Responsiveness of Routing Protocols to Link Failures and Repairs*

Cengiz Alaettinoğlu, A. Udaya Shankar†, Klaudia Dussa-Zieger, Ibrahim Matta

Department of Computer Science
University of Maryland
College Park, Maryland 20742

June 1, 1993

Abstract

Responsiveness of adaptive next-hop routing protocols has usually been studied in terms of the message (or hop-count) complexity of adapting to a topology change. These measures ignore the effect of workload. In this paper, we study the relationship between these "workload-independent" measures and workload-dependent measures such as delay, throughput, and data load (fraction of network capacity used by workload packets). We examine both transient characteristics (e.g. recovery times and overshoots) and steady-state characteristics (e.g. average values) of these workload-dependent measures. Our overall conclusions are that for networks like NSFNET (1) the routing load (fraction of network capacity used by routing packets) has a close correlation to the workload-independent measures, and (2) there is no correlation between the routing load and system performance.

* This work is supported in part by RADC and DARPA under contract F30602-90-C-0010 to UMIACS at the University of Maryland, and by National Science Foundation Grant No. NCR 89-04590. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, RADC, or the U.S. Government. Computer facilities were provided in part by NSF grant CCR-8811954.

†Also with Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742.

Contents

1 Introduction

In adaptive next-hop routing, each link has a cost that is regularly updated to reflect the link up/down status and the traffic flowing through the link. By using these *link costs*, a routing protocol adapts its next-hops to account for topology changes, i.e. failures, repairs and link cost changes.

The responsiveness of adaptive routing protocols has received much attention [16, 10, 21, 4, 2, 23]. These studies usually consider a succession[1] of topology changes that leaves the system in a state, say s_0, where the routes are not shortest. Assuming that there are no further topology changes, they obtain the number of messages (*message complexity*) and the number of successive hops of message propagations (*hop-count complexity*) required for the system to reach a new state, say s_1, where the routes are shortest with respect to the link costs at s_0.[2]

However, in reality when routes change, the traffic pattern changes. As a result, when link costs are next updated these routes are not optimal with respect to the new link costs; that is, the link costs at s_1 are different from the link costs at s_0, and the routes at s_1, although optimal with respect to the link costs at s_0, are not optimal with respect to the link costs at s_1. Thus the process of adapting the next-hops is repeated, until stable and optimal routes are attained; that is, after s_1, the systems enters a new state s_2, then s_3, and so on, until a stable state s_n, where n depends upon the workload, the link cost function, the routing algorithm, etc. It is the duration and cost of this adaptive behavior that we are concerned with in this paper.

The users of the network are interested in real-time performance measures like delay, throughput, which are *workload-dependent* rather than indirect measures such as message and hop-count complexities which are *workload-independent*. It is the workload that causes the feedback between route changes and link cost changes.

This paper examines the relationship between workload-independent and workload-dependent measures. We consider (1) the transient behavior of routing algorithms after a link failure (and

[1] Often just one topology change.

[2] [23] compares SPF, Distributed Bellman-Ford and DUAL algorithms with respect to measures such as number of paths with loops after a node/link status change using simulations. However, it has no workload and assumes unit link delays and zero processing time at the nodes.

a link repair) by studying the recovery time, overshoots, and undershoots in delay, throughput and load; and (2) the steady-state performance of routing algorithms while they are subjected to Poisson link failures and repairs. Our studies are done using a routing simulator called MaRS [1]. We consider uniform and hotspot-with-background workloads. Some very preliminary results for Poisson failures and repairs and uniform workload was reported in [20].

We consider three routing protocols: Shortest Path First (SPF) [15], Merlin-Segall (MS) [16, 19], and Extended Bellman-Ford (ExBF) [4]. SPF is a link-state routing algorithm, whereas MS and ExBF are distance-vector routing algorithms. (They are described in section 2.) The space requirement at each node is $O(N^2)$ for SPF, and $O(N \times e)$ for MS and ExBF, where N is the number of nodes in the network and e is the average degree of a node. The worst-case message complexity is $O(E)$ for SPF, $O(H \times E)$ for MS, and $O(2^N)$ for ExBF, where E is the number of network links, and H is the length of a maximum length shortest route between any two nodes in hops. The hop-count complexity is $O(D)$ for SPF, $O(H^2)$ for MS, and $O(N)$ for ExBF, where D is the network diameter in hops.

Our observations indicate that routing load is very small compared to data load, and is a negligible fraction of the overall network capacity. We also do not find any correlation between routing load and system performance. We observed that the recovery behavior for instantaneous routing load is not related to the recovery behavior for instantaneous delay and throughput. We believe that the routing load reflects the workload-independent measures (i.e. message and hop-count complexities).

The paper is organized as follows. In section 2, we describe SPF, MS, and ExBF. In section 3, we describe the parameters of the simulations. In section 4, we present our simulations and observations on the transient behaviors of routing algorithms after a link failure and a link repair. In section 5, we present our simulations and observations on the performance of routing algorithms subject to Poisson failures and repairs. We give our conclusions in section 6.

2 Overview of Routing Algorithms

In next-hop routing, each data packet has its destination node id, and each node (other than the destination) maintains a neighboring node id, referred to as *next-hop*, to forward the packets to. The next-hop of a node can also be *nil* which means that the node does not know how to reach the destination. A data packet for a destination for which the next-hop is *nil* is dropped.

Every routing algorithm does the following functions: (a) each node maintains for each of its outgoing links a dynamic *link cost* which is updated whenever the link status changes or regularly according to the traffic flowing through the link; (b) link cost information is disseminated across the network; (c) based on received link cost information, each node maintains and regularly updates its next-hop for each destination. Different routing algorithms differ in how they implement (b) and (c).

Link-State Algorithms

In the link-state approach, each node maintains a view of the network topology with a cost for each link. To keep these views up-to-date, each node regularly broadcasts the link costs of its outgoing links to all other nodes using a protocol such as flooding (i.e. it sends link cost information to its neighbors, they forward this information to their neighbors and so on). As a node receives this information, it updates its view of the network topology and applies a shortest path algorithm to choose its next-hop for each destination. Note that some of the link costs in a node's view can be old because of long propagation delays, partitioned network, etc. Such inconsistent views of network topology at different nodes might lead to loops in the next-hops. However, these loops are *short-lived*, because they disappear in the time it takes a message to traverse the diameter of the network.

SPF is a link-state algorithm where (1) each node calculates and broadcasts the costs of its outgoing links periodically or whenever a failure/repair occurs, and (2) Dijkstra's shortest path algorithm [6] is applied to the view of the network topology to determine next-hops.

Distance-Vector Algorithms

In the distance-vector approach, for each destination d, every node i maintains a set of distances $\{D_i(j)\}$ where j ranges over the neighbors of i. Node i treats neighbor k as a next-hop for a packet destined for d if $D_i(k)$ equals $\min_j\{D_i(j)\}$. The quantity $\min_j\{D_i(j)\}$ is referred to as the minimum distance to d. The intention is that from any node i the succession of next-hops should lead to d; furthermore $D_i(j)$ should equal the sum of the costs of the links on the path from i to d where the first hop is to j, and the rest of the path is the succession of the next-hops. To keep these distances up-to-date, each node regularly updates the cost of its outgoing links, and hence its distances and next-hops. If this causes a change in a minimum distance, it sends the new minimum distance to its neighbors; if as a result a minimum distance of a neighbor changes, the neighbor repeats the process.

