Combined Performance and Availability Analysis of a Switched Network Application

Steve Hunter* Teebu Philip† Kishor Trivedi ‡

Center for Advanced Computing and Communications
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291

Abstract

As switched networks providing services to end users become more commonplace, integral components of these networks must have an increasing level of dependability. One area of interest is determining the optimal number of network servers required for a switched network application by taking into consideration the characteristics of the network traffic and failure rate of the servers. A stochastic reward net model is developed for this purpose. Such models allow performance and availability measures to be combined.

Index Terms: ATM Networks, Network Modeling, Performability.

1 Introduction

As ATM becomes more commonplace in the work environment and services are made available to provide the transition from legacy LANs to switched networking, a need arises for *network servers* that can provide these services. While from an installation and operation point of view it is easiest to maintain services of this type on a single server, this approach doesn't necessarily provide the best solution from a performance and reliability perspective. In order to determine the appropriate number of servers, it is necessary to model the system. Analytic models are a cost-effective method often used to jointly optimize performance and reliability. Joint analysis of these characteristics has become common and is termed *performability modeling.* In this paper, a model is developed to determine the optimal number of network servers, N, for a switched network application. This is done by fixing the total amount of processing power of all the servers [6, 13], taking into account the failure rate of the servers, and also considering the characteristics of LAN traffic. This paper is a sequel to modeling a switching network access node as presented in [7].

*On leave from IBM Corp., Research Triangle Park, NC.

†Supported in part by an IBM fellowship.

‡Supported in part by the National Science Foundation under grant number EEC-9418765. Paper presenter.

For example, if the application/service is *LANE* [1], an access node in the network can be viewed as a LAN Emulation Client (LEC) that requires service from a LAN Emulation Server (LES) with a specified probability. This type of network can be modeled with Stochastic Reward Nets (SRNs) so that desired performability attributes can be obtained. Modeling of this type is becoming more important due to increased interest in issues such as scalability, redundancy, and dependability [2].

Several measures of system effectiveness have been proposed that combine the effects of failure/repair with system performance. First is the effect of increased load on the failure rates of the individual processors [8]. Another measure is throughput-oriented availability; however, failure is a relatively rare event that averages fault-free behavior with the behavior under faults. Therefore, this approach is not a very useful measure of system effectiveness [6]. Similarly, the mean response time that includes the effect of faults and queueing suffers from the same difficulty [13, 15]. In [6], Geist has woven the variance and the mean of the response time in the perceived mean. Tail probabilities of the response time distribution will also 'equitably' show the effect of performance and availability, but in spite of some recent progress in this direction [14], computing the response time distribution is difficult.

One measure used for this paper is the total loss probability of requests. It is the probability of rejection of an incoming task due either to a *system down* event or *system overloaded* event [17]. This measure appears to equally reflect performance and availability of the system. Another measure used which is associated to the latency of a request being serviced is *buffer utilization*. It is simply the expected queue length divided by the total queue size.

The paper is outlined as follows: Section 2 gives a brief overview of ATM along with some of the network services to be offered. Section 3 describes the network configurations to be analyzed. Section 4 provides background on stochastic Petri nets. In Section 5 the implementation of the network configurations are given using Petri nets [16]. Section 6 gives results for loss probability and buffer utilization with pure performance models and Section 7 adds availability to the models by allowing failures and repairs among the LAN Emulation Servers. The conclusion sums up what is learned by this exercise.

2 ATM Switching

One of the advantages of ATM switching is its ability to forward packets as small fixed length cells over a point to point circuit. This feature drastically reduces the amount of memory and processor power required at intermediate nodes and also reduces the latency of the network. However, the most important advantage of ATM is seen in the fact that it is a universal, service-independent switching and multiplexing technique that is totally independent of the underlying transmission technology and the transmission speed [4]. This ability makes ATM highly flexible for supporting existing and future network technologies.

2.1 Types of ATM Service

With the large number of established LANs that exist today, a method of migrating the existing LANs to ATM backbones is needed. At least two techniques are being pursued in doing this. One technique is "LAN Emulation" as described by the ATM Forum LAN Emulation SubWorking Group (LANE SWG)

[1]. The second technique, chosen by the IETF (Internet Engineering Task Force), is "Classical IP and ARP over ATM" and is described in RFC 1577[1]. This service operates at the network layer and provides internetworking with legacy LANs using IP routing. The background of these two techniques is presented in some detail in [10, 18].

For the purposes of this paper, the importance of these two proposals is that they both require service to be located somewhere within the network. This service could be located on a single server or, by using the same aggregate amount of processing power, the service could be distributed across multiple servers to which would provide better performability. Since both proposals are somewhat similar, the LAN Emulation technique and terminology will be the focus of the model presented.

2.2 LAN Emulation

The specification for LAN Emulation [1] was adopted in March 1995 by the ATM Forum. The basic idea is to implement a connectionless MAC service over a connection oriented circuit. This is combined with a server to provide the multicast MAC service.

