SRI International

Technical Report • March 1993

A REPORT ON STOCHASTIC FAIRNESS QUEUEING (SFQ) EXPERIMENTS

Barbara A. Denny, Computer Scientist
Information and Telecommunications Sciences Center

Project 8600
ITAD-8600-TR-93-62

Prepared for:

NASA Ames Research Center
Moffett Field, California 94035

Attn: Dr. Henry Lum, Code RI, M/S: 244-7

and

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington, Virginia 22203-1714

Attn: Dr. Paul Mockapetris

Approved by:

Boyd C. Fair, Director
Information and Telecommunications Sciences Center

Michael S. Frankel, Vice President and Director
Information, Telecommunications, and Automation Division

333 Ravenswood Avenue • Menlo Park, CA 94025-3493 • (415) 326-6200 • FAX (415) 326-5512 • Telex 334486

CONTENTS

LIST OF FIGURES AND TABLES iii

1 EXECUTIVE SUMMARY 1

2 INTRODUCTION 2

3 SFQ CONFIGURATION 2

4 EXPERIMENTS 3

4.1 FAIR UTILIZATION 5

4.1.1 Objective and Procedure 5

4.1.2 Data Description 6

4.1.3 Results 6

4.2 STARVATION PREVENTION 11

4.2.1 Objective and Procedure 11

4.2.2 Data Description 12

4.2.3 Results 12

4.3 GRACEFUL DEGRADATION 16

4.3.1 Objective and Procedure 16

4.3.2 Data Description 17

4.3.3 Results 17

4.4 RESOURCE USAGE 27

4.4.1 Objective and Procedure 27

4.4.2 Data Description 28

4.4.3 Results 28

5 CONCLUSIONS 31

FIGURES

1 Experiment Topology 4
2 Fair Utilization Traffic Flow 5
3 Starvation Prevention Traffic Flow 11
4 Graceful Degradation Traffic Flow 16
5 Resource Usage Traffic Flow 27

TABLES

1 Fair Utilization Results 6
2 Starvation Prevention Results 12
3 Graceful Degradation Results From Time 0 to 30 Seconds 23
4 Graceful Degradation Results From Time 30 to 60 Seconds 23
5 Graceful Degradation Results From Time 60 to 90 Seconds 24
6 Graceful Degradation Results From Time 90 to 120 Seconds 24
7 Graceful Degradation Results From Time 120 to 245 Seconds 25
8 Graceful Degradation Results From Time 245 to 275 Seconds 25
9 Graceful Degradation Results From Time 275 to 305 Seconds 26
10 Graceful Degradation Results From Time 305 to 335 Seconds 26
11 Resource Usage 30

1 EXECUTIVE SUMMARY

SRI International (SRI) has developed an improved queueing algorithm, known as Stochastic Fairness Queueing (SFQ), for best-effort traffic (i.e., traffic that does not require any guaranteed service). SFQ is a probablistic variant of strict fair queueing where instead of a single queue being allocated per flow, a fixed number of queues are used and a hash function maps the IP source and destination to a particular queue. A seed to the hash function is also perturbed occasionally to help distribute the flows amongst different queues when more than one flow maps to the same queue during the lifetime of the flow. SFQ provides "fair" access by trying to ensure that each flow from source to destination host obtains equal access to the available bandwidth.

This report covers a series of experiments performed on DARTnet* evaluating the behavior and performance of SFQ against a FIFO queueing discipline. These experiments were designed to show SFQ's advantages and performance, and include tests demonstrating

- Fair utilization of available resources
- Starvation prevention
- Graceful degradation under overload conditions
- Resource usage.

The details of each experiment, including objective, procedures, data, and results, are presented in Subsections 3.1 through 3.4.

In general, the experiments do show that SFQ is better than FIFO queueing at allocating bandwidth equally among a set of flows. SFQ also prevents a stream from dominating the available bandwidth, which seems to be a tendency with FIFO queueing (i.e., if a flow demands more than its share of the available bandwidth, with FIFO queueing that stream receives a disproportionate amount when compared to flows demanding less than their share). Furthermore, SFQ seems to reward "nice" users of the network by providing a lower variance in delay and more throughput when their resource demand is less than their available share. Both SFQ and FIFO queueing seem to degrade fairly well as the network becomes saturated and to recover well as the network becomes less congested. Not unexpectedly, FIFO queueing is a little more efficient than SFQ—the delays are less and the throughput slightly higher because SFQ requires more processing. However, the performance difference between the two queueing disciplines is relatively small.

However, the experiments do point out some interesting behavior. FIFO queueing can behave better than SFQ with seed perturbation. We recommend further evaluation of the hash function and the seed perturbation technique. There are probably weaknesses in their current selection that cause this unexpected behavior. SFQ also seems to possess good scaling properties. To verify this, more experiments with a larger number of streams from more hosts need to be executed and examined, including the staggered introduction of streams. Staggering the streams may prove important, because graphs in the degradation experiment revealed some unexpected increases and decreases in throughput, which should be examined. This may again be due to the interaction of the hash function with the seed perturbation but it may also be related to some other unknown problem.

*DARTnet is a T1 testbed network sponsored by DARPA.

2 INTRODUCTION

This report summarizes a set of experiments comparing first-in, first-out (FIFO) queueing and Stochastic Fairness Queueing (SFQ). Historically, FIFO queueing has been the discipline in general use in routers. A single queue is used for all packets, which are serviced in a first-come, first-served manner. However, FIFO queueing often exhibits unfair behavior in the presence of multiple streams, especially during times of overload. A one-to-one mapping between queues and streams, with round-robin bitwise service of these queues, would eliminate this problem, but this algorithm, know as strict fair queueing, is expensive in terms of processing and space requirements. SFQ is a probabilistic variant of strict fair queueing. Instead of requiring that each flow have its own queue, SFQ has a fixed number of queues and uses a hashing function to map the IP source and destination address into one of the queues. Packets are entered into their assigned queues in a FIFO manner and are removed in a round-robin fashion between all nonempty queues (in the current implementation, the round-robin service is on a packet-by-packet basis). A seed to the hash function is occasionally perturbed, to allow a redistribution of the address pair mapping. This mapping redistribution is done to ensure that flows are not consistently mapped into the same queue, so that a well-behaved source is not penalized by an ill-behaved source if the flows happen to map to the same queue at some point in time. For a more complete description of SFQ, see the referenced paper by Paul E. McKenney.*

3 SFQ CONFIGURATION

SFQ has many parameters that can be tuned to improve its performance. These parameters include

- Individual queue depth
- Total number of queues
- Hash function
- Seed perturbation technique.

The setting of these parameters for these experiments was somewhat arbitrary, because the experiment design was on a small enough scale that these factors did not affect the outcome significantly in most cases. More research needs to be done to determine the optimum choice for larger scenarios. The parameters chosen for this set of experiments are described next.

The choice of the hash function is crucial to good behavior, because the hash function is responsible for distributing the packets among the queues. If the hash function is poor, many different flows map into the same queue and the behavior approaches that achieved with FIFO queueing. The current implementation provides five different hash functions using XOR or rotate operations. For the set of addresses used in the experiments, each hash function displayed similar behavior. We therefore chose an XOR type of hash function.

---

*McKenney, P.E. 1991. "Stochastic Fairness Queueing," in *Internetworking: Research and Experience*, Vol. 2, pp. 113-131.

The individual queue depth was set to 100. This parameter value was chosen to match the queue depth provided by the DARTnet kernel with its FIFO queueing discipline. Queue depth is a compile time constant and can be changed easily.

The total number of queues used is also an important factor in the behavior of SFQ. As mentioned earlier, too few queues result in behavior similar to FIFO; too many queues, and the behavior, and space overhead, resemble strict fair queueing if a suitable hash function is used. To ensure that the condition arises where more than one flow maps to the same queue during the lifetime of an experiment (i.e. a collision occurred during the hash computation), the number of queues was limited to 9. This number is controlled by a compile-time constant and is easy to change; however, in our implementation, the number of queues has to be equal to 2^m+1, where m is an integer. We are using a software implementation of the modulo operator, which works only for these values, because the current SPARC* architecture lacks the hardware support necessary for an efficient implementation. The modulo operator is used to reduce the result of the hashing function to the desired range.

As previously mentioned, collisions into the same queue will occur for the host addresses used in the experiments below. Thus, the experiments will show the results with and without seed perturbation. To preserve packet ordering, the current implementation increments the seed when all the queues are empty. In a congested router, it is anticipated that this will be infrequent; therefore, different techniques for seed perturbation need to be developed and tested.

4 EXPERIMENTS

The experiments were executed on DARTnet, a DARPA research testbed network. DARTnet is a cross-country T1 network† that connects research sites via T1 tail circuits. The routers and most of the hosts are SPARC 1+, with the exception of two hosts, MM6 and Malarky, which are SPARC 2s. Figure 1 illustrates the portion of the network used in these experiments.

The objectives of the experiments were to show the benefits and performance of SFQ. These experiments therefore demonstrate

- Fair utilization of available resources
- Starvation prevention
- Graceful degradation under overload conditions
- Resource usage.

Each experiment was run at least twice to verify its repeatability; however, in this report we will include only a single case. Each run of an experiment consisted of executing the experiment with a special kernel on the routers that support SFQ, with and without seed perturbation in most cases, and with the standard DARTnet kernel, which provided the FIFO queueing mechanism. The version number of the DARTnet kernel was 6. All traffic streams originated from machines using

*All product names mentioned in this report are the trademarks of their respective holders.

†Due to hardware constraints, the network operates at 1.334 Mb/s instead of 1.536 Mb/s.

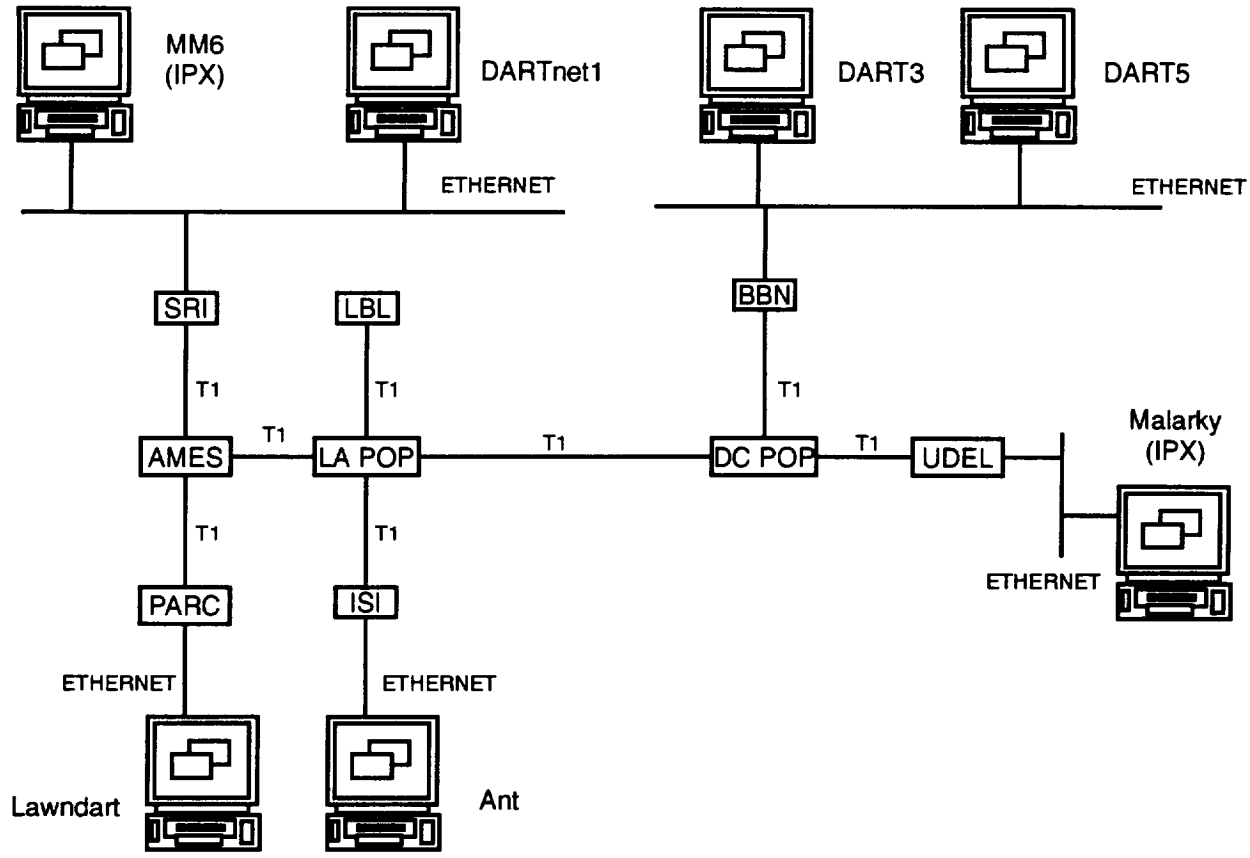


Figure 1. Experiment Topology

version 6 of the DARTnet kernel. Each stream in the experiment ran for 245 seconds, but the total time of the experiment was 335 seconds, due to the staggering of the start times for each stream. All experiments were designed to overload the link so that the network's behavior in times of congestion could be observed.

The DARTnet traffic generator (TG) was used to create the traffic streams. TG is an SRI-developed tool for creating high-quality and repeatable experiments on packet-switched networks. It executes as a source and sink program that enables experimenters to generate one-way traffic streams and gather statistical data about the transmission and reception of each stream. The TG is driven by a control language (script) that specifies different operating modes, protocols, addressing functions, traffic parameters, and execution times. At present, packet lengths and packet offer rates can be specified according to the following distributions: constant, uniform, exponential, and 2-state Markov. The delay and throughput for each of the experiments, therefore, is measured from user process to user process.