The above distance-vector algorithm is the classical Distributed Bellman-Ford algorithm [3, 8]. It is well known that this algorithm, in addition to short-lived next-hop loops, can have *long-lived* loops of duration proportional to link cost changes [18]. MS and ExBF have mechanisms to avoid long-lived loops.

MS is a distance-vector algorithm which avoids both short-lived and long-lived loops. For each destination, MS guarantees that the next-hops on the nodes that can reach the destination form a tree rooted at the destination. MS attains loop-free paths by coordinating the next-hop updates for each destination as a diffusion computation [7] which is started by the destination. When a failure/repair occurs, a request message is sent to each destination. Periodically or upon receiving a request message, the destination starts a new diffusion computation[3].

ExBF is another distance-vector algorithm which avoids long-lived loops but not short-lived loops. For each destination d, every node i maintains, in addition to the set of distances $\{D_i(j)\}$, a set of prefinal nodes $\{P_i(j)\}$, where j ranges over the neighbors of i. The intention is that $P_i(j)$ is

[3] The diffusion computation is as follows. The destination node starts the computation by sending the distance zero to all of its neighbors, and waits for distance messages from all its neighbors, at which point the computation is over. Every node i, other than the destination, does the following: upon receiving a distance message for the destination from its next-hop, it calculates its distance, and sends its distance to all of its neighbors except its next hop. After receiving distance messages from all its neighbors, node i chooses its new next-hop, and sends the new distance to its old next-hop. If node i's next-hop is *nil*, it chooses the first neighbor that reports a finite distance as its next-hop.

the last node before d on the next-hop path from i to d through j. Because node i has the prefinal node for every destination, it can obtain a complete path $s_0, s_1, \ldots s_{k-1}, s_k$ to d via j, where $s_0 = j$, $s_k = d$, and for all l, s_l is the prefinal node for destination s_{l+1}. Furthermore, it turns out that after short-lived inconsistencies die out, this "prefinal node" path equals the "next-hop" path, and $D_i(j)$ is its cost. ExBF avoids long-lived loops by the following trick: When node i disseminates distance changes for destination d, it reports ∞ as its distance to any neighbor in its prefinal node path. Periodically or whenever a failure/repair occurs, link costs are re-calculated and if there is a change in the cost of its outgoing links, a node starts a computation in which its distances are updated to reflect that change.

3 Simulation Parameters

In this section, we describe the range of parameters exercised in our simulations. Our simulation studies were done on a recently developed discrete-event simulator, MaRS (Maryland Routing Simulator). It provides a flexible platform for the evaluation and comparison of network routing algorithms. It allows the user to define a network configuration, control its simulation, log the values of selected parameters, calculate average and instantaneous performance measures, and save, load and modify network configurations. The scheduling of events can be done using random number generators, a trace file, or both. MaRS is implemented in C on a UNIX environment. It has an optional graphical (X Window System) interface. It is available in public domain by anonymous ftp from `ftp.cs.umd.edu` [1].

A network configuration consists of a *physical network* (nodes and links), a *routing algorithm* and a *workload*. In this paper, the physical network models the NSFNET-T1-Backbone illustrated in Figure 1. The routing algorithms considered are SPF, MS and ExBF.

The *workload* is defined in terms of *connections*. Each connection is associated with a *source* node and a *sink* node. The source produces packets destined for the sink. The packets are forwarded across the network and consumed by the sink. In this paper, we use FTP connections (a major NSFNET application [11]). An FTP connection incorporates a static send window-based flow control mechanism and an acknowledgment-with-retransmission mechanism using roundtrip time estimates. Retransmissions can be due to node or link failures, or buffer space limitation, or *nil* next-hops. We use this simple flow control scheme to keep the interactions between routing and flow control algorithms minimal. With a more sophisticated flow control mechanism, such as slow-start in TCP, the interaction would be more complicated, and we believe it would be hard to draw conclusions about the performance of the routing algorithm itself.

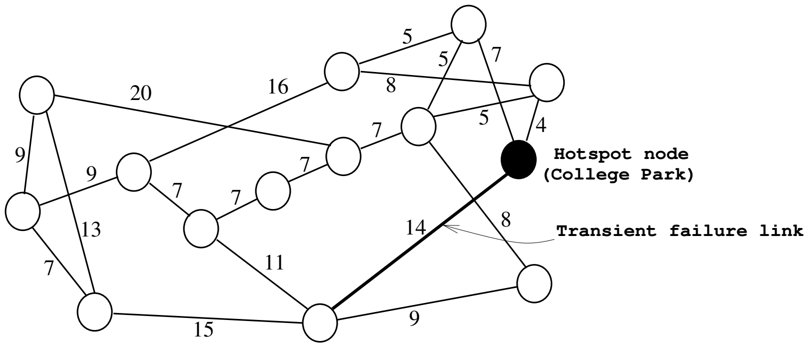

Figure 1: NSFNET-T1-Backbone (14 nodes, 21 bidirectional links, average degree 3).

Assumptions and parameters used in this paper were predetermined either consistently with those made in the literature, e.g. [15] [12] [9] [5], or from statistics provided by Merit/NSFNET Information Services, or by simulations with MaRS.

Physical Network

Link propagation delays in milliseconds are indicated in Figure 1. Also indicated is the node which will be a *hotspot* in some of our simulations, and an incident link which will be used in our transient failure simulations. All links have the same bandwidth, 1.5Mbit/sec. All nodes have adequate buffer space for buffering packets awaiting processing and forwarding. Workload packets can be processed (includes parsing the packet header, consulting the routing table, and adding the packet to the appropriate outgoing packet queue) in parallel. The processing time equals 1 msec. Routing packets have priority over workload packets in the nodes' outgoing packet queues.

Routing packets received at a node are processed sequentially. The size and processing time of routing packets depend on the algorithm. For SPF, a link-state packet sent by a node is $16 + 8e$ bytes long, where e is the number of neighbors of the node; it is processed in 6 msec. For MS, there are two types of packets: one conveying a distance, which is 20 bytes long and processed in 3 msec, and one carrying a request for initiating a new computation, which is 16 bytes long and processed in 2 msec. For ExBF, a distance-vector packet sent by a node is $24 + 12n$ bytes long, where n is the number of destinations for which a distance has been received; it is processed in 4.5 msec.

Link Cost Function

The link cost function used in this paper is the hop-normalized-delay function [12]. It monitors the *average packet delay* (queueing and transmission) and *average packet transmission time* for the link during the last update period. Assuming an $M/M/1$ model, it calculates the link utilization. The link utilization is first exponentially averaged with the previous value of the average link utilization. The result is scaled with a slope and an offset and then subjected to minimum and maximum bounds. Then its value is adjusted so that the difference from the previous value of the link cost is less than a movement limit. The resulting number is the new cost of the link. The parameters of hop-normalized-delay function in this paper is as follows: offset equals 0, slope equals 10, minimum limit equals 1, maximum limit equals 10, and movement limit equals 1. The link-cost update period is uniformly distributed with mean 10 seconds and standard deviation 1 second.

