or due to missed deadlines at a node in transit. Since soft real-time applications require less stringent service, in general, higher network utilization can be achieved in networks servicing such applications. Our present discussion is largely relevant to soft real-time applications.

3. Scheduling Schemes for Real-Time Traffic

In a packet-switched network, each communication link is statistically shared among many traffic applications. Typically, packets are queued and scheduled for transmission on the *First-In-First-Out (FIFO)* basis, owing to its simplicity and ease of implementation. The FIFO algorithm applies the same priority to all packets, independent of their performance objectives.

Recently, there have been several packet scheduling algorithms proposed for real-time traffic in packet-switching networks. Some networks use *Static Priority (SP)* algorithm that is designed to provide different services to different connections. In this scheme, packets are assigned a fixed priority at the source before they enter the network. Subsequently, at every node in the network, the packet with the highest priority of all queued packets is chosen next for transmission. The SP algorithm effectively provides a way to segregate applications into different classes thereby giving some more importance to some classes than to others. The main drawback of the SP algorithm and the FIFO algorithm is that they do not support the fact that the urgency of a packet is a function of time. The urgency could depend on the amount of delay the packet has encountered thus far in the network and also how much time is still remaining before the delivery deadline of the packet is reached. Zhang and Ferrari [26] have addressed this issue in their *Rate-Controlled Static-Priority Queueing (RCSP)* scheme. Bandwidth reservation [20], a variation of the SP algorithm, involves a portion of the network bandwidth being reserved for the transmission of high-priority packets at a desired rate. Under this scheme, users declare their traffic characteristics at connection setup time and bandwidth enforcement mechanisms ensure that they are honored. A violation of the traffic specification by any connection can lead to

unnecessarily large queuing delays, which lead to an overall under-utilization of the network. More recently, a static priority scheduling scheme for ATM networks has been proposed in [13].

Jackson [10] proposed the *Earliest Deadline First (EDF)* algorithm, which considers the problem of scheduling jobs with different deadlines. According to this algorithm, packets are assigned deadlines before they enter the network; the packet with the earliest deadline of all queued packets is chosen for transmission next at each node in the network. Deadline scheduling is optimal in the sense that given a set of packets with deadlines, if they are schedulable under any scheduling policy, they can also be scheduled under the deadline scheduling policy [14]. With a proper deadline assignment and buffering scheme, high network utilization can be achieved.

However, there are three problems associated with deadline scheduling. First, there is no way of accommodating any changes in the load in the network into the packet scheduling mechanism, as deadlines are assigned to packets at the source alone. Hence, EDF scheduling is complicated in a network due to the difficulties of decomposing an end-to-end delay requirement into per-hop delay bounds. Second, deadline scheduling is no longer optimal when the set of packets to be scheduled is not completely schedulable. This is very likely the case in a network supporting soft real-time applications. The third problem with deadline-based scheduling is that, if a packet is transmitted before its deadline at a certain node, then the time that was saved by early transmission should be made available to the packet at subsequent possibly more congested nodes along its path to the destination. There is no simple way of incorporating this into the earliest deadline scheduling scheme. Zhang [27] addressed this issue by assigning logical arrival times to packets dynamically and calculating their deadlines using the logical arrival time. The problem with this method is that it expects a constant end-to-end delay along the route of the packet, which is estimated at channel establishment time. Also, this method is valid only for static routing.

Peha and Tobagi [17] considered a scheduling problem where each arriving packet is of fixed size and has an arbitrary deadline and weight associated with it. They developed a non-real-time scheduling algorithm that minimizes the weighted average loss rate of traffic at all nodes. Since the Peha and Tobagi algorithm requires *a priori* knowledge of the arrival times of the packets, it is impractical for real implementation and is meant to be a standard for comparison purposes only. Zhang [25] surveyed packet scheduling algorithms for guaranteed performance service in packet-switching networks. More recently, Kweon and Shin [12] proposed a cell-multiplexing scheme for the real-time communication service in ATM networks. In this paper, we focus on

servicing real-time traffic with the goal of ensuring that packets reach their destinations as close as possible to their delivery deadlines and at the same time smoothing network load fluctuation by distributing traffic over the entire network.

4. Least Laxity Scheduling

In this section, we sketch our scheduling algorithm. The goal is to maximize the number of packets that reach their destination before their deadline has passed and minimize the number of packets that are dropped in transit. The scheduling strategy should be able to recognize the changing load on the network and be able to react to it and to transmit first those packets that have to travel along more congested paths to their destination.

The central idea behind our scheme is as follows. Of all the packets competing to be transmitted along a link from a node, the packet with the most urgency should be transmitted ahead of the rest of the packets. To elucidate our ideas, consider an arbitrary network of n nodes with a switch at each node. Let nodes be connected to neighboring nodes through two one-way channels (called a link) - one in each direction. Consider a connection from node s to node d as shown in Figure 2.