4.1 FAIR UTILIZATION

4.1.1 Objective and Procedure

This experiment was designed to show that streams receive equal portions of the available bandwidth. In this experiment, four equal UDP streams are created; these streams occupy 60 percent of the link capacity of a T1 line. The traffic distribution for each stream is identical: each stream's offer rate is exponentially distributed with a interarrival mean of 0.007692 seconds (~130 packets per second) and a constant size of 782 bytes (including UDP and IP headers). The source, destination, and direction for each stream is as follows (see Figure 2):

- Dartnet1 to Malarky
- Lawndart to Dart5
- LBL router to Dart3
- MM6 to Ant.

At the start of the experiment, the stream from DARTnet1 to Malarky and the stream from MM6 to Ant collide with the initial seed value.

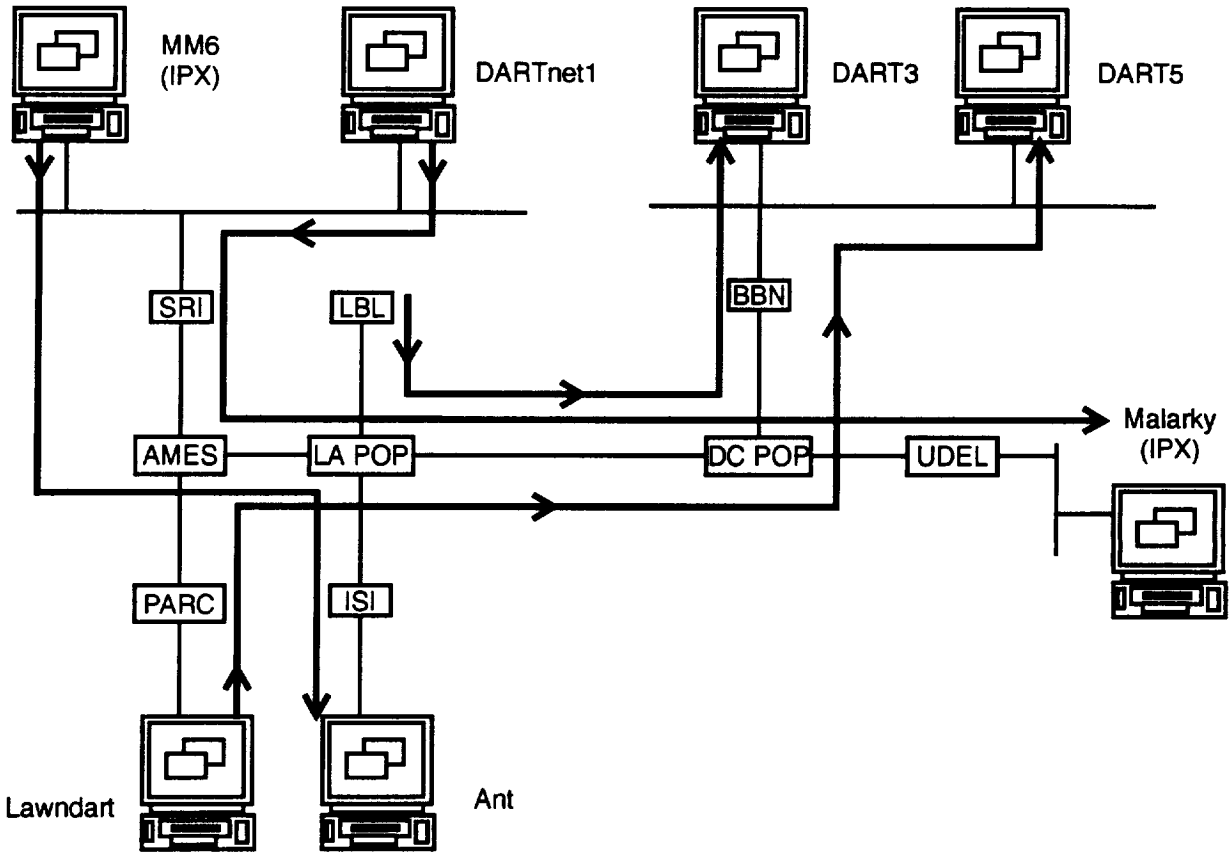


Figure 2. Fair Utilization Traffic Flow

4.1.2 Data Description

The graphs below show the offer rate, the SFQ throughput, and the FIFO throughput for each stream. The experiments were executed with and without seed perturbation: SFQ Experiment 29 ran without seed perturbation, and SFQ Experiment 49 ran seed perturbation.

Table 1 summarizes the results of the experiments, including the average offer rate, average throughput, average delay, and delay variance for each stream.

4.1.3 Results

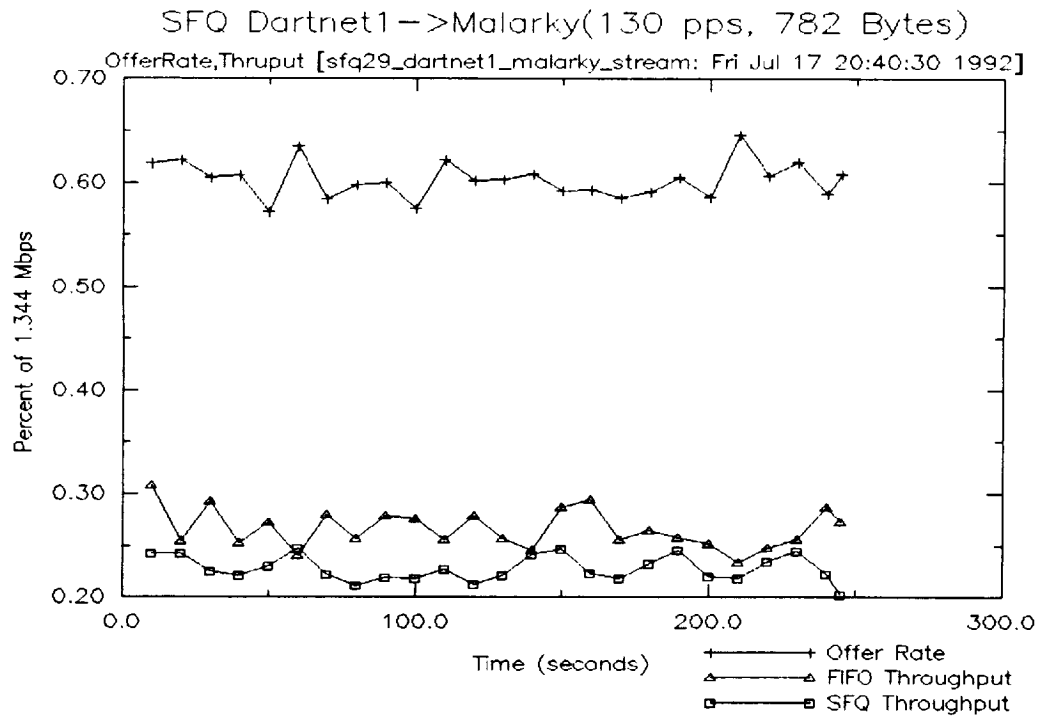
The results of this experiment show that SFQ does provide better equal access to the available resources, while FIFO queueing allows the LBL to Dart3 to dominate the resource. Furthermore, seed perturbation does seems to improve SFQ performance when collisions occur. It is important to note that the throughput bottleneck occurs on the link from AMES to LA and on the link from LA POP to the DC POP. At each of these points in the network, three streams are trying to share the same link. Under perfect utilization, each stream would receive 32.3% of link capacity.* In experiment 49 with SFQ, the utilization for each stream was 31.8%, 31.6%, 31.7%, and 31.8%.

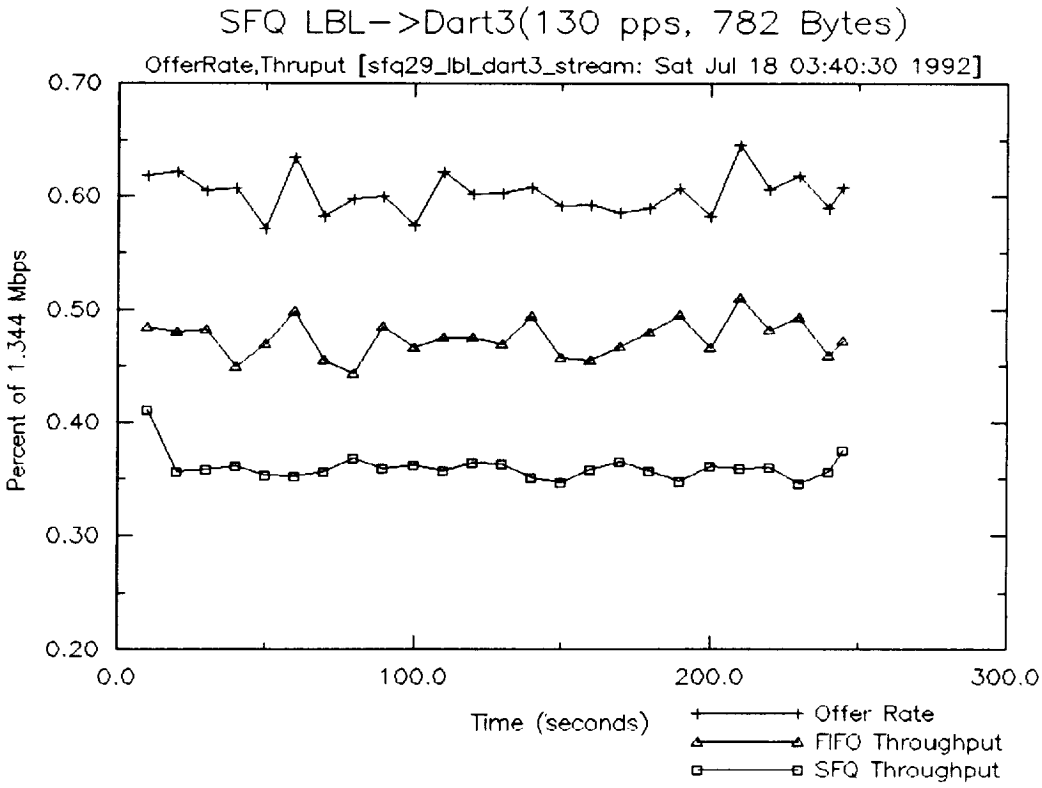
Table 1. Fair Utilization Results

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo29	60.26	26.64	1.4704	0.0057
	sfq29	60.25	22.85	1.5186	0.0074
	sfq49	60.25	31.84	2.8637	0.1215
LBL to Dart3	fifo29	60.25	47.49	0.5367	0.0007
	sfq29	60.26	36.00	1.3415	0.0095
	sfq49	60.26	31.62	1.5221	0.0095
Lawndart to Dart5	fifo29	60.25	23.08	1.0198	0.0029
	sfq29	60.25	36.15	2.2988	0.0384
	sfq49	60.25	31.72	1.9565	0.0849
MM6 to Ant	fifo29	60.26	34.73	0.9608	0.0024
	sfq29	60.25	24.71	1.4666	0.0071
	sfq49	60.25	31.84	2.4286	0.0398

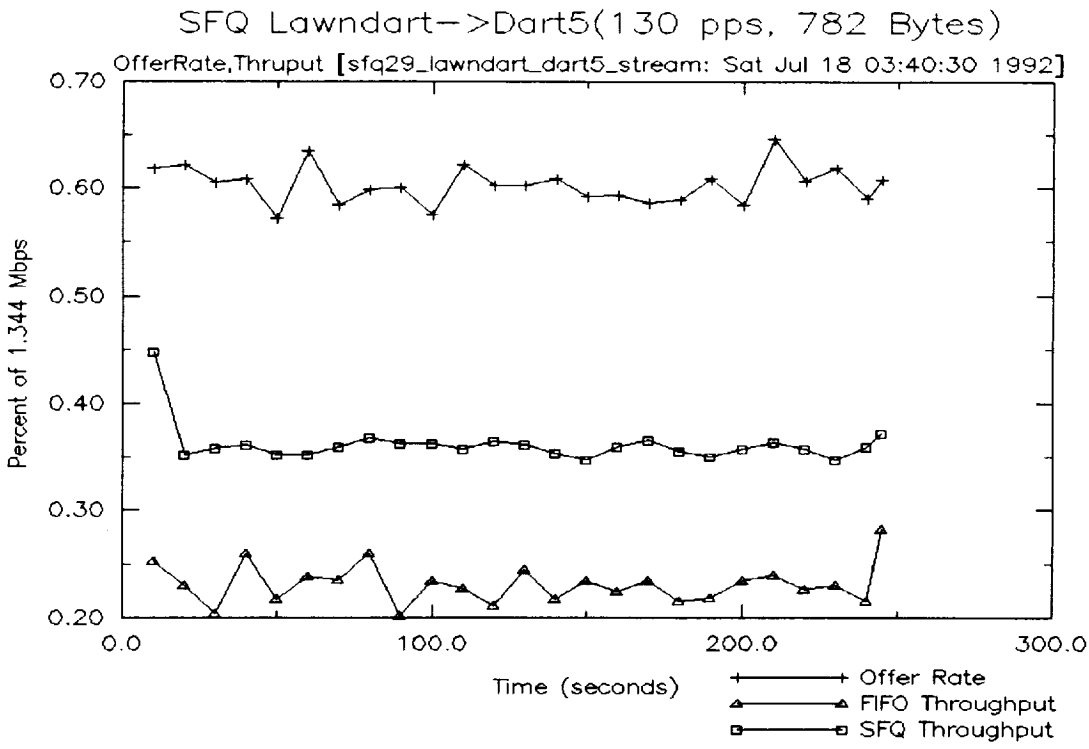
*In previous baseline measurements of the DARTnet FIFO kernel, the throughput for a packet size of 750 bytes was 96.9% of the link capacity.