Workload Distribution

We consider two types of source-sink pair distributions for the FTP connections:

- *Uniform workload* with parameter U indicating the average number of FTP connections between every two nodes, and
- *Hotspot-with-background workload* with parameters H and B, where H is the average number of FTP connections from every node to the hotspot node, and B is the average number of FTP connections between every pair of nodes excluding the hotspot node.

Each connection has data packet size equal to 512 bytes, inter-packet generation time equal to 150 msec, and window size for flow control equal to 8 packets. All connections start when the simulation begins and are unending.

4 Responsiveness to a Link Failure and a Link Repair

To study the responsiveness of routing algorithms to a link failure and a link repair, we look at transient performance measures such as overshoots, undershoots, and recovery times of various instantaneous measures after a link failure, or a link repair. An *instantaneous measure* is a function of time updated periodically every 500 msec. An instantaneous measure at time t is based on statistics collected during the 500 msec period preceding t. We consider the following instantaneous measures:

- *Instantaneous Throughput at time t.* Total number of data bytes acknowledged in the period preceding t divided by the length of the period.
- *Instantaneous Delay at time t.* Total delay of data packets acknowledged in the period preceding t divided by the number of data packets acknowledged in this period.
- *Instantaneous Data (Routing) Load at time t.* Fraction of the network capacity used by workload (routing) packets in the period preceding t. (The network capacity is the sum of all link capacities.)

We define *recovery time* of an instantaneous measure to be the time needed for the measure to reach its steady-state value after a link failure, or a link repair. The overshoot (undershoot) of an instantaneous measure is its maximum (minimum) value above (below) the steady-state value during the recovery time. Recovery times indicate how quickly a routing algorithm adapts to the topology changes. Overshoots and undershoots indicate how badly it performs while trying to adapt to the topology changes. Such kinds of measures are regularly used in control systems literature to characterize the dynamic response of systems [17].

For any measure x that we present in this paper (including instantaneous measures), we have computed 95% confidence intervals using the method of independent replications [14, 22] (i.e. $x = (x_1 + x_2 + \cdots + x_n)/n$, where $x_1, x_2, \ldots, x_n$ are the measures obtained using the different independent simulation runs). In almost all cases, the size of the confidence intervals is less than 10%. In some cases, it is higher than 10%. Even when the size of the confidence interval is highest, our conclusions are not affected.

To examine the responsiveness of the routing protocols, we use hotspot-with-background workload and subject only one link to failure (the transient failure link indicated in Figure 1). It is failed at time 100 seconds and repaired at time 200 seconds.

Figures 2 shows the instantaneous delay, throughput, and routing load versus time, for $B = 1$ and $H = 2$. Figures 3 through 8 show the same for $B = 1$ and $H = 3$ through 8.

We begin our observations by examining the behavior of instantaneous delay. Before the failure (at time 80 seconds), it is around a steady-state value (e.g. 65 msecs for $H = 2$). As a result of the link failure at time 100 seconds, it increases to a new steady-state value (e.g. 71 msecs for $H = 2$). The recovery time of the instantaneous delay after the failure is very small (less than 1 second for $H = 2, 3, 4$; 6 seconds for $H = 5, 6$; 7 seconds for $H = 7$; and 8 seconds for $H = 8$), and there are no undershoots and overshoots except for $H = 7, 8$. For $H = 7, 8$, there is an undershoot followed by an overshoot.

When the link repairs at time 200 seconds, the instantaneous delay shoots up and decreases slowly to the steady-state value before the link failure (except for $H = 6$ where it approaches a different steady-state value). With increasing H, the overshoots increase, and the recovery time first increases and then starts decreasing. The maximum recovery time is reached at $H = 5$ (e.g. it is between 20-35 seconds for $H = 2$, between 90-116 seconds for $H = 5$, between 45-65 seconds for $H = 8$)[4]. Note that the recovery time can be as high as 12 link-cost update periods.

Our explanation for the high overshoots and large recovery times after the link repair is that the repaired link becomes very attractive to many source-destination pairs since it reports a low cost.

Similar conclusions can be drawn for instantaneous throughput. Regarding the behavior of instantaneous routing load, after the link failure, it shoots up and falls down to the same steady-state value as before. The same happens after the link repair as well. The recovery time in instantaneous routing load for ExBF can be as high as 50 seconds whereas it is less than 5 seconds for SPF and MS. We believe that the instantaneous routing load is highly related to the workload-independent measures (message and hop-count complexities). We observe no relation between the

[4] The workload $H = 6$ corresponds to saturation in steady-state if all links were always operational; the workload $H = 5$ corresponds to saturation in steady-state if the failed link did not exist and all links were always operational.

behavior of instantaneous routing load and the behaviors for instantaneous delay and throughput.

Our comparison of the routing algorithms is as follows: After a link failure, with respect to the recovery times, undershoots and overshoots in instantaneous throughput and delay, the three algorithms are equivalent; with respect to instantaneous routing load, SPF and MS are equivalent and superior to ExBF. After a link repair, with respect to the recovery times in instantaneous throughput and delay, SPF is superior, followed closely by ExBF, while MS is much worse; with respect to overshoots in instantaneous delay and undershoots in instantaneous throughput, MS is the best, followed by SPF, and then ExBF; with respect to instantaneous routing load, the same observation as for the link failure holds.

5 Performance for Poisson Failures and Repairs

In Poisson failures, a link is characterized by an average *uptime* and an average *downtime*; the uptimes and downtimes are independent and exponentially distributed. All links in the network which are subject to failure have the same average uptime and average downtime. The failures of the network are characterized by two indirect parameters, the *up+down time* (UDT), and the *failure intensity* (FI). UDT is the sum of the average link uptime and average link downtime; thus, a small UDT means a high frequency of topology changes. FI is the average number of failed links in the network at any instant. These indirect parameters are varied by changing the average link uptime and average link downtime. For example, if all 21 links in the network are subject to failure, then an average link uptime of 4 seconds and an average link downtime of 3 seconds yield UDT equal to 7 ($= 4 + 3$) and FI equal to 9 ($= (3/7) \times 21$).

We consider performance measures of throughput, delay, data load and routing load. These measures are based on statistics collected over a large *measurement interval*, which is the duration of the simulation except for an initial "startup interval" (to eliminate transient effects due to empty initial network). Thus:

- *Throughput.* Total number of data bytes acknowledged during the measurement interval divided by the length of the measurement interval.
- *Delay.* Total delay of all data packets acknowledged during the measurement interval divided

by the number of data packets acknowledged in this interval, where *delay* of a data packet is defined to be the time difference between sending a packet for the first time and receiving the corresponding acknowledgement.

- *Data (Routing) Load.* Fraction of the network capacity used by workload (routing) packets during the measurement interval.

Our simulations are divided into two groups: uniform workload and hotspot-with-background workload. With the uniform workload, every link in the network is subject to failure. With the hotspot-with-background workload, only the links incident to the hotspot node are subject to failure.

In all simulations, the routing load is very small compared to the data load and a negligible fraction of the overall network capacity. Hence, we do not show any plot of it.