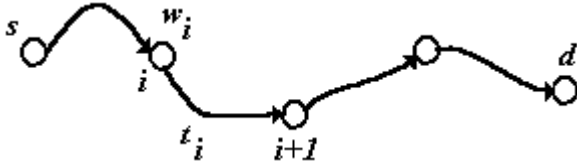


Figure 2. A simple connection

w_i is a measure of the queuing delay experienced by a packet at node i before it is transmitted on the outgoing channel at the node i and t_i represents the propagation time taken for the transmission of a packet on the channel. t_i is independent of the traffic load on the network whereas obviously w_i is a function of the network load, especially the load on the outgoing channel of node i . w_i is typically the average of the delays suffered by the transmitted packets in the last time period at the node and link in question. Two protocols need to be discussed - one to establish a new route between a source and a destination, the other to handle a packet in transit along the route between a source-destination pair.

Steps involved in route establishment:

1. Select a route p from source s to destination d .
2. Get average delays at each node ($w_i + t_i$) along the route p from the global topology table of source s .
3. Compute $C_p = (\sum_{i \in p} w_i + \sum_{i \in p} t_i)$ for every node and link along path p .
4. Let D_p be the requested deadline of the packet, that is, if a packet is produced at time t at the source, then it has to reach the destination before time $t + D_p$. If $D_p \geq$

C_p , allow the connection to be established and follow the steps involved in packet transmission to handle the first packet to be transmitted.

Steps involved in packet transmission:

1. Compute the remaining deadline $D_p = D_p - (w + t)$ at current node, where $w + t$ is the delay at the previous node and link along the path p .
2. Compute C_p for the portion of the path p from current node to destination by using the global topology table of the current node if C_p has not been estimated already.
3. Calculate laxity $L_p = D_p - C_p$ and do one of the following steps according to L_p .
 - 3.1 If $L_p = 0$, transmit the current packet immediately.
 - 3.2 If $L_p < 0$, discard the current packet as deadline cannot be satisfied.
 - 3.3 If $L_p > 0$, insert the current packet into the queue for its outgoing link in sorted order according to its laxity L_p and transmit the packet with the lowest value of L_p in the queue across the link to neighbor.

5. Load-Balanced Routing

In this section, we sketch our method, called the Load-Balanced Routing (LBR), routes data packets to the destination. Each node creates data packets and receives data packets from its neighbors. The node should forward these data packets to its neighbors so that the cost of the path traversed by each data packet is as small as possible, while at the same time attempting to distribute these data packets evenly throughout the network to smooth the network traffic fluctuation, avoid congestion and thus increase network throughput. Our scheme is based on the link-state routing algorithm [15]. Here is the basic algorithm of LBR.

Steps in the LBR algorithm

Given a source s and a destination node d ,

1. LBR selects a set S of nodes from the network nodes.
2. For each data packet to be sent from a source node s to a destination node d , the proposed routing scheme chooses randomly an intermediate node e among the nodes in S .
3. LBR routes the packet via the shortest-distance (or least-cost) path from s to e .
4. Then, LBR routes the packet via the shortest-distance (or least-cost) path from e to d .

The question to be considered next is how to select a set of candidates for the intermediate node. We can use simply all the network nodes for a set of candidates for an intermediate. In this case, our routing scheme is called Load-Balanced Routing via Full Randomization (LBR-FR).

LBR-FR may increase the effective bandwidth between each pair of nodes up to the capacity of minimum cut separating the pair, which is the upper bound on the available bandwidth between these two nodes [7].

LBR-FR has a shortcoming, especially for pairs of nodes of short distance; it is possible that a data packet is routed to the destination via a very long path, much longer than a shortest path from the source to the destination. Clearly, using paths that are excessively long (or expensive) will waste network resources.

To remedy this problem, we introduce a parameter k , in order to exclude nodes from being candidates for an intermediate node that are “too far away” (or too expensive to reach them) from the source. The set of candidates is restricted to all the nodes whose distance (or cost) from the source is at most k . When we use the parameter k to control the candidate set for an intermediate node, our routing scheme will be called Load-Balanced Routing via Bounded Randomization (LBR-BR).

Choosing an appropriate value for the parameter k is crucial for the performance of the algorithm. Choosing too small a value will exclude nodes that are far away (expensive to reach them) from the source from being candidates for an intermediate node, but it will increase the likelihood of a bottleneck. On the other hand, choosing too large a value may waste network resources by routing packets via excessively long (or expensive) paths, but it will increase the effective bandwidth up to the capacity of the minimum cut separating each pair of nodes. To reach a compromise between these two extremes, the parameter k may be chosen to be the average of the distance (or cost) to each node reachable from the source [4]:

$$\bullet k = \frac{1}{n} \sum_{i=1}^n \text{dist}(s, d_i)$$

where d_i is a node in the network and s is the source node.

This value is a constant for the source s until the route update occurs. When we use this value for the parameter k , LBR-BR will be called LBR-BR1. This value, however, has shortcomings. It limits the effective bandwidth between each pair of the nodes in the network to less than the capacity of the minimum cut separating the pair. The static value of k is too strong a restriction for a pair of nodes with a long path length and too weak a restriction for a pair of nodes with a short path length.