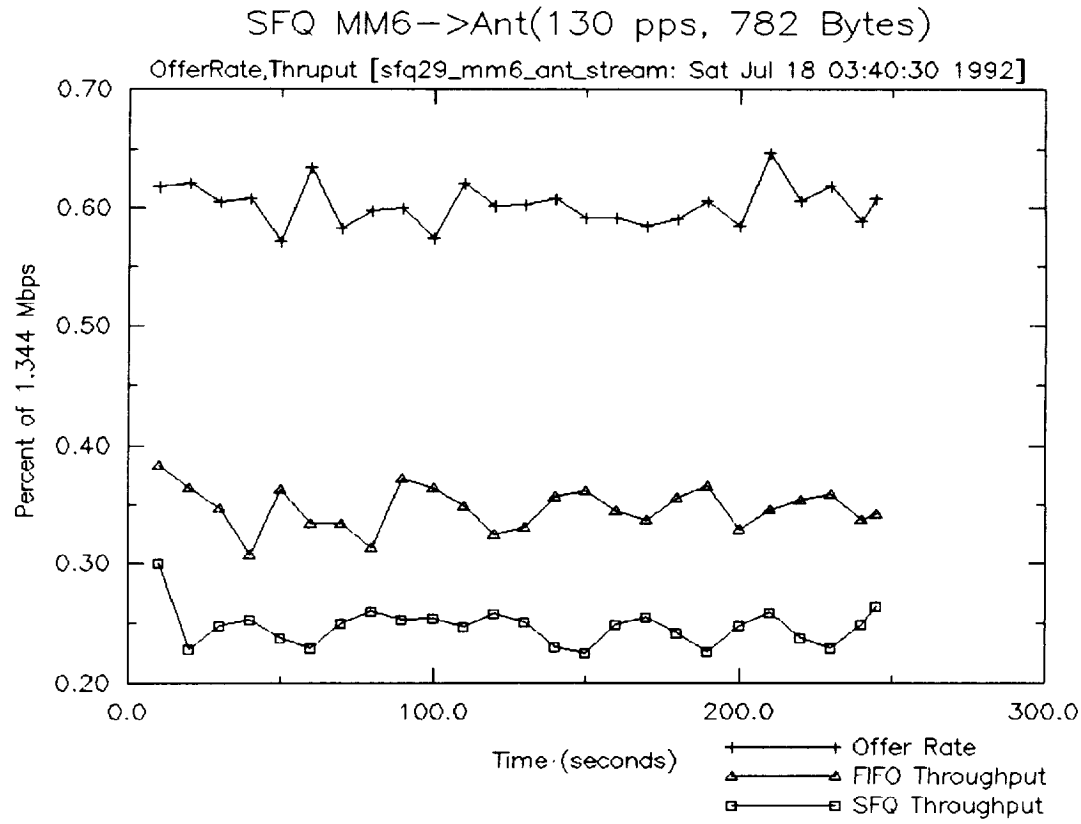
SFQ EXPERIMENT 29



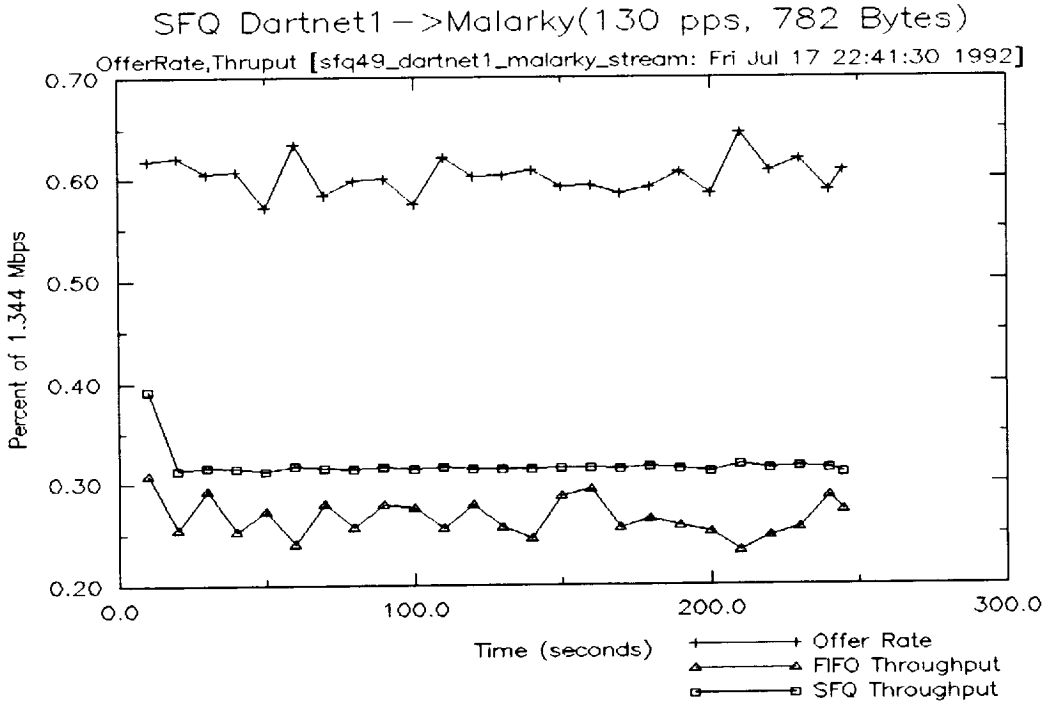


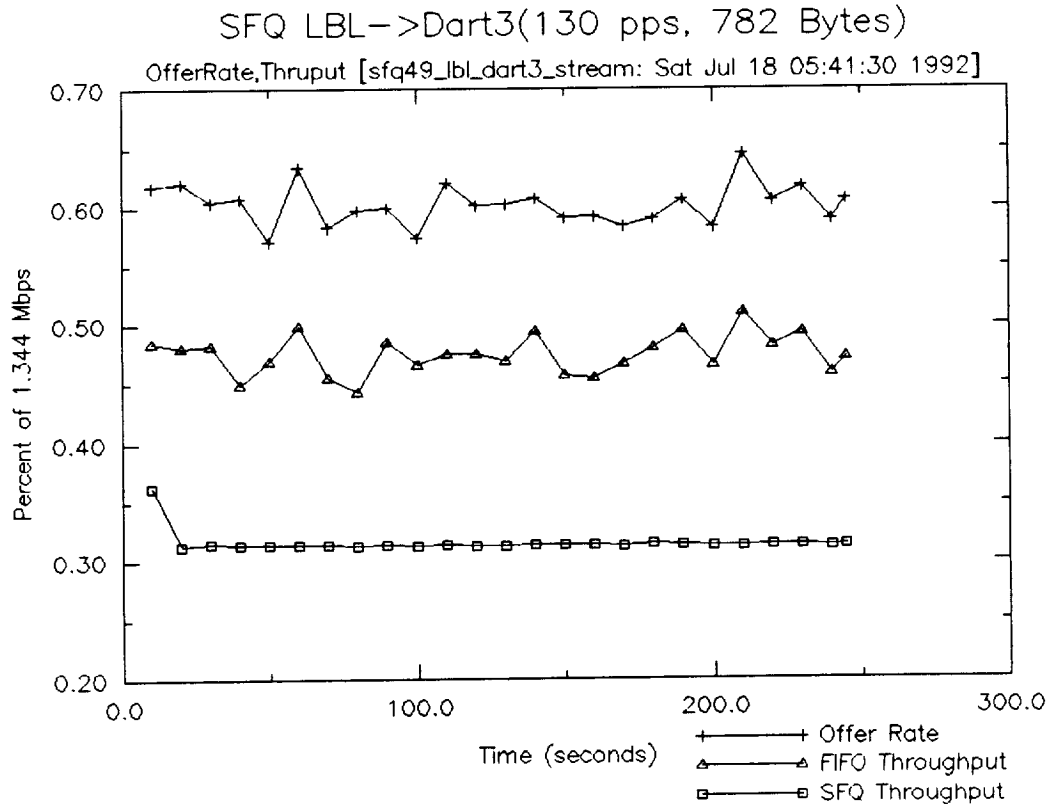
SFQ EXPERIMENT 29



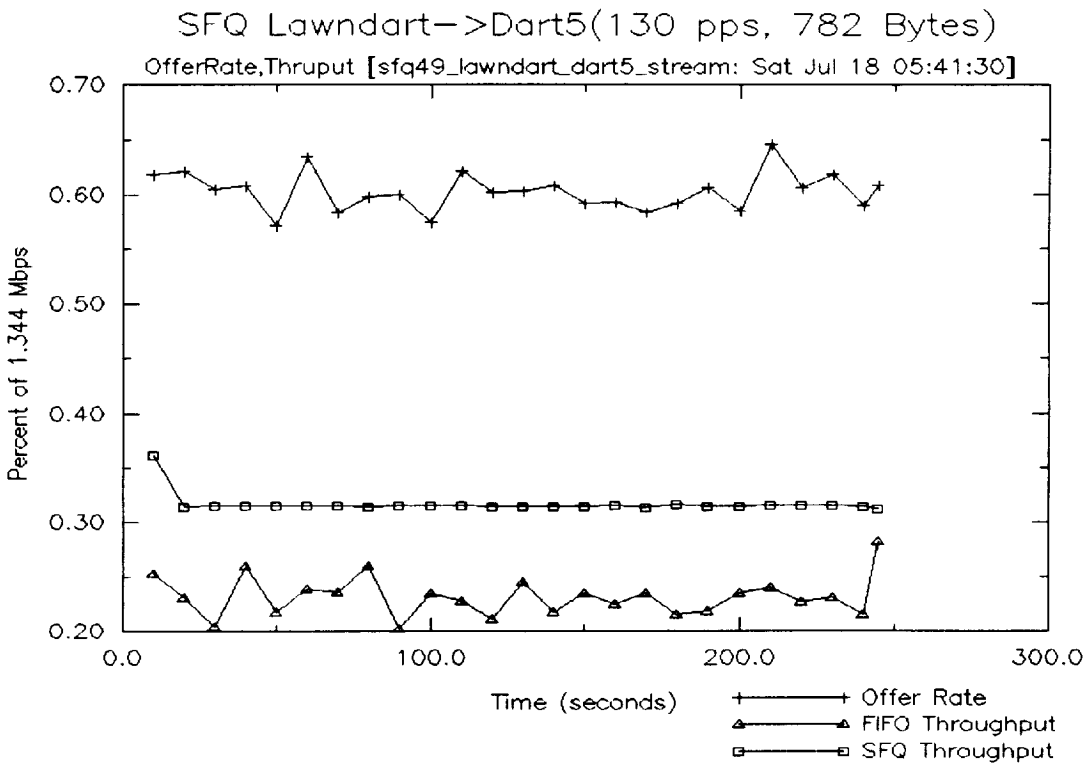


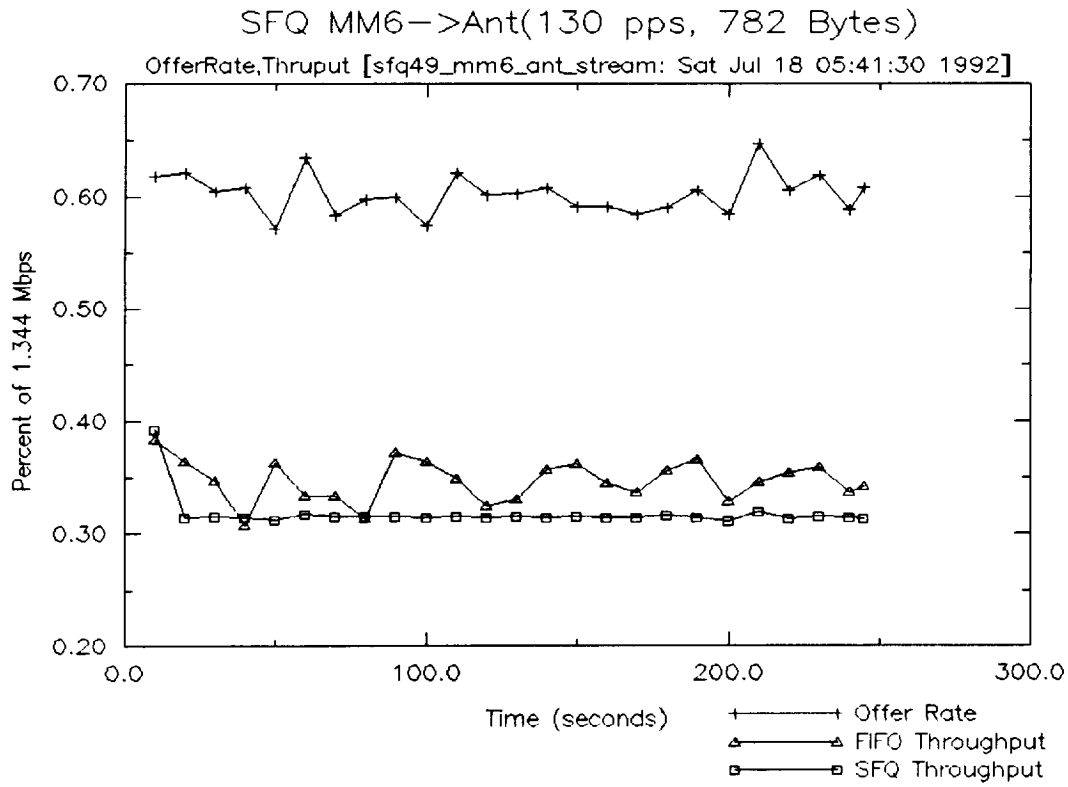
SFQ EXPERIMENT 49





SFQ EXPERIMENT 49





4.2 STARVATION PREVENTION

4.2.1 Objective and Procedure

This experiment was designed to show SFQ's ability to prevent starvation. In the experiment, three unequal UDP streams are created. The source, destination, and direction of each stream are as follows (see Figure 3):

- Lawndart to Ant
- MM6 to Dart5
- Dartnet1 to Malarky.