Uniform workload

Varying uniform workload U for fixed UDT and FI

The first three graphs in Figure 9 show data load, throughput, and delay versus U in the range 0.25 to 5, for $UDT = 21$ and $FI = 1$. The fourth graph shows the delay versus throughput. As the workload increases, the data load and throughput first increase linearly, and then level off as the system becomes saturated. The delay increases very slowly at first, and then very rapidly after the saturation point. Observe that as the workload increases, the network behaves more as a closed queueing network [13]. The reason for this is that the number of packets in the network is upper bounded because each connection can have at most one send window's worth of packets in the network[5]. MS performs worse than the other two algorithms (more than 50% worse in delay for $U = 1$ and $U = 1.5$). ExBF is worse than SPF by at most 20% delay (for $U = 2$). SPF performs the best.

[5] Note that at high workload when the system is saturated, packets are being produced at a higher rate than they are admitted into the network because of flow control. Thus, the produce window of some sources is expanding without bound. However, the network itself (including the send windows) reaches steady-state.

Varying UDT for fixed FI and uniform workload U

Figure 10 shows data load, throughput and delay versus UDT in the range 5.25 to 105, for $U = 1$ and $FI = 1$. SPF and ExBF are almost insensitive to changes in UDT. With increasing UDT (i.e. with less frequent topology changes), the data load and delay for MS decreases. This is because MS is spending a smaller fraction of time in diffusion computations during which some next-hops are set to *nil* causing packets to be dropped. In any case, MS performs much worse than the other two algorithms both in delay and data load.

Varying FI for fixed UDT and uniform workload U

Figure 11 shows data load, throughput and delay versus FI in the range 0.5 to 8, for $U = 1$ and $UDT = 21$. As failure intensity increases, data load increases (workload packets now take longer routes, hence they use more resources), then levels off, and then decreases (at high FI, next-hops to many destinations are *nil*, hence workload packets destined to these destinations are dropped at the source and thus do not consume resources). Delay first increases slowly, then sharply, and then at a slower rate. The difference between the algorithms is at first small, then increases, and then decreases. This is because with low number of failed links, good routes can still be found; and with high number of failed links, any route chosen is as bad. With moderate FI, the choice of routes becomes more critical.

The reason for the decrease in data load is that with many links failed, the network may become partitioned and many destinations become unreachable; hence packets are dropped at the source node. Algorithms are about equivalent in throughput. MS has higher data load. MS also has higher delay except for very high FI values where the algorithms are equivalent.

Our comparison of the algorithms is as follows: The three algorithms are equivalent with respect to throughput. ExBF and SPF are equivalent with respect to data load, while MS has higher data load. The algorithms differ highly in delay. SPF has the smallest delay, followed closely by ExBF, and then MS.

Hotspot-with-background workload

Varying hotspot workload H for fixed background workload B, UDT and FI

The first three graphs in Figure 12 show data load, throughput, and delay versus H in the range 2 to 8, for $B = 1$, $UDT = 6$, and $FI = 1$. The fourth graph shows the delay versus throughput. The curves follow the typical behavior of closed queueing networks. For almost all values of H, MS has higher data load, lower throughput and lower delay than SPF and ExBF which are almost equivalent.

Varying UDT for fixed FI and hotspot-with-background workload (H, B)

Figure 13 shows data load, throughput, and delay versus UDT in the range 1.5 to 15, for $B = 1$, $H = 4$ and $FI = 1$. The behavior is similar to the behavior in the uniform workload case, except here the throughput for MS increases with less frequent topology changes. For all values of UDT, MS has higher data load, lower throughput, and lower delay than the other two algorithms which have similar data load and throughput. SPF has lower delay than ExBF (e.g. ExBF has 40% higher delay than SPF for $UDT = 3$).

Varying FI for fixed UDT and hotspot-with-background workload (H, B)

Figure 14 shows data load, throughput, and delay versus FI in the range 0.25 to 2.5, for $B = 1$, $H = 4$ and $UDT = 6$. The behavior is similar to the behavior in the uniform workload case, except here for low values of FI, ExBF has much higher delay than the other two algorithms. For almost all values of FI, MS has higher data load, lower throughput, and lower delay than the other two algorithms. SPF and ExBF have approximately equivalent data load and throughput.

Our explanation for MS having the lowest delay, while having lower throughput and higher data load, is that in response to a link failure, MS sets next-hop entries for some destinations to *nil* to avoid loops entirely (while the other algorithms allow short-lived loops). These entries stay *nil* until the diffusion computation terminates. While a next-hop entry for a destination is *nil*, packets received for that destination are dropped, which in turn causes retransmissions.

Our comparison of the algorithms is as follows: ExBF and SPF are equivalent with respect to throughput and data load. MS has lower throughput and higher data load. The algorithms differ highly in delay. MS has the smallest delay, followed by SPF, and then ExBF. However, since for the same H, MS has lower throughput. By looking at the delay versus throughput graph (instead of delay versus H), we conclude that SPF is the best, followed by ExBF, and then MS. SPF is superior to ExBF especially for small up+down time and small failure intensity values.

6 Conclusions

We observed that the recovery behavior for instantaneous routing load is not related to the recovery behavior for instantaneous delay and instantaneous throughput. We believe that the behavior of instantaneous routing load after a link failure or a repair reflects the workload-independent measures.

Even though ExBF has high message complexity, our simulations show that it has a comparable routing load to SPF in steady-state, and that the overall routing load is very small compared to data load and a negligible fraction of the overall network capacity. We also do not find any correlation between routing load and system performance. Hence, reducing message complexity and routing load should not be a primary criteria in proposing new routing algorithms.

The diffusion computation mechanism used by MS to avoid short-lived loops causes MS to drop too many packets in response to link failures. In spite of allowing short-lived loops, SPF and ExBF perform better than MS. This implies that short-lived loops do not degrade performance.

Regarding the transient behavior of routing algorithms after a link failure and a link repair, we found SPF and ExBF perform comparable while MS is much worse. Regarding the average performance of routing algorithms while they are subjected to Poisson failures and repairs, SPF performs the best, followed by ExBF, and then MS.

Our simulations were limited to only one network topology, which although realistic was not very large. Simulations on more topologies are required to further analyze the routing algorithms, and to see how the performance scales with network parameters such as the number of network nodes, network diameter, average nodal degree, etc.

References

[1] C. Alaettinoğlu, K. Dussa-Zieger, I. Matta, and A. U. Shankar. MaRS (Maryland Routing Simulator) – Version 1.0 User's Manual. Technical Report UMIACS-TR-91-80, CS-TR-2687, Department of Computer Science, University of Maryland, College Park, MD 20742, June 1991.

[2] B. Awerbuch. Shortest Paths and Loop–Free Routing in Dynamic Networks. In Proc. *ACM SIGCOMM '90*, pages 177–187, Philadelphia, Pennsylvania, September 1990.

[3] D. Bertsekas and R. Gallager. *Data Networks*, pages 297–333. Prentice-Hall, Inc., 1987.

[4] C. Cheng, R. Riley, S. P. R. Kumar, and J. J. Garcia-Luna-Aceves. A Loop-free Bellman-Ford Routing Protocol Without Bouncing Effect. In Proc. *ACM SIGCOMM '89*, pages 224–237, September 1989.

[5] W. Chou, A.W. Bragg, and A.A. Nilson. The Need for Adaptive Routing in the Chaotic and Unbalanced Traffic Environment. *IEEE Transactions on Communications*, 29:481–490, April 1981.

[6] E. Dijkstra. A Note on Two Problems in Connection with Graphs. *Numer. Math.*, 1:269–271, 1959.