To remedy this problem of the static value for the parameter k , we may consider a dynamic choice of parameter k and choose the value of the parameter k more “intelligent” to be fair to all the source-destination pairs:

$$\bullet k = \text{dist}(s, d) * \frac{\text{MAX}(\text{dist}(s, d_i)) - 1}{\text{MAX}(\text{dist}(s, d_i))}$$

where d_i is a node in the network, s is the source node and d is the destination node.

This parameter k dynamically changes according to the length of the shortest path from the source s to the destination d . When we use this value for the parameter k , LBR-BR will be called LBR-BR2. In this paper, we choose this dynamic value for the parameter k .

6. Implementation Details

In this section, we present a possible way of implementing our real-time routing protocol. We choose the LBR-BR2 algorithm as a routing algorithm in the network, since it has the best performance among the three LBR algorithms (LBR-FR, LBR-BR1 and LBR-BR2) [5]. Our load-balanced routing algorithm is based on a link-state algorithm [15]. The routing component of each node maintains a routing table (RT). The routing table specifies which particular neighbor a packet should be sent to if it is headed to a particular destination (or intermediate node). Note that packets to different destinations (or intermediate nodes) may need to be sent to the same neighbor of the current node if their routing table entries are the same. RT contains the cost (average delay) of reaching every destination through a chosen neighbor.

To accomplish the real-time load-balanced routing, each packet must carry at least four pieces of information: the destination d , the intermediate node e , a bit b and a delivery deadline D_p . Bit b indicates whether the packet has not yet reached node e ($b = 0$) or has already passed through node e ($b = 1$). The delivery deadline D_p is initialized by the delay bound, by which the packet should be delivered to the destination d , from the time when it was introduced into the network at the source node.

Therefore, the operation of the protocol is as follows. Initially, the source node s sends the packet with $b = 0$, and routes it in the direction of node e . As long as $b = 0$, the packet keeps being routed along the network until it reaches node e . At node e , b is updated to 1, and the packet is routed towards node d . As long as $b = 1$, the packet keeps being routed along the network until it reaches node d , where it is delivered. Whenever a packet arrives at each node along the path p to the node d , its D_p is updated to $D_p = D_p - (w + t)$ at current node, where $w + t$ is the (queuing and propagation) delay at the previous node and link along the path p . If a packet arrives at a node along the path to the destination d after its D_p has expired, the packet is considered lost. For each packet dropped due to buffer overflow in transit, its source node retransmits with its current delivery deadline D_p .

Thus, at any point in time, a node routes packets using the entry in the routing table for the destination or intermediate node in question. Each node maintains its local topology information in a local topology table, storing the cost associated with each of its local links. Also, each network node maintains a global topology

information in its global topology table, storing the cost (average queuing and propagation time) of each link in the network. Whenever the local topology changes, the corresponding entries in the local topology table are updated. To keep a view of the entire network topology up-to-date, each node periodically sends routing packets to all the other nodes with a copy of the node's local link status information. On receiving a routing packet, a node updates its own global topology table to reflect the new cost of each link on the network. Dijkstra's shortest path algorithm [9] is applied to the global topology table to determine next hops. By using this information, the node updates its routing table to re-route packets along less expensive paths to their destination. For each packet to be sent from source to destination, we apply Dijkstra's shortest path algorithm to the view of the global network topology to determine the expected cost of the selected path between each pair of nodes. The data structures originally proposed in [15] were expanded to include timing and cost information.

To reduce the computation that needs to be done on receipt of a packet to be routed, each node maintains a buffer for each of its outgoing links to hold packets waiting to be transmitted. Entries in this buffer are maintained in a queue sorted according to their laxity L_p . Packets ahead in the queue have a lesser laxity than packets behind them. Packets are taken from the front of the queue for transmission. Thus, the more urgent packets leave the network node before the less urgent ones. A new packet is inserted into the buffer at the appropriate position based on its laxity.

7. Simulation Model

Our simulation studies were done on the Maryland Routing Simulator (MaRS) [1], which is a network simulator developed at the University of Maryland, with an interest in studying the performance of different network routing algorithms under different simulated loads. We added timing constraints to the simulator and deadlines to the simulated traffic to create Real-Time MaRS (RT MaRS). A network configuration consists of a physical network, a real-time routing and scheduling algorithm, and a workload.

The real-time routing and scheduling algorithms are SFR-LL, SFR-EDF, SFR-FIFO, LBR-LL, LBR-EDF, and LBR-FIFO. Each SFR uses a link-state routing protocol [11] as its routing protocol. SFR-LL, SFR-EDF and SFR-FIFO use Least laxity First, Earliest Deadline First and First In First Out as a scheduling algorithm at each node, respectively. Each LBR uses our load-balanced routing protocol (LBR-BR2) as its routing protocol. LBR-LL, LBR-EDF and LBR-FIFO use Least laxity First, Earliest Deadline First and First In First Out as a scheduling

algorithm at each node, respectively. To get a better understanding of our load-balanced routing protocol, we compare the performance of the three LBR protocols against three SFR protocols. To get a better understanding of our scheduling scheme (Least Laxity First), the performance of LBR-LL and SFR-LL are compared against LBR-FIFO and LBR-EDF and against SFR-FIFO and SFR-EDF, respectively.