Any station that is connected directly to the ATM network and is a participant in the ATM service contains a "LAN Emulation Client" (LEC). These stations can be workstations, servers, bridges, routers, or another type of user of the ATM service. The LECs primary function is to provide the MAC connectionless service by using the AAL5 for communication between ATM stations and UNI signaling services to set up and tear down VCCs to establish the communication.

Due to its dual nature, a LEC is identified by its 6 byte IEEE MAC address, for MAC emulation and a 20 byte ATM address, for its ATM attachment. When the LEC receives a LAN packet, it searches a resolution table to map the MAC destination address to a VPI/VCI pair that corresponds to a "Data Direct VCC" to the receiving LEC station. If the MAC destination address is unknown, a "LE Address Resolution Protocol (LE-ARP)" request is sent by the source LEC to the "LAN Emulation Server" (LES) asking for the ATM address. With this address a "Data Direct VCC" can be set up by the LEC.

The primary function of the LES is to resolve MAC addresses, via the LE-ARP request. These requests from the LECs are sent over the "Control Direct VCC" to the LES. Having resolved the request, the LES forwards the request to the destination LEC, which has the responsibility of responding to the LES with an LE-ARP reply containing its own ATM address. The LES forwards the reply to the source LEC and a connection is set up.

3 Network Description

The network being analyzed consists of some number of LECs and LESs. The LECs receive LAN traffic and will deliver it to a host across its ATM link using the techniques described by the LAN Emulation specification. Similarly, packets are received from the ATM link and are forwarded to the appropriate host/LAN.

As a packet enters a LEC, an address search is started and with some probability the search is successful. This probability depends on the size of the cache and the retention technique used and can be estimated or determined by measurements. If the search isn't successful, the LE_ARP procedure is started. In any case, the packet is forwarded to the ATM interface once the forwarding address is found.

As previously mentioned, the LES function can be performed across some number, N, of network servers. For the case of multiple servers, common and separate queue configurations, as shown in Figure 1, are evaluated. The difference in these two techniques from the network perspective is the LES function with a common queue has a centralized network location while the same function with a separate queue is most likely distributed across the network. These two techniques are compared in some detail in [9].

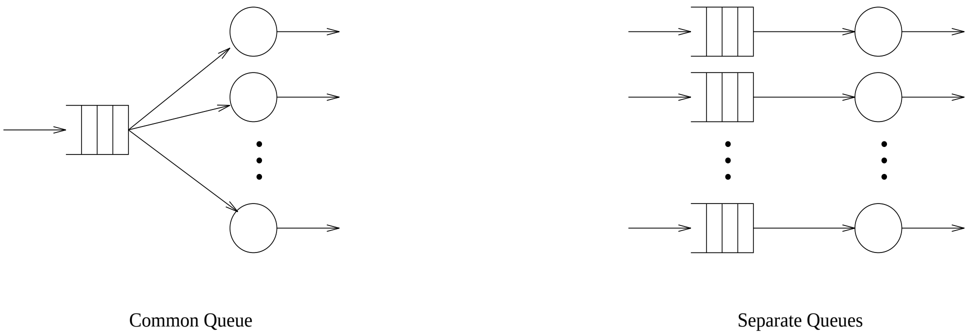


Figure 1: Queue Configurations

The goal of this paper is to determine the best design topology for the LES function by taking into account packet loss, product failure and queue structure in order to optimize performability.

4 Stochastic Petri Net

A Petri net [16] is a directed bipartite graph with two disjoint and finite sets of nodes: *places* and *transitions*. In a graphical representation, the places are depicted as circles and the transitions by rectangles (bars). A place is an *input* to a transition if there is a directed edge called an *input arc* from the place to the transition. Similarly, a place is an *output* to a transition if there is a directed edge called an *output arc* from the transition to the place. *Tokens*, depicted by dots, are associated with places and the movement of these tokens represents the dynamic behavior of the system. The tokens move based upon the *firing* of transitions. A transition is *enabled* to fire if each of its input places contains at least one token. Upon firing, one token from each input place is removed and in each of the output places, one token is deposited.

A *marking* of a Petri net is the distribution of tokens in the set of places. Thus, firing of a transition results in a new marking. Each marking defines a state of the system. If the number of tokens in the net is bounded, then the number of markings is finite. A marking is said to be *reachable* from an original marking if there is a sequence of firings starting from the original firing which result in that marking. The *reachability set(graph)* of a Petri net is the set of markings that are reachable from the initial marking.

A stochastic Petri net (SPN) is a Petri net with an exponentially distributed delay associated with firing of each of its transitions. A SPN can be used to model the temporal behavior of a system along with its functional behavior.

