Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

HIERARCHICAL TRAFFIC MODELING AND ANALYSIS

Y.-D. Lin
M. Gerla

July 1993
CSD-930023

Hierarchical Traffic Modeling and Analysis

Ying-Dar Lin[1]
Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024

Abstract

We first characterize traffic patterns in terms of statistical properties, transmission protocols, and traffic models. Traffic models serve as the computational bases for admission control and bandwidth allocation. To have accurate computations, we propose a detailed and realistic traffic model, HAP (Hierarchical Arrival Process) which is shown to be a special class of MMPP (Markov Modulated Poisson Process). Packets generated from HAP are modulated by processes at user, application, and message levels. HAP's long-term and short-term queueing behaviors are analyzed. Delay under HAP traffic can be well over tens of times higher than Poisson traffic. Congestion may persist for minutes. HAP's dramatic short-term behavior explains the occasional congestion in the real networks. Conventional traffic models, however, do not exhibit this behavior. With these results, we give implications for broadband network control.

1 Introduction

A recent trace-driven simulation study [FL91] indicates that real computer network traffic has extreme traffic variability on time scales ranging from milliseconds to months, which is not captured by conventional Poisson or MMPP traffic models. The study shows that, when the real traffic is fed into a network simulator, the short-term performance behavior is much worse than the one predicted using analytic traffic models. During congested periods, congestion persists and losses can be significant. There is, indeed, a *gap* between the behavior under real traffic and that produced by analytic traffic models.

[1]The author is now with Dept. of Computer and Information Science at National Chiao-Tung University, Taiwan.

Traffic results from the interaction of various traffic sources. Thus, network traffic depends on the way traffic sources behave locally and pairwisely. We often characterize traffic pattern, or traffic behavior, using the following parameters: locality, correlation, and burstiness. Locality deals with the geographical distribution of traffic sources. High locality (i.e. most of the traffic is contributed by a small fraction of communicating node pairs) results in performance degradation if resources are allocated uniformly across node pairs. Correlation and burstiness are more directly related to the packet arrival process which is the main focus of our study. A correlated packet stream means that packet arrival instances are not independent from each other, while a bursty packet stream means that there is a high variability in the packet interarrival times. The general observation is that higher correlation leads to higher burstiness and thus longer delay.

As we take a close look at the packet stream correlation, we realize that the long-term correlation depends on the user and application behavior, while the short-term correlation is regulated by the interaction of protocol parameters and communication device parameters. In fact, there are many processes modulating a single packet arrival stream. Users arrive at and leave from a computer. Users may invoke applications while staying in the system. Applications can generate traffic of various types: interactive, file transfer, virtual memory paging, process swapping, image transfer, and future real-time applications like voice, video, and multi-media. Each traffic source type uses a specific protocol suite to transmit its packet stream. For example, interactive traffic usually goes through the character stream protocol like TCP, while file transfer traffic uses a block operation on a datagram protocol like RPC-on-UDP. Finally, the capacity of the host machines and the network determine how fast the packet streams can be injected into the network.

Motivated by the above observations, a new computer network packet arrival process, HAP (Hierarchical Arrival Process), capturing both long-term and short-term correlations is proposed in this paper. The objective of this HAP model is to replace Poisson and general MMPP models for more accurate performance analysis and simulation. A HAP has three levels – user, application, and message. A set of parameters describe the arrival and departure processes at each level. The model captures the fact that a packet arrival process is modulated by its upper-level arrival processes. HAP is a formal generalization of ON-OFF type traffic models [Hui88,Sch88,Kue89] where a burst can arrive only when the call it belongs to is active. In fact, the ON-OFF model is a 2-level HAP with only one message type. We can model a computer packet stream as a 3-level HAP while modeling packet streams from some other different environments as 2-level HAPs.

In this report, we first survey the previous work on traffic pattern studies in section 2. Section 3 characterizes traffic patterns in three aspects, namely statistical properties, transmission protocols, and formal analytical models. A simulation study on NFS traffic is also reported. The new model, HAP, for packet arrivals is presented in section 4. HAP is an object-oriented model which generalizes the ON-OFF type traffic models. We map HAP to MMPP and analyze its queueing behavior. Numerical results are reported in section 5. Simulation study is conducted to observe

HAP's short-term behavior. HAP is a class of traffic models which exhibit realistic and interesting performance patterns. Its dramatic behavior matches and explains the congestion in the real networks. From these results, section 6 examines some issues in broadband network control. Parts of the report are published in [LHDJ92,LTHG93,Lin93].

2 Previous Work

Most of the previous studies related to traffic patterns fall in two classes: network measurements and analysis, and variability modeling of packet arrival processes. The former conducts traffic monitoring and intensive analysis on the collected measurements to draw some observations, the latter uses point processes to model packet arrivals and studies the traffic burstiness and correlation.

2.1 Network Measurement and Analysis

Local area networks are the most common environments where traffic measurements are conducted due to the fact that LANs are broadcast media where traffic can be passively monitored without disrupting the network. Back in 1980, Shoch and Hupp [SH80] performed a measurement study of Ethernet traffic. They observed that Ethernet is lightly loaded and the host interfaces are the performance bottleneck. Recently, Boggs et al. [BMK88] explored the limit behavior of Ethernet using artificial traffic. Gusella's [Gus90] measurements on diskless workstation traffic observed 1Mbps bandwidth requirement between SUN-3s with a full workload generated by a user running processes in a window environment. This shows that traffic from a small set of workstations with NFS (Network File System) and window applications can easily saturate Ethernets. The performance bottleneck is shifting from the host to the network. Leland and Wilson [LE90][LW91] used a high-resolution monitor to study traffic behavior and its effect on delay and loss performance at different time scales. Measurement studies have been classified and reviewed by Pawlita [Paw88].

2.2 Variability Modeling of Packet Arrival Processes

Peak to mean ratio is not an accurate measure of traffic burstiness since a small spike of arrivals will make this measure very large although traffic is not so bursty most of the time. Siriram and Whitt [SW86] first used the index of dispersion for intervals to characterize the burstiness of the superposition of voice and data arrival processes. As a simple Poisson process is inadequate to model various types of traffic sources, Heffes and Lucantoni [HL86] applied a Markov Modulated Poisson Process (MMPP) to model the dependencies in voice and data sources. However, Leland and Wilson's [LW91] study showed that even a MMPP has its index of dispersion converge to a fixed value while the real traffic is still monotonically increasing.

Jain and Routhier [JR86] used the packet train model to describe the correlation within a packet stream. However, no suggestion was given in how to divide a packet stream into packet trains and what to do if it can be done. Song and Landweber [SL88] proposed to modify the protocol layers on hosts in order to optimize the transmission and receipt of packet trains. In their scheme, similar protocol processing functions are applied to all packet associated with a single bulk transfer. But no suggestion on how to choose the train size was given, either. Cruz [Cru87] defined a data stream as satisfying a "burstiness constraint" if the amount of data from the stream contained in any interval of time is less than a value that depends on the length of those intervals. He developed a calculus for obtaining bounds on delay based on input data streams that satisfy a burstiness constraint. Hui [Hui88,Hui90] characterized traffic states by three levels, namely, the packet level, the burst level, and the call level. The duration of a call may last from a few minutes to several hours. The burst length, on the other hand, is in the order of milliseconds. Cells in ATM are of fixed length with a transmission time of 2.83 μsecs at 150 Mbps. He proposed a multilayer bandwidth allocation algorithm.

3 Traffic Pattern Characterization

In this section, we study traffic patterns from three different viewpoints, namely, statistical properties, transmission protocols, and formal traffic models.

Packet traffic is bursty and non-uniformly distributed over the geographical coverage. Its statistical properties are the indicators showing how strong its patterns are. Statistically speaking, we can have several measures for packet traffic. Each measure represents one property. The measures we study include locality, correlation, burstiness, and predictability. These properties greatly influence the network performance. Traffic control and resource management schemes should carefully consider these properties.

Transmission protocols are the actual building blocks of traffic patterns. Different types of applications, transmitted by different protocols, exhibit different traffic statistics. It is the transmission protocol that determines how packets are injected into the network. Packets generated from different applications are transmitted through different protocols. We classify applications to interactive, file transfer, image, voice, and video transfer. Packet trace generated from the transmission protocols are analyzed.

For real-time control, formal traffic models are necessary for bandwidth allocation, admission control, and congestion control problems. These models describe and predict how traffic sources behave and are used to compute required bandwidth, admissible connections, and other parameters. We survey the commonly used formal analytical models.

3.1 Statistical Properties: The Traffic Characteristics

Different types of applications and protocols generate different workloads. In turn, different workload characteristics result in different performance behaviors. Data traffic is very different from voice traffic. It still requires very many users to flood a voice network but a small set of computers are capable of severely taxing the resources of a data network. Also, the complexity of traffic dynamics is much higher in a data network than in a voice network. Thus, characterizing traffic patterns for different types of applications is important for the performance studies. In an effort to better understand traffic statistics, we characterize traffic patterns in four aspects, namely, locality, correlation, burstiness, and predictability.

3.1.1 Locality

More than 80% of the traffic is contributed by less than 20% of communicating pairs, ie. traffic is not uniformly distributed. This observation implies that uniform resource assignment will cause performance degradation and fairness problems. It is essential to capture this distribution in order to optimize the network configuration. A good indicator for the strength of locality is the pair $(C\%, P\%)$ where $C\%$ of traffic is contributed by $P\%$ of communicating pairs. A typical degree of locality is shown in Figure 1.

C %	P %
10	0.9
20	1.3
30	1.9
40	2.7
50	3.6
60	4.7
70	6.8
80	9.7
90	16.5
95	28.4
98	41.2

Figure 1: Traffic Locality

3.1.2 Correlation

It is well understood that packet arrival process is not any Poisson-based process. Poisson is a memoryless process where the arrival time for the next packet is independent from how long the previous packet has arrived, ie. there is no correlation between these two arrivals. However, the

real packet arrival process is not memoryless. Packets tend to arrive in bursts within which packets are highly correlated. A burst hierarchy even exists with various levels. These correlations are determined by protocols, applications, and users where short-term correlation is determined by protocols and long-term correlation is determined by the user behaviors. The degree of correlation within an observed packet stream depends on the above three factors and the multiplexing effect. Multiplexing two packet streams will reduce the degree of correlation within the resulting stream.

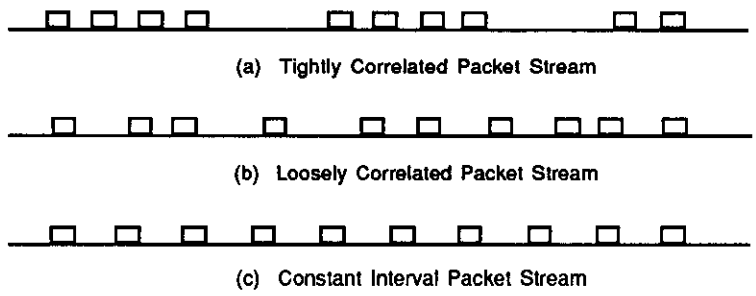

Figure 2: Packet Streams

Figure 2 shows three types of packet streams where (a) is a tightly correlated stream possibly generated by an application, (b) is a stream with little or no correlation, possibly generated by two or more applications, and (c) is a constant stream which is unlikely to happen in data networks. Note, (b) is unlikely to be seen when the load level is low and medium. When the load is very high, the chance that two streams from two applications will interleave increase. In fact, (b) is the kind of stream that a Poisson process generates.

3.1.3 Burstiness

If we consider the burstiness in terms of different time scales, different degrees of burstiness exist on different time scales, from milliseconds to thousands of seconds. It is common to have peak to mean ratio over 250 for 1-millisecond intervals and 10 for 1-second intervals. Moreover, highly correlated stream tend to generate more bursty patterns and in turn have inferior delay and loss performance. In figure 3, (a) is a highly correlated stream and has the highest peak to mean ratio over the other two. (c) has a value of peak to mean ratio of one. The performance ordering for these three is (c), (b), and (a), although they all have the same load level. That is, the results of performance studies using constant or Poisson arrival process will be too optimistic.

Three measures are usually used as indicators of burstiness degree: peak to mean ratio, coefficient of variation, and index of dispersion. Note that, in some cases, peak to mean ratio is not a good indicator because one single spike can make the ratio large although the stream is not bursty

most of the time. Coefficient of variation for arrivals in a given interval size is the ratio of the standard deviation in the observed number of arrivals in the interval of that size to the observed mean number of arrivals in the interval of that size, ie. $\sigma_N/\overline{N}$ where N is the number of arrivals in that interval. Similarly, coefficient of variation for interarrival time in a given interval size is $\sigma_X/\overline{X}$ where X is the interarrival time in the interval of that size [CL66]. A Poisson process has $\sigma_N/\overline{N}$ converge to zero when the interval size is over seconds while actual traffic has a value of one at the interval size of hundreds of seconds [LW91]. Index of dispersion for arrivals and interarrival time in intervals of a given size are defined as $Var(N)/\overline{N}$ and $Var(X)/\overline{X}$, respectively. A Poisson process has dispersion of one all the time and batch Poisson converges to about 6 when the interval size is over seconds, while actual traffic has monotonically increasing dispersion throughout a time span of 6 orders of magnitude [LW91].

3.1.4 Predictability

In most of the networks, a temporal cycle exists in the traffic distribution. A cycle hierarchy may even exist. For example, one week is a cycle and one day is a subcycle within that cycle. Being able to keep track of the distribution cycle will enable dynamic configuration management which tunes the network dynamically according to the cycle. Index of dispersion for arrivals does converge to an upper bound when the size of interval is on the order of days. [LW91] Study is needed to see how well this measure can detect a cycle and the strength of the cycle.

A good estimation of traffic demand can improve the resource allocation strategy. A dynamic resource pre-allocation scheme can be used if the system is able to capture the repetitive cycle behavior and predict for the next cycle. However, traffic in many networks is difficult to estimate. Based on the Law of Large Number, the predictability increases as the size of the user population increases. Since a small set of users is supported in a LAN, LAN traffic is very bursty with high degree of fluctuation. If we take a snapshot of a LAN, the number of current communicating pairs is very small. Thus, LAN traffic can only be predicted over a long time scale, on the order of days.

The above characterization is done at the numerical level. It helps us in clarifying what kinds of traffic patterns may exist in a network and their strength. It also suggests a way to model traffic patterns and reproduce them. In fact, a traffic model is proposed in next section to reproduce traffic for performance studies. However, traffic patterns in logical forms are still needed for our network management schemes.

3.2 Transmission Protocols: The Building Blocks

In this section, we study the packet traces generated from various types of applications. We look at the applications running on the TCP/IP protocol stack.

3.2.1 Protocol Hierarchy

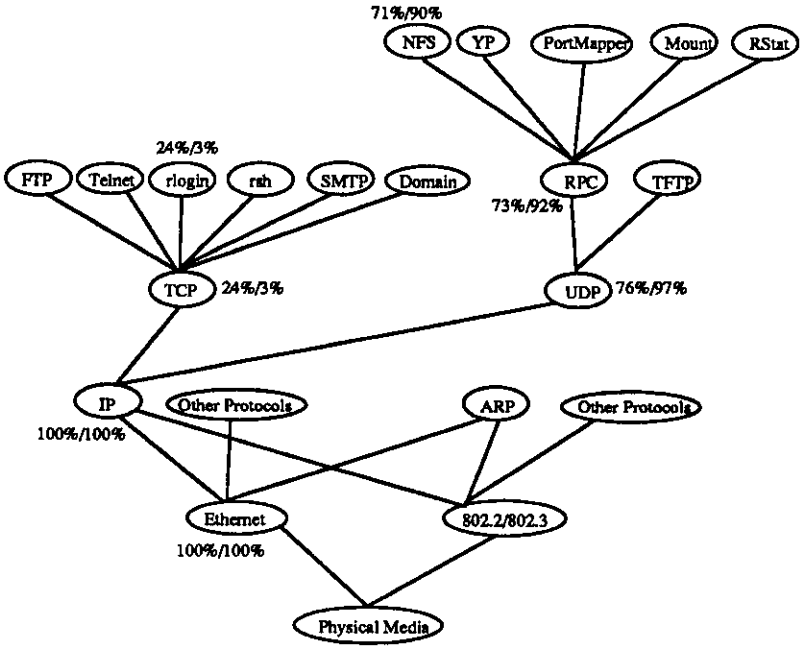

Figure 3: TCP/IP Protocol Stack

Figure 3 decompose packet traffic into the TCP/IP protocol hierarchy. A measurement of a 3-hour period is taken from an Ethernet. It shows that NFS (Network File System) and rlogin (remote login) are two major traffic contributors. NFS contributes 71% of transmitted packets and 90% of transmitted bytes, while rlogin contributes 24% of packets and 3% of bytes. NFS is a file transfer application and rlogin is an interactive application. This protocol stack does not show protocols for graphics, voice, and video applications. Graphics applications running on X window use RPC (Remote Procedure Call) directly. Protocols supporting voice and video applications are in the experimental phase [TP91,Mil91,Nic90]. For these real-time applications, guaranteed performance on the packet switching networks is the objective that requires further researches and experiments. It is one of the most active research areas.