In our simulation, the assumed physical network is the NSFNET topology given in Figure 3. All links have a bandwidth of 1.5 Mbits/sec. We assumed that there are no link or node failures. Each node has a buffer space of 50,000 bytes. The processing time of a data packet at each node equals 1 msec. In order to calculate the cost of a path in the network, we use average delay of reaching every destination. The propagation delay of each link is 1 msec.

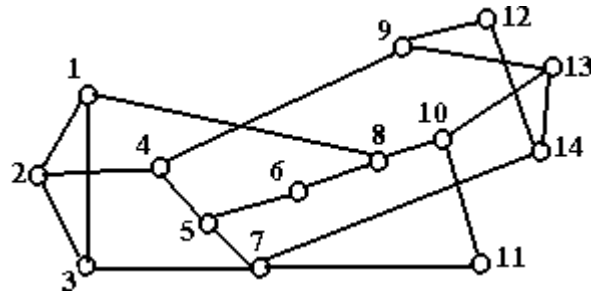


Figure 3. NSFNET Topology: 14 nodes, 21 bi-directional links, average degree 3

The workload consists of FT (file transfer protocol) connections and telnet connections. A connection is a real-time communication session established between end-user applications at source and destination nodes. All FTP and telnet connections have the following parameters: the data packet length equals 512 bytes, the inter-packet generation time is 1 or 10 msec, and the window size is 500 packets. Traffic was introduced into the network by the FTP and/or telnet connections at the nodes they were attached to. Network traffic consisted of data packets sent from the source of a connection to the destination and response packets sent from the destination of the connection to the source. Further, each source and destination node send acknowledgements for data packets received. Also present in the network are routing packets, which are sent periodically to update the state of the network. All data and response packets were randomly assigned a deadline at the source from the following set:

{100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900, 2000}, where the unit of each value is msec.

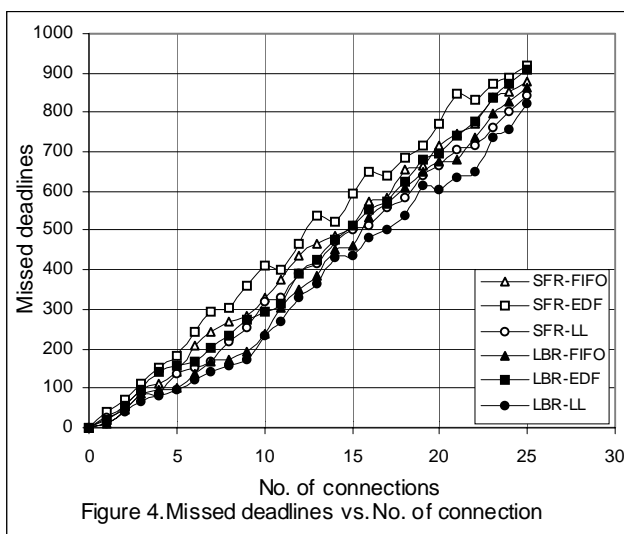
The situation we want to simulate is a network that receives packets from source nodes with deadline requirements to reach destination nodes. Another constraint added to the connections is that the number of packets sent by a telnet/FTP source is not allowed to

exceed the number of packets that have been acknowledged by the corresponding destination by more than a fixed amount. This was incorporated to prevent undue flooding of the network and as a means of flow control. A packet that reaches its destination after its deadline had passed is considered a failure and dropped. Also, at a node in transit, if it is detected that it is impossible to meet the deadline of the packet under the current network load, the packet is dropped. This makes the network resources more available to service other packets and ease the strain on the shared network resources. Connections start when the simulation begins and they are considered never-ending.

The measurement interval of each simulation is 100,000 msec. We consider the following performance measures:

- *Missed deadline.* The total number of packets dropped due to deadline miss during the measurement interval.
- *Fraction of missed deadline.* Missed deadline divided by missed deadline plus the total number of packets acknowledged during the measurement interval.
- *Packet drop.* The total number of packets dropped due to buffer overflow during the measurement interval.
- *Throughput.* The total number of data bytes acknowledged during the measurement interval divided by the length of the measurement interval.
- *Packet delay.* The total delay of all packets acknowledged during the measurement interval divided by the number of packets acknowledged during the measurement interval.
- *Link utilization.* The data service rate divided by the link bandwidth.

8. Simulation Results



Figures 4 and 5 show the missed deadline and the fraction of the missed deadline versus the number of

connections, respectively. The missed deadline is the total number of the packets that were dropped because their deadlines had been expired. The missed deadline and the fraction of the missed deadline in each LBR protocol are lower than in its corresponding SFR both when the number of connections is low and high.