The stream from Lawndart to Ant occupies 95% of the link capacity of a T1 line. This stream is exponentially distributed, with an interarrival mean of 0. 004902 seconds (~204 packets per second) and a constant packet size of 782 bytes (including UDP and IP headers). The streams from MM6 to Dart5 and Dartnet1 to Malarky each occupy 20% of the link capacity of a T1 line. Each stream is exponentially distributed, with an interarrival mean of 0.023256 seconds (~43 packets per second) and a constant packet size of 782 bytes. At the start of the experiment, the streams from Lawndart to Ant and from MM6 to Dart5 collide with the initial seed value.

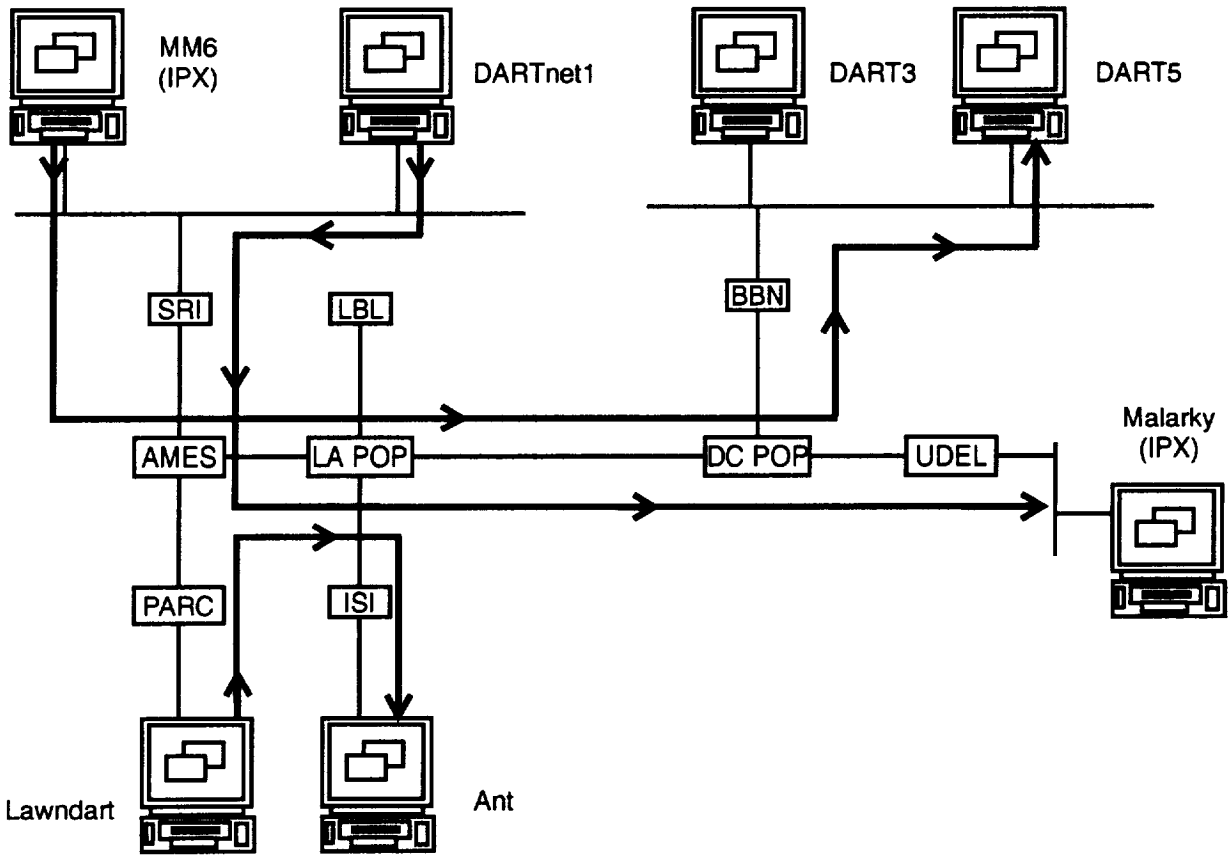


Figure 3. Starvation Prevention Traffic Flow

4.2.2 Data Description

Each graph below shows the offer rate, the SFQ throughput, and the FIFO throughput for each stream. The experiments were executed with and without seed perturbation: SFQ Experiment 28 ran without seed perturbation and SFQ Experiment 47 ran with seed perturbation.

Table 2 summarizes the results of the experiments, including average offer rate, average throughput, average delay, and delay variance for each stream.

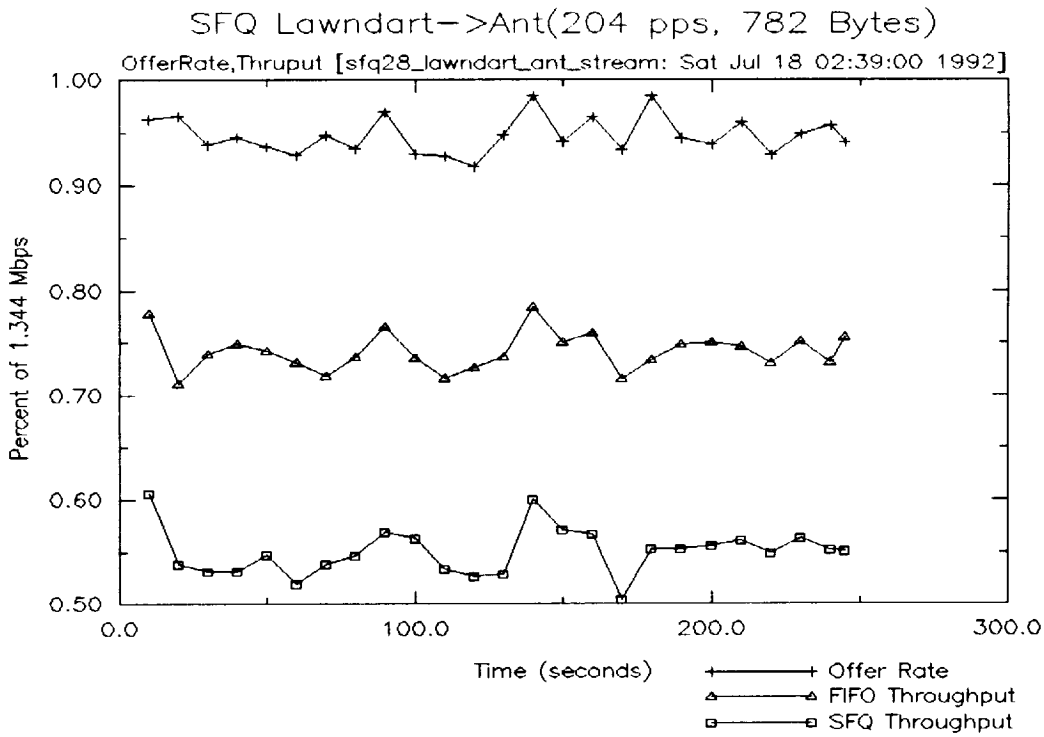
4.2.3 Results

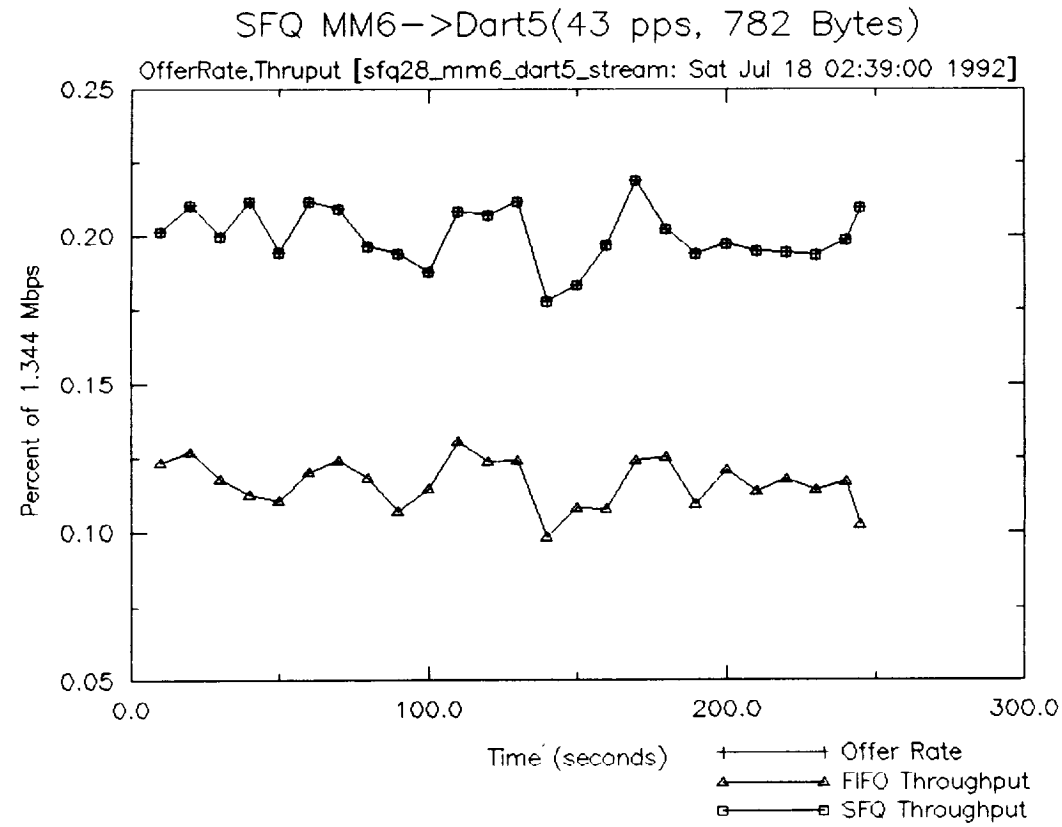
This experiment does show that SFQ helps prevent starvation by ensuring that those streams that demand less than their fair share of the available bandwidth receive their entire quota; while with FIFO queueing, they receive a disproportionate amount. In particular, with FIFO queueing, the stream that demanded ~95% of capacity received 74%, approximately 78% of its request. However, the two streams that wanted only 20% of the bandwidth received ~11%, or only 58% of their requested utilization. The variance in delay for the smaller streams was also less with SFQ than with FIFO: SFQ thus rewards "nice" users of the network.

Table 2. Starvation Prevention Results

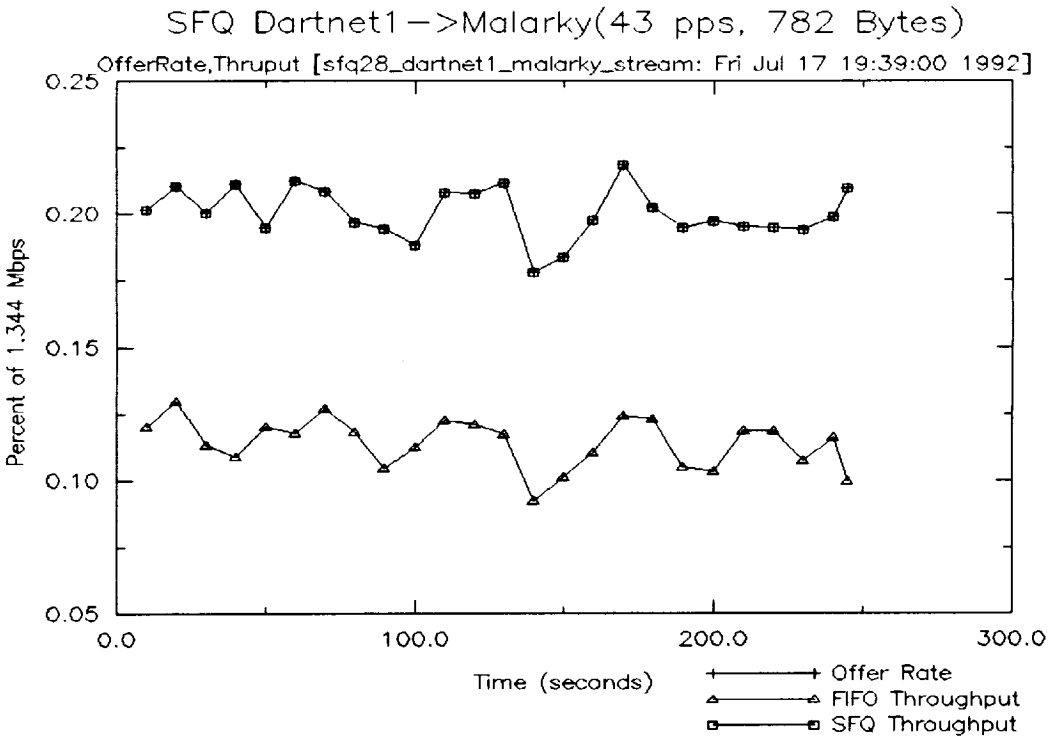
STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Lawndart to Ant	fifo27	94.72	74.12	0.5727	0.0057
	sfq28	94.72	55.05	1.0888	0.0272
	sfq47	94.72	55.07	1.0897	0.0271
MM6 to Dart5	fifo27	20.01	11.68	0.5383	0.0007
	sfq28	20.01	20.01	0.0853	0.0004
	sfq47	20.01	20.01	0.0770	0.0003
Dartnet1 to Malarky	fifo27	20.01	11.44	0.5377	0.0008
	sfq28	20.01	20.01	0.0796	0.0003
	sfq47	20.01	20.01	0.0796	0.0004