3.2.2 Interactive Applications

Interactive applications using rlogin are usually terminal sessions involving character echoplexing by the remote host and the display of ASCII test. As shown in Figure 3, rlogin is supported by TCP which is a byte-stream oriented protocol. Pointers to positions in the byte-stream are acknowledged individually. As a result, a series of exchange of packet pairs between source and destination continues. As each packet carries only one-byte data, these numerous packets only carry a small fraction of total bytes in the network. The interval between two consecutive packets from the source station depends on the interval between keystrokes, which is quite dependent on the human user. Thus, the resulting trace is not as foreseeable as the other protocols like file transfer protocols. Figure 4 is a typical trace for a rlogin session.

Figure 4: Packet Trace of rlogin Session

3.2.3 File Transfer Applications

FTP (File Transfer Protocol) and NFS are two major file transfer protocols in the TCP/IP protocol stack [KW89,Ste90,San91]. FTP is used for human-initiated file transfer between hosts. It contributes only a very few fraction of transmitted packets. NFS plays a very important role in current distributed systems and contributes most traffic as we can see in Figure 3. It is designed to integrate file systems in a distributed computing system and make them appear as one file system. That is, it supports transparency of file locations.

We now illustrate how NFS works. If the file requested by a file access is not on the local host, this file access request is then sent from the local host, NFS client, to the remote host, NFS server. As shown in Figure 3, NFS invokes RPC (Remote Procedure Call) to transfer the requests and responses between clients and servers. RPC, in turn, uses UDP which uses the IP protocol to transfer packets [Ste90]. When a read file request is generated at the client according to the distributions of read request interarrival time and request size, several NFS "lookup" requests are used to traverse the remote directory and a NFS "getattr" request to get the file location and attributes. Several block operations are then performed to retrieve the file one block at a time. The procedure to handle a write file request is similar except that blocks of data are transferred from a client to its server.

The block size can range from 8K-byte to 512-byte which is adjustable according to a window

flow control protocol. A NFS request/response is encapsulated by a RPC header and then by a UDP header. The limitation of UDP packet length is 64K-byte. So basically a block can fit in one UDP packet and then one IP packet. However, the maximum frame size on Ethernets is 1518-bytes. Thus, a block of 8K-bytes is then fragmented into 6 Ethernet frames and a block of 2K-bytes is fragmented into 2 Ethernet frames. Figure 5 illustrates these packet length calculations. Figure 6 represents two packet traces, one for NFS read and one for NFS write. Unlike rlogin, the interval between two packets is quite dependent on the protocol behavior and the transmission devices.

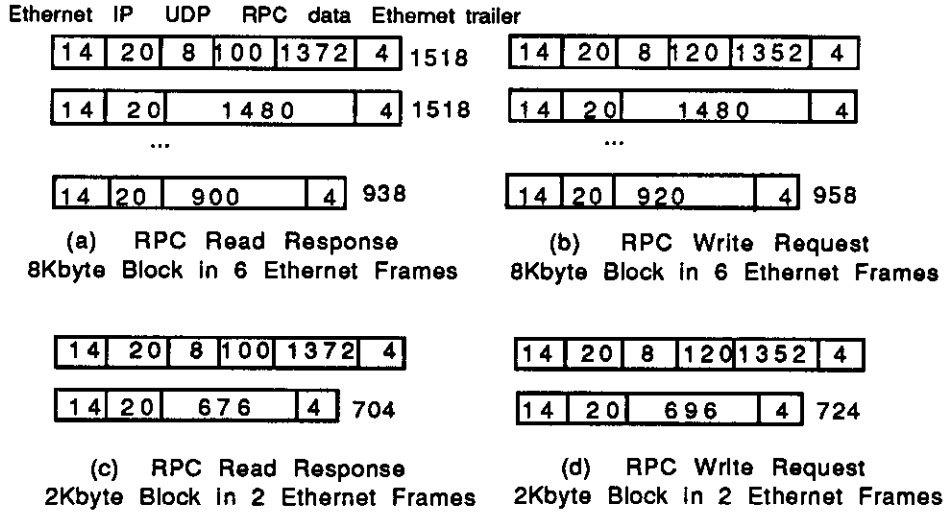

Figure 5: NFS Packet Length on Ethernets

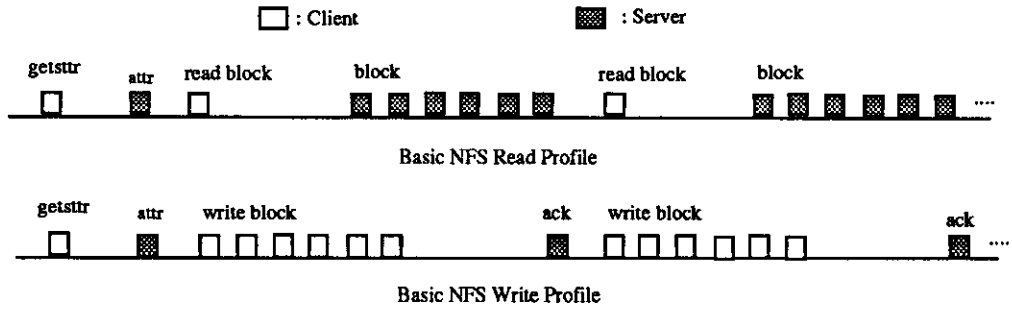

Figure 6: Packet Trace of NFS Session

FTP uses TCP and avoids fragmentation at IP layer by sending small amount of data in each TCP packet. Thus, FTP generates packet streams which are less bursty than NFS's.

3.2.4 Image/Voice/Video Applications

Graphics applications are similar to file transfer applications, but now the amount of data to be transferred is fixed. For example, an image produces a burst of 10 Mbits which are sent in 10,000 packets each of 1000 bits. In TCP/IP protocol stack, X window image is transferred by TCP which supports byte-stream communication. Thus, its packet trace is similar to FTP's, except that an image burst has a fixed length. The well-known X protocol is designed to communicate all the information necessary to operate a window system over a single asynchronous bi-directional stream of 8-bit byte [KW89]. X can work with any lower layer protocol as long as it is bidirectional and delivers bytes in sequence and unduplicated between a server and a client process.

Although voice and video protocols are in experimental phase, we can foresee their packet traces by examining their traffic generating processes. Voice calls with silent removal communicate in alternating states of idle and burst after the call is set up. Each talk spurk generate a burst. If these variable-length bursts are transferred by TCP, the packet trace is similar to FTP's.

Compressed full motion video has a continuously changing bit rate which depends on the amount of motion and the degree of granularity of the picture. Usually, a differential, VBR (Variable Bit Rate) [VPV88] encoding scheme is used. For 30 picture frames per second and 250,000 pixels per frame, its has a peak bit rate of 10 Mbps and an average bit rate of 5 Mbps [Mag87]. Unlike voice with silence removal and still picture graphics, there is no alternating states between idle and burst periods. Thus, its packet trace is quite different from the other applications. Voice and image applications are more bursty than variable rate video applications, although a single video connection has a much higher average bit rate.

3.3 A Simulation Study of NFS Traffic

After examining packet traffic's statistical properties and its transmission protocols, we would like to study the statistical properties of individual protocol's packet trace. As surveyed in section 1, there are many measurement studies on packet traces. Here we resort to simulation study to have a controllable environment. We simulate NFS traffic to evaluate its burstiness and SMDS (Switched Multi-megabit Data Service) [BCR89] access path to study how much NFS traffic can be supported without violating the performance constraints.

3.3.1 SMDS for LAN Interconnection

Comparing with high-speed networking in the intra-premises environment, wide-area, inter-premises networks like Internet are contrasts and bottlenecks to extend distributed systems to a inter-

premises domain. Under this bandwidth constraint, only primitive applications like remote file transfer (eg. "ftp", "smtp") and remote login (eg. "telnet", "rlogin") can be supported with acceptable performance. Thus, the bottleneck between local area networks and wide area networks has to be eliminated. Current solution for high-speed LAN interconnection over a metropolitan area is a private T1 (1.544 Mbps) network. Customers lease T1 links to have a point-to-point connection between two separately located LANs. Extra lines have to be leased if inter-enterprise data sharing is required.

SMDS is a public packet-switched service for high-speed data sharing. A subscriber accesses SMDS via a DS1 or DS3 link. DS1 is a 1.544 Mbps link with approximately 1.173 Mbps maximum effective throughput after the physical framing and SIP protocol overhead has been removed, while DS3 is a 45 Mbps link with approximately 34 Mbps maximum effective throughput available to the subscriber. A logical private network can be created by the source address and destination address screening features. Inter-enterprise data sharing is achieved by adjusting the screening features. Unlike the T1 solution, the switching tasks are shifted to the SMDS carrier where a packet can be routed in an optimal path. The service access point protocol is SIP(SMDS Interface Protocol) which is actually DQDB as specified in the IEEE 802.6 standard.

SMDS is targeted to achieve high performance LAN interconnection. A list of performance requirements are documented in [BCR89]. Among them, delay criteria for individually addressed packets are, for 95% of all packets delivered, the delay from first-bit-out at one SNI(Subscriber Network Interface) to last-bit-in at another SNI should be less than 20 msec, 80 msec, and 140 msec for DS3/DS3, DS1/DS3, and DS1/DS1 configurations, respectively. DS3/DS3 here means two CPEs on both sides of SMDS are using DS3-based access paths. Loss ratio for individually addressed packets should be less than 0.01%. However, this connectionless packet-switched service is subject to congestion and performance degradation just like any other connectionless packet-switched networking service, especially when the network is designed to interconnect LANs, where traffic is extremely bursty. Some applications can tolerate this degraded delay and loss performance, while the others can not. Similarly, some applications tend to generate very bursty traffic patterns and congest the network, while the others do not. Thus, analysis and experiments are desired to foresee the applications that are suitable to run across SMDS and evaluate SMDS's ability to support bursty applications.

3.3.2 NFS Traffic Simulator

We generate NFS packet traces according to the profiles in Figure 6. In order to calibrate the profiles, data was gathered from two LANs connected by DS1 access links to a prototype SMDS switch and is shown in Figure 6 as a basic NFS read/write profile. [LHDJ92] This profile, along with a similar NFS/SMDS write profile, is combined with application configurations (the number and

location of mainframes, diskless workstations, and diskful workstations on the network). Estimated NFS request distributions (of inter-arrival time and file size) for each type of station are fed into the NFS traffic simulator. For clients on this site, the packets from the clients to their remote servers are extracted into a packet trace. Similarly, for servers on this site, the packets from the servers to their clients are dumped into a trace. These packet traces produced by different stations are aggregated and sorted by packet timestamp. A simple CSMA/CD backoff is applied to adjust the packet timestamps when Ethernet access contention occurs.

A single queue is used to simulate a SMDS access path which is a DS1 or DS3 link. The buffer size throughout this experiment is assumed to be fixed at 250 packets. The access delay is defined to be from the time the packet arrives at the Ethernet transmitter queue to the time its transmission into SMDS is finished. It includes possible Ethernet queueing, backoff, and transmission delay, DS1/DS3 queueing and transmission delay. For the purpose of comparing DS1 and DS3 performance, we have set the delay and loss requirements for the access link to be the same as the SMDS network switching delay and loss specified in [BCR89].

3.3.3 Results: Burstiness and Performance

The reproduced packet trace has embedded correlations within the packet stream. However, another characteristic to verify is how bursty the reproduced traffic is. We need to compare the reproduction of our model with real traffic and evaluate the perspective of this approach.

Unlike file transfer protocols (e.g. "ftp"), NFS is designed to integrate file systems in a distributed computing system and make them appear as one file system. As a result, a tighter performance requirement is desired. Also, due to its block operation, NFS traffic contains highly correlated packet streams and has strong burstiness. The results here should provide SMDS subscribers information about how much NFS traffic, with its high performance requirement, tight correlation, and strong burstiness, can be supported using DS1 and DS3 access links.

Burstiness Analysis

In this analysis, a DS1 link with the configuration of 21 active stations (3 mainframes, 6 diskless workstations, and 12 diskful workstation) is studied. The reproduced traffic has 5.91% Ethernet utilization, 50.41% DS1 utilization, and average values of 96 packets/sec, 73913 bytes/sec, 10358 μs inter-arrival time, 28.73 queue length upon arrival, 43236 μs packet delay, 8.19% delay violation, and 3.47% packet loss. Each packet trace simulates NFS traffic for a duration of 1200,000,000 μs (20 minutes). For a burstiness analysis, we need to compute the values at different time scales. Peak/Mean ratio and coefficient of variance of inter-arrival time for different interval sizes are plotted in Figure 7. (Note that some axes are logarithmic.) This shows that the reproduced traffic

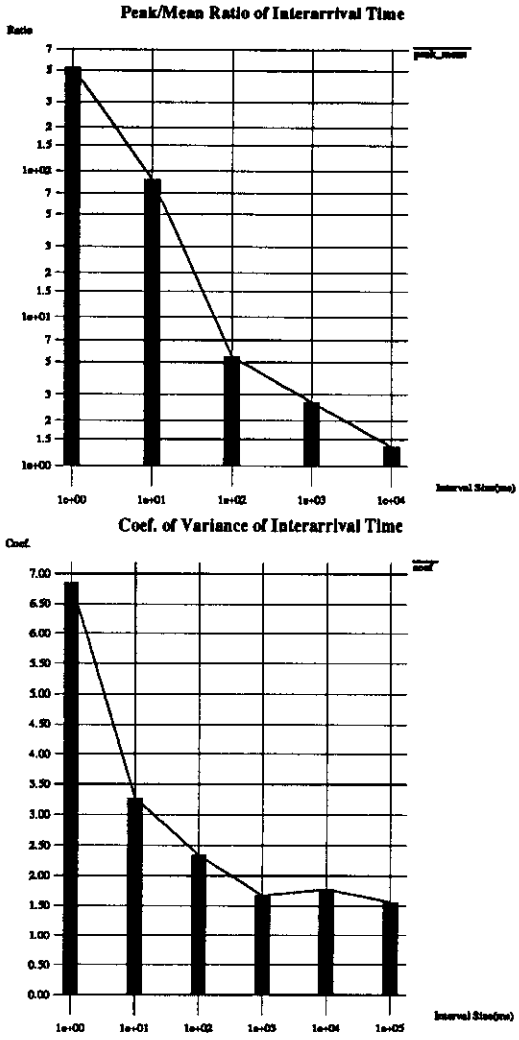

Figure 7: Burstiness of Packet Interarrival Time

has high degree of burstiness in time scales ranging from milliseconds to 10 seconds, compared to Poisson and batch Poisson which converge in 1-sec intervals [LW91].

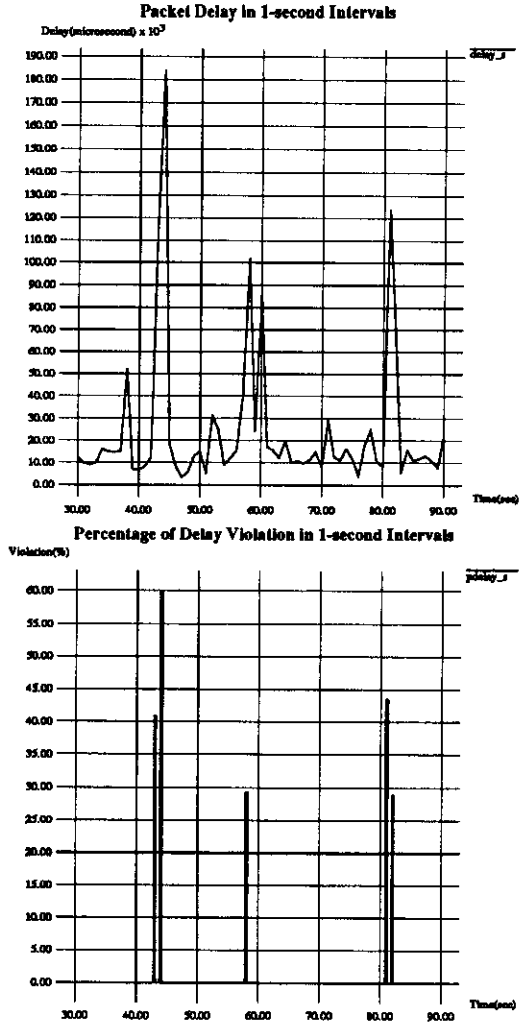

Figure 8: Average Delay and Delay Violation Ratio

Figure 8 displays packet delay and delay violation ratio averaged over 1-sec intervals for part of a packet trace representing a 20 minute simulation. Between the 30th second and 90th second, only one spike at about the 44th second has average delay over the requirement, 140 ms. However, there are 5 intervals with individual packet delay violation ratios over 25%. This indicates that average delay, as a performance measure, is not as precise as delay violation ratio. Furthermore, it was found that during the simulated 20 minutes there are 1-ms, 10-ms, 100-ms, and 1-sec intervals where all the packets violate the delay requirement. That is, we can still find for the busiest 1-sec intervals during the 20 minutes where all the packets have delay over 140 ms. Similarly for packet loss, it is shown in Figure 9 that many 1-sec intervals have 100% packet loss.