Generalized stochastic Petri nets (GSPNs) are an extension to SPNs and allow immediate transitions to have zero firing delay or exponentially distributed firing delay [12]. GSPNs also include an *inhibitor* arc to allow a transition to be disabled by being connected to a place with at least as many tokens as the multiplicity of the arc. Both SPNs and GSPNs have been shown to be equivalent to CTMCs.

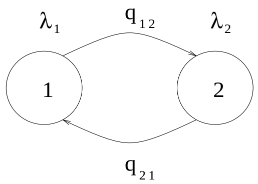
The specification of a system using GSPN's can be tedious and troublesome. To remedy this, Ciardo et al. [3] introduced several structural extensions to GSPNs. Variable multiplicity arcs, enabling functions (also known as *guards*) for transitions, marking dependent arc multiplicities and user defined transition priorities. The resulting net with all these extensions and the capability of assigning a real valued reward rate to any marking is termed as a stochastic reward net (SRN). These enhancements allow a model to keep a simpler representation of the system at the net level [11].

5 Model of Network Application

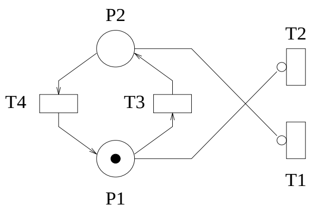
For the networking configurations shown in Figure 1, it is assumed that each LES has its own repair facility and is independent of failure and repair of any other component. Also, if an LES should fail, all incoming requests are directed to an active LES. The models for these two configurations are described in the following sections, but first the traffic model is described.

5.1 Traffic Model

The input traffic streams are modeled with independent Markov-Modulated Poisson Processes (MMPPs). The MMPP is used in a wide variety of applications due to their ability to capture the bursty nature of network traffic [5]. For the purpose of this paper a two-state MMPP as shown in Figure 2 is used.



Two-state MMPP



Petri Net Model

Figure 2: Traffic Model

The Petri net configuration for the traffic model is also shown in Figure 2 and corresponds to the MMPP if firing rates for *transitions* $T1$, $T2$, $T3$, and $T4$ are $exp(\lambda_1)$, $exp(\lambda_2)$, $exp(q_{21})$, and $exp(q_{12})$, respectively. To include burstiness in the model, the parameters $\lambda_1 = 3000$ and $\lambda_2 = 1000$ are used. If no burstiness is desired, then the parameters are set to $\lambda_1 = 2000$ and $\lambda_2 = 2000$. The parameters for q_{12} and q_{21} are the same so that each MMPP state has the same occupancy probability.

5.2 Common Queue Model

The SRN used for the common queue model is shown in Figure 3. The approach used combines the failure and repair behavior of the LESs into one model and is used to determine the total processing capability.

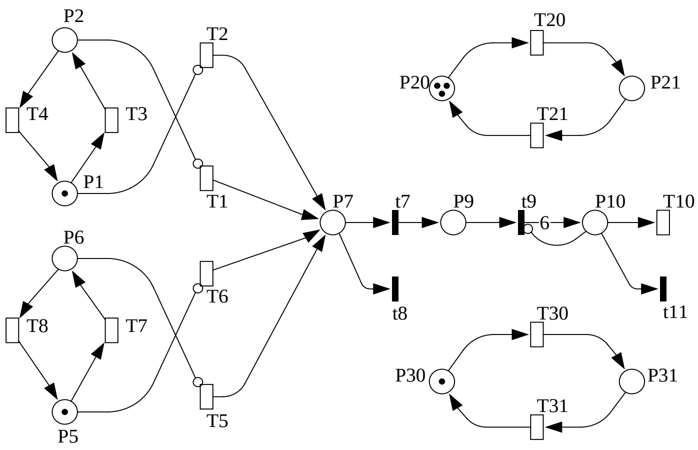


Figure 3: Common Queue SPN

Transitions $T1$, $T2$, $T5$, and $T6$ are used to generate the requests to be serviced. The activation of these *transitions* are controlled by *places* $P1$, $P2$, $P5$, and $P6$ and *transitions* $T3$, $T4$, $T7$, and $T8$. All of these work together, as described in the previous section, to represent input traffic from two LANs.

Place $P7$ catches all requests being sent to the servers. If there is adequate buffer space, these requests are forwarded to *place* $P9$, through *transition* $t7$; otherwise, the requests are purged (i.e., the buffer is full), via *transition* $t8$. Once the requests are buffered in *place* $P9$, they are serviced by *transition* $T10$.

The addition of *transition* $t9$, and *place* $P10$, with the appropriate *arcs* joining them, allows the service time of a request to be modified by including Erlang stages. This addition causes the service time to approximate a deterministic value, as more stages are added, while maintaining the overall mean service time.