SFQ EXPERIMENT 28

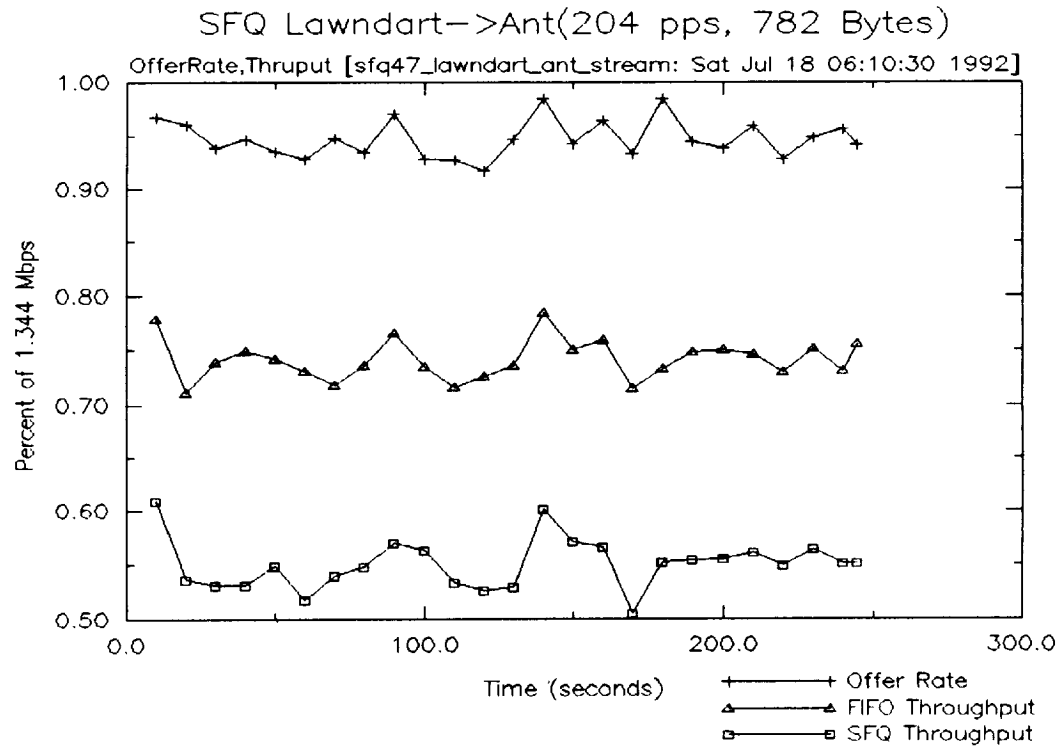




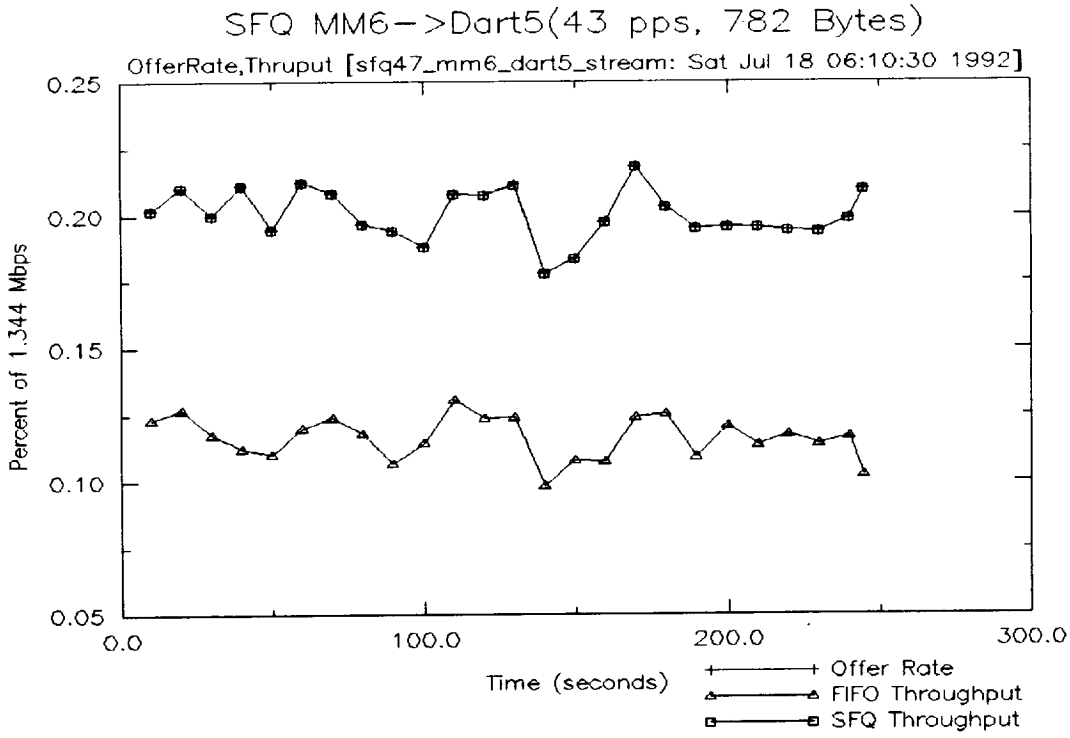
SFQ EXPERIMENT 28

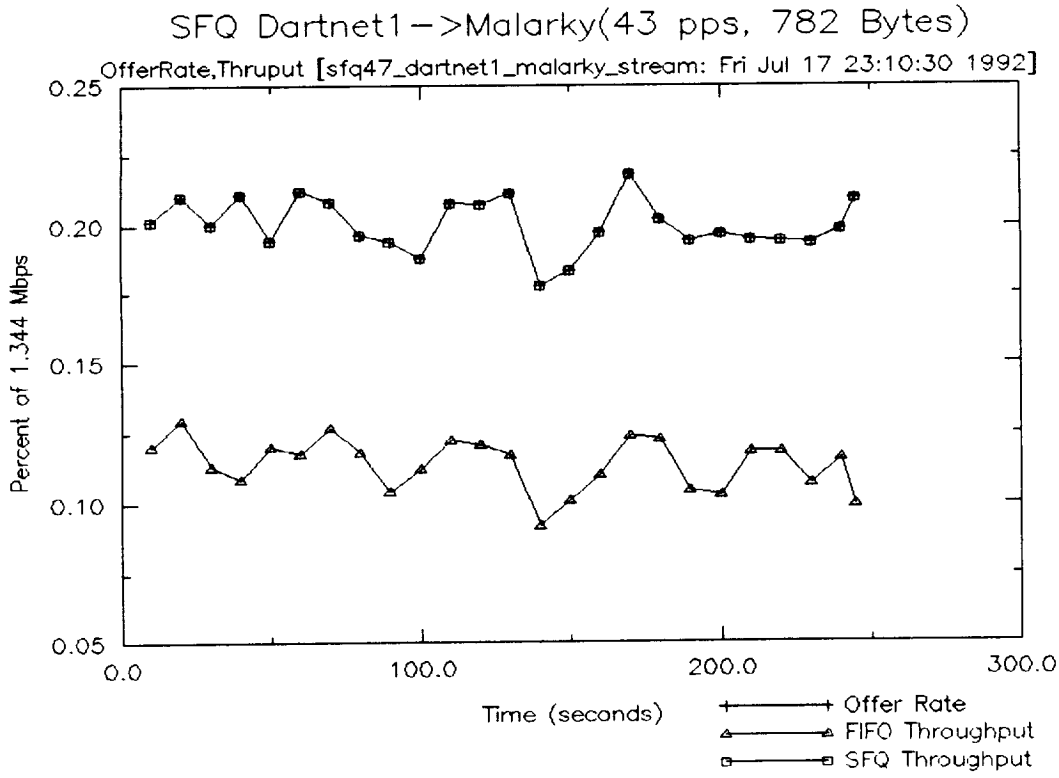


SFQ EXPERIMENT 47



SFQ EXPERIMENT 47





4.3 GRACEFUL DEGRADATION

4.3.1 Objective and Procedure

This experiment was designed to show that SFQ degrades gracefully in periods of overload; each stream receives its fair share of the available bandwidth as more streams are added. In the experiment, four equal UDP streams were created; the streams occupy 60 percent of the link capacity of a T1 line. The traffic distribution for each stream is identical: each stream's offer rate is exponentially distributed, with a interarrival mean of 0.007692 seconds (~130 packets per second) and a constant size of 782 bytes (including UDP and IP headers). A stream is added every 30 seconds after the previous one. The source, destination, and direction of each stream, in the order in which the streams were created, is as follows (see Figure 4):

- Dartnet1 to Malarky
- LBL router to Dart3
- Lawndart to Dart5
- MM6 to Ant.

At the start of the experiment, the stream from DARTnet1 to Malarky and the stream from MM6 to Ant collide with the initial seed value.

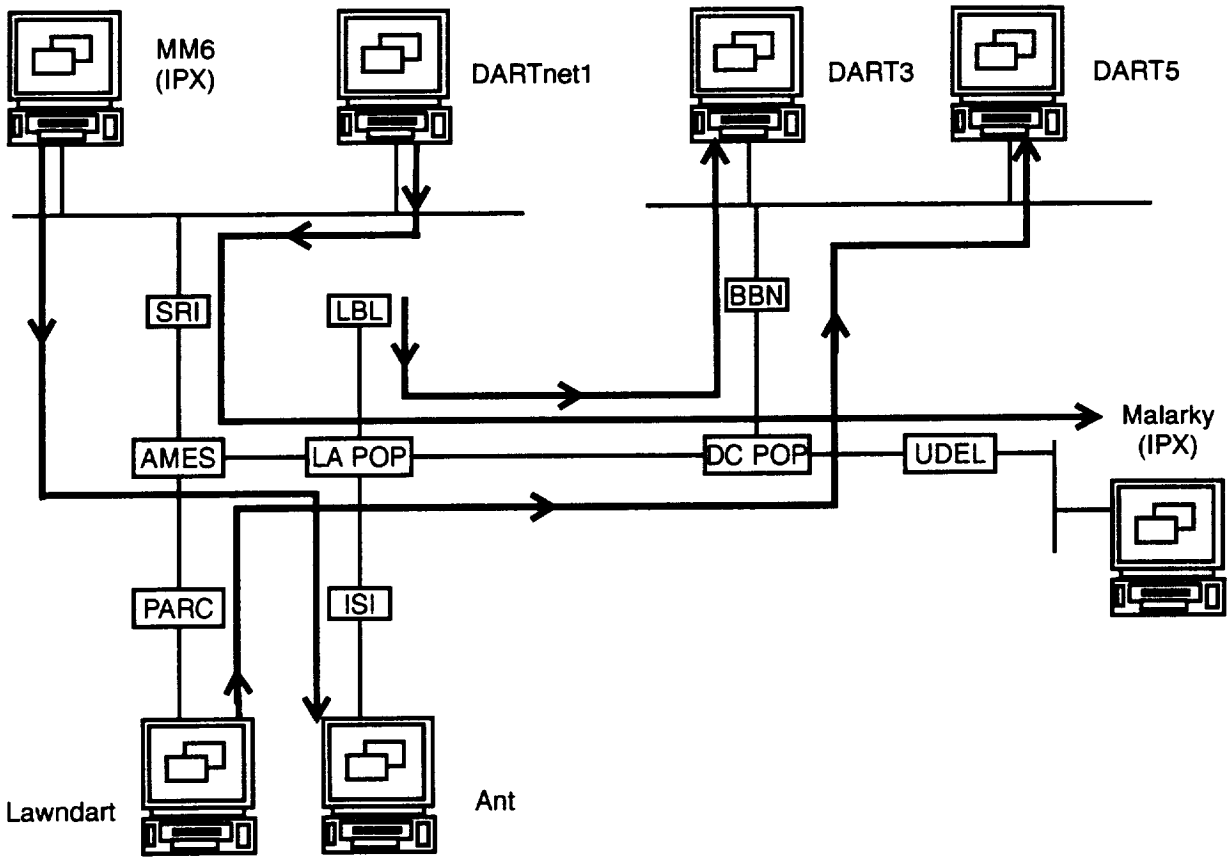


Figure 4. Graceful Degradation Traffic Flow

4.3.2 Data Description

The graphs below show the offer rate, the SFQ throughput, and the FIFO throughput for each stream. The experiments were executed with and without seed perturbation: SFQ Experiment 31 ran without seed perturbation and SFQ 41 ran with seed perturbation.

Tables 3 through 10 summarize the progression of the experiment, by including the average offer rate, average throughput, average delay, and delay variance for each significant time period for every stream. The time periods displayed are related to the times when streams are added and deleted, including a time period when all streams are executing simultaneously. To summarize,

- Table 3 shows the first 30 seconds, when only the Dartnet1 to Malarky stream exists.
- Table 4 shows the second 30-second interval, when the stream from LBL to Dart3 has been added.
- Table 5 shows the third 30-second interval, when the stream from Lawndart to Dart5 has been added.
- Table 6 shows the fourth 30-second interval, when the stream from MM6 to Ant has been added.
- Table 7 shows the next 125 seconds, when all the streams are running.
- Table 8 shows the next 30 seconds, when the stream from DARTnet to Malarky has dropped out and only the streams from LBL to Dart3, Lawndart to Dart5, and MM6 to Ant are running.
- Table 9 shows the next 30 seconds, when only the streams Lawndart to Dart 5 and from MM6 to Ant are present.
- Table 10 shows the final 30 seconds, when the stream from MM6 to Ant is the only one left.

As in the fair utilization experiment, the throughput bottleneck occurs on the links from AMES to LA and from LA POP to DC POP. The bottleneck on the AMES to LA link affects the streams from Dartnet1 to Malarky, Lawndart to Dart5, and MM6 to Ant; while the bottleneck on the link from LA POP to DC POP affects the streams from Dartnet1 to Malarky, LBL to Dart3, and Lawndart to Dart5. At each of these locations, the streams are trying to share the same link.