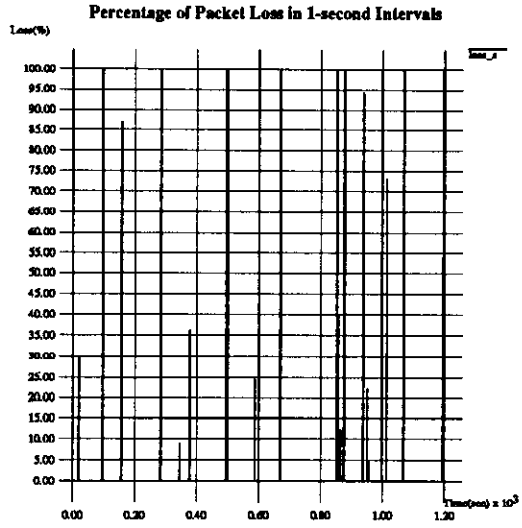

Figure 9: Peak Loss Ratio

This burstiness analysis has two implications:

- The performance violation ratio is a more precise measure than the averaged value when a QOS (Quality Of Service) is to be specified.
- Data traffic is extremely bursty, and long averaging intervals can not capture its short-term behavior and performance.

Combining the above observations, Figure 10 represents an example of how alternative *interval performance requirements* for delay and loss might be specified as a function of the averaging interval size. Smaller interval sizes are given wider range to accommodate stronger burstiness.

Interval Size	Delay (95%)	Loss
1-hour	140 ms	0.01%
1-min	180 ms	0.5%
1-sec	250 ms	10%

Figure 10: Interval Performance Criteria

Performance Analysis

Figure 11 summarizes the simulation results for DS1 and DS3 access lines. The entry DS1(2,4,8), for example, means both source site and destination site are using DS1 access paths and there are 14 active stations (2 mainframes, 4 diskless workstations, 8 diskful workstation) on both sites. DS1 can support NFS traffic of 4.05% Ethernet utilization, which is 34.54% DS1 utilization, with overall

delay violation ratio at 1.29% and a loss ratio at 0.21%, while DS3 can support 41.54% Ethernet utilization, which is 11.81% DS3 utilization, with delay violation ratio at 0.839% and zero packet loss.

	%	%	pkt	byte	micro-s	pkt	micro-s	%	%
Configuration	Ether_util	Link_util	Pkt/s	Byte/s	Inter_pkt	Queuelen	Delay	Violation	Loss
DS1(1,2,4)	2.17	18.48	34	27099	28990	0.89	10468	0	0
DS1(2,4,8)	4.05	34.54	65	50641	15308	6.79	17429	1.29	0.21
DS1(3,6,12)	5.91	50.41	96	73913	10358	28.73	43236	8.19	3.47
DS1(4,8,16)	7.99	68.09	127	99829	7818	82.01	145822	28.84	20.11
DS3(1,2,4)	2.18	0.62	22	27207	43745	0.018	1247	0	0
DS3(2,4,8)	4.02	1.14	42	50225	23631	0.035	1255	0	0
DS3(3,6,12)	6.35	1.81	66	79415	14989	0.058	1302	0	0
DS3(4,8,16)	8.53	2.42	90	106573	11112	0.081	1338	0	0
DS3(10,20,40)	20.77	5.90	218	259623	4574	0.188	1663	0.0038	0
DS3(20,40,80)	41.54	11.81	438	519277	2286	0.338	3010	0.839	0
DS3(30,60,120)	62.62	17.79	659	782729	1520	0.461	9123	10.986	0

Figure 11: Performance of DS1/DS3 for Various Configurations

3.4 Classic Traffic Models: The Computational Bases

For traffic control and resource management, we need models that describe and predict traffic source behaviors. Queueing analysis for bandwidth allocation and admission control is drawn from these models. In this section, we survey the commonly used models for performance analysis.

3.4.1 Constant-Bit-Rate Model

In this model, packets are periodically transmitted according to its average bit rate. The average bit rate is the same as the peak bit rate. Therefore, the peak to mean ratio is one and the index of dispersion is zero.

Eckberg [Eck79] derived an algorithm for computing the exact delay distribution of a single server queue with a number of sources with the same periodic arrival process and deterministic service times. Virtamo and Roberts [VR89] derived bounds for the buffer dimensioning of an ATM multiplexer with a superposition of a large number of periodic sources as its input.

3.4.2 Poisson Model

This classic model has been used extensively due to its simplicity in analytical derivations. However, this model has its limitation. Packets generated from Poisson processes have exponentially distributed interarrival times. That is, packet arrivals are independent from each others, which is not true in the real world. The index of dispersion of Poisson processes is always one, while the real traffic has a monotonically increasing index of dispersion [FL91].

Schoute [Sch88] compared the permitted loading of five simple queueing systems and derive simple decision rules for admission control. The permitted loading is as follows: M/M/1 (0.539), D/M/1 (0.746), M/D/1 (0.773), U/D/1 (0.908), D/D/1 (0.999).

3.4.3 ON-OFF Model

An ON-OFF source alternates between two states: ON and OFF. When the source is in the ON state, it generate packets at the peak rate, B_p. When it is in the OFF state, it keeps silent. In Figure 12, we assume that ON and OFF periods are exponentially distributed with means T and S, respectively. We have mean bit rate

$$B_m = \frac{B_p \times T}{T + S}.$$

If we define the peak to mean bit rate ratio, B_p/B_m, as b, we know the mean OFF period

$$S = T(b - 1).$$

We can model an ON-OFF source as a two state Markov chain as shown in Figure 13.

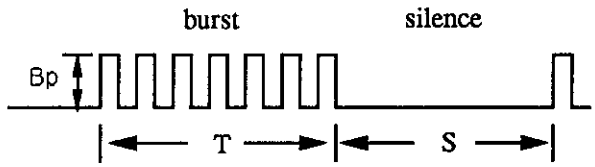

Figure 12: ON-OFF Traffic

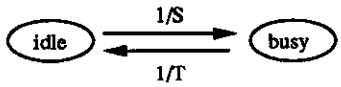

Figure 13: Markov Chain for an ON-OFF Traffic Source

Voice sources with silence removal, graphics sources, and typical data sources can be modeled as ON-OFF sources. ON-OFF sources tend to generate very bursty traffic. Furthermore, longer ON periods combined with a higher peak bit rate result in a even higher degree of burstiness.

3.4.4 Variable-Bit-Rate Model

Due to the differential encoding scheme [VPV88], the bit rate of a video source changes on a frame by frame basis between a minimum and a maximum bit rate. Maglaris et al [Mag89] proposed to model a VBR video source by a Markov chain and suggested that the superposition of identical ON-OFF sources may also be used to represent the VBR video source.

The approach to model VBR sources by Markov chains has drawn a significant amount of research activities. MMPP (Markov Modulated Poisson Process), which is a class of VBR models, has been applied to approximate both voice and video sources [DL85,DL86,HL86]. Figure 14 is a typical MMPP model where the bit rate at state i is λ_i.

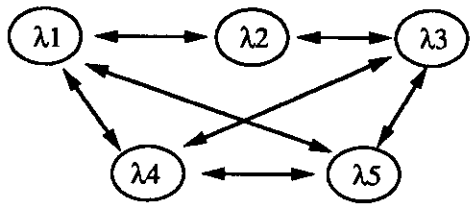

Figure 14: Markov Modulated Poisson Process

In next section, we propose a new traffic model which captures the correlation among packet arrivals. This model is shown to be a special class of MMPP.

4 Modeling Traffic Arrivals as HAP

Section 4.1 formally presents the HAP model and its parameters. In section 4.2, we show that HAPs are multi-dimension, infinite-state MMPPs. In section 4.3 and 4.4, three solutions are used to analyze HAP's queueing performance. HAP simulator is briefed in section 4.5.

4.1 HAP: Hierarchical Arrival Process

4.1.1 HAP with User, Application, and Message Levels

A HAP is a message arrival process at a network node. At a node, users arrive and depart at a specified arrival rate and departure rate. That is, users arrive according to an interarrival time distribution and stay in the system for a duration that also has a specified distribution. During his/her presence in the system, the user may invoke several types of applications. Each type of application is invoked at a specified rate, and remains active for an interval which has a specified distribution. Again, during the active interval, the application generates several types of messages, at different rates and with different message size distributions. Messages may be fragmented into packets or cells in the transmission network. This depends on the transmission protocols.

There may be many users in the system. And each user is invoking several applications which belong to one application type or several application types. Again, each application has generated none or several messages which belong to one message type or several message types. It is also possible, in this model, that a user has departed but the application this user invoked may be still active. This may happen, in a computer, when the user issues an application as a background process and leaves.

To formally model a HAP, we use Figure 15 as a concise representation for the HAP model. The *HAP model parameters* are defined below. Suppose that the reciprocal of each parameter is the mean of its distribution.

λ : user interarrival time distribution
μ : user service time distribution
λ_i : interarrival time distribution for application type i
(Application arrivals are enabled only during user service time)
μ_i : service time distribution for application type i
λ_{ij} : interarrival time distribution for message type j of application type i
(again, messages are generated during application life span)
μ_{ij} : service time distribution for message type j of application type i

where $i = 1 \ldots l$ and $j = 1 \ldots m_i$. That is, a user can simultaneously invoke up to l different types of applications. Each application type i can generate m_i types of messages.

Figure 15 is a *containment hierarchy* of the *object class* "user" in the terminology of the object-oriented methodology. A user arrival produces an *object instance* of the object class "user". Several instances of "user" can coexist at a network node. Each "user" instance, in turn, forks its own child instances which belong to the classes "application i, $i = 1 \ldots l$". Each "application i, $i =$

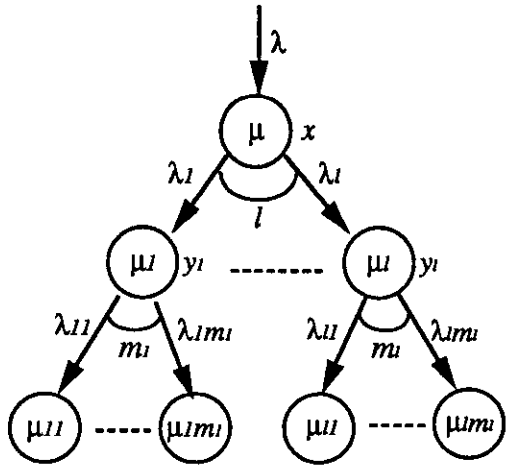

Figure 15: The HAP model

$1 .. l$" instance, belonging to a "user" instance, autonomously forks its own child instances which are of the classes "message ij, $i = 1 .. l$ and $j = 1 .. m_i$". Figure 16 is an example of two instances of the "user" class where two nodes overlap if they belong to the same class/type.

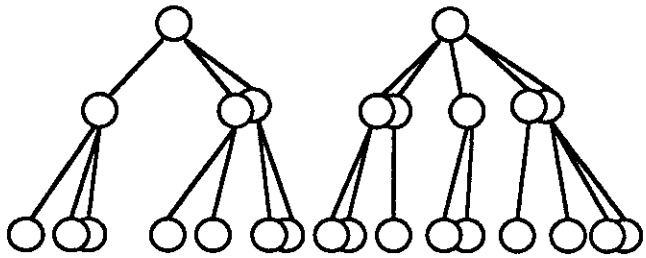

Figure 16: Two user instances of HAP class

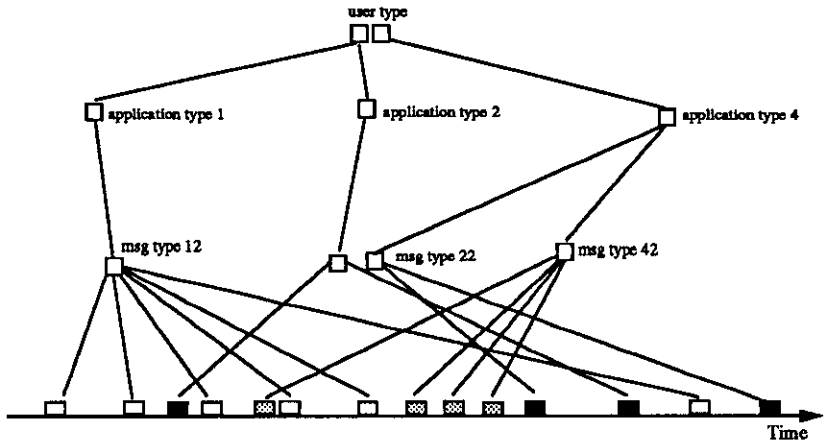

Figure 17: Decomposing a message stream into a HAP

If a HAP on a network node dumps its messages into the network, this message stream contains complex correlations between messages. In fact, we can decompose a message stream into a *burst*

hierarchy as shown in Figure 17 where message instances of the same color are of the same message type.

4.1.2 HAP with Client-Server Interaction

In a computer network, traffic occurs due to the interactions among traffic sources. These interactions can be studied using the client-server interaction model where requests generated by the clients are sent over the network to the servers. Upon receiving the requests, the servers may send back the responses to the clients. These responses may, in turn, trigger next requests from the clients. That is, a sequence of requests and responses between a client and a server may be generated once the original request is issued by the client. Let us take "rlogin" as an example. A user running "rlogin" application on a node issues a command, which is considered as a request, to the remote node. The result, which is considered as a response, from the remote node may trigger the user to issue another command in this "rlogin" application.

A message generated from a HAP process is actually a request from a client on a network node to a server on another network node. To incorporate the client-server interactions into HAP, we modify the original HAP model to the HAP-CS (Client-Server) model in Figure 18 with the additions of the parameters,

μ_{ij}^{c} : service time distribution for request message type j of application type i
μ_{ij}^{s} : service time distribution for response message type j of application type i
p_{ij}^{c} : probability that request message type j of application type i will trigger a response
p_{ij}^{s} : probability that response message type j of application type i will trigger another request

where $i = 1 \ldots l$ and $j = 1 \ldots m_i$.

4.1.3 Example HAPs

Figure 19(a) is an example HAP with 4 application types and 5 message types. Message types A, B, C, D, E represent interactive, file transfer, image transfer, voice call, and compressed video, respectively. They may use different protocols for transmission. For example, interactive data usually uses TCP on top of IP, transparent file transfer by NFS/RPC/UDP/IP, image by RPC/UDP/IP, voice and video by some new protocols. Note that a message is usually fragmented into many packets or cells during transmission.

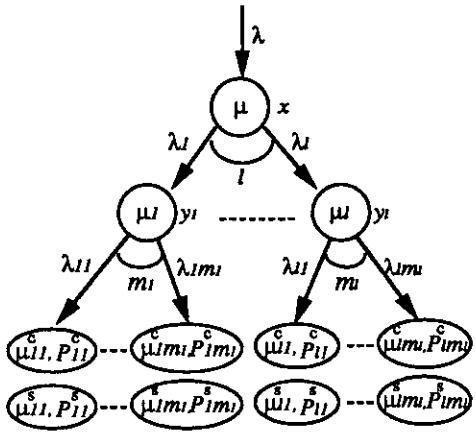

Figure 18: HAP with client-server interaction

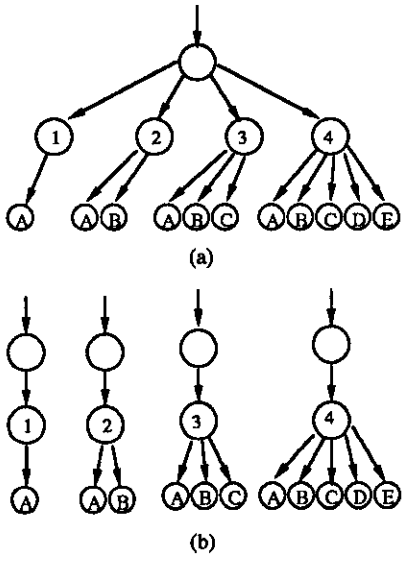

Figure 19: Example HAPs

Application type 1 is a common programming environment where users read or edit files, check directories, etc. Type 2 is possibly a database query environment where only short interactive data is transferred. Type 3 is a graphics-intensive application where fixed size images are transferred. Type 4 is a multi-media application where all message types are possible. The example assumes that all users are homogeneous, while in reality we may have several types of users. In Figure 19(b), we divide the original HAP into four HAPs representing four heterogeneous user types.

4.2 Mapping HAP to MMPP

MMPP is a doubly stochastic Poisson process where the arrival rate is modulated by the state of its embedded continuous-time Markov chain. Early studies on the queueing behavior with this kind of arrival process with algorithmic solutions can be found in [Neu78A,Neu78B,Neu81]. Recently, 2-state MMPP has been used extensively to approximate a superposition of various arrival processes like data, voice, and video traffic [Hef80,HL86,SW86]. A complicated algorithmic solution is applied to a 2-state-MMPP/G/1 queue [HL86]. However, this 2-state MMPP is only an approximate traffic model for the analysis purpose. Besides, a general MMPP is not an appropriate model for computer network traffic.

