Load-Balanced Routing and Scheduling for Traffic with Deadlines in Packet-Switched Networks

Sangman Bak[1], Albert M. K. Cheng[1], Jorge A. Cobb[2] and Ernst L. Leiss[1]

Abstract

Future computer networks are expected to carry bursty data 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 routing and scheduling scheme for data traffic with deadlines. The proposed scheme randomly distributes the traffic load over a set of selected 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 a set of paths chosen between the source-destination pair to remove bottlenecks caused by the single-path routing. 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 misses 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 more 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, such as video conferencing, remote medical imaging, video-on-demand, and multimedia mail [12]. Many of these multimedia applications will have stringent performance requirements such as delay, throughput and packet-loss rate. Among these performance requirements, the delay requirement is essential to real-time applications. Also, these applications do not consider 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 it was 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

[1]Department of Computer Science, University of Houston, Houston, TX 77204-3475, {smbak, cheng, coscel}@cs.uh.edu.
[2]Department of Computer Science, University of Texas at Dallas, Dallas, TX 75083-0688, jcobb@utdallas.edu.

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 [10] from real-time process scheduling to packet scheduling at a switch in a packet-switched network.

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 remain 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 any link along the shortest path from the source to the destination.

If the network uses shortest routing protocols to carry 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 paper, we consider the case of a network for data traffic with deadlines, where the objective is to minimize the packet loss both due to deadline misses and due to buffer overflow at some node along the chosen path. 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 [6], 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.

Several multiple-path routing techniques for data traffic without deadlines have been proposed to increase the effective bandwidth between each pair of nodes and to attempt thereby to improve performance ([2], [7], [13], [15], [16]). These routing protocols improve performance by routing data packets via multiple paths to the destination. In general, they first use a selected shortest path to route data packets from source to 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 algorithm. Several randomized multiple-path routing schemes ([5], [11], [14]) for data traffic without deadlines 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 [9], Kwan and Ramanathan proposed a

multiple-path routing scheme for data traffic with deadlines. 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 recent papers ([3] and [4]), we proposed a family of load-balanced routing schemes for data traffic without deadlines. The family of routing protocols improves network performance by distributing randomly the traffic load over a set of selected paths to the destination. The proposed family was formulated for an IP (Internet Protocol) network with an irregular topology.

In this paper, we propose a load-balanced routing and scheduling scheme for data traffic with deadlines. The proposed scheme distributes the traffic load over a set of selected 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 deadlines have 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. In Section 2, we survey past approaches to transmission scheduling. Section 3 sketches the least laxity scheduling algorithm, which we use for packet scheduling. Section 4 sketches the load-balanced routing protocol. In Sections 5 and 6, we present the simulation model and our results. In Section 7, we draw conclusion and mention future work.

2. Scheduling Schemes for Traffic with Deadlines

In a packet-switched network, each communication link is statistically shared among many connections. 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.

The main drawback of the FIFO algorithms 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.

Jackson [8] 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 EDF scheduling policy [10]. 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 in 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. The third problem with EDF scheduling is that, if a packet is transmitted before its deadline at a certain node, then the time that was saved by this 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 EDF scheduling scheme.

3. 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 to minimize the number of packets that are dropped in transit.

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 1.

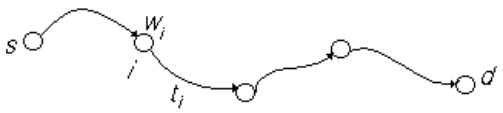

Figure 1. A simple connection

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 the 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 the current node to the destination by using the global topology table of the current node if C_p has not been estimated already.
3. Calculate the laxity $L_p = D_p - C_p$ and do one of the following steps according to the value of L_p.
   3.1 If $L_p = 0$, transmit the current packet immediately.
   3.2 If $L_p < 0$, discard the packet as the 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 the neighbor.

4. Overview of Load-Balanced Routing

In this section, we sketch how our method, called 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 traffic fluctuation, avoid congestion, and thus increase network throughput. Here is the basic idea of LBR:

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 [6].

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 will waste network resources.

To remedy the above mentioned problem of LBR-FR, 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*.

Choosing too small a value for *k* will exclude nodes that are far away (expensive to reach) 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 (LBR-BR1) [3]:

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

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

The value is a constant for the source *s* until the route update occurs. It, however, limits the effective bandwidth between each pair of the nodes to less than the capacity of the minimum cut separating the pair. This static value of *k* may be 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 choose the value of the parameter *k* to be fair to all the pairs of network nodes and consider a dynamic choice of parameter *k* (LBR-BR2) [4]:

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

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

The parameter *k* changes dynamically according to the length of the shortest path from the source *s* to the destination *d*. In this paper, we choose this dynamic value for the parameter *k*.