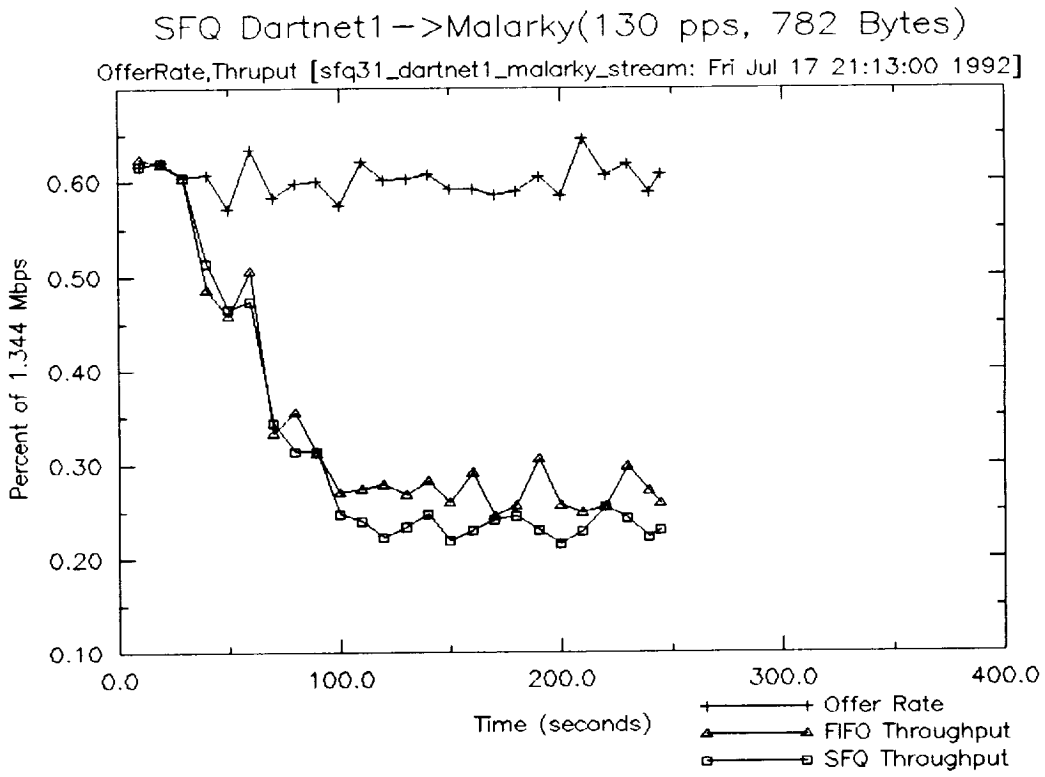
4.3.3 Results

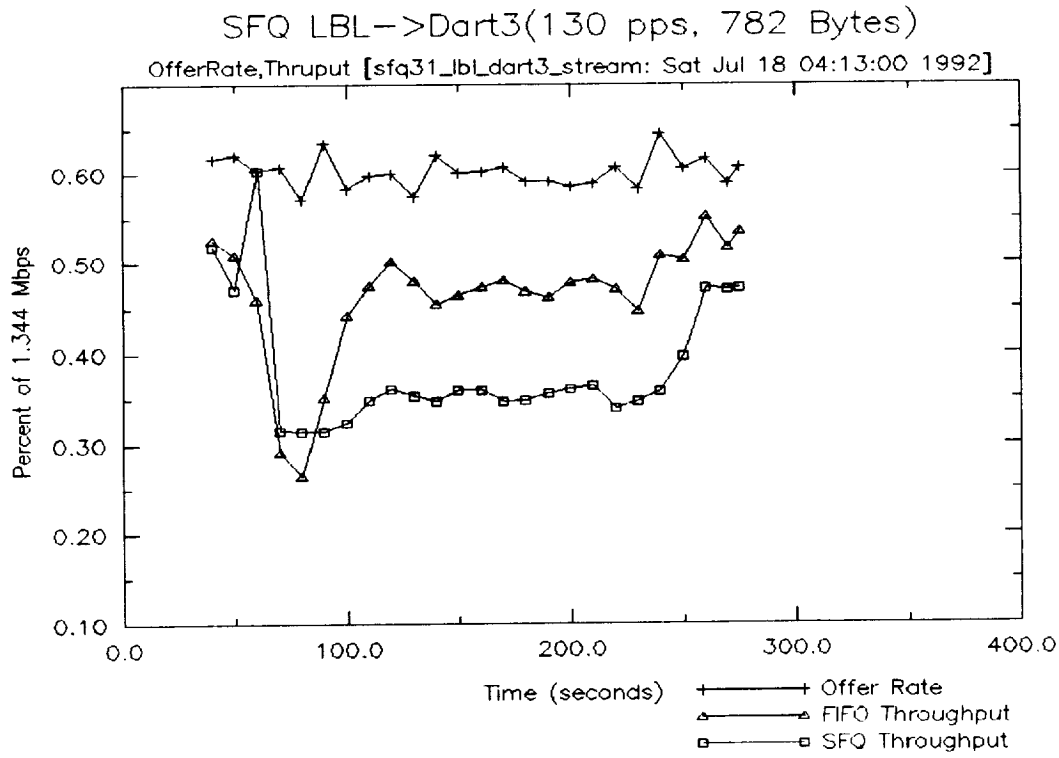
The results of this experiment should be the following:

1. As the first three streams are added, the amount of throughput on each stream should decrease proportionately to the number of streams running, since they all use the LA POP to DC POP link.
2. As the fourth stream (MM6 to Ant) is added, there should be little effect on the first three streams, since the bottleneck has not substantially changed.
3. As Dartnet1 to Malarky drops out, the throughput on all the remaining streams should reach approximately 50%.
4. As LBL to Dart3 drops out, there should be no real effect on the two remaining streams, because the AMES-LA bottleneck now comes into play.
5. As the Lawndart to Dart5 stream drops out, throughput on the stream from MM6 to Ant should match the offer rate.

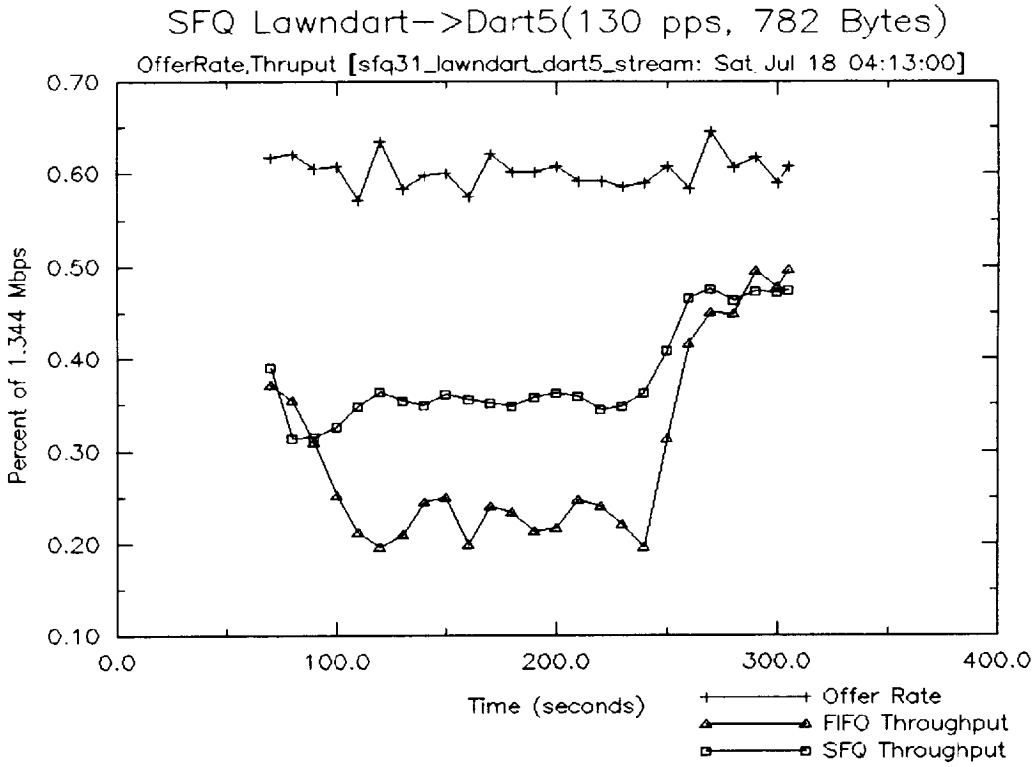
In general, both SFQ and FIFO exhibited the behavior described above. However, SFQ seems to be "more" fair overall than FIFO queueing. The range of throughputs achieved under SFQ more closely matched the ideal; under FIFO queueing there may be a tendency for a stream to "dominate" the available resources under heavy utilization (see Table 7). Not unexpectedly, "strict" fair queueing (SFQ with no collisions) behaved better than SFQ with seed perturbation (see Table 5). However, in one instance, FIFO queueing behaved better than SFQ with seed perturbation (see Table 5). This underlines the need to find optimal hashing functions and good seed perturbation techniques.