The model shown in Figure 3 is an example of a 6-stage Erlang, if the *firing* rate of *transition* $T10$ is

$$\#processors\_active\ *\ individual\_processing\_rate\ *\ 6$$

with the

$$individual\_processing\_rate = \frac{aggregate\_processing\_rate}{\#processors}$$

and the

$$aggregate\_processing\_rate = 3600.$$

The model for failure and repair is represented by *places* $P20$ and $P21$ with *transitions* $T20$ and $T21$. *Place* $P20$ shows the initial number of servers available by the number of *tokens* it contains; therefore, failures occur at rate $\#P20 * \lambda_{fail}$, where λ_{fail} is the *firing* rate of *transition* $T20$. Similarly *Place* $P21$ represents the number of servers that have failed and are repaired at $\#P21 * \mu_{repair}$, which corresponds to *transition* $T21$. The queue failure and repair are represented by *transitions* $T30$ and $T31$, respectively.

Similar to *transition* $t8$, *transition* $t11$ is used to purge requests held in the queue; however, these requests are removed only when all servers have failed. For both $t8$ and $t11$, *guards* are used to activate the *transitions* for the appropriate circumstances.

5.3 Separate Queue Model

The approach used for the common queue model to combine all the servers into a single transition cannot be used for the separate queue model, as shown in Figure 4. Therefore, the numerical results of the separate queue model are more state intensive for the same number of LESs.

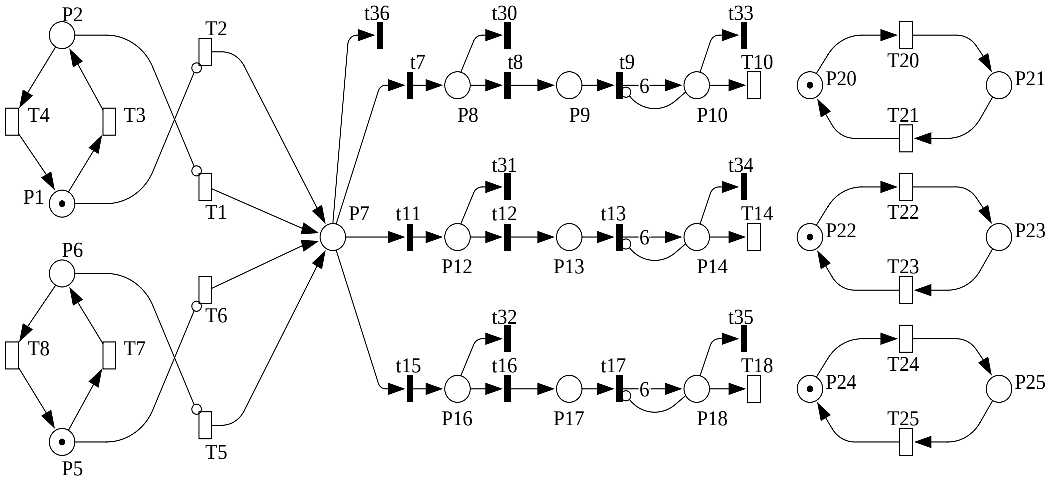


Figure 4: Separate Queue SPN with $N = 3$

The acceptance policy of the requests is changed for this model, such that the requests, as well as the total amount of buffer, are evenly distributed over all active servers. Therefore, the requests are lost on a per server basis.

Figure 4 is an example of 3 servers with each using 1/3 of the total buffer space. When a request enters the system and is buffered into *place* $P8$, $P12$, or $P16$, it is accepted by the firing of *transition* $t8$, $t12$, or $t16$, respectively. If accepted, the request is queued into the corresponding *place* $P9$, $P13$, or $P17$ and is serviced by *transition* $T10$, $T14$, or $T18$. If an individual queue is full, a request is purged by the appropriate *transition* $t30$, $t31$, or $t32$.

Similar to the case with the common queue, the addition of *transitions* $t9$, $t13$, and $t17$ and *places* $P10$, $P14$ and $P18$ allow the service time of a request, at each server, to be modified by including Erlangian stages. The model shown in Figure 4 is an example of a 6-stage Erlang, if the *firing* rate of *transitions* $T10$, $T14$, and $T18$ is $individual\_processing\_rate * 6$ with the $individual\_processing\_rate$ and the $aggregate\_processing\_rate$ as previously defined.

Three submodels for the failure and repair of the servers are represented by *places* $P20$-$P25$ and *transitions* $T20$-$T25$. For example, if *place* $P20$ contains a token, then the queue, represented by *place* $P10$, receives requests and *transition* $T10$ processes them. For this case the rate of failure, represented by *transition* $T20$ is λ_{fail}; therefore, *place* $P21$ contains a token if the server has failed with repair rate μ_{repair}, represented by *transition* $T21$. Since the processors and queues are separated in this model, the failure of the queues is included as part of the server failure and repair submodels.

Also similar to the common queue, if an individual queue has failed, requests are purged from its queue and incoming requests are routed to other servers until the failed server is repaired. This is the purpose of *transitions* $t33 - t35$. For the separate queue configuration, it is necessary to add *transition* $t36$ to purge all incoming requests when all servers are in a failed state. All of *transitions* $t30 - 36$ are activated under the appropriate circumstances by the use of *guards*.