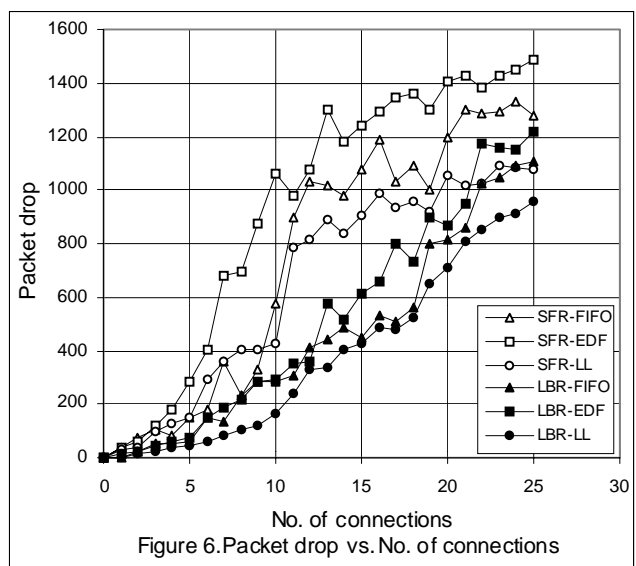
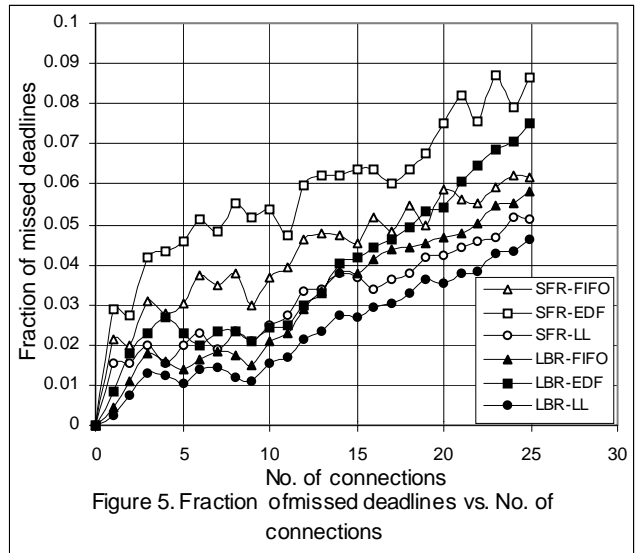


Figure 6 shows the packet drop versus the number of connections. The packet drop is the total number of packets dropped due to buffer overflow at each node when the network became congested. The packet drop in each LBR protocol is lower than in its corresponding SFR at all network loads. In three Figures 4,5 and 6, LBR-LL has the best performance with respect to missed deadline, the fraction of the missed deadline and the packet drop at most network loads.

Figure 7 shows throughput versus the number of connections. The throughput in all the routing protocols in general increases as the number of connections increases. With respect to throughput, each of the three LBR algorithms performs significantly better than its corresponding SFR algorithm at all loads. This is because our schemes make effective use of the multiple paths between each pair of nodes in the network. The LBR-LL outperform all the other algorithms with respect to network throughput. The throughput generally increases linearly except around the saturation points. The system is saturated when the number of connections is around 2, 10 and 19 in all the schemes.

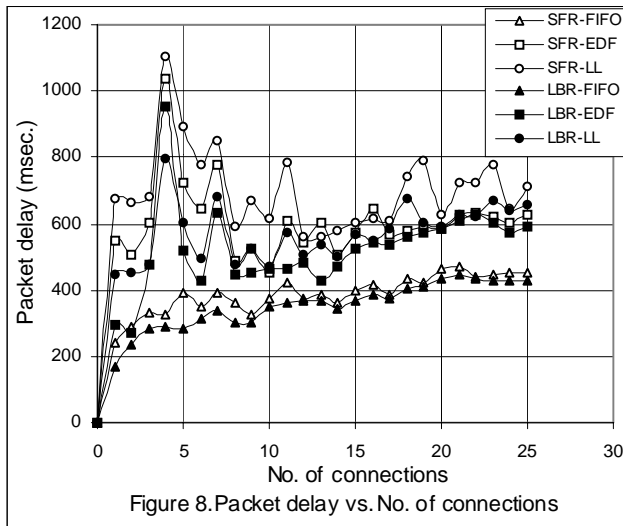
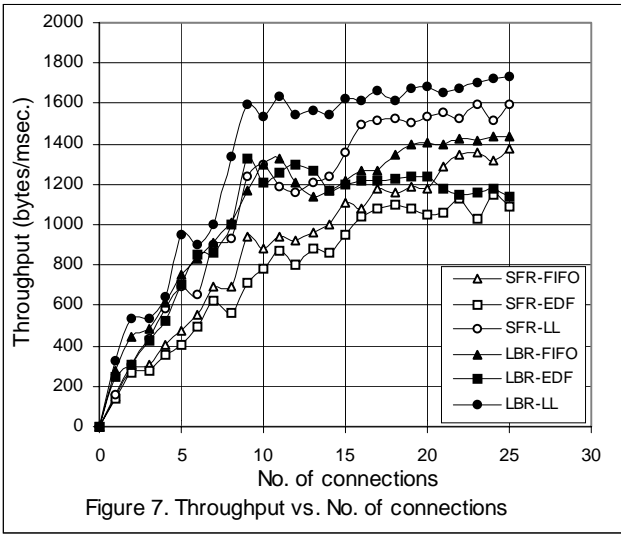