[7] E. Dijkstra and C. Scholten. Termination Detection for Diffusing Computations. *Information Processing Letters*, 11(1):1–4, 1980.

[8] L. Ford and D. Fulkerson. *Flows in Networks*, pages 297–333. Prentice-Hall, Inc., 1962.

[9] S. A. Heimlich. Traffic Characterization of the NSFNET National Backbone. In Proc. *Proc. USENIX '90*, pages 207–227, Washington, D.C., January 1990.

[10] J. M. Jaffe and F.H. Moss. A Responsive Distributed Routing Algorithm for Computer Networks. *IEEE Transactions on Communications*, COM-30(7):1758–1762, July 1982.

[11] D. Johnson. NSFnet Report. In Proc. *of the Nineteenth Internet Engineering Task Force*, pages 377–382, University of Colorado, National Center for Atmospheric Research, December 1990.

[12] A. Khanna and J. Zinky. A Revised ARPANET Routing Metric. In Proc. *ACM SIGCOMM '89*, pages 45–56, September 1989.

[13] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. John Wiley and Sons, Inc., 1976.

[14] Stephen S. Lavenberg. *Computer Performance Modeling Handbook*. Academic Press, 1983.

[15] J. M. McQuillan, I. Richer, and E. C. Rosen. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[16] P. M. Merlin and A. Segall. A Failsafe Distributed Routing Protocol. *IEEE Transactions on Communications*, COM-27(9):1280–1287, September 1979.

[17] K. Ogata. *Discrete-Time Control Systems*. Prentice-Hall, Inc., 1987.

[18] B. Rajagopalan and M. Faiman. A New Responsive Distributed Shortest-Path Routing Algorithm. In Proc. *ACM SIGCOMM '89*, pages 237–246, September 1989.

[19] A. Segall. Advances in Verifiable Fail-Safe Routing Procedures. *IEEE Transactions on Communications*, COM-29(4):491–497, April 1981.

[20] A. U. Shankar, C. Alaettinoğlu, I. Matta, and K. Dussa-Zieger. Performance Comparison of Routing Protocols Using MaRS: Distance-Vector versus Link-State. In Proc. *ACM SIGMETRICS/PERFORMANCE '92*, New Port, Rhode Island, June 1992.

[21] K. G. Shin and M. Chen. Performance Analysis of Distributed Routing Strategies Free or Ping-Pong-Type Looping. *IEEE Transactions on Computers*, 1987.

[22] H. A. Taha. *Operations Research : An Introduction*. Macmillan publishing Co., second edition, 1976.

[23] W. Zaumen and J.J. Garcia-Luna-Aceves. Dynamics of Distributed Shortest-path Routing Algorithms. In Proc. *ACM SIGCOMM '91*, September 1991.

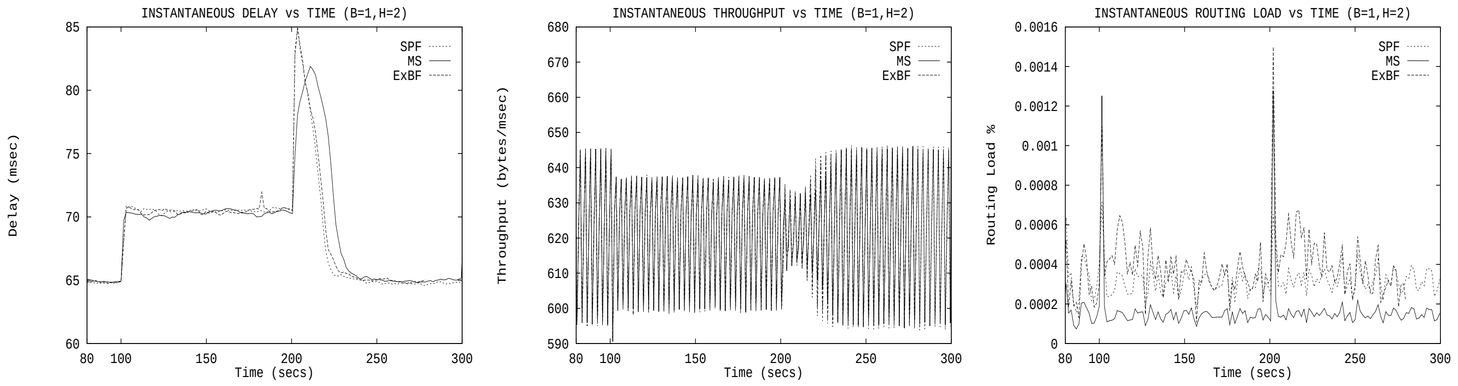

Figure 2: Responsiveness to a link failure and repair for $B = 1$ and $H = 2$.

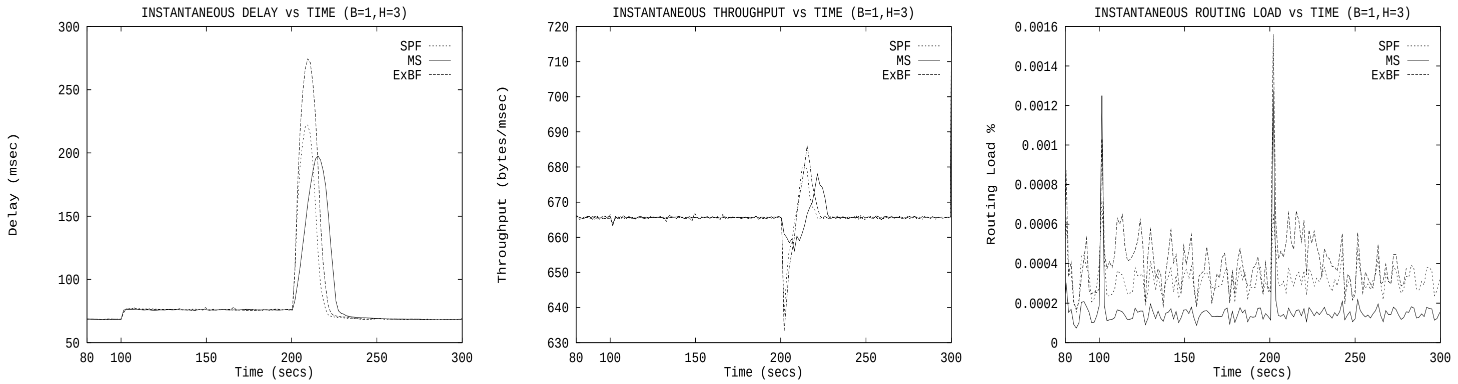

Figure 3: Responsiveness to a link failure and repair for $B = 1$ and $H = 3$.