6 Performance-only Model Results

To better understand the effects of failure later, the first runs of the model use performance criteria only. That is, *transitions* $T20$, $T22$ and $T24$ (where applicable) of the Petri nets shown in Figures 3 and 4 are disabled; therefore, all loss of requests is due to buffer overflow.

6.1 Non-Bursty Traffic

The first results are for a common buffer configuration with no burstiness present in the arrival stream. For this case, distributing the processing power over several processors has no effect and the resulting B_{util} are 0.0166666675359 and 0.0143518522382 for $k = 1$ and $k = 6$, respectively. The resulting P_{loss} is zero, due to precision. The results for no burstiness and a separate buffer configuration are shown in Table 1. For this case distributing the processing power does have an effect on the steady-state results. P_{loss} increases steadily with the number of servers and B_{util} increasing up to 4/5 servers for E_1/E_6, but then decreasing, due to the high loss rate of requests. As expected, based on this criteria the common queue configuration is the better technique.

6.2 Bursty Traffic

The results for bursty traffic and a common buffer configuration again has no effect when distributing the processing power over several processors and the resulting B_{util} are 0.0175623744726 and 0.0150103112683 for $k = 1$ and $k = 6$, respectively. The resulting P_{loss} is zero. The results for the separate buffer configuration with burstiness present in the arrival stream is shown in Table 2. Comparing Table 2 with the Table 1, P_{loss} increases steadily with the number of servers and B_{util} increasing up to 3/4 servers for E_1/E_6, instead of as before, and then again decreasing, due to the high loss rate of requests. Again, based on this criteria the common queue configuration is the better technique.

Plots of P_{loss} are shown in Figure 5 for the separate queue configuration with both bursty and non-bursty traffic applied.

Table 1: Model Parameters and Results for Non-Bursty Traffic with Separate Buffers and No Failure

Parm	**Processing**		**SS Results:** E_k **with** $k = 1$		**SS Results:** E_k **with** $k = 6$	
Set	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
7	3600	1	0.0166666675359	0.0000000000000000	0.0143518522382	0.0000000000000000
8	1800	2	0.0658534765244	0.0007623434066770	0.0481253191829	0.0000225901603699
9	1200	3	0.1666666716340	0.0909090638161000	0.1684107780460	0.0581717491150000
10	900	4	0.1630001366140	0.2778126597400000	0.1774601936340	0.2571524381640000
11	720	5	0.1567199528220	0.4115199446680000	0.1678081303830	0.4014772176740000
12	600	6	0.1365079283710	0.5079365074630000	0.1440338194370	0.5007636249070000

Table 2: Model Parameters and Results for Bursty Traffic with Separate Buffers and No Failure

Parm	**Processing**		**SS Results:** E_k **with** $k = 1$		**SS Results:** E_k **with** $k = 6$	
Set	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
19	3600	1	0.0175623744726	0.0000000000000000	0.0150103112683	0.0000000000000000
20	1800	2	0.0720643252134	0.0017126798629800	0.0545707903802	0.0001589655876160
21	1200	3	0.1649379134180	0.1014494299890000	0.1663605719800	0.0702752470970000
22	900	4	0.1589946150780	0.2847732305530000	0.1710152775050	0.2624130845070000
23	720	5	0.1534353643660	0.4152586460110000	0.1635769307610	0.4031346440320000
24	600	6	0.1341051012280	0.5105206370350000	0.1412726193670	0.5016635358330000

7 Combined Performance and Availability Results

By including availability into the model, the *transitions* $T20$, $T22$ and $T24$ (where applicable) of the Petri nets shown in Figures 3 and 4 do *fire*; therefore, requests are lost due to both buffer overflow and system failure.

The mean time to failure (MTTF) used for this model for the processors and queues, respectively, are 1 failure per year and 1 failure per 6 years. The mean time to repair (MTTR) for both processors and queues is 4 hours.

7.1 Non-Bursty Traffic

The results, shown in Table 3, uses a common buffer configuration with no burstiness present in the arrival stream. From these results it is seen that distributing the processing power over several processors now has an effect. For the service time of both cases the best solution for P_{loss} is to use a 2 servers, but this causes a slight rise in B_{util}. This rise is reduced some by going to 3 or more servers. The results with no burstiness and a separate buffer configuration are shown in Table 4. If the service time is E_k with $k = 1$, the best solution for P_{loss} is to use a single server; however, if the service time becomes more deterministic with $k = 6$, then the best solution is to use 2 servers assuming a significant rise in B_{util} is allowable. For this case, the rise in B_{util} continues to increase by adding more servers. Comparing these results, the best P_{loss} occurs with 2 servers, each with a more deterministic service time.