Figure 8 shows the packet delay versus the number of connections. Each SFR algorithm exhibits higher delay oscillations than its corresponding LBR protocol. The three LBR protocols have lower packet delay than their corresponding SFR algorithms, respectively, during the

entire measurement interval except when the number of connections is 20 and 21 in SFR-FIFO and LBR-FIFO. LBR-LL has the lowest packet delay at most times during the measurement interval. However, in both Figures 7 and 8, the curves of each SFR and its corresponding LBR tend to converge as the number of connections increases. This is to be expected, since even a single path routing algorithm would tend to distribute load over the entire network (from a global perspective), as the number of connections approaches $n \cdot (n-1)$, where n is the number of the network nodes.

| Link | SFR-FIFO | SFR-EDF | SFR-LL | LBR-FIFO | LBR-EDF | LBR-LL |
|----------|----------|----------|---------|----------|----------|---------|
| (1,2) | 0 | 0 | 0 | 0.20397 | 0.210305 | 0.20860 |
| (2,1) | 0 | 0 | 0 | 0.241087 | 0.217396 | 0.17039 |
| (1,3) | 0 | 0 | 0 | 0.009301 | 0.010095 | 0.01187 |
| (3,1) | 0 | 0 | 0 | 0.278631 | 0.236989 | 0.27402 |
| (1,8) | 0 | 0 | 0 | 0.320923 | 0.254089 | 0.24971 |
| (8,1) | 0 | 0 | 0 | 0.026664 | 0.028229 | 0.03234 |
| (2,3) | 0.711985 | 0.692581 | 0.57226 | 0.546585 | 0.57591 | 0.63328 |
| (3,2) | 0.075119 | 0.036123 | 0.03486 | 0.152213 | 0.14705 | 0.13127 |
| (2,4) | 0.995844 | 0.878438 | 0.86205 | 0.996593 | 0.867884 | 0.85951 |
| (4,2) | 0.37534 | 0.19706 | 0.24797 | 0.395367 | 0.31592 | 0.31157 |
| (3,7) | 0 | 0 | 0 | 0.434933 | 0.352121 | 0.37690 |
| (7,3) | 0 | 0 | 0 | 0.081884 | 0.087776 | 0.09696 |
| (4,5) | 0.286073 | 0.26071 | 0.29849 | 0.442404 | 0.515311 | 0.40615 |
| (5,4) | 0.338314 | 0.162643 | 0.22928 | 0.486971 | 0.403995 | 0.44149 |
| (4,9) | 0.987519 | 0.874845 | 0.87170 | 0.993542 | 0.869145 | 0.86914 |
| (9,4) | 0.201792 | 0.152588 | 0.16612 | 0.220512 | 0.186782 | 0.19999 |
| (5,6) | 0.381952 | 0.381952 | 0.38195 | 0.222418 | 0.213627 | 0.22222 |
| (6,5) | 0.042359 | 0.042359 | 0.04235 | 0.151933 | 0.137168 | 0.12323 |
| (5,7) | 0.042359 | 0.042359 | 0.04235 | 0.041267 | 0.040038 | 0.03973 |
| (7,5) | 0.382532 | 0.382532 | 0.38253 | 0.3257 | 0.32 | 0.31969 |
| (6,8) | 0 | 0 | 0 | 0.03724 | 0.030003 | 0.03232 |
| (8,6) | 0.001195 | 0.001195 | 0.00119 | 0.307029 | 0.296141 | 0.27412 |
| (7,11) | 0.042359 | 0.042325 | 0.04235 | 0.039936 | 0.039526 | 0.03915 |
| (11,7) | 0.382566 | 0.382566 | 0.38256 | 0.194936 | 0.211968 | 0.22207 |
| (7,14) | 0 | 0 | 0 | 0.25856 | 0.195277 | 0.21958 |
| (14,7) | 0 | 0 | 0 | 0.036111 | 0.039068 | 0.03842 |
| (8,10) | 0 | 0 | 0 | 0.239176 | 0.158856 | 0.17261 |
| (10,8) | 0.001195 | 0.001195 | 0.00119 | 0.215284 | 0.199135 | 0.19761 |
| (9,12) | 0.312422 | 0.301943 | 0.32593 | 0.329529 | 0.343108 | 0.32061 |
| (12,9) | 0.021333 | 0.0185 | 0.01955 | 0.372911 | 0.285168 | 0.31648 |
| (9,13) | 0.174114 | 0.176265 | 0.23968 | 0.160819 | 0.163792 | 0.11893 |
| (13,9) | 0.097124 | 0.064144 | 0.07508 | 0.056852 | 0.047962 | 0.03736 |
| (10,11) | 0 | 0 | 0 | 0.000444 | 0.001058 | 0.00099 |
| (11,10) | 0.001195 | 0.001195 | 0.00119 | 0.189781 | 0.172954 | 0.16599 |
| (10,12) | 0 | 0 | 0 | 0.17367 | 0.141687 | 0.15854 |
| (12,10) | 0 | 0 | 0 | 0.055142 | 0.056482 | 0.05659 |
| (10,13) | 0 | 0 | 0 | 0.155887 | 0.098987 | 0.09850 |
| (13,10) | 0 | 0 | 0 | 0.061223 | 0.05336 | 0.06050 |
| (12,14) | 0 | 0 | 0 | 0.013297 | 0.014258 | 0.01734 |
| (14,12) | 0 | 0 | 0 | 0.238592 | 0.176469 | 0.20019 |
| (13,14) | 0 | 0 | 0 | 0.022711 | 0.024741 | 0.02104 |
| (14,13) | 0 | 0 | 0 | 0.019866 | 0.018159 | 0.01877 |
| Total | 5.854692 | 5.09352 | 5.22074 | 9.751893 | 8.757987 | 8.76593 |
| Average | 0.139397 | 0.121274 | 0.12430 | 0.232188 | 0.208523 | 0.20871 |
| Variance | 0.063976 | 0.051889 | 0.04944 | 0.05054 | 0.040935 | 0.04110 |