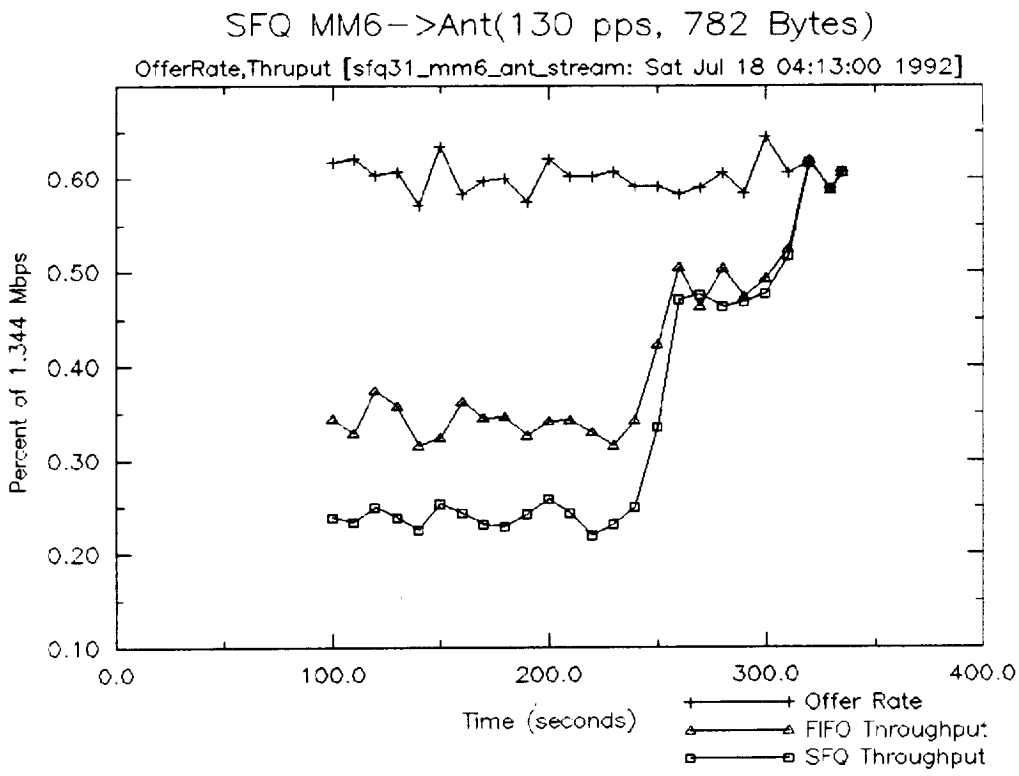
SFQ EXPERIMENT 31



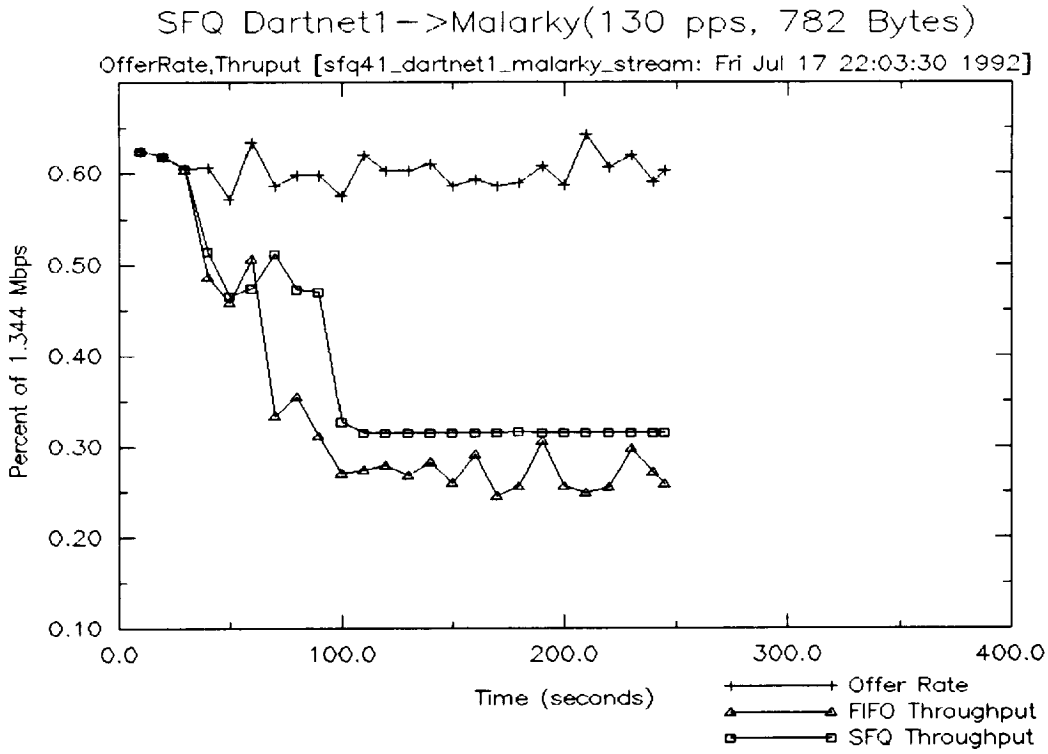


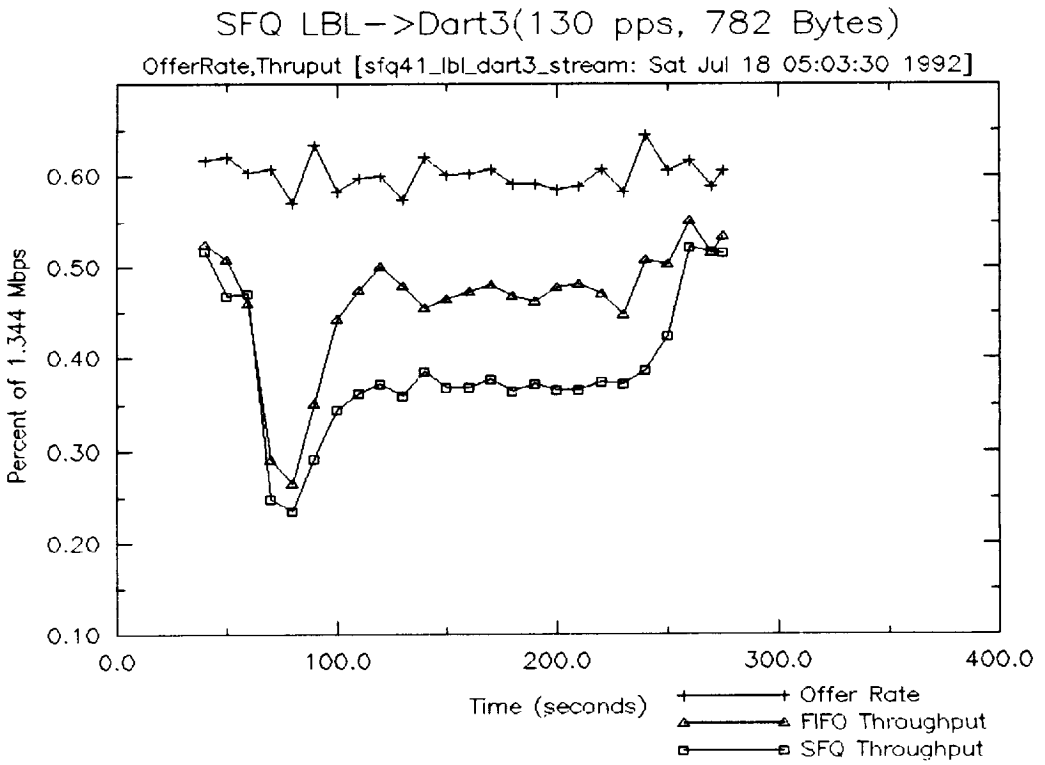
SFQ EXPERIMENT 31



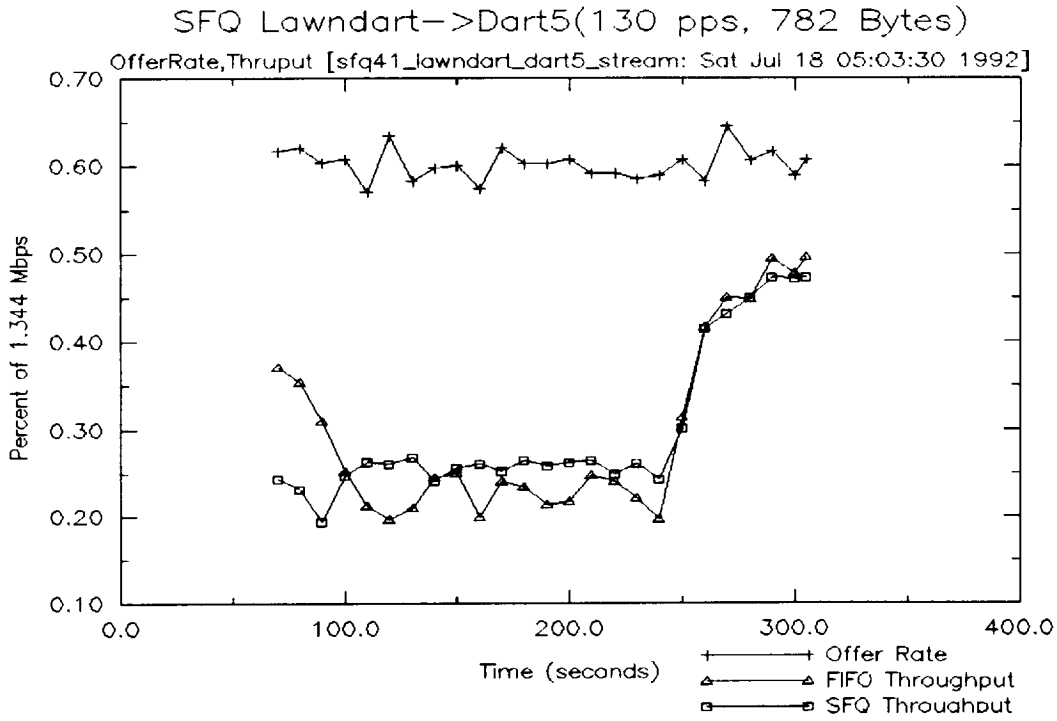


SFQ EXPERIMENT 41





SFQ EXPERIMENT 41



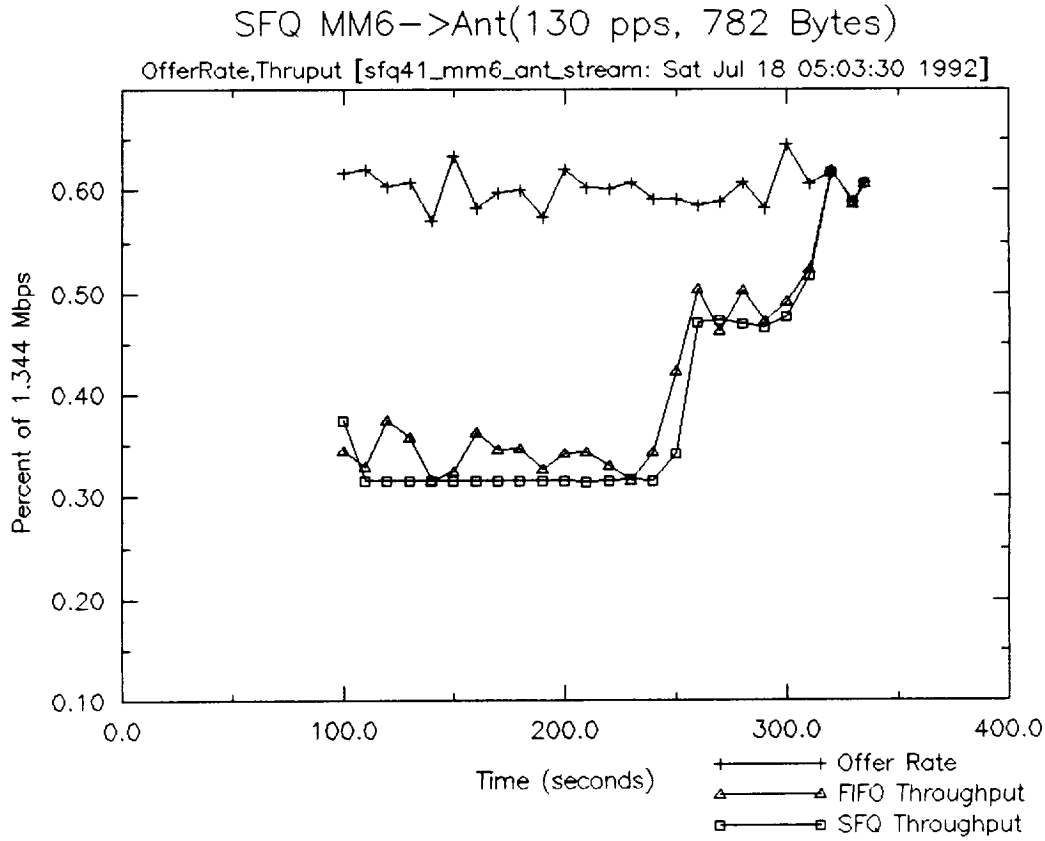


Table 3. Graceful Degradation Results From Time 0 to 30 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo31	61.61	61.61	0.0605	0.000044
	sfq31	61.47	61.47	0.0583	0.000049
	sfq41	61.63	61.63	0.0588	0.000051

*Time is approximate.

Table 4. Graceful Degradation Results From Time 30 to 60 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo31	60.39	48.35	0.5212	0.00503
	sfq31	60.43	48.53	0.9591	0.04639
	sfq41	60.39	48.47	0.9537	0.04045
LBL to Dart3	fifo31	61.45	49.79	0.5179	0.00557
	sfq31	61.45	48.58	0.9650	0.04815
	sfq41	61.46	48.56	0.9605	0.0429

*Time is approximate.

Table 5. Graceful Degradation Results From Time 60 to 90 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo31	59.38	33.34	0.99504	0.00743
	sfq31	59.38	32.43	2.4168	0.05466
	sfq41	59.39	48.52	1.8293	0.05488
LBL to Dart3	fifo31	60.45	30.24	0.5426	0.00004
	sfq31	60.45	31.44	1.5327	0.00016
	sfq41	60.45	25.85	1.0463	0.00005
Lawndart to Dart5	fifo31	61.47	34.43	0.9950	0.00896
	sfq31	61.47	33.98	2.3087	0.26848
	sfq41	61.47	22.31	1.9382	0.05580

*Time is approximate.

Table 6. Graceful Degradation Results From Time 90 to 120 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo31	59.97	27.42	1.4656	0.00532
	sfq31	59.95	23.69	1.6404	0.06492
	sfq41	59.95	31.90	1.9922	0.00157
LBL to Dart3	fifo31	59.38	47.29	0.5370	0.00012
	sfq31	59.38	34.47	1.3963	0.00701
	sfq41	59.38	36.03	0.8097	0.00358
Lawndart to Dart5	fifo31	60.43	22.02	1.0237	0.00010
	sfq31	60.43	34.62	2.2364	0.00730
	sfq41	60.43	25.71	2.2667	0.00098
MM6 to Ant	fifo31	61.47	34.96	0.9547	0.00582
	sfq31	61.46	24.07	1.4622	0.00585
	sfq41	61.46	33.50	1.8455	0.13713

*Time is approximate.

Table 7. Graceful Degradation Results From Time 120 to 245 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Dartnet1 to Malarky	fifo31	60.25	26.94	1.4840	0.00021
	sfq31	60.20	23.42	1.5336	0.00026
	sfq41	60.21	31.54	2.0037	0.00025
LBL to Dart3	fifo31	60.19	47.28	0.5370	0.00013
	sfq31	60.20	35.46	1.3625	0.00223
	sfq41	60.19	37.23	0.7953	0.00008
Lawndart to Dart5	fifo31	59.66	22.62	1.0238	0.00012
	sfq31	59.62	35.48	2.3359	0.00245
	sfq41	59.66	22.62	1.0238	0.00012
MM6 to Ant	fifo31	60.04	33.93	0.9735	0.00012
	sfq31	60.03	23.91	1.4822	0.00010
	sfq41	60.01	31.53	1.9654	0.00024

*Time is approximate.

Table 8. Graceful Degradation Results From Time 245 to 275 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
LBL to Dart3	fifo31	60.01	53.26	0.5177	0.00109
	sfq31	60.02	46.52	1.0397	0.00238
	sfq41	60.01	51.16	0.5553	0.00207
Lawndart to Dart5	fifo31	61.29	42.86	0.9837	0.00778
	sfq31	61.83	46.63	2.0102	0.00273
	sfq41	61.79	41.23	1.5244	0.01237
MM6 to Ant	fifo31	58.70	48.47	0.5099	0.00276
	sfq31	58.85	46.30	1.0008	0.00304
	sfq41	58.85	45.49	1.0091	0.01186

*Time is approximate.

Table 9. Graceful Degradation Results From Time 275 to 305 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Lawndart to Dart5	fifo31	60.01	43.82	0.5380	0.00029
	sfq31	60.01	47.23	1.0401	0.002752
	sfq41	60.01	47.30	1.0317	0.000472
MM6 to Ant	fifo31	61.79	48.68	0.5028	0.00016
	sfq31	61.78	47.17	0.9937	0.00070
	sfq41	61.77	47.22	0.9941	0.00075

*Time is approximate.

Table 10. Graceful Degradation Results From Time 305 to 335 Seconds*

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
MM6 to Ant	fifo31	60.01	60.01	0.0313	0.001457
	sfq31	60.01	59.78	0.0551	0.01628
	sfq41	60.01	59.81	0.0567	0.01707

*Time is approximate.

4.4 RESOURCE USAGE

4.4.1 Objective and Procedure

SFQ Experiment 24 experiment is similar in design to the experiment on starvation prevention (see Subsection 3.2). However, instead of three UDP streams flowing in the same direction, this experiment used two UDP streams flowing in one direction, and another stream in the reverse direction. This was to done to test SFQ's performance in relation to FIFO queueing. It also provided a stress test of the network, since previous experiments with cross streams failed (blackouts occurred).The source, destination, and direction of each stream are as follows (see Figure 5):

- Lawndart to Ant
- MM6 to Dart5
- Malarky to Dartnet1.