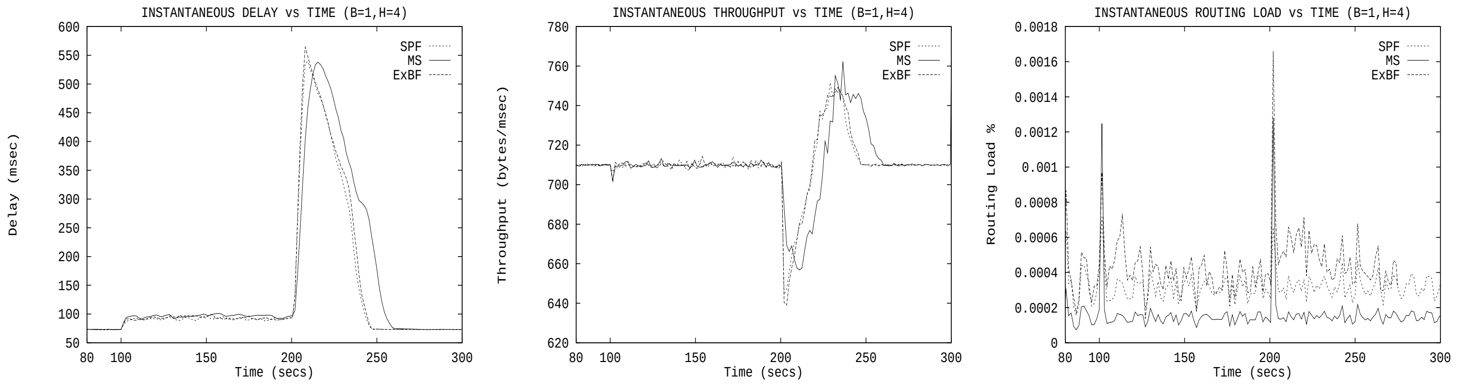

Figure 4: Responsiveness to a link failure and repair for $B = 1$ and $H = 4$.

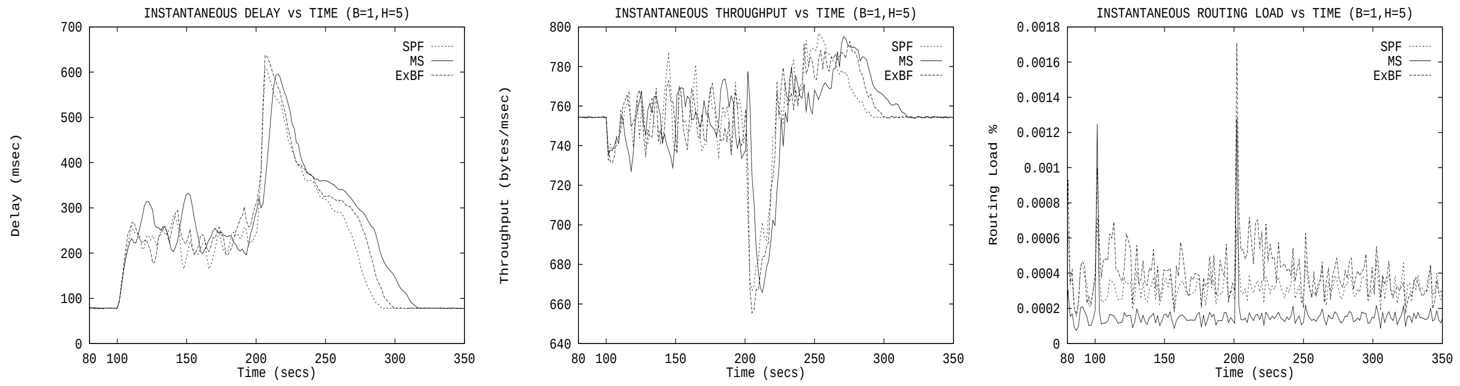

Figure 5: Responsiveness to a link failure and repair for $B = 1$ and $H = 5$.

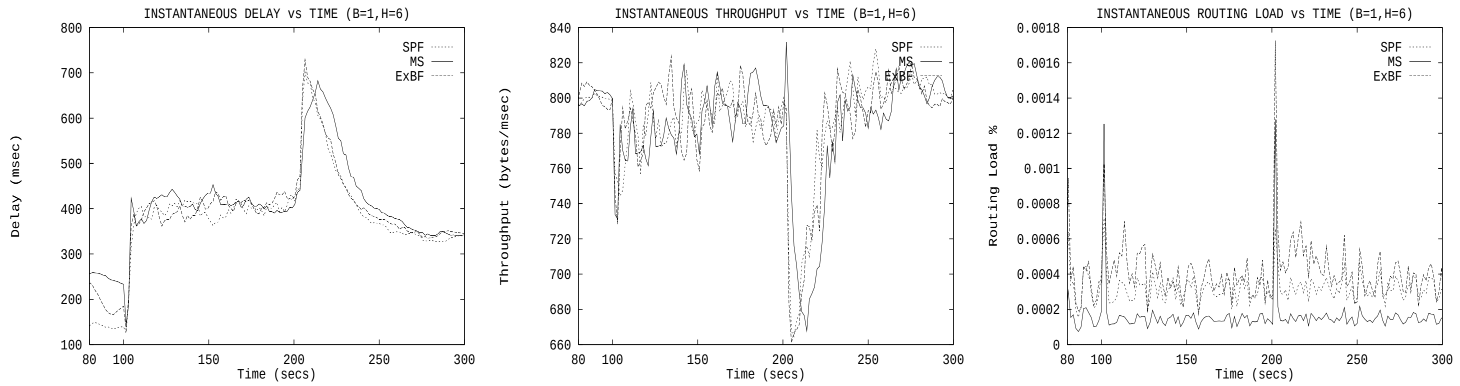

Figure 6: Responsiveness to a link failure and repair for $B = 1$ and $H = 6$.

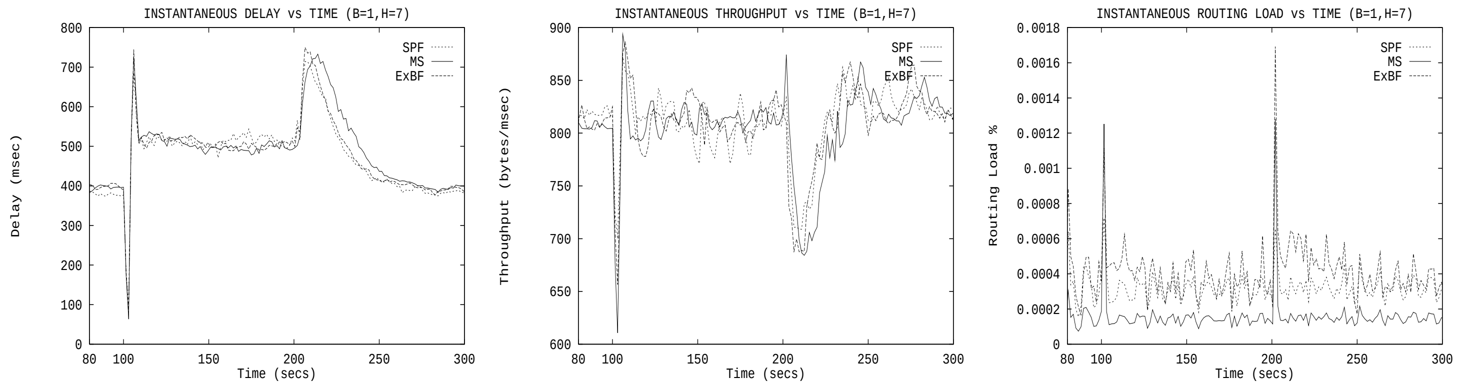

Figure 7: Responsiveness to a link failure and repair for $B = 1$ and $H = 7$.