Table 3: Model Parameters and Results for Non-Bursty Traffic with Common Buffer and Failure

Parm	**Processing**		**SS Results: E_k with $k=1$**		**SS Results: E_k with $k=6$**	
Set	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
1	3600	1	0.0166577398777	0.000534892082214	0.0143441809341	0.000534296035767
2	1800	2	0.0167100057006	0.000076472759247	0.0143812941387	0.000076413154602
3	1200	3	0.0166882872581	0.000076293945313	0.0143673680723	0.000076234340668
4	900	4	0.0166836511344	0.000076293945313	0.0143642462790	0.000076234340668
5	720	5	0.0166816972196	0.000076293945313	0.0143629228696	0.000076234340668
6	600	6	0.0166806112975	0.000076293945313	0.0143621806055	0.000076234340668

Table 4: Model Parameters and Results for Non-Bursty Traffic with Separate Buffers and Failure

Parm	**Processing**		**SS Results: E_k with $k=1$**		**SS Results: E_k with $k=6$**	
Set	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
7	3600	1	0.0166577510536	0.000534296035767	0.0143441883847	0.000533759593964
8	1800	2	0.0658534392715	0.000762641429901	0.0481252893806	0.000022888183594
9	1200	3	0.1666666716340	0.090909063816100	0.1684107929470	0.058171749115000
10	900	4	0.1630001366140	0.277812659740000	0.1774602085350	0.257152438164000
11	720	5	0.1567199528220	0.411519944668000	0.1678081303830	0.401477217674000
12	600	6	0.1365079432730	0.507936507463000	0.1440338194370	0.500763624907000

7.2 Bursty Traffic

For the common buffer configuration with burstiness present, the results of the comparisons are the same, but with just a slightly higher result for P_{loss}. These results are shown in Table 5. Also, for the separate buffer configuration with burstiness present, the results, shown in Table 6, of the comparisons are the same; however, there is a more drastic difference between the cases for 1 and 2 servers for the first service time distribution than there was with non-bursty traffic. The opposite effect occurs (i.e., the difference becomes less) when the service time becomes more deterministic. Comparing these results, the best P_{loss} again occurs with 2 servers, each with a more deterministic service time.

Plots of P_{loss} are shown in Figure 6 for both the common queue and separate queue configurations with bursty traffic applied. Note the effect failure has in comparison to Figure 5.

8 Combined Performance and Availability Results 2

A second set of results were obtained after changing the failure rate of the queue to 1 failure in 2 years instead of 1 failure in 6 years, as before. All other parameters remain the same.

8.1 Non-Bursty Traffic

Table 7 shows the new results for non-bursty traffic and a common queue. These results show that P_{loss} increases somewhat for a single server, but drastically for 2 or more servers. B_{util} increased slightly for all cases. Table 8 shows the new results for non-bursty traffic and a separate queue. There is very little

Table 5: Model Parameters and Results for Bursty Traffic with Common Buffer and Failure

Parm Set	Processing		SS Results: E_k with $k=1$		SS Results: E_k with $k=6$	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
13	3600	1	0.0175529588014	0.0005351305007930	0.0150022823364	0.000534534454346
14	1800	2	0.0176111254841	0.0000764727592468	0.0150449229404	0.000076413154602
15	1200	3	0.0175864491612	0.0000762939453125	0.0150280026719	0.000076234340668
16	900	4	0.0175812151283	0.0000762939453125	0.0150243053213	0.000076234340668
17	720	5	0.0175790097564	0.0000762939453125	0.0150227453560	0.000076234340668
18	600	6	0.0175777859986	0.0000762939453125	0.0150218755007	0.000076234340668

Table 6: Model Parameters and Results for Bursty Traffic with Separate Buffers and Failure

Parm Set	Processing		SS Results: E_k with $k=1$		SS Results: E_k with $k=6$	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
19	3600	1	0.0175529718399	0.000534474849701	0.0150022916496	0.000533938407898
20	1800	2	0.0720642879605	0.001713037490840	0.0545707568526	0.000159323215485
21	1200	3	0.1649378985170	0.101449429989000	0.1663605570790	0.070275247097000
22	900	4	0.1589946150780	0.284773230553000	0.1710152775050	0.262413084507000
23	720	5	0.1534353643660	0.415258646011000	0.1635769307610	0.403134644032000
24	600	6	0.1341051012280	0.510520637035000	0.1412726193670	0.501663535833000

difference with multiple servers, the result for a single server increased some and is approximately the same as the single server common queue. Comparing these results, the best P_{loss} occurs with separate queues with 2 servers, each with a more deterministic service time.