As we can see, HAP is a class of MMPP whose arrival rate is determined by the state of an $(l+1)$-dimension infinite-state Markov chain where transitions only happen between neighboring states. The state variable of this Markov chain is represented as

$$(x, y_1, y_2, ..., y_l)$$

where x is the number of user instances and y_i is the number of instances of application type i, $i = 1..l$. The arrival process in this state is Poisson with rate $\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}$. Figure 20 shows the Markov chain for this MMPP. We have an $(l+2)$-dimension, infinite-state Markov chain including one dimension for z where z is the number of message instances in the system when we assume $\mu_{ij} = \mu''$ and feed this HAP into an exponential service queue. If we assume $\lambda_i = \lambda'$, $\lambda_{ij} = \lambda''$, $\mu_i = \mu'$, and $m_i = m$, this MMPP is reduced to a 2-dimension one, as illustrated in Figure 21, with state variable (x, y) where y is the total number of instances of all application types. The arrival rate in this state is $ym\lambda''$. If we feed this simplified HAP into an exponential service queue, we get a 3-dimension Markov chain with state variable (x, y, z) where z is the number of message instances in the system. However, there is no closed-form solution for this Markov chain. Even if we use Z-transform to solve a 2-level HAP, we still only get a partially differential Z-transform equation, due to the fact that z depends on y (and also y depends on x). For HAP, a special class of MMPP, we resort to the G/M/1 approach as an approximation, in Solution 1 and 2, to solve HAP/M/1, which is presented in the following sections.

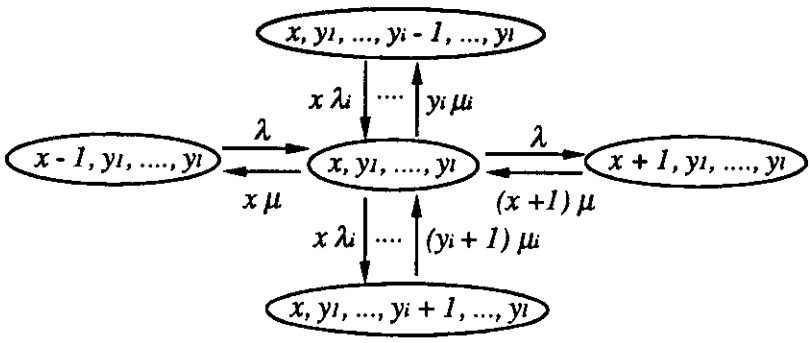

Figure 20: Embedded Markov chain for HAP

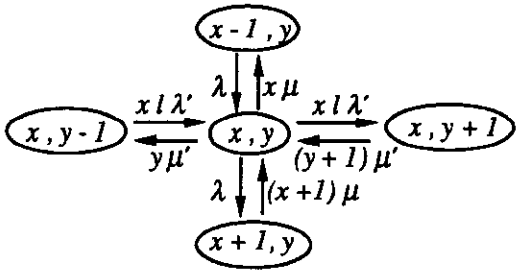

Figure 21: Simplified embedded Markov chain for HAP

4.3 Queueing Performance Analysis

After presenting HAP and HAP-CS models, we are ready to do some analysis. We first map a HAP to a $(l+1)$-dimension, infinite-state MMPP. As there is no closed form solution for queueing performance under MMPP traffic, we resort to three algorithmic solutions. Solution 0 is a brute-force iterative approach, while Solution 1 and 2 are approximate solutions. These approximations are shown to be good under three constraints on parameters. In Solution 2, we obtain a closed-form formula for message interarrival time distribution, which speeds up the computation dramatically. We further study the characteristics of superposition of multiple HAPs using this formula. To keep the analysis tractable, we assume, unless stated, that all the HAP model parameters are exponentially distributed for the rest of this paper. In the analysis part of this paper, we analyze the interarrival and queueing performance at the message level, not at the packet level, because the pattern of packet streams really depends on the adopted transmission protocols.

Solution 0: brute force

We intend to iteratively compute the steady-state probabilities of the $(l+2)$-dimension Markov chain and obtain $\overline{z}$, mean number of messages in the system, and $\overline{\lambda}$, mean arrival rate. From $\overline{z}$ and $\overline{\lambda}$, we can compute T, mean message delay, using Little's result.

From the $(l+2)$-dimension Markov chain, we can write down the state transition equation as

$$P(x, y_1, ..., y_l, z)[\lambda + x\mu + \sum_{i=1}^{l}(x\lambda_i + y_i\mu_i) + \mu'' + \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}]$$

$$= \lambda P(x-1, y_1, ..., y_l, z) + (x+1)\mu P(x+1, y_1, ..., y_l, z)$$

$$+ \sum_{i=1}^{l}[x\lambda_i P(x, y_1, .., y_i - 1, .., y_l, z) + (y_i + 1)\mu_i P(x, y_1, .., y_i + 1, .., y_l, z)]$$

$$+ \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij} P(x, y_1, ..., y_l, z-1) + \mu'' P(x, y_1, ..., y_l, z+1). \quad (1)$$

Although this Markov chain has infinite states, we have to limit the number of states to run the iterative algorithm. With several trials, we can set reasonable bounds for x, y_i, and z so that boundary states have probabilities very close to 0. As this is a continuous-time Markov chain, there is no loop transition from a state back to itself. Thus, we can set the probabilities of out-of-bound adjacent states to 0 to avoid using a large set of boundary equations for all special boundary conditions.

Solving HAP/M/1

The algorithm starts from initializing the state probabilities and then many iterations to reach the steady-state probabilities. Initially, we set the state probabilities as

$$P(x, y_1, ..., y_l, z) = \frac{|x| - x}{\sum_{k=1}^{|x|} k} \times \prod_{i=1}^{l} \frac{|y_i| - y_i}{\sum_{k=1}^{|y_i|} k} \times \frac{|z| - z}{\sum_{k=1}^{|z|} k}$$

where $|x|$ is the number of x's possible values, etc. For each iteration, we recompute probabilities for the states with $x + y_1 + ... + y_l + z = k$, starting from $k = 0$ to 1, 2, etc. At the end of each iteration, state probabilities are normalized to make sure they sum up to 1. The algorithm stops when $\frac{P_{n+1} - P_n}{P_n} \leq \epsilon$ for all states, where P_n is the probability in the nth iteration and ϵ is a small real number. From the steady-state probabilities, $\overline{z}$ and $\overline{\lambda}$ can be computed. By Little's result, $\overline{z} = \overline{\lambda}T$, message delay, T, can be obtained. The experience with this brute-force approach is that the bound for z is much larger than the ones for x and y_i. That is because we have only one server at the message level while there are, logically, infinite servers at the user and application levels. For large x and y_i, the states with even larger z have taken over most of the probability. A large number of states makes the convergence to steady-state probabilities difficult.

4.4 Approximate Analysis

4.4.1 Solution 1: Steady State Probability

Given the computational difficulty in Solution 0, we now drop the z dimension of the $(l+2)$-dimension Markov chain, which results in the Markov chain in Figure 20. We use the same approach to obtain $P(x, y_1, ..., y_l)$ and then $\overline{\lambda}$. Suppose that the state transition rates are small compared to the message arrival rate, $\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}$, when the system stay in state $(x, y_1, ..., y_l)$, the message interarrival time distribution can be approximated as

$$a(t) = \sum_{x=0}^{\infty} \sum_{y_l=0}^{\infty} \cdots \sum_{y_1=0}^{\infty} \tilde{P}(x, y_1, ..., y_l) \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij} e^{-\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij} t} \tag{2}$$

where $\tilde{P}(x, y_1, ..., y_l)$ is the probability that, given a message, it is generated during the system's stay in state $(x, y_1, ..., y_l)$. It is the probability that the system stays in state $(x, y_1, ..., y_l)$ weighted by the message arrival rate of that state. Thus, we can write

$$\tilde{P}(x, y_1, ..., y_l) = \frac{P(x, y_1, ..., y_l) \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}}{\sum_{x=0}^{\infty} \sum_{y_l=0}^{\infty} \cdots \sum_{y_1=0}^{\infty} P(x, y_1, ..., y_l) \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}}. \tag{3}$$

Note that the denominator is actually $\overline{\lambda}$. As we express, in Solution 1 and also Solution 2, the message interarrival time as a distribution and redraw arrivals from this distribution, correlation between subsequent arrivals is lost. Only Solution 0 preserves this correlation.

Solving HAP/M/1

Being able to compute $a(t)$, the problem now is to compute mean delay of HAP/M/1. We use the approach for G/M/1 [Kle75]; namely if we can obtain σ in the equation $A^*(\mu'' - \mu''\sigma) = \sigma$ where $A^*(s)$ is the Laplace transform of message interarrival time, we can plug it into the equation for delay, $T = \frac{1}{\mu''(1-\sigma)}$, and waiting time distribution, $W(y) = 1 - \sigma e^{-\mu''(1-\sigma)y}$. Using Little's result, $\overline{N} = \overline{\lambda}\, T$, we can also solve $\overline{N}$, the mean queue length, including the one in service, if we know $\overline{\lambda}$. Since $A^*(s) = \int_0^{\infty} a(t) e^{-st} dt$, we can compute $A^*(\mu'' - \mu''\sigma)$ by integration. We resort to the following algorithm to solve σ.

σ-Algorithm:

Step 1 : Pick any value between 0 and 1 for σ, say 0.5.

Step 2 : Calculate $A^*(\mu'' - \mu''\sigma)$ by integrating $\int_0^{\infty} a(t) e^{-(\mu'' - \mu''\sigma)t} dt$ where a(t) is from Equation 2.

Step 3 : If $|A^*(\mu'' - \mu''\sigma) - \sigma| < \epsilon$, stop. Otherwise, pick σ as $\frac{A^*(\mu'' - \mu''\sigma) + \sigma}{2}$ and go to Step 2.

Note that as σ increases, $\int_0^\infty a(t)e^{-(\mu''-\mu''\sigma)t}dt$ also increases. Thus, $A^*(\mu''-\mu''\sigma)$ has a positive correlation with σ. This means each time we take the average of $A^*(\mu''-\mu''\sigma)$ and σ as the new σ value their difference gets smaller. This shows that the algorithm will converge.

The mean message delay, T, for this G/M/1 queueing system is then $\frac{1}{\mu''(1-\sigma^*)}$ where σ^* is the result of σ-algorithm. For the mean queue length, we have

$$\overline{N} = \overline{\lambda}T = \frac{\frac{\lambda}{\mu}\sum_{i=1}^{l}\frac{\lambda_i}{\mu_i}\sum_{j=1}^{m_i}\lambda_{ij}}{\mu''(1-\sigma^*)}$$

where $\overline{\lambda}$ is the denominator of Equation 3. For G/M/1, once we have solved σ^*, the waiting time distribution is easily expressed as $W(y) = 1 - \sigma^* e^{-\mu''(1-\sigma^*)y}$.

4.4.2 Solution 2: Conditional Probability

Solution 2 is the same as Solution 1 except that we can now obtain closed-form formulas for $a(t)$ and $\overline{\lambda}$ by conditional probability. We now derive $\overline{\lambda}$, mean numbers of users and applications, and $a(t)$.

Mean message arrival rate

Suppose there are x user instances and y_i instances of application type i, $i = 1 \ldots l$, the mean message arrival rate, $\overline{\lambda}$, is

$$\overline{y_1\sum_{j=1}^{m_1}\lambda_{1j} + \ldots + y_l\sum_{j=1}^{m_l}\lambda_{lj}} = \sum_{i=1}^{l}\overline{y_i}\sum_{j=1}^{m_i}\lambda_{ij}.$$

If we condition on x and, since there is no restriction on y_i, model application arrivals and departures as M/M/∞ [Kle75], we can obtain $\overline{y_i}$ as

$$\sum_{x=0}^{\infty}\sum_{y_i=0}^{\infty} y_i \frac{(\frac{x\lambda_i}{\mu_i})^{y_i}}{y_i!}e^{-\frac{x\lambda_i}{\mu_i}}P_x$$

where P_x is the probability that there are x user instances. Note that conditioning on x and y_i and the approximation as M/M/∞ are good only when arrival and departure rates at user level are much smaller than the ones at application level, which again are much smaller than the ones at message level. Again, since there is no restriction on x, we can model user arrivals and departures as M/M/∞. Thus, P_x is $\frac{(\frac{\lambda}{\mu})^x}{x!}e^{-\frac{\lambda}{\mu}}$. Clearly we have

$$\overline{y_i} = \sum_{x=0}^{\infty}\frac{x\lambda_i}{\mu_i}\frac{(\frac{\lambda}{\mu})^x}{x!}e^{-\frac{\lambda}{\mu}} = \frac{\lambda_i}{\mu_i}\frac{\lambda}{\mu}.$$

Thus, we have

$$\overline{\lambda} = \frac{\lambda}{\mu}\sum_{i=1}^{l}\frac{\lambda_i}{\mu_i}\sum_{j=1}^{m_i}\lambda_{ij}. \tag{4}$$

If we assume $\lambda_i = \lambda'$, $\mu_i = \mu'$, and $\lambda_{ij} = \lambda'' \; \forall \; i$ and j, we have

$$\overline{\lambda} = \frac{\lambda}{\mu}\frac{\lambda'}{\mu'}\lambda'' \sum_{i=1}^{l} m_i. \tag{5}$$

From Equation 5, we observe that merging or splitting the branches in this simplified HAP (i.e. $\lambda_i = \lambda'$, $\mu_i = \mu'$, and $\lambda_{ij} = \lambda''$) will not change its $\overline{\lambda}$ as long as we keep the same number of leaves in its HAP object class. Figure 22 shows three HAP's with the same $\overline{\lambda}$, which is $4\frac{\lambda}{\mu}\frac{\lambda'}{\mu'}\lambda''$ (from Equation 5). However, their degrees of burstiness should be different. Intuitively, one would think that the order of burstiness is (c) > (b) > (a) because the arrival rate is $4\lambda''$ when a single application instance is active in (c) while it is $2\lambda''$ when a single application instance is active in (a).

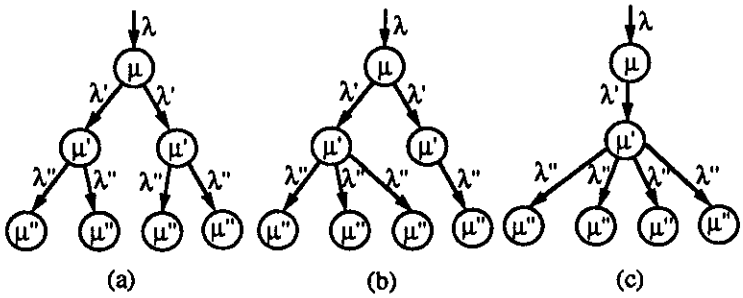

Figure 22: Three HAPs with equivalent mean arrival rate

Mean numbers of users and applications

As pointed out, the arrivals and departures of user and application instances can be modeled as M/M/∞. Thus, the mean number of user instances, $\overline{x}$, is $\frac{\lambda}{\mu}$ and the mean number of application instances, $\overline{y}$, is

$$\overline{y_1} + ... + \overline{y_l} = \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i}\frac{\lambda}{\mu} = \frac{\lambda}{\mu}\sum_{i=1}^{l}\frac{\lambda_i}{\mu_i} \tag{6}$$

If $\lambda_i = \lambda'$ and $\mu_i = \mu'$, $\overline{y}$ becomes $l\frac{\lambda}{\mu}\frac{\lambda'}{\mu'}$.