5. 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. We added timing constraints to the simulator and deadlines to the simulated traffic. A network configuration consists of a physical network, a routing and scheduling algorithm, and a workload.

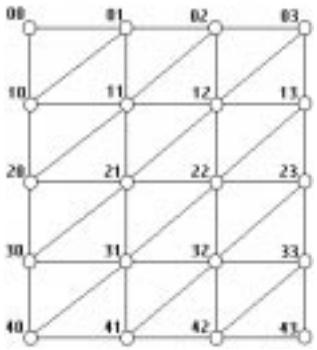

Figure 2. Network topology: 20 nodes, 43 bi-directional links, average degree of 4.3

The routing and scheduling algorithms are SFR-LL, SFR-EDF, SFR-FIFO, LBR-LL, LBR-EDF, and LBR-FIFO. Each SFR uses a shortest-path routing protocol 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 the three SFR protocols. To get a better understanding of our scheduling scheme (Least Laxity First), the performances 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 network topology given in Figure 2. All links have a bandwidth of 1.5 Mbits/sec. We assume that there are no link or node failures. Each node has a buffer space of 50,000 bytes (97 packets, each packet having 512 bytes). In order to calculate the cost of a path in the network, we use the average queuing and propagation delay of reaching every destination. The propagation delay of each link is 1 msec.

The workload consists of FTP (file transfer protocol) connections. A connection is a communication session established between end-user applications at source and destination nodes. All FTP connections have the following parameters: the packet length equals 512 bytes, the inter-packet generation time is 1 msec, and the window size is 500 packets. Traffic is introduced into the network by the FTP connections at the nodes they were attached to. Traffic consists 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 acknowledgments 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. 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 more network resources available to other packets and eases the demands on the shared network resources. Connections start when the simulation begins and are considered never-ending.

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

- *Missed deadlines*. The total number of packets dropped due to deadline misses during the measurement interval.
- *Fraction of missed deadlines*. Missed deadlines divided by missed deadlines plus the total number of packets acknowledged during the measurement interval.
- *Packet drops*. 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.

6. Simulation Results

Figures 3 and 4 show missed deadlines and the fraction of missed deadlines versus the number of connections, respectively. Missed deadlines is the total number of the packets that were

dropped because their deadlines had been expired. Missed deadlines and the fraction of missed deadlines in each LBR protocol are lower than in its corresponding SFR both when the number of connections is low and high.