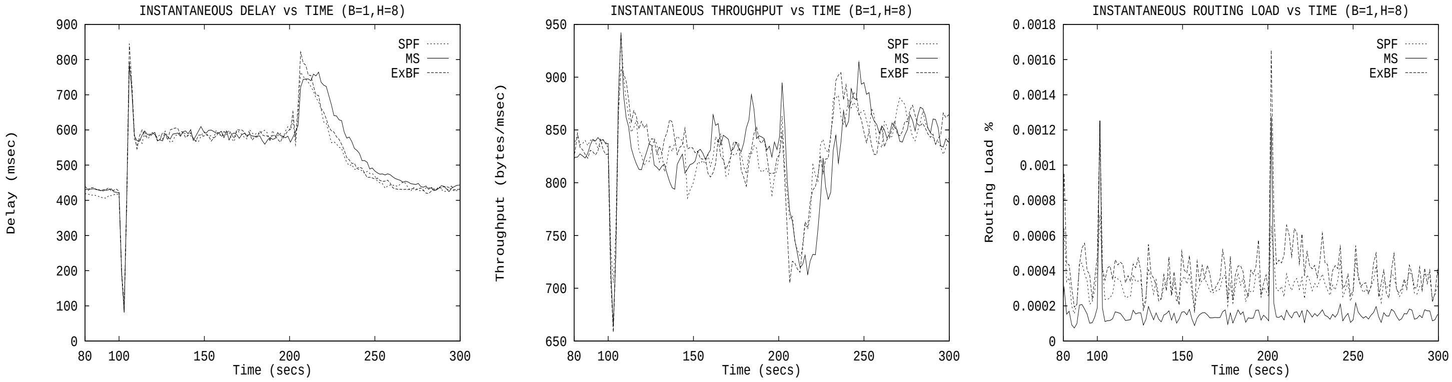

Figure 8: Responsiveness to a link failure and repair for $B = 1$ and $H = 8$.

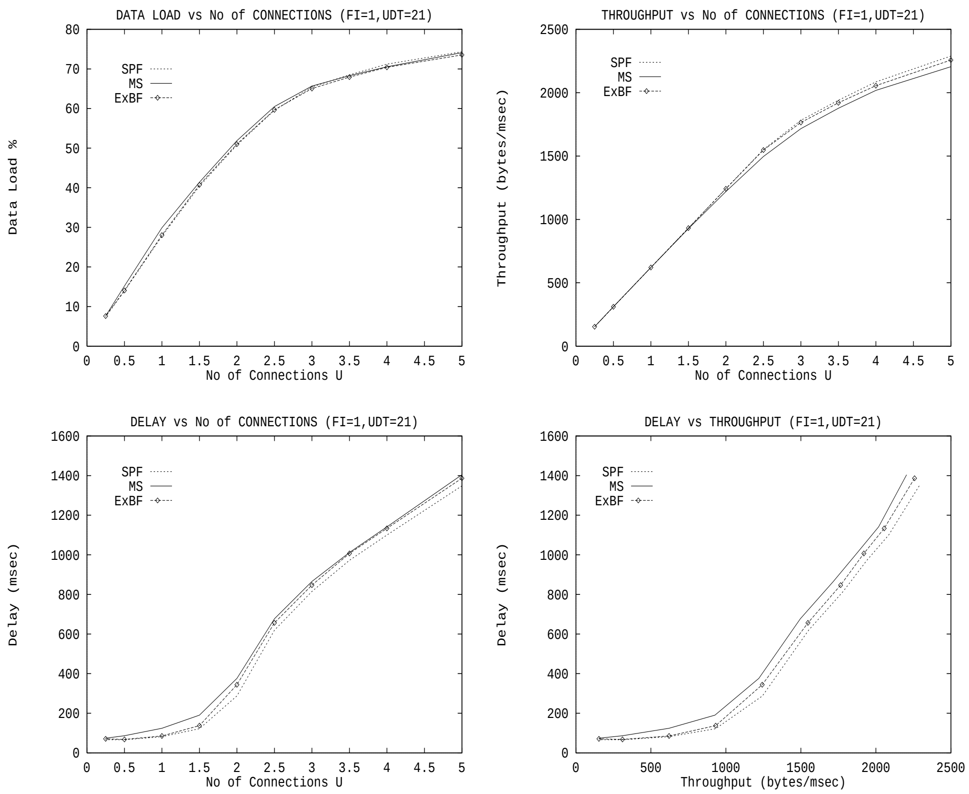

Figure 9: Poisson link failures and repairs, uniform workload, varying U.

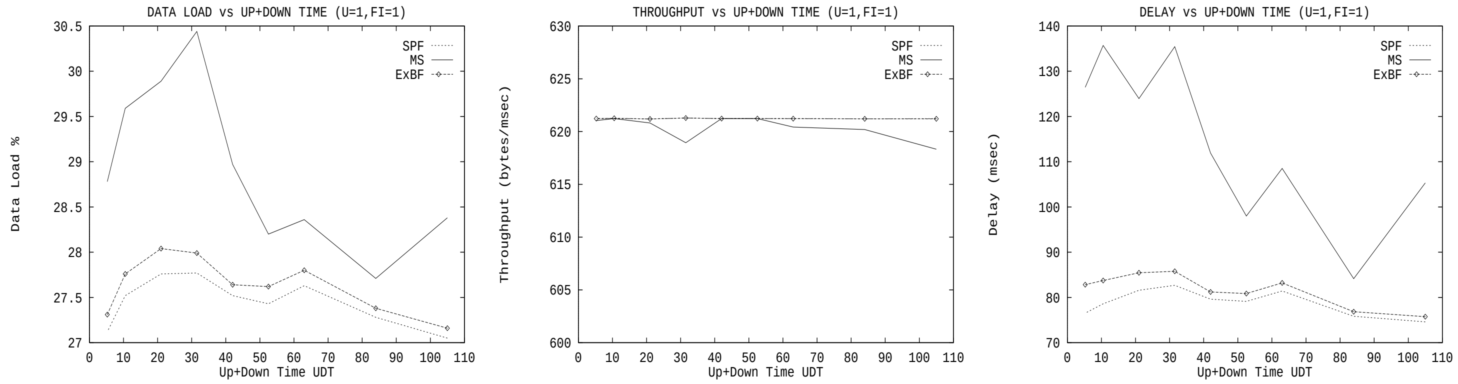

Figure 10: Poisson link failures and repairs, uniform workload, varying UDT.

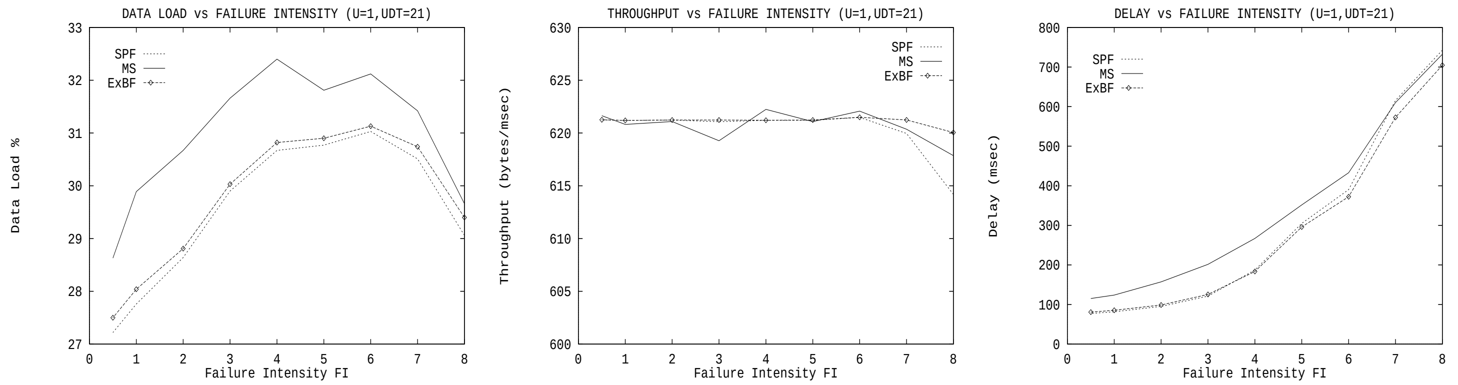

Figure 11: Poisson link failures and repairs, uniform workload, varying FI.