Message interarrival time distribution

To solve the message interarrival time distribution, $a(t)$, we try to solve $A(t)$ first since it is easier. We first condition on x and then condition on y_i, $i = 1 \; .. \; l$. When HAP is in state $(x, y_1, ..., y_l)$, the message arrival process is Poisson with rate $\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}$. Using the conditional probability, on x and y_i, weighted by the message arrival rates just like Equation 2 and 3, we have

$$A(t) = \sum_{x=0}^{\infty}(\sum_{y_l=0}^{\infty} ... \sum_{y_1=0}^{\infty})(1 - e^{-\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij} t})$$

$$\times \frac{[\prod_{i=1}^{l} \frac{(\frac{x\lambda_i}{\mu_i})^{y_i}}{y_i!} e^{-\frac{x\lambda_i}{\mu_i}} \times \frac{(\frac{\lambda}{\mu})^x}{x!} e^{-\frac{\lambda}{\mu}}] \times \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}}{\sum_{x=0}^{\infty}(\sum_{y_l=0}^{\infty} \cdots \sum_{y_1=0}^{\infty})[\prod_{i=1}^{l} \frac{(\frac{x\lambda_i}{\mu_i})^{y_i}}{y_i!} e^{-\frac{x\lambda_i}{\mu_i}} \times \frac{(\frac{\lambda}{\mu})^x}{x!} e^{-\frac{\lambda}{\mu}}] \times \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}}.$$

The denominator is $\overline{\lambda}$ and its result turns out to be the same as Equation 4. As we carry out all the summations on 1, which is still 1, and the summation over y_1, we have

$$1 - \sum_{x=0}^{\infty} \sum_{y_l=0}^{\infty} .. \sum_{y_2=0}^{\infty} [\frac{x\lambda_1}{\mu_1} e^{-\sum_{j=1}^{m_1} \lambda_{1j} t} \sum_{j=1}^{m_1} \lambda_{1j} e^{\frac{x\lambda_1}{\mu_1}} e^{-\sum_{j=1}^{m_1} \lambda_{1j} t} + e^{\frac{x\lambda_1}{\mu_1}} e^{-\sum_{j=1}^{m_1} \lambda_{1j} t} \sum_{i=2}^{l} y_i \sum_{j=1}^{m_{ij}} \lambda_{ij}]$$

$$\times \prod_{i=2}^{l} \frac{(\frac{x\lambda_i}{\mu_i})^{y_i}}{y_i!} \times \frac{e^{-x\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i}} \times \frac{(\frac{\lambda}{\mu})^x}{x!} e^{-\frac{\lambda}{\mu}}}{\frac{\lambda}{\mu} \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}}.$$

We can extract $e^{\frac{x\lambda_1}{\mu_1}} e^{-\sum_{j=1}^{m_1} \lambda_{1j} t}$ from the above formula. Carrying out the summations over the other y_i's, we obtain

$$1 - \sum_{x=0}^{\infty} e^{x\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i}} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} \times x(\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} \sum_{j=1}^{m_i} \lambda_{ij}) \times \frac{e^{-x\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i}} \times \frac{(\frac{\lambda}{\mu})^x}{x!} e^{-\frac{\lambda}{\mu}}}{\frac{\lambda}{\mu} \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}}$$

which is reorganized to

$$1 - \sum_{x=0}^{\infty} \frac{x(\frac{\lambda}{\mu} e^{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} (e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} - 1)})^x}{x!} \times \frac{e^{-\frac{\lambda}{\mu}} \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} \sum_{j=1}^{m_i} \lambda_{ij}}{\frac{\lambda}{\mu} \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}}.$$

Finally, by carrying out the last summation, we reach the result

$$A(t) = 1 - \frac{e^{-\frac{\lambda}{\mu}} L(t) e^{\frac{\lambda}{\mu} L(t)} M(t)}{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}}, where \tag{7}$$

$$L(t) = e^{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} (e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} - 1)} \tag{8}$$

$$M(t) = \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} \sum_{j=1}^{m_i} \lambda_{ij}. \tag{9}$$

Note that $L'(t) = -L(t)M(t)$. We can also check this probability function: $A(t) \to 1$ as $t \to \infty$ and $A(t) = 0$ as $t = 0$. For the density function, $a(t)$, we differentiate Equation 7 to get

$$a(t) = \frac{e^{-\frac{\lambda}{\mu}} e^{\frac{\lambda}{\mu} L(t)}}{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}} [L(t)N(t) + L(t)M^2(t) + \frac{\lambda}{\mu} L^2(t)M^2(t)], where \tag{10}$$

$$N(t) = \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} (\sum_{j=1}^{m_i} \lambda_{ij})^2. \tag{11}$$

Note that $a(t) \to 0$ as $t \to \infty$ and

$$a(0) = \frac{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} (\sum_{j=1}^{m_i} \lambda_{ij})^2 + (1 + \frac{\lambda}{\mu})(\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij})^2}{\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} \sum_{j=1}^{m_i} \lambda_{ij}}.$$

Solving HAP/M/1

To solve HAP/M/1, we need to assume μ_{ij}, the message service time distribution, is equal to μ'' for all i and j. If we allow non-identical service time distributions at a queue, we have no product form solution [BCMP75]. However, as a(t) in Equation 10 is a complex function, obtaining the $A^*(s)$ and solving for σ will be intractable. Thus, we still use our σ-algorithm to solve σ.

Superposition of HAPs

A queue in a queueing network may have input traffic from several sources. That is, the arrival process to this queue is a superposition of several arrival processes. To analyze HAP on a queueing network, it is important to understand what arrival process it is if we superpose two or more independent HAP's. We first investigate the superposition of two HAP's with different sets of model parameters. And then generalize the result for the superposition of many HAP's. At last, we study the superposition of homogeneous HAP's where two HAP's are defined to be homogeneous if their model parameters, except λ, are identical.

$HAP_1 + HAP_2$

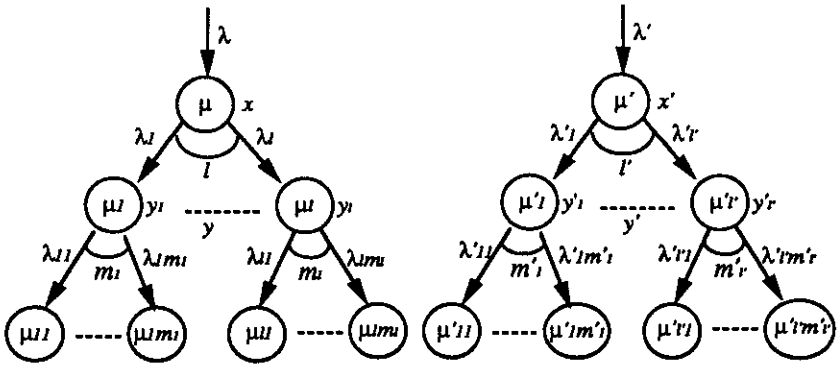

Figure 23: Two HAP's

In Figure 23, there are two HAP's with two different sets of model parameters. Note that x' and y_i' are defined as x and y_i. We need to know the interarrival time distribution, $A(t)$ and $a(t)$. Again, conditioning on x, x' and then on y_i, y_i', $A(t)$ is

$$\sum_{x=0}^{\infty} \sum_{x'=0}^{\infty} \sum_{y_1=0}^{\infty} \cdots \sum_{y_1=0}^{\infty} \sum_{y_{l'}'=0}^{\infty} \cdots \sum_{y_1'=0}^{\infty} (1 - e^{-(\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij} + \sum_{i=1}^{l'} y_i' \sum_{j=1}^{m_i'} \lambda_{ij}')t})$$

$$\times \prod_{i=1}^{l} \frac{(\frac{x\lambda_i}{\mu_i})^{y_i}}{y_i!} e^{-\frac{x\lambda_i}{\mu_i}} \prod_{i=1}^{l'} \frac{(\frac{x'\lambda_i'}{\mu_i'})^{y_i'}}{y_i'!} e^{-\frac{x'\lambda_i'}{\mu_i'}} \times \frac{(\frac{\lambda}{\mu})^x}{x!} e^{\frac{\lambda}{\mu}} \frac{(\frac{\lambda'}{\mu'})^{x'}}{x'!} e^{\frac{\lambda'}{\mu'}}.$$

As y_i's and y_i''s are independent sets of variables, we observe that the above formula is actually

$$A(t) = 1 - (1 - A_1(t))(1 - A_2(t)). \tag{12}$$

Again, differentiating $A(t)$ gives us

$$a(t) = a_1(t)(1 - A_2(t)) + a_2(t)(1 - A_1(t)) \tag{13}$$

or

$$a(t) = \frac{a_1(t)a_2(t)}{\frac{\lambda'}{\mu'}L_2(t)M_2(t)} + \frac{a_1(t)a_2(t)}{\frac{\lambda}{\mu}L_1(t)M_1(t)} \tag{14}$$

Generalization

Suppose now we have n HAPs and each HAP has its own model parameters. By the simple iterative derivation based on Equations 12 and 13, we can obtain

$$A(t) = 1 - \prod_{i=1}^{n}(1 - A_i(t)), and \tag{15}$$

$$a(t) = \sum_{i=1}^{n} a_i(t) \prod_{j=1, j\neq i}^{n} (1 - A_j(t)). \tag{16}$$

Superposition of homogeneous HAPs

From Equation 16 we know combining several heterogeneous HAP's whose model parameters are different results in a non-HAP. We want to know if combining several homogeneous HAP's whose model parameters except λ are the same results in a HAP. By plugging in the same model parameters, except λ which we use λ_1 through λ_n, into Equation 16, we get

$$a(t) = \frac{\sum_{i=1}^{n}\lambda_i}{\mu} e^{-\frac{\sum_{i=1}^{n}\lambda_i}{\mu}} L(t) e^{\frac{\sum_{i=1}^{n}\lambda_i}{\mu}L(t)} M(t). \tag{17}$$

Comparing with Equation 10, the result shows that the superposition is still a HAP where everything remains the same except λ is replaced with $\sum_{i=1}^{n}\lambda_i$. This is a very useful result which enable us to analyze a queueing network if all traffic sources have homogeneous HAP's.

HAP on queueing networks

After analyzing the HAP/M/1 and the superposition of HAP's, we are ready to apply some of the results to analyze open queueing networks where external traffic arrival processes are HAP's.

Modeling a network as a single queue

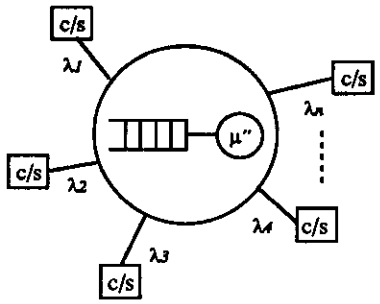

Figure 24: A single-queue network

Suppose that we model a carrier network as a single queue; that is, we are combining all queues into a single queue and, in the mean time, all processing, switching transmission, and propagation tasks onto a single network server. Indeed, this is a very abstract and simplified model. Figure 24 shows the single-queue network connecting n stations. Each station acts as both client and server, and the arrival process at each station is a HAP-CS as described in section 2. Suppose that these HAP-CS's are homogeneous and the user arrival rate for station i is λ_i, $i = 1...n$. Stations generate requests into the queue and the network server switches the requests to the destination stations. There is a probability that the requests will trigger responses back to the request sources. These responses, which are also sent over the network, may again trigger next requests with a probability. Assuming both the response and request triggering probabilities are p, and both response and request service time distributions are exponential with mean $1/\mu''$, we want to analyze this single-queue network.

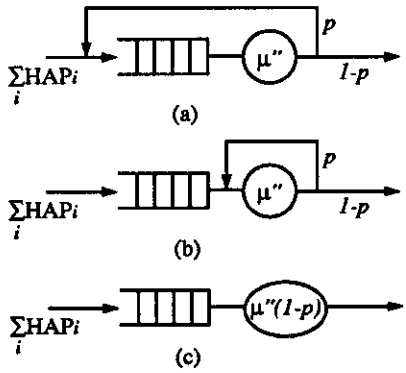

Figure 25: Queueing models for the single-queue network

This queue actually can be modeled as Figure 25(a) where the arrival process is a superposition of n homogeneous HAP's and a message has probability p to go back to the tail of the queue, i.e. trigger another message into the network. From Equation 17, we know that this superposition is a still HAP with user arrival rate $\sum_{i=1}^{n} \lambda_i$. Furthermore, it can be shown that, in Figure 25, (a), (b), (c) have the same mean delay although the delay distributions are different. Thus, the mean delay

for this single-queue network is $\frac{1}{\mu''(1-p)(1-\sigma)}$ where σ can be obtained from $A^*(\mu''(1-p)-\mu''(1-p)\sigma)=\sigma$ where $A^*(s)$ is the transform of $a(t)$ in Equation 17.

Open queueing networks with no loops

Next we attack the problem of analyzing open queueing networks with external HAP traffic sources. If we can solve the mean delay at queue i, T_i $i=1...M$ where M is the number of queues in the network, we can use the result, $T=\sum_{i=1}^{M}\frac{\lambda_i T_i}{\gamma}$ [Kle76], to obtain the mean end-to-end delay, T. Note that the above equation is very general and is applicable to any open queueing networks.

The first problem we encounter is: what is the departure process of a HAP/M/1 queue? It would be nice if it is still a HAP. Suppose that the interdeparture time distribution at queue i is $d_i(t)$. As the probability that the queue is not empty upon an arrival instance, $1-r_0$ (which is σ in G/M/1), is equal to the probability that the queue is not empty upon a departure instance, $1-d_0$, for any queueing system with unit changes of $N(t)$ [Kle75], we can derive the transform of $d_i(t)$, by conditioning on the state of the queue upon a departure instance, as

$$D_i^*(s)=\sigma\frac{\mu''}{s+\mu''}+(1-\sigma)A_i^*(s)\frac{\mu''}{s+\mu''}. \tag{18}$$

From this, we can obtain the equation for the density function,

$$d_i(t)=\sigma\mu''e^{-\mu''t}+(1-\sigma)a_i(t)\otimes b(t) \tag{19}$$

where $b(t)=\mu''e^{-\mu''t}$ and $a_i(t)\otimes b(t)=\int_0^t a_i(t')b(t-t')dt'$. Using integration, we get

$$D_i(t)=\int_0^t d_i(t')dt'=\int_0^t \sigma\mu''e^{-\mu''t}dt'+\int_0^t\int_0^{t'} a_i(t'')b(t'-t'')dt''dt'. \tag{20}$$

Unfortunately, this $d_i(t)$ is no longer a distribution of HAP! If we feed this departure process to another queue, we will have to solve σ for the equation $A^*(\mu''-\mu''\sigma)=\sigma$ where $A^*(s)$ is now replaced by Equation 18. Still, we can get the mean delay and waiting time distribution for the second queue.

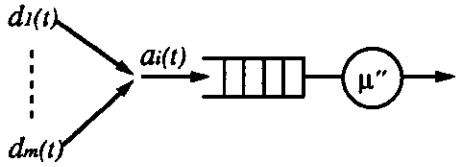

Figure 26: Superposition as an arrival process

However, as shown in Figure 26, if the arrival process to queue i is a superposition of several independent HAP's and non-HAP's, $d_j(t), j=1..m$, we will have to compute the interarrival time distribution, $a_i(t)$, of this superposition. To obtain this new $a_i(t)$, we start from its $A_i(t)$, which is

$$A_i(t)=1-P\{t_a>t\}=1-P\{t_{a_1}>t\}\times...\times P\{t_{a_m}>t\}=1-\prod_{j=1}^{m}(1-D_j(t)).$$

Thus, we have

$$a_i(t) = A_i'(t) = \sum_{j=1}^{m} d_j(t) \prod_{k=1, k\neq j}^{n} (1 - D_k(t)) \tag{21}$$

where $d_j(t)$ and $D_k(t)$ can be obtained from

1. Equation 19 and 20 if they represent a departure process of a HAP/M/1 queue,
2. Equation 10 and 7 if they are from an external HAP traffic source, or
3. Equation 19 and 20, except that $a(t)$ in these two equations depends, recursively, on the input process of the previous queue of queue i, if they have external HAP's that are more than one-hop away.

Figure 27 illustrates these three cases. Once we have the above method to compute $a_i(t)$, we can solve σ_i which, in turn, gives us T_i. Hence, the mean end-to-end delay, T, is $\sum_{i=1}^{M} \frac{\lambda_i T_i}{\gamma}$ where λ_i, the mean arrival rate to queue i, can be calculated by combining all i's arrivals and γ is the sum of mean arrival rates of all n external HAP's.

The problem left now is: what if these $d_j(t)$'s depend on each other(i.e. there are loops in the network flow)? As our external arrival processes, HAP's, are not Poisson, we can no longer use Jackson's theorem [Jac57] where each queue can be treated as an M/M/1 queue. Thus, we have no solution for this general queueing network so far!

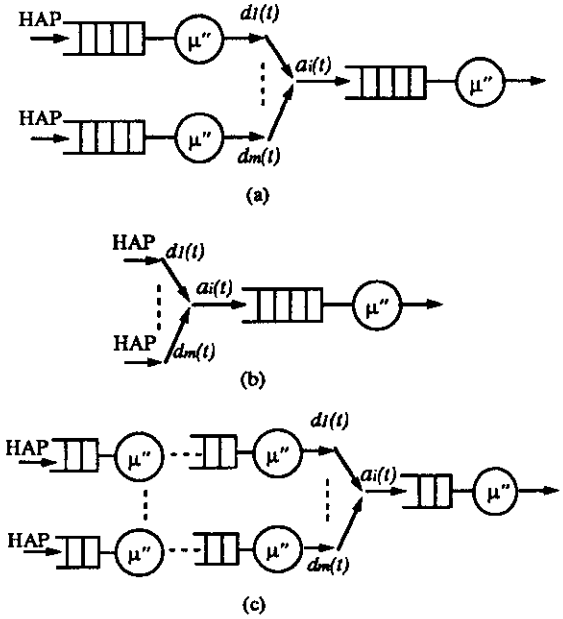

Figure 27: Distance to external HAP's

4.5 HAP Simulation

HAP has a hierarchical structure where a process may fork many child processes. We now briefly describe its simulation on a single server queue.

This simulator is event-driven and continuous-time. Events are stored in an *event queue* and sorted by their timestamps. Each time an event is fetched, simulation time is advanced to this event's timestamp. There are *arrival* events and *departure* events for user, application, and message instances. An arrival event marks the generation of a user process, an application process, or a message. A departure event signals that the service for its corresponding arrival event is finished. This means that a user has quitted the system, an application has been terminated, or a message has been transmitted by the server.