8.2 Bursty Traffic

The results for bursty traffic and a common queue are shown in Table 9. These results show that P_{loss} increases somewhat for a single server, but drastically for 2 or more servers. B_{util} increased slightly for all cases. The results for bursty traffic and a separate queue are shown in Table 10. There is very little difference with multiple servers, the result for a single server increased some and is approximately the same

Table 7: Model 2 Parameters and Results for Non-Bursty Traffic with Common Buffer and Failure

Parm Set	Processing		SS Results: E_k with $k=1$		SS Results: E_k with $k=6$	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
1	3600	1	0.0166551955044	0.000687360763550	0.0143419941887	0.000686645507812
2	1800	2	0.0167074538767	0.000229001045227	0.0143791018054	0.000228762626648
3	1200	3	0.0166857372969	0.000228822231293	0.0143651776016	0.000228583812714
4	900	4	0.0166811030358	0.000228822231293	0.0143620558083	0.000228583812714
5	720	5	0.0166791491210	0.000228822231293	0.0143607333302	0.000228583812714
6	600	6	0.0166780631989	0.000228822231293	0.0143599910662	0.000228583812714

Table 8: Model 2 Parameters and Results for Non-Bursty Traffic with Separate Buffers and Failure

Parm Set	**Processing**		**SS Results: E_k with $k=1$**		**SS Results: E_k with $k=6$**	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
7	3600	1	0.0166552066803	0.0006867051124570	0.0143420025706	0.0006859898567200
8	1800	2	0.0658534318209	0.0007628798484800	0.0481252819300	0.0000231266021729
9	1200	3	0.1666666716340	0.0909090638161000	0.1684107929470	0.0581717491150000
10	900	4	0.1630001366140	0.2778126597400000	0.1774602085350	0.2571524381640000
11	720	5	0.1567199528220	0.4115199446680000	0.1678081303830	0.4014772176740000
12	600	6	0.1365079432730	0.5079365074630000	0.1440338194370	0.5007636249070000

Table 9: Model 2 Parameters and Results for Bursty Traffic with Common Buffer and Failure

Parm Set	**Processing**		**SS Results: E_k with $k=1$**		**SS Results: E_k with $k=6$**	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
13	3600	1	0.0175502765924	0.000687658786774	0.0149999950081	0.000686883926392
14	1800	2	0.0176084339619	0.000229060649872	0.0150426281616	0.000228822231293
15	1200	3	0.0175837632269	0.000228881835938	0.0150257116184	0.000228583812714
16	900	4	0.0175785291940	0.000228881835938	0.0150220142677	0.000228583812714
17	720	5	0.0175763219595	0.000228881835938	0.0150204543024	0.000228583812714
18	600	6	0.0175751000643	0.000228881835938	0.0150195844471	0.000228583812714

as the single server common queue. Comparing these results, the best P_{loss} occurs with separate queues with 2 servers, each with a more deterministic service time.

9 Conclusion

Based on the results, it is seen that including the possibility of server failure does make a difference on P_{loss}. Also, it is shown that having multiple servers for a virtual network can show significant improvement. However, the amount of improvement can depend on the service time distribution and configuration type of the servers. It is also learned that the potential burstiness of traffic has less effect than what a person might expect.

Table 10: Model 2 Parameters and Results for Bursty Traffic with Separate Buffers and Failure

Parm Set	**Processing**		**SS Results: E_k with $k=1$**		**SS Results: E_k with $k=6$**	
	μ_{proc}	#Procs	B_{util}	P_{loss}	B_{util}	P_{loss}
19	3600	1	0.0175502896309	0.000686883926392	0.0150000052527	0.000686168670654
20	1800	2	0.0720642730594	0.001713275909420	0.0545707456768	0.000159502029419
21	1200	3	0.1649378985170	0.101449429989000	0.1663605570790	0.070275247097000
22	900	4	0.1589946150780	0.284773230553000	0.1710152775050	0.262413084507000
23	720	5	0.1534353643660	0.415258646011000	0.1635769307610	0.403134644032000
24	600	6	0.1341051012280	0.510520637035000	0.1412726193670	0.501663535833000

References

[1] ATM Forum, LAN Emulation SWG Drafting Group, "LAN Emulation Over ATM Specification - Version 1.0," Jan., 1995.

[2] *Issue on Dependability of Networking Services*, IEEE Communication Magazine, June 1993.

[3] Ciardo, G., Muppala, J., and Trivedi, K., "SPNP: Stochastic Petri Net Package," *Proceedings International Conference on Petri Nets and Performance Models, pages 142-150*, Kyoto, Japan, December, 1989.

[4] Denzel, Wolfgang E., "High-Speed ATM Switching," IEEE Communication Magazine, pp. 26, February 1993.

[5] Fischer, W. and Meier-Hellstern, K., "The Markov-Modulated Poisson Process (MMPP) Cookbook," Performance Evaluation, Vol. 18, No. 2, 1992, pp. 149–171.

[6] Geist, R. Stevenson, D. and Allen, R., "The Perceived Effect of Breakdown and Repair on the Performance of a Multiprocessor System," *Performance Evaluation*, Vol. 6, 1986, pp. 249-260.