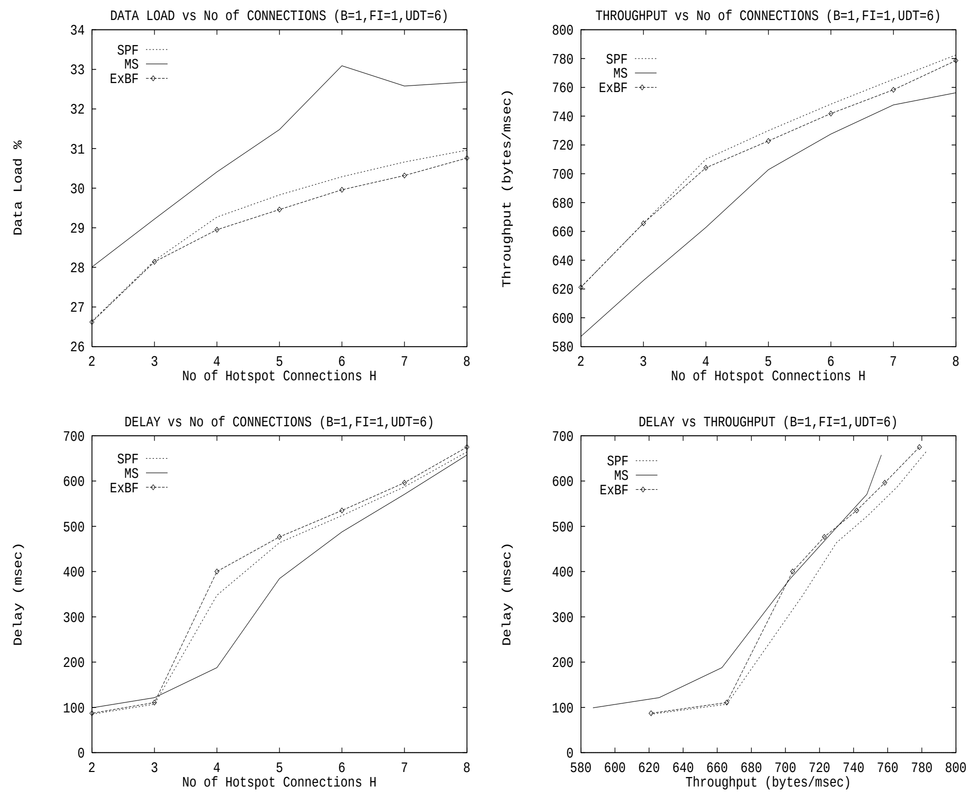

Figure 12: Poisson link failures and repairs, hotspot-with-background workload, varying H.

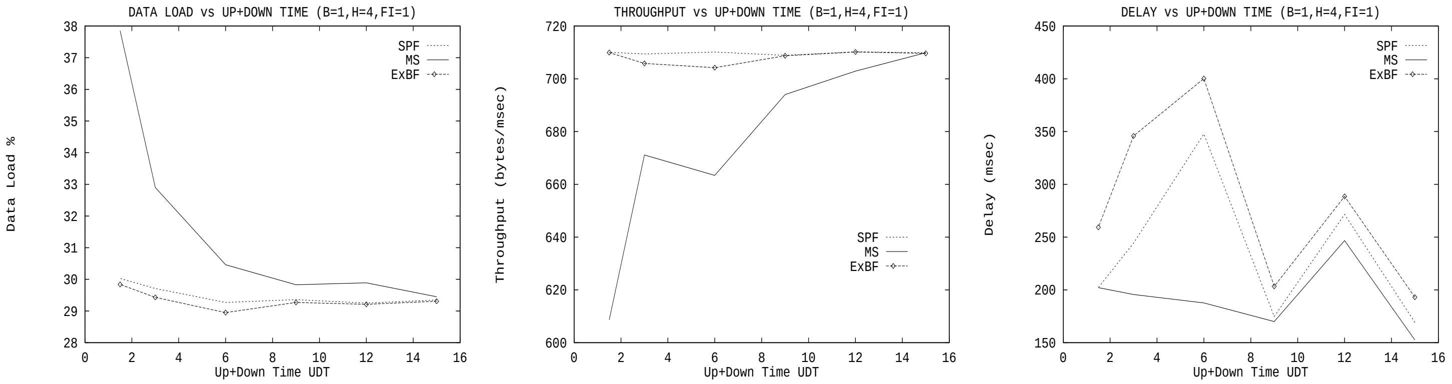

Figure 13: Poisson link failures and repairs, hotspot-with-background workload, varying UDT.

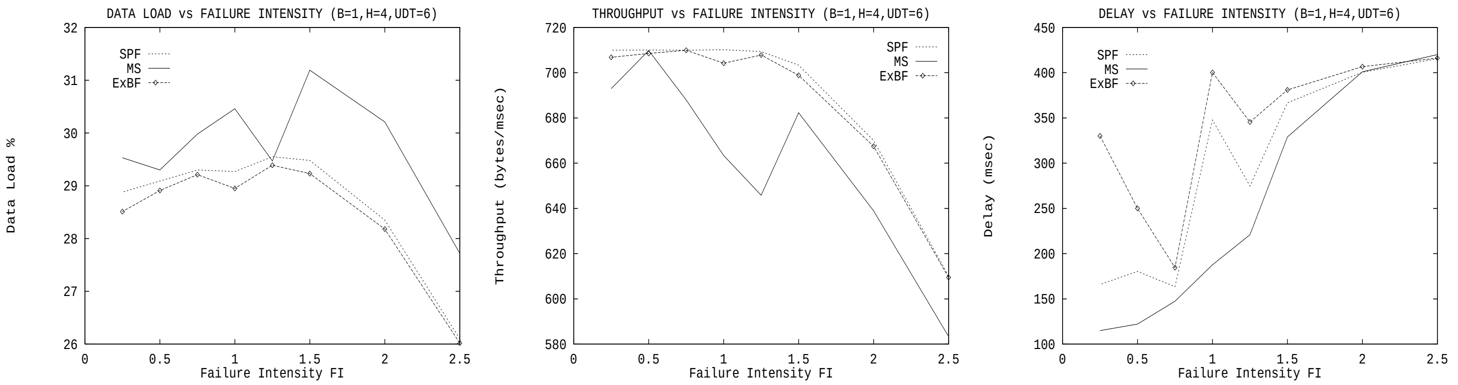

Figure 14: Poisson link failures and repairs, hotspot-with-background workload, varying FI.