Initially, the first set of arrival events are generated, one event for each type in the HAP. This includes the first user arrival, its first application arrivals of all application types, and their first message arrivals of all message tyes. Note that the timestamp of a child arrival event must be within its parent event's life which spans from the arrival time to the departure time, while the timestamp of a child departure event does not have to fall in its parent event's life span. As the departure times of users and applications do not depend on the server, the departure events are generated when their corresponding arrival events are generated. The departure times of messages, however, depend on the server and can be determined only when the message reaches the server.

Depending on its type, an event fetched from event queue is processed in different ways:

user arrival Another user arrival event, its first set of descendant arrival events, and the corresponding user and application departure events are generated and inserted into event queue.

application arrival Another application arrival event, its first set of child arrival events, and the corresponding application departure event are generated and inserted into event queue.

message arrival Another message arrival event is generated and inserted into event queue. The fetched message is appended to the tail of the server queue. If it happens to arrive to an idle server, its message departure event is generated and inserted into event queue.

user or application departure Update statistics.

message departure Update statistics. If the departure leaves a non-empty server, the departure event for the next message is generated and inserted into event queue.

5 Numerical Results

Section 5.1 presents the basic numerical results regarding the accuracy of the approximate solutions. Section 5.2 discusses HAP's message interarrival time distribution. HAP's long-term and short-term behaviors are examined in section 5.3 and 5.4, respectively. Impacts of HAP parameters on delay patterns are studied in detail in section 5.5. We compare HAP with MMPP and Poisson. The impacts of arrival and departure processes at different levels on queueing performance are investigated. The effect of a simple flow control scheme at user and application levels is studied. We also compare 2-level HAPs with 3-level HAPs. Finally, the behavior of a realistic example HAP with heterogeneous applications is studied.

Assuming that $\lambda_i = \lambda'$, $\mu_i = \mu'$, and $\lambda_{ij} = \lambda''$ for all i and j, we study the accuracy of the solutions under different user, application, and message parameters. We use the following set of parameters to study message interarrival time distribution and the queueing behavior on an exponential server queue: $\lambda = 0.0055$, $\mu = 0.001$, $\lambda' = 0.01$, $\mu' = 0.01$, $\lambda'' = 0.1$, $\mu'' = 20$, $l = 5$, $m = 3$. According to Equation 4, $\overline{\lambda} = \frac{0.0055}{0.001} \times \frac{0.01}{0.01} \times 0.1 \times 5 \times 3 = 8.25$ which is the same as the result from Solution 0 and simulations. For this set of parameters, we have $\sigma = 0.50$ by Solution 0, 1, and 2, $\rho = 0.42$, mean delay for HAP/M/1 = 0.55 by Solution 0 and simulation, and 0.1 by Solution 1 and 2, mean delay for M/M/1 = 0.085 (HAP's delay is 6.47 times higher by Solution 0 and simulation, and 17.65% higher by Solution 1 and 2). Solution 1 and 2 are within 1% difference between each others. As we can see, the approximation error due to the loss in correlation, when we express the message interarrival time as a distribution, is very significant.

5.1 Accuracy of the Solutions

Solution 0 in section 3 takes about 2 weeks, on a SUN-4/280 minicomputer without other long running process, to compute the delay for a given set of parameters because of the large bound required in z dimension. Solution 1 takes around 7 hours, while Solution 2 only takes 5 to 7 minutes.

From the derivation of Solution 1 and 2, we observe that three constraints on the parameters are required to obtain good results by Solution 1 and 2:

1a. Message-level arrival and departure rates have to be much larger than the upper-level arrival and departure rates.

1b. Lower-level arrival and departure rates have to be much larger than the upper-level arrival and departure rates.

2. Gap between the rates of neighboring states in the underlying Markov chain can not be too large.

3. Traffic load is light.

Condition 1a and 1b are for Solution 1 and 2, respectively. Condition 1a are due to the derivation of Equation 2 which is valid only when the message arrival rate, $\sum_{i=1}^{I} y_i \sum_{j=1}^{m_i} \lambda_{ij}$, is large compared to the state transition rate of the Markov chain in Figure 20 and the message arrival rate does not have a sudden big change when the state transition happens. (The message departure rate, of course, should be larger than the message arrival rate.) Condition 1b is a tighter condition than 1a because, in Solution 2, Equation 7 is obtained by first conditioning on x and then on y_i. This conditional probability is valid only when x is seldom changed compared to y_i. Even if x or y is changed, we do not want a big jump in the message arrival rate, which results in condition 2.

Condition 3 is due to the loss in correlation. This loss becomes very serious when the system utilization gets higher.

The above three conditions are just guidelines. In our trials, we observe that if (1) lower-level rates are 5 times larger than the upper-level rates, (2) message arrival rate of one state in the underlying Markov chain is not more than double or less than a half of the rate of its neighboring states (which limits the value of m), and (3) the resulting σ for HAP/M/1 is less than 30%, Solution 1 and 2 have approximation errors less than 5%. As we can expect, Solution 1 is a better approximation when condition 1a is satisfied but condition 1b is not. Surprisingly, Solution 1 and 2 are almost the same, with less than 1% difference between them, when the tighter condition, 1b, is satisfied.

5.2 Message Interarrival Time

For the same traffic level, HAP exhibits a higher degree of burstiness than Poisson. In Figure 28, both curves of $a(t)$ have the same $\overline{\lambda}$ (7.5). We plot HAP's $a(t)$ with Equation 10 and compute the HAP's $\overline{\lambda}$ using Equation 4 to find the equivalent load-level Poisson process. Integration of $a(t)$ is 1 for both curves and integration of $t * a(t)$, which is $\overline{t}$ or $\frac{1}{\lambda}$, have the same value (1/7.5). HAP has higher $a(0)$, 9.28, than Poisson's 7.5. But they intersect at two points: $t = 0.077$ and 0.53. The intersection at 0.53 is illustrated in Figure 29.

The interpretation of these two intersections is that HAP tends to have a higher percentage of messages arriving with rather short interarrival times while Poisson has a higher percentage of messages arriving with medium interarrival times. As HAP's $a(t)$ has a longer tail, it also has a slightly higher percentage of messages with relatively large interarrival times. It is important to note that these two intersections make both curves have the same $\overline{t}$ and hence $\overline{\lambda}$. If they only intersect at $t = 0.077$, this HAP will have smaller $\overline{t}$ and hence larger $\overline{\lambda}$ because integration of $t * a(t)$ for HAP will be smaller than the one for Poisson. Integration of $t * a(t)$ of HAP's tail after 0.53

compensates the discrepancy of the front part.

Compared to Poisson, HAP's arrivals tend to spread out from the mean interarrival time (0.133). If the difference of message arrival rates of neighboring states in the Markov chain is large, arrivals will spread out from the mean interarrival time even more. The short interarrival times represent the time intervals between messages of the same burst, while the longer interarrival times are the time intervals between bursts at the application and user levels.

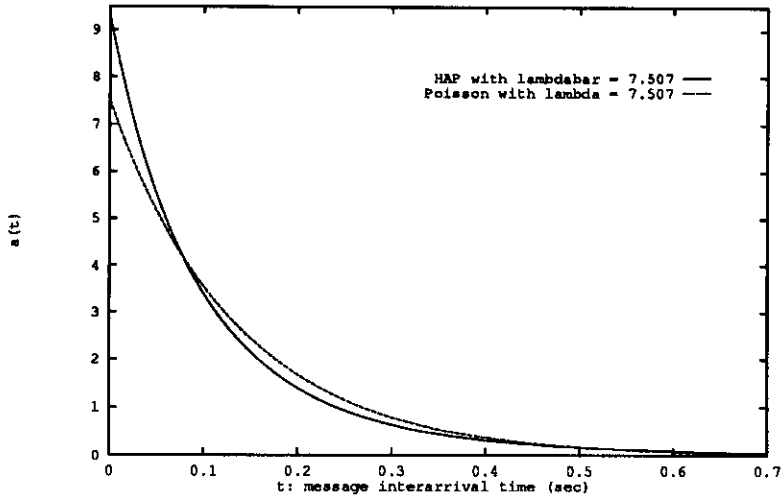

Figure 28: Message interarrival time distribution

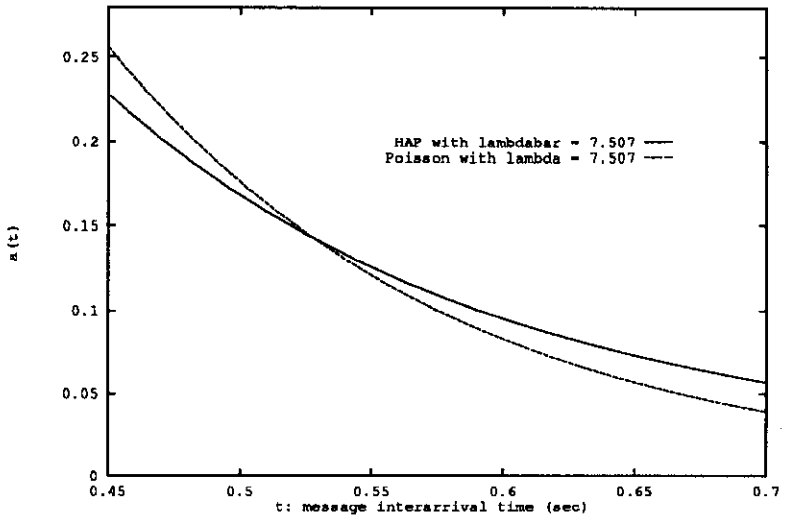

Figure 29: Tail of message interarrival time distribution

5.3 Long-term Queueing Behavior

As we feed this multi-level highly correlated message stream into an exponential service queue, we can expect a higher queueing delay compared to Poisson. This is obvious because a message is more likely to enter the queue with a group of messages which may be correlated by the same user instance, application instance, and even message instance.

In Figure 30 and 31, we obtain average delay, probability that an arrival finds the server busy (σ), and utilization (ρ) by Solution 0 and simulations (which have less than 5% difference between each other). Note that our starting set of parameters are $\lambda = 0.0055$, $\mu = 0.001$, $\lambda' = 0.01$, $\mu' = 0.01$, $\lambda'' = 0.1$, $\mu'' = 17$, $l = 5$, and $m = 3$. In Figure 30, when μ(message), which is equivalent to the server capacity, is 30 messages per second on the average, HAP's delay is only 15.22% higher than Poisson's. But HAP's delay becomes 200 times higher than Poisson's when the utilization is 64%! In the meantime, the difference between σ and ρ grows as we increase the utilization. In Figure 31, we adjust the load, by changing λ(user), while keeping the server capacity fixed.

μ (message)	σ	ρ	delay (HAP)	delay(Poisson)
30	0.33	0.28	0.053	0.046
23	0.43	0.36	0.152	0.068
20	0.50	0.42	0.550	0.085
19	0.52	0.44	0.950	0.093
17	0.58	0.49	3.20	0.114
15	0.65	0.55	11.80	0.148
13	0.74	0.64	45.50	0.221

Delay performance, $\overline{\lambda} = 8.25$

Figure 30: Average Delay versus Server Capacity

hereafter vanishes.

One interesting question arises: how bursty a HAP can be? As we will point out in next section, burstiness increases significantly when the gap between message arrival rates of neighboring states in the underlying Markov chain increases. For the given set of parameters, this gap is not large, especially when y_i is large, as we have only one message type ($\lambda_{ij} = \lambda''$). Unfortunately, a large gap between neighboring states violates our condition 2 for using Solution 1 and 2. In next section, an example HAP with heterogeneous message types, with slightly larger gap between neighboring states, exhibits a higher burstiness.

λ (user)	$\bar{\lambda}$	σ	ρ	delay (HAP)	delay(Poisson)
0.001	1.52	0.187	0.089	0.074	0.065
0.002	3.01	0.275	0.177	0.086	0.071
0.003	4.49	0.362	0.264	0.127	0.080
0.004	5.99	0.450	0.352	0.35	0.091
0.005	7.48	0.537	0.440	1.65	0.105
0.0055	8.25	0.581	0.486	3.20	0.114
0.006	9.01	0.625	0.530	5.73	0.125
0.007	10.50	0.706	0.618	17.50	0.154
0.008	11.95	0.783	0.704	47.50	0.199

Delay performance, μ (message) = 17

Figure 31: Average Delay versus Message Arrival Rate

5.4 Short-term Queueing Behavior

To explain why HAP's average delay goes up dramatically as we increasing the utilization, we now look at HAP's short-term queueing behavior in the simulation.

Figure 32 shows that HAP simulation is difficult to converge. It is not easy to determine the criteria to stop the simulation due to its fluctuation which is far more serious than Poisson. There are two reasons for this. The first one is that HAP compounds processes at quite different time scales. User arrival and departure have time scales of tens of minutes, while message arrival and departure have time scales of milliseconds.

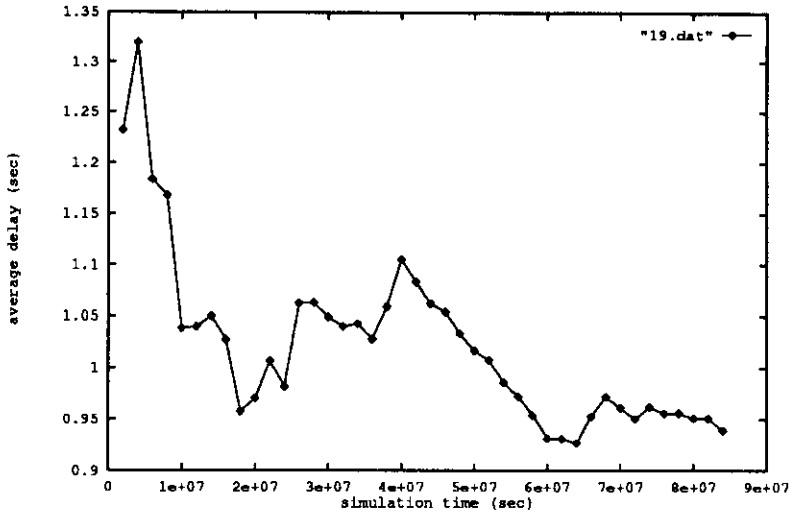

Figure 32: Fluctuation of HAP Simulation

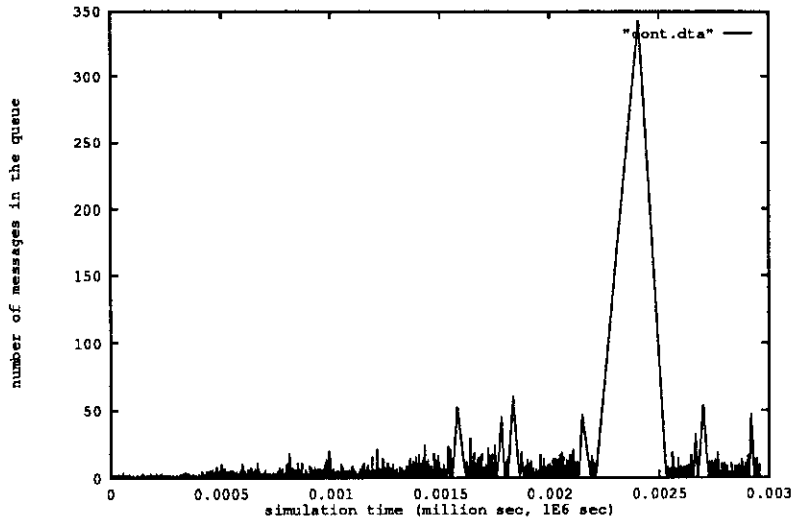

Figure 33: Arrivals and Departures in a One-hour Interval

The second reason is that, from time to time, serious congestions happen in the HAP simulation. These congestions are due to the long bursts which are illustrated as the big *mountains* in Figure 33 (in a one-hour period) where the number of messages in the queue is traced. In the extreme case as shown in Figure 34, we have a *mountain* which has the peak number of messages over 17,000 and lasts about 80 minutes. (In our simulations, Poisson's peak number of messages only reaches 29!) These mountains cause the fluctuation in simulations and make the average delay go up significantly. For this extreme mountain, we trace the numbers of users and applications in Figure 35 and 36, respectively. At the beginning of this long burst, there are 13 users and 49 applications while the averages are 5.5 and 27.5, respectively. Under a large number of users or applications, the chance to have an upcoming long burst is high.

Figure 37 lists the statistics of busy/idle period and height of the mountain (maximum number of messages in a busy period) for HAP and Poisson. Both HAP and Poisson have mean busy period/(mean busy period + mean idle period) around 55%. Mean busy/idle period and height of HAP are only slightly higher than Poisson's, while the variances of HAP are much higher (618, 15, and 66 times higher than Poisson's for busy period, idle period, and height, respectively). Besides, we observe that, for the same simulation length, the number of mountains of HAP is 19% smaller than Poisson's.