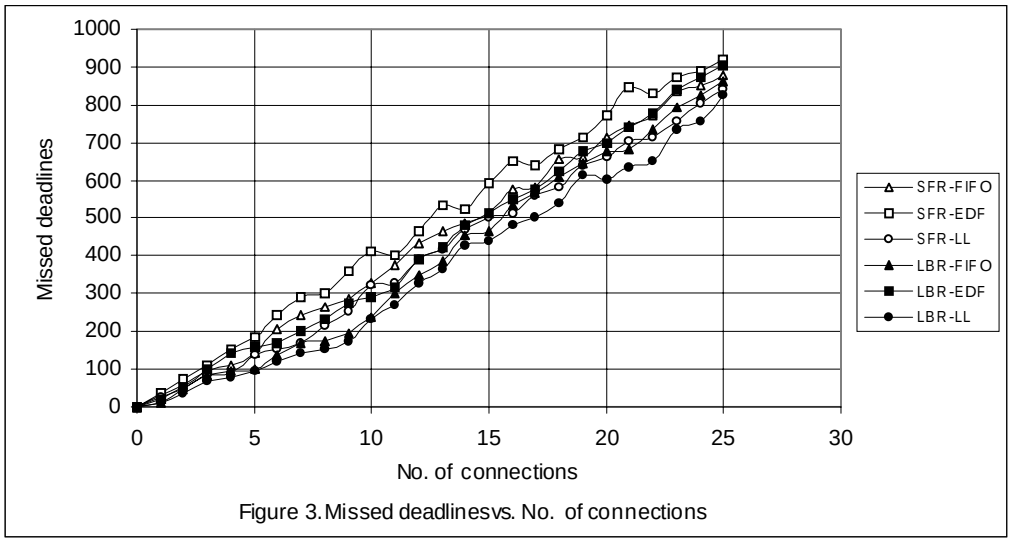

Figure 3. Missed deadlines vs. No. of connections

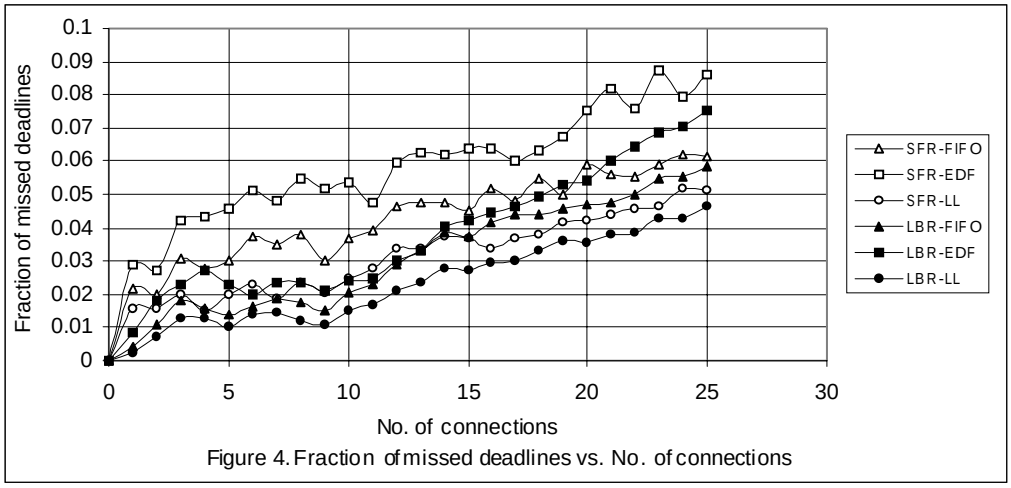

Figure 4. Fraction of missed deadlines vs. No. of connections

Figure 5 shows packet drops versus the number of connections. Packet drops is the total number of packets dropped due to buffer overflow at each node when the network becomes congested. Packet drops in each LBR protocol are lower than in its corresponding SFR at all network loads. In the Figures 3, 4 and 5, LBR-LL has the best performance with respect to missed deadlines, the fraction of missed deadlines and packet drops at most network loads.

Figure 6 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 outperforms 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.