Table 1. Link utilization (Number of FTP/telnet connections: 7)

We shall now discuss our observations and the interesting insights obtained from the results.

The performance of each of the Earliest Deadline First schemes (LBR-EDF and SFR-EDF) is considerably worse

than its two corresponding schemes (FIFO and LL). The EDF schemes perform poorly when the network has to schedule a set of packets that are not completely schedulable, as is the case here. Both the EDF schemes (LBR-EDF and SFR-EDF) and the FIFO schemes (LBR-FIFO and SFR-FIFO) do not have any estimate of the urgency of the packets to reach their destinations. This estimate is a function of the deadline of the packet, the distance to the destination, and the load along the path to the destination. The EDF schemes may end up retaining the more urgent packets at the nodes and sending less urgent packets ahead. The FIFO schemes are impartial and can at best perform as well as the LL schemes. This is because the urgency or laxity of the packet is the basis of the LL schemes, which schedule more urgent packets for transmission ahead of less urgent ones and hence outperform the other two schemes (EDF and FIFO). As the connections receive acknowledgements for packets sent earlier, they send new packets into the network that is operating at its peak efficiency. Thus, the connections are prevented from flooding the network with packets that would otherwise be dropped at subsequent nodes. Hence the packet loss stays in a constant range even when the simulation is allowed to run for a longer time for the EDF scheme.

Table 1 shows the average link utilization during the measurement interval when the number of connections is 7. By inspecting the link utilization over the whole network we can see that the LBR protocols distribute the data packets more uniformly over the whole network than all SFR protocols. Also, the total and the average of the link utilization indicate that the LBR protocols use network resources more productively than the SFR protocols (Refer to Figure 7).

9. Conclusions

We presented a real-time load-balanced routing and scheduling scheme (LBR-LL) to distribute the data traffic randomly over all available paths to a destination in the network for data load balancing and schedules competing packets at a switch in a packet-switched network. Our simulation results show that each of the LBR schemes (LBR-FIFO, LBR-EDF and LBR-LL) has improved performance with respect to packet loss, throughput, and link utilization at most times during the measurement interval, compared with its corresponding SFR scheme (SFR-FIFO, SFR-EDF and SFR-LL), which is based on a conventional shortest-path routing protocol. The routing protocol of the proposed real-time scheme is simple and suffers from little control overhead; it randomly chooses one node, whose cost to reach it is within k , as an intermediate node. It is not concerned with the current traffic status of the network. When the traffic is ver-

balanced in the whole network, since our routing method may use a path that is more expensive than the least-cost path, the gain from load balance may be smaller than the cost resulting from a longer delay. From the result of our simulation, we conclude that our routing scheme has the following advantage over existing schemes:

- Better utilization -- The improved network throughput, the lower packet loss and traffic load balancing achieved by using the proposed LBR schemes is indicative of better utilization than the SFR routing schemes.

Regarding our scheduling algorithm (Least-laxity scheduling algorithm), SFR-LL outperformed its corresponding other schemes (SFR-FIFO and SFR-EDF) and LBR-LL outperformed its corresponding other schemes (LBR-FIFO and LBR-EDF). The least-laxity scheduling algorithm increased network throughput at the expense of non-urgent packets' delay. The results of our simulations lead us to believe that the proposed scheme has an advantage over existing schemes for transmission scheduling:

- Better utilization -- The improved network throughput and the lower packet loss achieved by using the proposed LL schemes is indicative of better utilization than the FIFO or the ED schemes.

Our proposed real-time routing and scheduling scheme (LBR-LL) has the best performance among all six LBR and SFR schemes during the measurement interval with respect to throughput, packet loss and link utilization.