Heights of the mountains directly affect the waiting times of new arrivals, while lengths of busy periods directly determine the number of arrivals that will have waiting times of this order. A high mountain with a short width means only a small number of arrivals suffer congestion. Having larger mean busy period and height, and even much higher variances, HAP has a much larger number of arrivals suffering serious congestion. For the studied case, comparing with Poisson's,

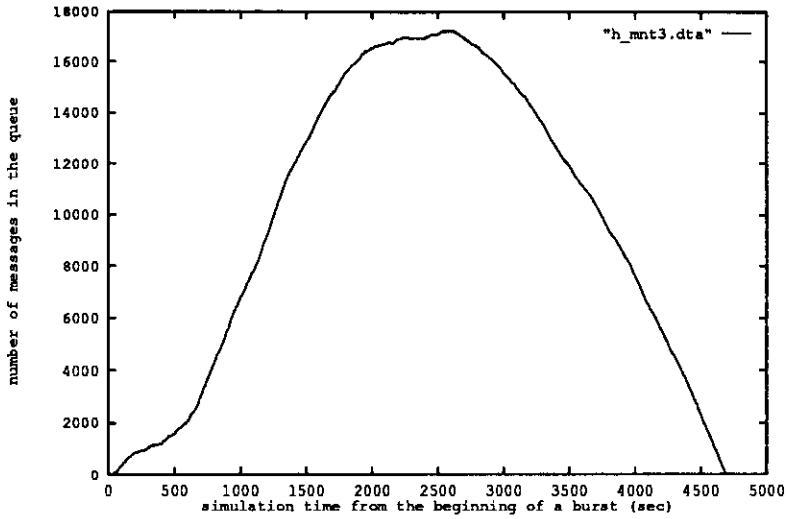

Figure 34: The Peak Busy Period

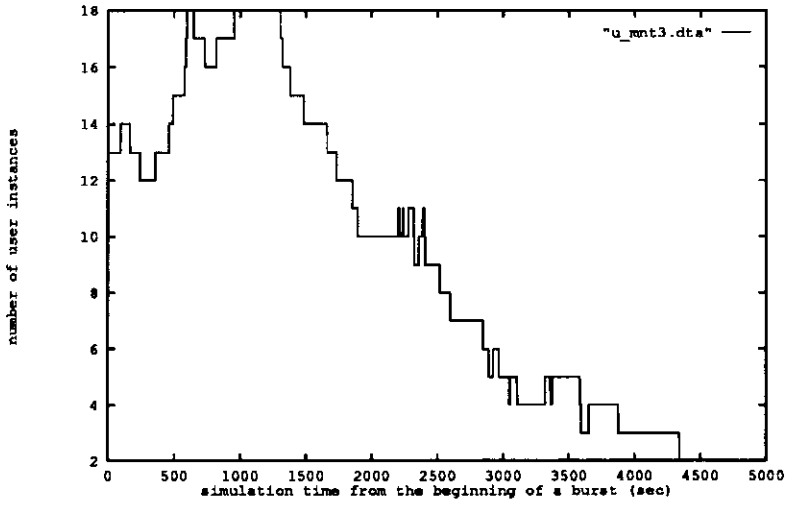

Figure 35: User Arrivals and Departures in the Peak Busy Period

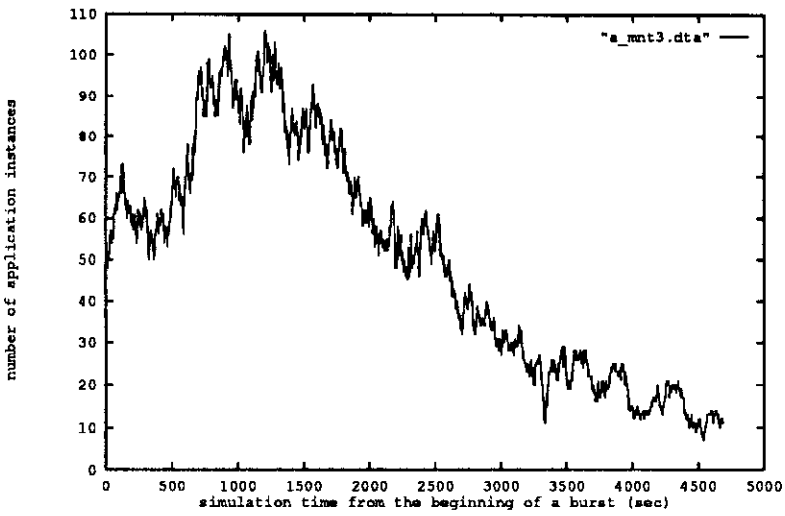

Figure 36: Application Arrivals and Departures in the Peak Busy Period

Mean/Variance	Busy period	Idle period	Height
HAP	0.194/47.189	0.157/0.224	1.717/110.795
Poisson	0.148/0.076	0.121/0.015	1.695/1.662

$\overline{\lambda} = 8.25,\ \mu$ (message) = 15

Figure 37: Busy and Idle Periods: HAP versus Poisson

HAP's variance for busy period is extremely high, while its variance for height is medium high. This implies that this HAP have a very larger number of arrivals suffering medium waiting times (many medium high mountains with very long widths).

The average delay increases dramatically due to the occasional long bursts. But why does HAP tend to generate such long bursts? The explanation is that HAP compounds correlated processes into one process to generate arrivals, which increases burstiness. As these processes are correlated, to the same parent process for example, the chance that they are active simultaneously is much higher. This is totally different from multiplexing independent arrival processes, which reduces burstiness.

5.5 Adjusting HAP Parameters

We now use Solution 1 and 2 to study various topics: comparing HAPs with 2-state MMPPs, levels of modulating processes, arrival versus departure processes, bounding the numbers of users and applications, and multiplexing HAP with non-HAP traffic. Here we are interested in observing the *trend* of the results by adjusting HAP parameters, not the quantitative differences. Solution 1 and 2 are not good approximations when the utilization is over 30%. For many of the studies, we use the set of parameters described at the beginning of Section 4 as a starting point to adjust different parameters.

5.5.1 Comparison of HAP/M/1, MMPP/M/1, and M/M/1

For a 2-state MMPP, the burstiness increases as the gap between λ_1 and λ_2 increases. One interesting question is: for two MMPPs with the same $\overline{\lambda}$, whether a MMPP with high probability staying at high rate or the one with high probability staying at low rate has a higher burstiness. We try the following two cases: $(\lambda_1 = 10, \lambda_2 = 20, r_{12} = 0.1, r_{21} = 0.025, \mu = 40)$ and $(\lambda_1 = 15, \lambda_2 = 30, r_{12} = 0.025, r_{21} = 0.1, \mu = 40)$ (both have $\overline{\lambda} = 18$). The result shows that the second case has higher burstiness. That is, for MMPPs of the same $\overline{\lambda}$, the ones with a large gap between λ_1 and λ_2 and high probability staying at low rate have higher burstiness. Figure 38 is a table with three combinations.

In fact, we can create an artificial MMPP for some specified burstiness. There are also procedures to fit a MMPP to a measured trace [Gus91] or superposition of packetized voice sources [HL86] so that their first, second, and third moments are matched. However, HAP parameters can be easily calibrated naturally without any artificial fitting techniques. Besides, it requires further studies to see if the performance results by the resulting MMPP are accurate under all traffic characteristics.

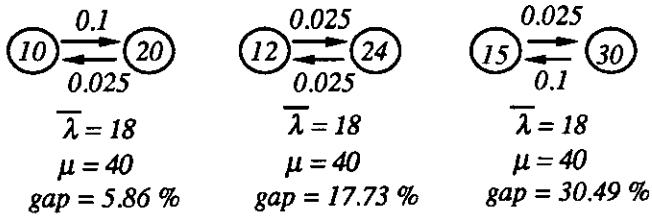

Figure 38: Delay gaps with Poisson for three MMPPs

5.5.2 Levels of Modulating Processes

Starting from the original set of parameters, we keep increasing and decreasing, one at a time, arrival and departure rates, by 5%, of processes at user, application, and message levels. In Figure 39, we find that adjusting λ' and λ'' has larger impact on burstiness than adjusting λ. Note that the X axis is $\overline{\lambda}$ in order to have compare their behaviors under the same $\overline{\lambda}$. It is interesting that λ' and λ'' have the same effect on burstiness. On the other hand, adjusting λ has a larger impact on $\overline{\lambda}$ than adjusting λ' which has also a larger impact than adjusting λ''. The results show that adjusting upper-level arrival processes has more impact on $\overline{\lambda}$ while changing lower-level arrival processes has more impact on burstiness. The explanation is that adjusting λ'' directly affects the message arrival rate. Adjusting λ, from Equation 4 which is a good approximation when the parameters satisfy our three constraints, should have the same effect on $\overline{\lambda}$. The result suggest that there is still a minor difference which desires further explanation. The results for the starting point at high load still hold although the differences vanish. The only exception is that there is no difference between the influences of λ' and λ'' on burstiness, which requires further studies and explanation.

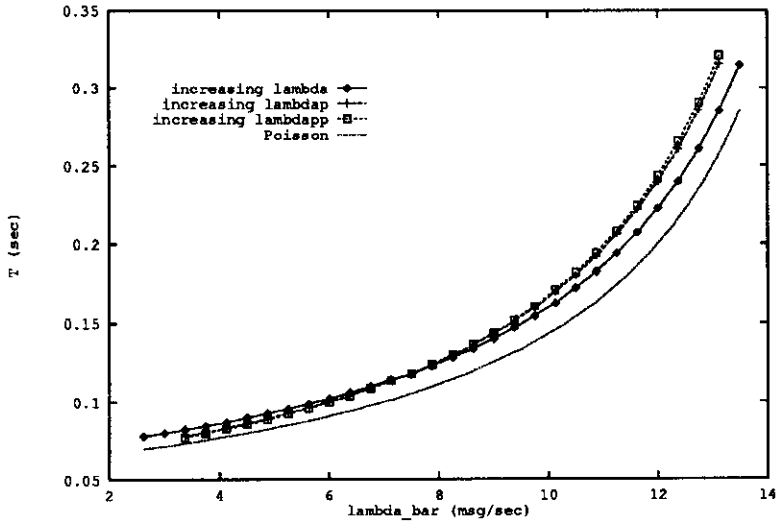

Figure 39: Levels of arrival processes

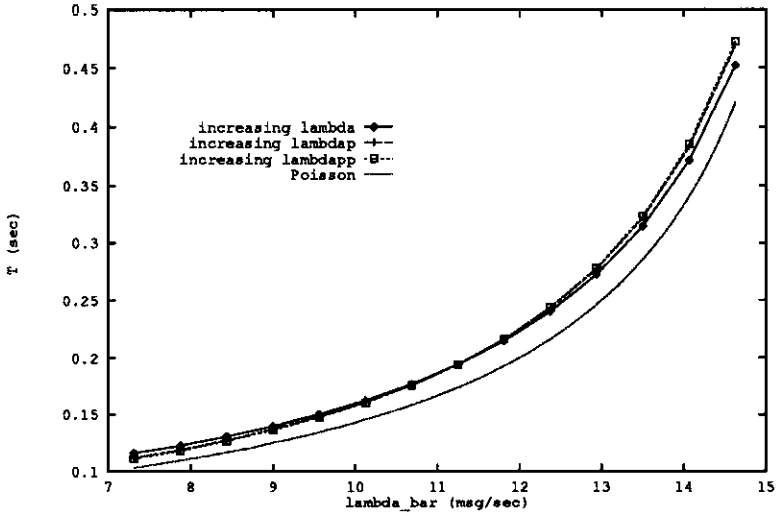

Figure 40: Levels of arrival processes during heavy load

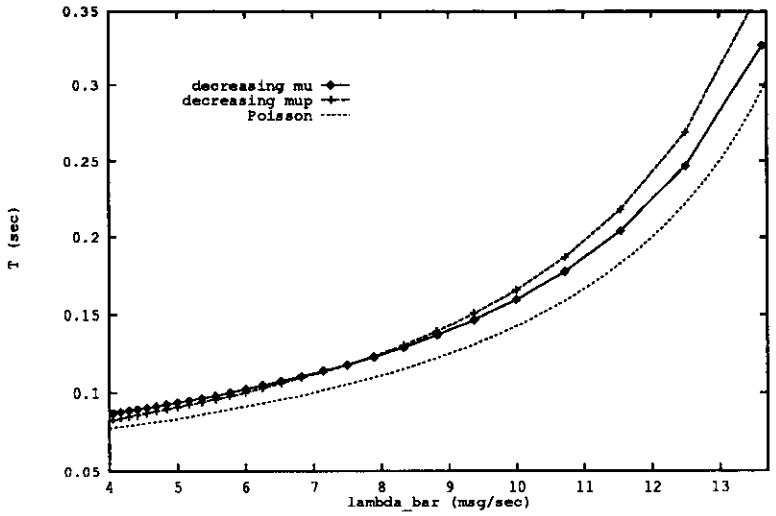

Figure 41: Levels of departure processes

The same observation also applies to the departure processes as shown in Figure 41 where μ'' is not compared because it is a capacity parameter instead of a loading parameter. When we adjust μ'', we find that its impact on the gap between HAP and Poisson is dramatic. As shown in Figure 42, when we reduce the server capacity, HAP's delay increases much faster than Poisson. The gap can be over 25%, which doubles the one for starting point. On the other hand, increasing the server capacity reduces this gap. The insight is that HAP's performance is more sensitive to the server capacity than Poisson. Underestimating the required bandwidth for the given HAP workload results in more serious performance violation than underestimating the one for Poisson workload.

μ''	9.35	11.90	17.0	22.95	36.55
σ	84.22%	68.82%	50%	38.17%	24.54%
gap	25.37%	18.59%	11.89%	8.88%	5.33%

$\overline{\lambda} = 7.5$

Figure 42: Effect of server capacity on the gap

5.5.3 Arrival and Departure Processes

For arrival and departure processes at the same level, adjusting any one has the same effect on burstiness. Two curves for delay versus $\overline{\lambda}$ simply coincide. However, adjusting departure rate has larger impact on $\overline{\lambda}$ than adjusting arrival rate. This can be easily shown by Equation 4. Take the application level with $\lambda' = 0.1$ and $\mu' = 0.1$ as an example. Increasing λ' by 10% and decreasing μ' by 10% result in two set of parameters, respectively: $(\lambda' = 0.11, \mu' = 0.1)$ and $(\lambda' = 0.1, \mu' = 0.09)$. The former has a factor of $\frac{0.11}{0.1}$, in $\overline{\lambda}$ as shown in Equation 4, which is slightly smaller than the latter which has a factor of $\frac{0.1}{0.09}$. Thus, adjusting μ' has a larger impact on $\overline{\lambda}$.

One interesting question is: what if we adjust arrival and departure processes at the same level simultaneously by the same factor? From Equation 4, $\overline{\lambda}$ remains the same. However, the burstiness differs. The result shows that increasing both by the same factor of 10% decreases the delay by about 1% and vice versa. The interpretation is simple. The arrivals that come frequently but go quickly generate shorter bursts than the arrivals, with equivalent $\overline{\lambda}$, that come infrequently but stay longer.

5.5.4 Effect of Flow control

One simple flow control scheme on HAP is to limit the numbers of current users and applications. We know that this scheme for sure will reduce $\overline{\lambda}$. But will it reduce the burstiness? Figure 43

says it will and it reduces more as $\overline{\lambda}$ increases. The bounds we put for the numbers of users and applications are 12 and 60, respectively, while originally they are set to 60 and 300 that are large enough for computing the unbound cases.

This result suggests that a simple admission control is effective in reducing delay and provides a chance to support a higher $\overline{\lambda}$ for a given delay criteria. The interpretation of this result is that bounding the numbers of users and applications also bounds the burst length. As we can foresee, bounding the number of applications should be more effective than bounding the number of users. However, this simple admission control can not bound the number of messages. That is, we have no control at the message level.

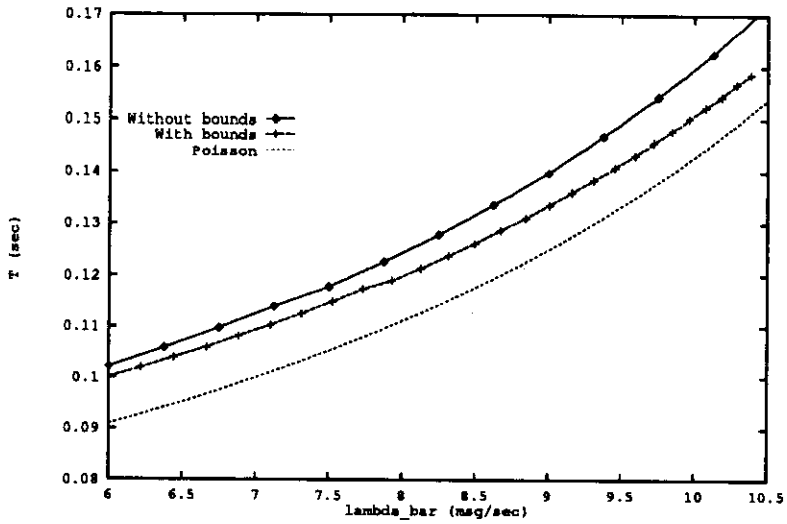

Figure 43: Effect of bounding users and applications

5.5.5 Multiplexing HAP with Non-HAP

The last topic to investigate is the multiplexing effect of HAP traffic with non-HAP traffic. The results here are simulation results, not analytical results. We take Poisson traffic as our non-HAP traffic. The server capacity is fixed at $\mu'' = 17$ and the mean arrival rate at $\overline{\lambda} = 8.25$.