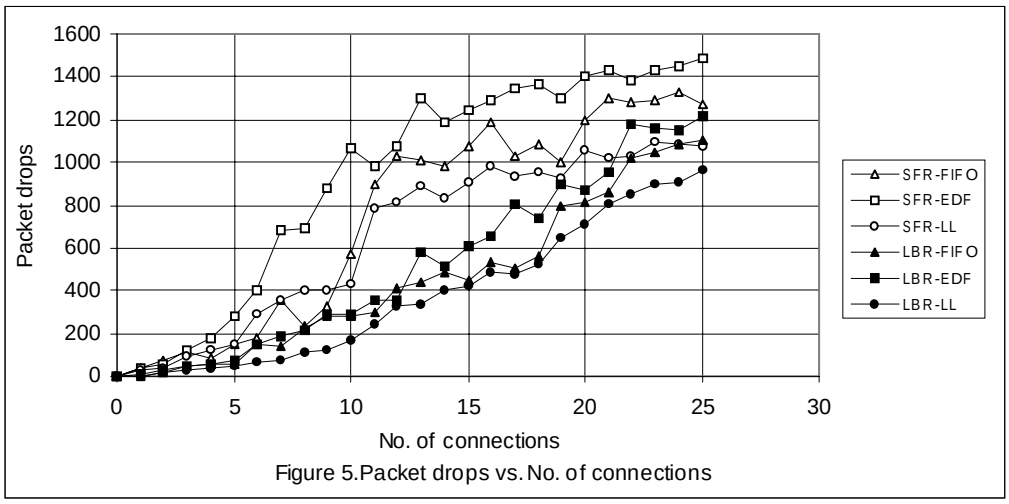

Figure 5.Packet drops vs. No. of connections

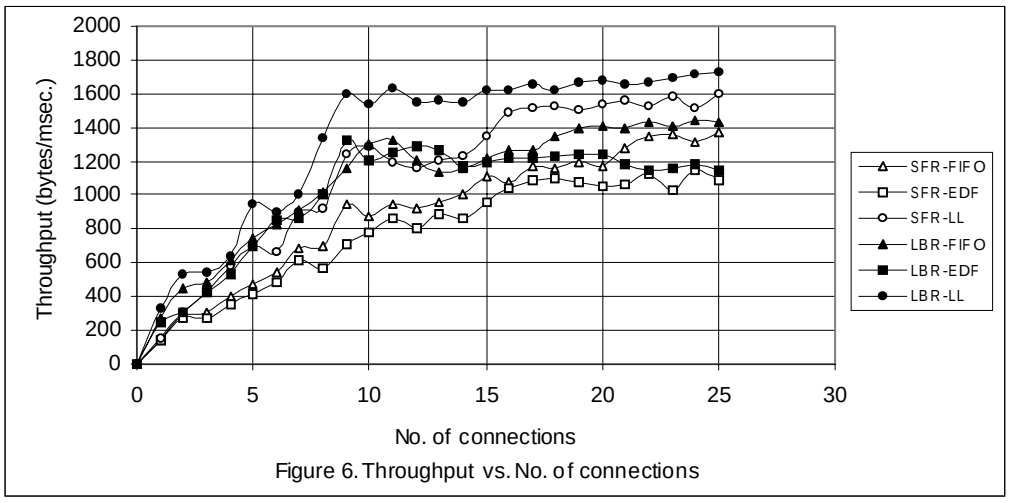

Figure 6. Throughput vs. No. of connections

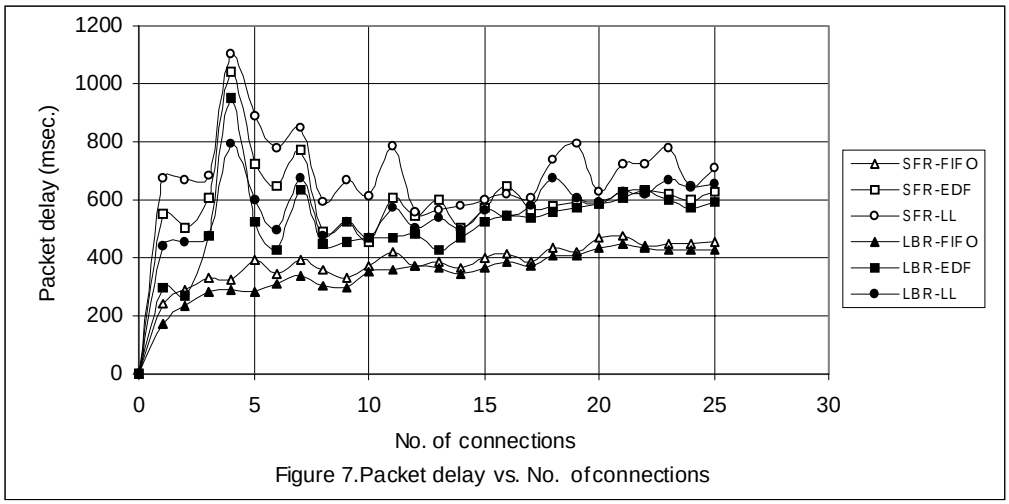

Figure 7.Packet delay vs. No. ofconnections

Figure 7 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 6 and 7, 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•(n-1), where n is the number of the network nodes.

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 the other 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 acknowledgments 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.

7. Conclusion and Future Work

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). The proposed scheme is simple and suffers from little control overhead. LBR-LL exhibits the best performance among the three LBR protocols at most times during the measurement interval. A direction of future work is to extend our load-balanced

routing and scheduling scheme to a real-time scheme, in which every packet in a data traffic should arrive by its deadline (guaranteed) or no more than a certain percentage of these packets of any stream may miss their deadline (statistical real-time).

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] 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.

[3] S. Bak and J. A. Cobb, *Randomized Distance-Vector Routing Protocol*, Proceedings of ACM Symposium on Applied Computing, San Antonio, Texas, Feb. 1999.

[4] S. Bak, J. A. Cobb, E. L. Leiss, *Load-Balanced Routing via Randomization*, Proceedings of CLEI, August 30, 1999.

[5] R. Cole, B.M. Maggs, F. Meyer auf der Heide, M. Mitzenmacher, A.W. Richa, K. Schroeder, R.K. Sitaraman, and B. Voecking, *Randomized Protocols for low-congestion circuit routing in multistage interconnection networks*, Procceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 378-388, May 1998.

[6] D. P. Bertsekas, *Linear Network Optimization: Algorithms and Codes*, The MIT Press, 1991.

[7] J. A. Cobb and M. G. Gouda, *Balanced Routing*, IEEE Proceedings of the International Conference on Network Protocols, 1997.

[8] J. R. Jackson, *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Science Research Project, UCLA, 1955.

[9] 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.

[10] 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.

[11] 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, July 1995.

[12] S. M. Rao and A. M. K. Cheng, *Scheduling and Routing of Real-Time Traffic in Packet-Switched Networks*, Technical Report, Department of Computer Science, University of Houston, Sep. 1997.

[13] D. Sidhu, R. Nair and S. Abdallah, *Finding Disjoint Paths in Networks*, Proceedings of the 1991 ACM SIGCOMM Conference, 1991.

[14] L. G. Valiant, *A Scheme for Fast Parallel Communication*, SIAM Journal on Computing, Vol. 11, No. 2, May 1982.

[15] Z. Wang and J. Crowcroft, *Shortest Path First with Emergency Exists*, Proceedings of the 1990 ACM SIGCOMM Conference, 1990.

[16] 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.