The streams from Lawndart to Ant and Malarky to Dartnet1 occupy 95% of the link capacity of a T1 line. Each of these streams is exponentially distributed, with an interarrival mean of 0.004902 seconds (~204 packets per second) and a constant packet size of 782 bytes (including UDP and IP

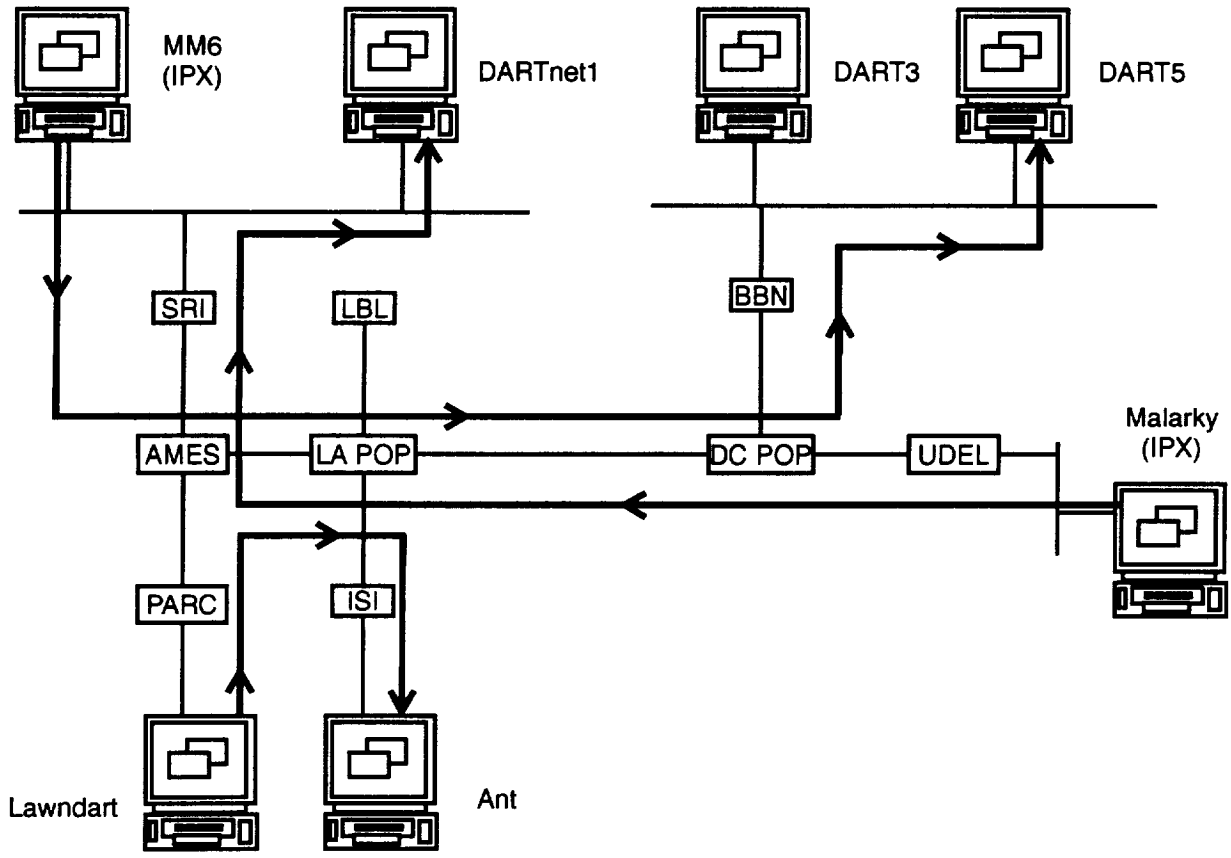


Figure 5. Resource Usage Traffic Flow

headers). The stream from MM6 to Dart5 occupies 20% of the link capacity of a T1 line. This stream is exponentially distributed, with an interarrival mean of 0.023256 seconds (~43 packets per second) and a constant packet size of 782 bytes.

4.4.2 Data Description

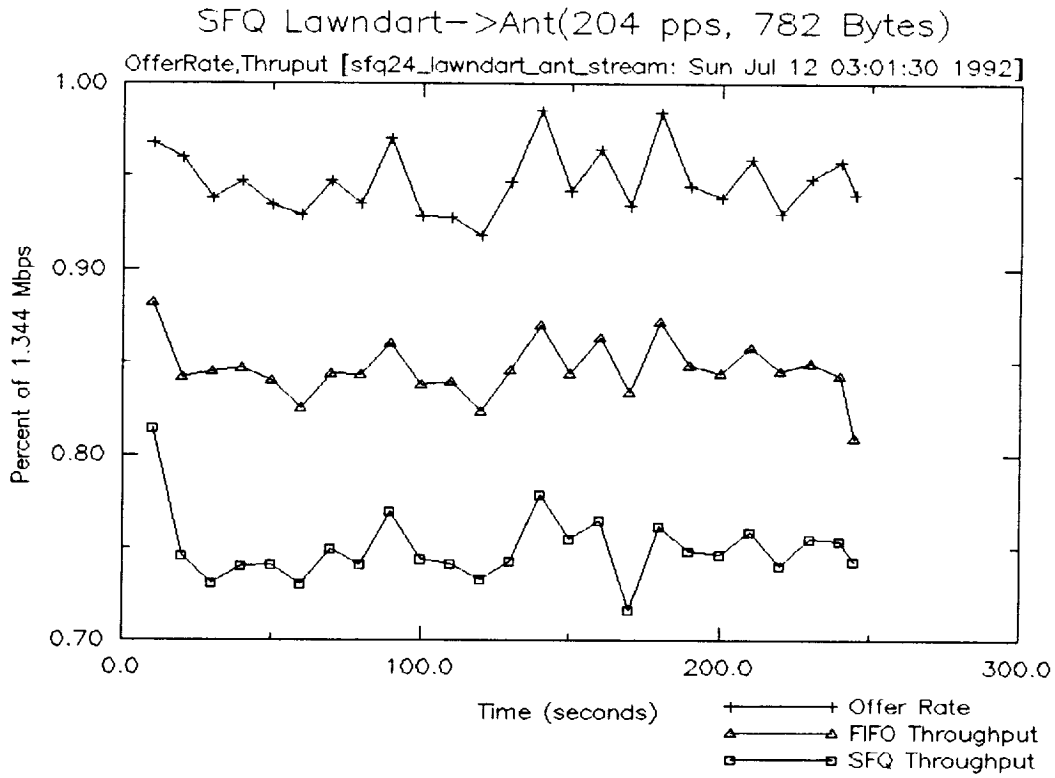
Each graph below shows the offer rate, the SFQ throughput, and the FIFO throughput for each stream. The experiment was executed without seed perturbation, which was not of interest.

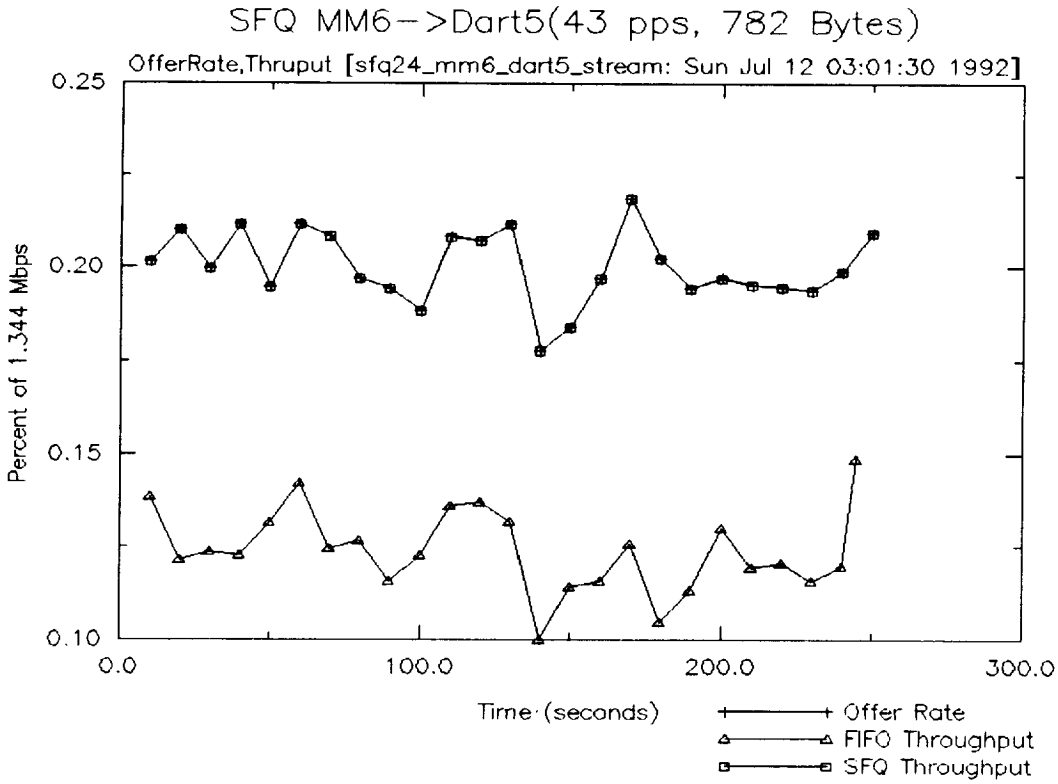
Table 11 following the graphs summarizes the results of the experiments, including average throughput, average delay, and delay variance for each stream.

4.4.3 Results

This experiment does show that SFQ does not significantly affect the throughput of the network, compared with FIFO queueing. At ~95% offer load, the average throughput for FIFO queueing was 94.68%, while for SFQ it was 94.46%. As in all experiments, the average delay and variance in delay is greater with SFQ than with FIFO, because SFQ requires more processing. However, when a stream is competing for scarce resources, the variance seems to be lower with SFQ for streams that are demanding less than their share.

SFQ EXPERIMENT 24





SFQ EXPERIMENT 24

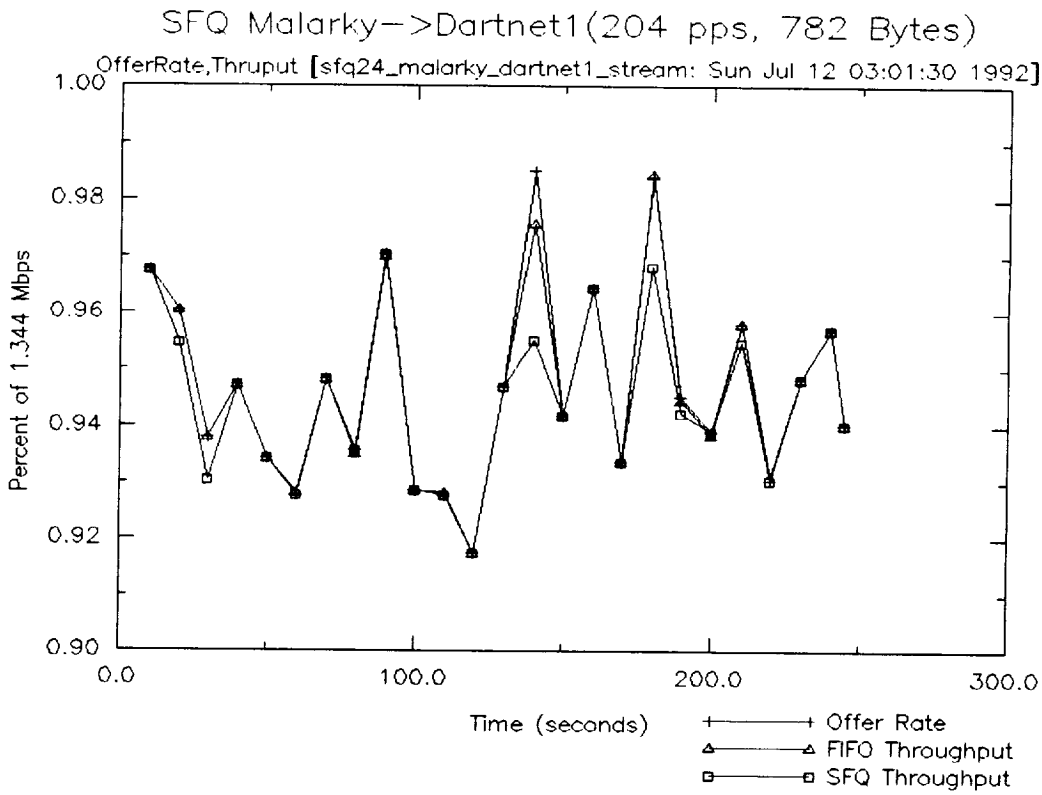


Table 11. Resource Usage

STREAM	EXPERIMENT NUMBER	AVERAGE OFFER RATE (percentage of 1.344 Mb/s)	AVERAGE THROUGHPUT (percentage of 1.344 Mb/s)	AVERAGE DELAY (seconds)	DELAY VARIANCE
Lawndart to Ant	fifo21	94.72	84.71	0.5714	0.0065
	sfq24	94.72	74.99	0.8562	0.0189
MM6 to Dart5	fifo21	20.01	12.34	0.5337	0.0015
	sfq24	20.01	20.01	0.0802	0.0001
Malarky to Dartnet1	fifo21	94.72	94.68	0.1455	0.0069
	sfq24	94.72	94.46	0.2808	0.0153

5 CONCLUSIONS

SFQ is an efficient queueing discipline for providing equal access to the available bandwidth. The isolation of the streams helps to ensure that no stream receives more than its fair share and that each stream degrades gracefully as more streams are added. SFQ also seems to possess very good scaling properties; but more work needs to be done to verify this. In particular, the choice of hash function and seed perturbation technique needs further investigation. The current choices may prove inadequate in a more stressful environment.