11. References

- [1] C. Alaettinoglu, K. Dussa-Zieget, I. Matta, O. Gudmundsson, and A.U. Shankar, *MaRS - Maryland Routing Simulator* Version 1.0. Department of Computer Science, University of Maryland, 1991.
- [2] G. Apostolopoulos, R. Guerin, S. Kamat, and S.K. Tripathi, *Quality of Service Based Routing: A Performance Perspective*, Proceedings of the 1998 ACM SIGCOMM Conference, Aug. 1998.
- [3] S. Bahk and, M. E. Zarki, *Dynamic Multi-path Routing and How it Compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks*, Proceedings of the 1992 ACM SIGCOMM Conference, Vol. 22, Oct. 1992.
- [4] S. Bak and J. A. Cobb, *Randomized Distance-Vector Routing Protocol*, Proceedings of ACM Symposium on Applied Computing, San Antonio, Texas, Feb. 1999.
- [5] S. Bak, J. A. Cobb and E. L. Leiss, *Load-Balanced Routing via Bounded Randomization*, Technical Report, Department of Computer Science, University of Houston, Apr. 1999.
- [6] R. Cole, B.M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A.W. Richa, K. Schroeder, R.K. Sitaraman, and B. Voeking, *Randomized Protocols for low-congestion circuit routing in multistage interconnection networks*,

- Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 378-388, May 1998.
- [7] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.
- [8] J. A. Cobb and M. G. Gouda, *Balanced Routing*, IEEE Proceedings of the International Conference on Network Protocols, 1997.
- [9] E. Dijkstra, *A Note on Two Problems in Connection with Graphs*, Numerische Mathematik, Vol. 1, pp. 269-271, 1959.
- [10] J. R. Jackson, *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, UCLA, 1955.
- [11] K.C Kwan and P. Ramanathan, *Multiple Route Real-Time Channels in Packet-Switched Networks*, Proceedings of IEEE Real Time Systems Symposium, pp. 74-83, 1994.
- [12] S. Kweon and K. G. Shin, *Providing Deterministic Delay Guarantees in ATM Networks*, IEEE/ACM Trans. On Networking, Vol. 6, pp. 838-850, Dec. 1998.
- [13] C. Li, R. Bettati and W. Zhao, *Static Priority Scheduling for ATM Networks*, IEEE proceedings of Real-Time Systems Symposium, pp. 264-273, Dec. 1997.
- [14] C. L. Liu and J. W. Layland, *Scheduling Algorithms for Multiprogramming in a Hard Real-time Environment*, Journal of the ACM, 20:46-61, Jan.1973.
- [15] J.M. McQuillan, Ira Richer and E.C. Rosen, *The New Routing Algorithm for the ARPANET*, IEEE Trans. on Communications, Vol. COM-28, NO. 5, pp. 711-719, Ma 1980.
- [16] T. Nesson and S. L. Johnsson, *ROMM Routing on Mesh and Torus Networks*, Proceedings of the 7th Annual ACM Symposium on Parallel Algorithms and Architectures, Jul 1995.
- [17] J.M. Peha and F.A. Tobagi, *Evaluating Scheduling algorithms for Traffic with Heterogenous Performance Objectives*. Proceedings of IEEE GLOBECOM, pp. 21-27, 1990.
- [18] S. M. Rao and A. M. K. Cheng, *Scheduling and Routing of Real-Time Traffic in Packet-Switched Networks*, Submitted to 8th IEEE Intl. Conf. On Computer Communications and Networks, Boston, MA, Oct. 1999.
- [19] D. Sidhu, R. Nair and S. Abdallah, *Finding Disjoint Paths in Networks*, Proceedings of the 1991 ACM SIGCOMM Conference, 1991.
- [20] K. Sriram, *Methodologies for Bandwidth Allocation, Transmission Scheduling and Congestion Avoidance in Broadband ATM Networks*, Computer Networks and ISDN Systems, Vol. 26, pp. 43-59, 1993.
- [21] I. Stoica and H. Zhang, *LIRA: An Approach for Service Differentiation in the Internet*, Proceedings of NOSSDAV 98, Jul. 1998.
- [22] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.
- [23] Z. Wang and J. Crowcroft, *Shortest Path First with Emergency Exists*, Proceedings of the 1990 ACM SIGCOMM Conference, 1990.
- [24] W.T. Zaumen and J.J. Garcia-Luna-Aceves, *Loop-Free Multipath Routing Using Generalized Diffusing Computations*, Proceedings of IEEE INFOCOM '98, San Francisco, California, Mar. 1998.
- [25] H. Zhang, *Service Disciplines for Guaranteed Performance Service in Packet-Switching Networks*, Proceedings of the IEEE, Vol. 83, No. 10, Oct. 1995
- [26] H. Zhang and D. Ferrari, *Rate-controlled Static-priority Queuing*, Proceedings of IEEE INFOCOM '93, San Francisco, California, Apr. 1993.
- [27] L. Zhang, *Virtual clock: A new Traffic Control Algorithm for Packet Switching Networks*, Proceedings of ACM SIGCOMM '90, Philadelphia, PA, pp. 19-29, Sep. 1990.