[7] Hunter, S.W., Fricks, R.M., "The Effects of LAN-like Input Traffic on a Switch Access Node," IEEE International Computer Performance and Dependability Symposium, Urbana-Champaign, IL – Sept. 4-6, 1996.

[8] Iyer, R.K., Rosetti, D.J., and Hsueh, M.C., "Measurement and Modeling of Computer Reliability as Affected by System Activity," *ACM Transactions on Computer Systems,* vol. 4, pp. 214 - 237, August 1986.

[9] Kleinrock, L., *Queueing Systems Volume I: Theory.* John Wiley and Sons, New York, 1976.

[10] Laubach, M., "Classical IP and ARP over ATM," RFC 1577, Jan. 1994.

[11] Malhotra, M. and Trivedi, K., Dependability Modeling Using Petri-Nets, *IEEE Transactions on Reliability,* Vol. 44, No. 3, pp. 428-440, Sept., 1995.

[12] Marsan, A., Balbo, G. and Conte, G., A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comp. Systems*, Vol 2. No. 2, May 1984, pp. 93-122.

[13] Mitrani, I. and King, P., "Multiserver Systems Subject to Breakdown: An Empirical Study," *IEEE Transactions on Computers*, vol. C-32, pp. 96–98, Jan. 1983.

[14] Muppala, J., Trivedi, K., Mainkar, M. and Kulkarni, V., "Numerical Computation of Response Time Distributions Using Stochastic Reward Nets," *Annals of Operations Research,* Vol 48, pp. 155-184, 1994.

[15] Nicola, V., Kulkarni, V. and Trivedi, K., "A Queueing Analysis of Fault-Tolerant Computer systems," *IEEE Transactions on Software Engineering*, vol. SE-13, pp. 363-375, March 1987.

[16] Peterson, J.L., *Petri Net Theory and the Modeling of Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1981.

[17] Trivedi, K., Ibe, O., Sathaye, A., and Howe, A., Should I Add a Processor?, *23rd Annual Hawaii Conference on System Sciences,* pp. 214-221, Jan. 1990.

[18] Truong, Hong Linh, Ellington, William W., Le Boudec, Jean-Yves, Meier, Andreas X., and Pace, J. Wayne, "LAN Emulation on an ATM Network," IEEE Communication Magazine, pp. 70-85, May 1995.

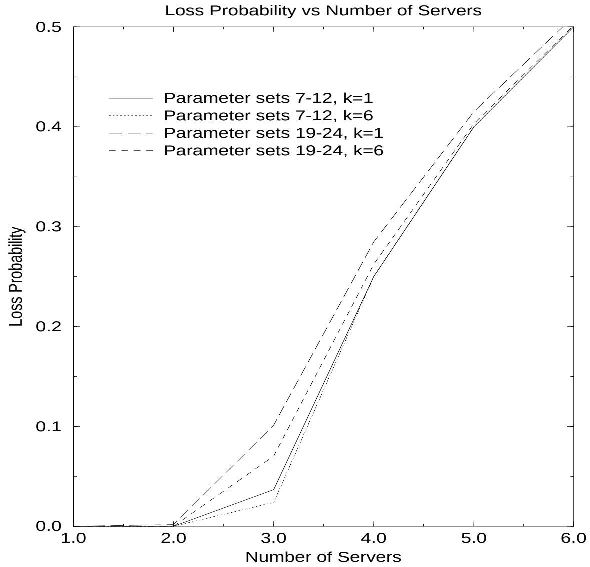


Figure 5: Loss Probability for Bursty and Non-bursty Traffic with Separate Queues and No Failure

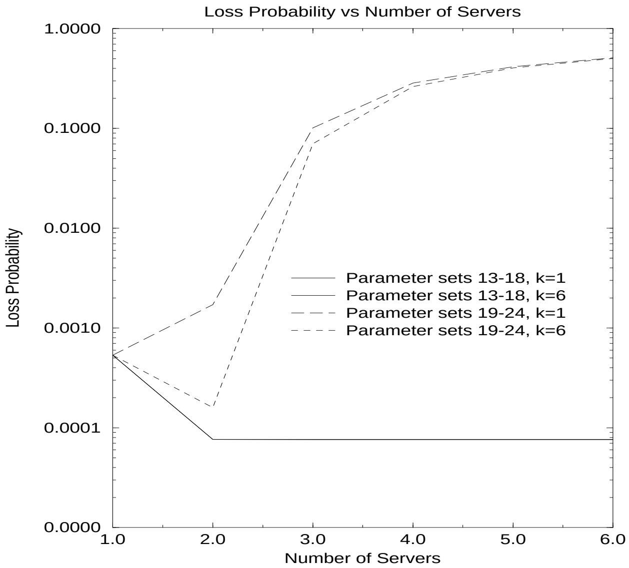


Figure 6: Loss Probability for Bursty Traffic with Separate and Common Queues and Failure