In Figure 44, HAP has delay of 3.2 seconds and Poisson has delay of 0.11 seconds when each of them is injected into the server without multiplexing. When each of them accounts for 50% of the input traffic, HAP's delay drops to 0.65 seconds and Poisson's delay increases to 0.3 seconds. That is, we can significantly reduce HAP's delay if the multiplexed non-HAP traffic can tolerate moderate delay increase. Given a delay criteria, we can determine the optimal percentage of multiplexed HAP traffic. This result suggests the way to deal with HAP's high burstiness under high server utilization.

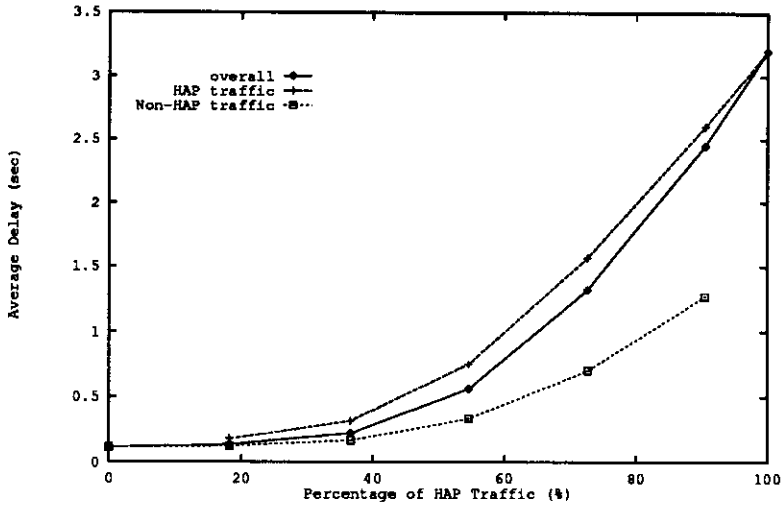

Figure 44: Multiplexing HAP with Non-HAP, $\overline{\lambda} = 8.25$

6 Implications for Broadband Network Control

We now summarize the results and discuss their implications for broadband network control. HAP traffic is very bursty. Congestion may persist for minutes. HAP generates long bursts from time to time and cause prolonged and serious congestions. This phenomena is not observed by previous formal traffic models, but is observed in the real traffic measurements [FL91]. HAP can serve as the computational base to estimate the admissible workload for a given bandwidth (admission control), or the required bandwidth for a given workload (bandwidth allocation).

The delay gap between HAP and Poisson increases significantly as the server utilization increases, which means HAP is very sensitive to the allocated bandwidth. Misengineering with underestimated bandwidth requirement results in a serious performance degradation which is much worse than what we can predict by the Poisson model. In high-speed networks, allocating appropriate bandwidth is much more effective than allocating more buffer space to reduce delay and loss [FL91]. For the studied HAPs, HAP's average delay is only tens of percentage higher than Poisson's if the utilization is under 30%. For this level of utilizations, fast computations are feasible by our Solution 2.

Burstiness increases as the gap between the arrival rates of neighboring states in the underlying Markov chain increases. That is, there is a change in message arrival rate when a state transition happens. Since state transitions in HAP only happen between neighboring states, HAP's burstiness is limited. However, as the number of states is infinite, the message arrival rate can change significantly as the process navigate this huge Markov chain. Reducing the number of states reduces the

burstiness. One simple way is to limit the number of users and applications. By the same argument, if a HAP contains very heterogeneous applications, which results in big gaps between neighboring states, burstiness is increased. The implication is not to multiplex heterogeneous applications on the same channel if these applications are *correlated* under the HAP. However, multiplexing HAP traffic with *independent* non-HAP traffic smooths out HAP's burstiness. Multiplexing is a way to have both high server utilization and low delay of HAP traffic.

To reduce burstiness at the message arrival level, we can design the end-to-end protocol to reduce the message arrival rate, which reduces the burst length at message level. A common scheme is window flow control. Now the only uncontrolled part is the message length. For message types like images and files, a block operation which transmits messages in smaller blocks by window flow control can be used [Ste90]. In fact, this block operation is used in NFS (Network File Systems) and X protocol. As images and files are considered the most bursty message types and they are connectionless applications, this block operation should remain. For ATM networks, however, control is only applied to the connection-oriented applications. Control on connectionless applications mentioned here have to be implemented on end stations.

7 Conclusion and Future Work

Traffic characterization is the fundamental problem confronted by network protocol designers and researchers. The importance of this issue will become more obvious in the future broadband packet-switching networks where algorithms for bandwidth allocation, admission control, and congestion control are based on the underlying traffic models.

We proposed HAP as a packet arrival process in computer networks. HAP's hierarchical structure truly reflects the correlation structure in packet streams. As a special class of MMPP, HAP is a variable-bit-rate model whose bit rate is determined by a $multi-dimension\ infinite-state$ Markov chain where transitions only happen between neighboring states. Compared to the commonly used two-state MMPP model where the physical meaning of these two states is unknown, HAP is superior in modeling packet traffic. However, the queueing analysis for HAP is especially difficult. We developed a brute-force solution and two approximate solutions. The brute-force solution needs extensive computation, while the approximate solutions lose some correlation when we write down the function for HAP's message interarrival time distribution and feed it into the G/M/1 solution. It is shown that this correlation loss is a serious problem and the approximate solutions are good only when the utilization is about 30% or less. We further run a series of simulations to verify the results and observe HAP's short-term behavior.

Some surprising results are obtained. For the given set of parameters, HAP's mean queueing delay can range from 15% higher than Poisson's when the utilization is about 28%, 10 times higher

than Poisson's when the utilization is about 44%, to 80 times higher than Poisson's when the utilization reaches 55%. This result is valid for both brute-force solution and simulations. In order to explain why HAP's delay can be so high, we observe HAP's short-term behavior by looking at its time series graphs of number of messages in the queue. These graphs show that large *mountains* occur from time to time. The highest "mountain" has height over 15,000 messages and width over 40 minutes. There are many mountains with height over 1,000 messages and width over 5 minutes. These large "mountains" cause the average delay go up dramatically and explain the observations in a trace-driven simulation study [FL91].

In that study, it shows that congestion persists and loss can be significant during congested periods, and consequences of misengineering can be serious. HAP, as a formal model, explains these phenomena in the real traffic. However, our study does not show that most congested periods are preceded by signs of impending danger. These large "mountains" are normally preceded or followed by very small "mountains". From this study, a general observation is drawn. An arrival process compounding many *correlated* processes tend to be more bursty, while multiplexing *independent* arrival processes together reduces burstiness.

Despite of the loss of some correlation in the approximate solutions, we use it to explore the change of delay patterns when we adjust HAP parameters. This is because that we are interested in the *trend* of delay change instead of how much it would change. Some results are as follows. Upper-level modulating processes has larger impact on the average bit rate while lower-level modulating processes has larger impact on burstiness and, in turn, delay. Adjusting departure rate of a modulating process has larger impact on bit rate than adjusting arrival rate. Simple flow control by limiting the user and application instances significantly reduces burstiness.

We believe that the work reported here will foster a substantial amount of future work in the areas of hierarchical packet traffic modeling. In this section, we identify the research issues that deserve further work.

An efficient analytical solution for HAP's queueing behavior is necessary to enable HAP to play a role in real-time control algorithms like bandwidth allocation and admission control. Matrix-geometric solutions are worth of pursuing. Transitions of HAP's Markov chain only happen between neighboring states, which means its transition matrix has a diagonal structure and some fast computation techniques can be applied. However, limited speedup in computation does not totally resolve the real-time control problems. Some approximation techniques might be necessary.

HAP's simulation is difficult to converge. However, many transient behavior and statistics can be obtained from the simulation. As the analysis part does not cover the protocol behavior, simulation is the resort to study the the feasibility of the protocols designed for specific applications.

Another issue is HAP's role in admission control of connection-oriented (CO) services and

design problem of connectionless (CL) overlay network in ATM networks. In a HAP, interactive, file transfer, and image transfer will use CL services for transmission through an ATM network, while real-time applications like voice and video will use CO services. Suppose that we use the HAP model to compute the admissible number of CO connections for each application type from the given HAP parameters and the performance requirement. A linear approximation technique [Hui88] can be used to compute the admissible call region. If we store this admissible call region in a admission decision table of each ATM network interface, the admission decision for an incoming VC (Virtual Connection) or VP (Virtual Path) request can be made by a table lookup. Further study on how to calibrate the HAP parameters for each application and the accuracy of this linear approximation is desired.

Interconnection of LANs/MANs to ATM networks is one of the first services to be offered in B-ISDN [BCR89]. CCITT Recommendations [CCI91A,CCI91B] describe the concept and the direct provision of CL capabilities within B-ISDN. Given the physical ATM network and the HAP parameters for both CL and CO applications, we can design the CL overlay network for CL services subject to a performance requirement. This problem is fairly complicate as it involves the dynamic interaction between CO and CL traffic. HAP can serve as a good model for either CL or CO traffic. We can model CL and CO traffic generated from LANs/MANs as 3-level HAPs while modeling some CO traffic like telephone traffic as 2-level HAPs (with only application and message levels).

References

[BCMP75] Baskett, F., K. M. Chandy, R. R. Muntz, and F. G. Palacios, *Open, Closed, and Mixed Networks of Queues with Different Classes of Customers,* Journal of the ACM, 22(2), 248-260, 1975.

[BCR89] Bellcore Communications Research, *Generic System Requirements in Support of Switched Multi-Megabit Data Service,* Bellcore Technical Advisory, TA-TSY-00772, October, 1989, Supplement 1, December 1990.

[BMK88] Boggs, D. R., J. C. Mogul and C. A. Kent, *Measured Capacity of an Ethernet: Myths and Reality,* Proceedings of SIGCOMM, August 1988.

[CCI91A] CCITT, *B-ISDN Service Aspects,* Recommendation I.211, Geneva, 1991.

[CCI91B] CCITT, *B-ISDN Functional Architecture,* Recommendation I.327, Geneva, 1991.

[CL66] Cox, D. R. and P. A. Lewis, *The Statistical Analysis of Series of Events,* Methuen, London, 1966.

[Cru87] Cruz, R. L., *A Calculus for Network Delay and a Note on Topologies of Interconnection Networks,* Ph.D. Dissertation, Report No. UILU-ENG-87-2246, University of Illinois, July 1987.

[DL85] Daigle, J. N. and J. D. Langford, *Queueing Analysis of a Packet Voice Communication System,* Proceedings of INFOCOM '85, p.18-26, 1985.

[DL86] Daigle, J. N. and J. D. Langford, *Models for Analysis of Packet Voice Communication Systems,* IEEE Journal on Selected Areas in Communications, 4(6):847-855, September 1986.

[Eck79] Eckberg, A. E., Jr., *The Single Server Queue with Periodic Arrival Process and Deterministic Service Time,* IEEE Transactions on Communications, 27(3):556-562, March 1979.

[FL91] Fowler, H. J. and W. E. Leland, *Local Area Network Traffic Characteristics, with Implications for Broadband Network Congestion Management,* IEEE Journal on Selected Areas in Communications, vol. 9, No. 7, September 1991.

[Gus90] Gusella, R., *A Measurement Study of Diskless Workstation Traffic on an Ethernet,* IEEE Transactions on Communications, vol. 38, No. 9, September 1990.

[Gus91] Gusella, R., *Characterizing the Variability of Arrival Processes with Indexes of Dispersion,* IEEE Journal on Selected Areas in Communications, vol. 9, No. 2, pp.203-211, February 1991.

[Hef80] Heffes, H., *A Class of Data Traffic Processes - Covariance Function Characterization and Related Queueing Results,* vol. 59, No. 6, The Bell System Technical Journal, 1980.

[HL86] Heffes, H. and D. M. Lucantoni, *A Markov Modulated Characterization of Packetized Voice and Data Traffic and Related Statistical Multiplexer Performance,* IEEE Journal on Selected Areas in Communications, vol. SAC-4, No. 6, pp.856-868, September 1986.

[Hui88] Hui, J. Y., *Resource Allocation for Broadband Networks,* IEEE Journal on Selected Areas in Communications, vol. 6, no. 9, pp. 1598-1608, December 1988.

[Jac57] Jackson, J. R., *Networks of Waiting Lines,* Operations Research, 5, 518-521, 1957.

[JR86] Jain, R. and S. A. Routhier, *Packet Trains – Measurements and a New Model for Computer Network Traffic,* IEEE Journal on Selected Areas in Communications, vol. SAC-4, No. 6, September 1986.

[Kle75] Kleinrock, L., *Queueing Systems, Vol. I: Theory,* pp.101-102, pp.176, pp.251-252, John Wiley & Sons, New York, 1975.

[Kle76] Kleinrock, L., *Queueing Systems, Vol. II: Computer Applications,* pp.320-322, pp.422-484, John Wiley & Sons, New York, 1976.

[KW89] Kochan, S. G. and P. H. Wood, *UNIX Networking,* p.93-133, p.285-337, Hayden Books, Indiana, 1989.

[Kue89] Kuehn, P. J., *From ISDN to IBCN (Integrated Broadband Communication Network),* Proceedings of the World Computer Congress IFIP '89, pp. 479-486, San Francisco, 1989.

[LW91] Leland, W. E. and D. V. Wilson, *High Time-Resolution Measurement and Analysis of LAN Traffic: Implication for LAN Interconnection,* INFOCOM 91.

[Lin93] Lin, Y. D., *Network Traffic Patterns: Models and Heuristics,* Ph.D. Dissertation, University of California, Los Angeles, June, 1993.

[LTHG93] Lin, Y. D., T. C. Tsai, S. C. Huang, and M. Gerla, *HAP: A New Model for Packet Arrivals,* To appear in Proceedings of ACM SIGCOMM '93, San Francisco, September 1993; also to appear in Journal of High Speed Networks in late 1993.

[LHDJ92] Lin, Y. D., S. Holmgren, C. Druce, F. James, *Traffic Pattern Reproduction and Manipulation for LAN-to-LAN SMDS Experiments,* Proceedings of ICC '92, Chicago, 1992.

[Mag87] Maglaris, B., D. Anastassiou, P. Sen, G. Karlsson, and J. Robbins, *Performance Analysis of Statistical Multiplexing for Packet Video Sources,* Proceedings of IEEE GLOBECOM, Tokyo, Nov. 1987.

[Mag89] Maglaris, B., P. Sen, and Rikli, *Models for Packet Switching of Variable-bit-rate Video Sources,* IEEE Journal on Selected Areas in Communication, vo. 7, no. 5, June 1989.

[Mil91] Milazzo, P. G., *Shared Video under UNIX,* USENIX, Nashville, TN, Summer 1991.

[Neu78A] Neuts, M. F., *The M/M/1 Queue with Randomly Varying Arrival and Service Rates,* Opsearch, 15, No. 4, pp.139-157, 1978.

[Neu78B] Neuts, M. F., *Further Results on the M/M/1 Queue with Randomly Varying Rates,* Opsearch, 15, No. 4, pp.158-168, 1978.

[Neu81] Neuts, M. F., *Matrix-Geometric Solutions in Stochastic Models, An Algorithmic Approach,* John Hopkins University Press, Baltimore and London, 1981.

[Nic90] Nicolaou, C., *An Architecture for Real-Time Multimedia Communication Systems,* IEEE Journal on Selected Areas in Communications, vol. 8, no. 3, April 1990.

[Paw88] Pawlita, P. F., *Two Decades of Data Traffic Measurements: A Survey of Published Results, Experience and Applicability,* Proceedings of the 12th International Telecommunication Conference, Torino, June 1988.

[San91] Santifaller, M., *TCP/IP and NFS Internetworking in a UNIX Environment,* p.55-69, p.121-157, Addison-Wesley, 1991.

[Sch88] Schoute, F. C., *Simple Decision Rules for Acceptance of Mixed Traffic Streams,* Proceedings of the 12th International Teletraffic Congress, Torino, 1988.

[SH80] Shoch, J. F. and J. F. Hupp, *Measured Performance of an Ethernet Local Network,* Communications of the ACM 23, 12, 711-721, December 1980.

[SL88] Song, C. and L. H. Landweber, *Optimizing Bulk Data Transfer Performance: a Packet Train Approach,* Proceedings of ACM SIGCOMM, 1988.

[SW86] Sriram, K. and W. Whitt, *Characterizing Superposition Arrival Processes in Packet Multiplexers for Voice and Data,* IEEE Journal on Selected Areas in Communications, vol. SAC-4, No. 6, pp.833-846, September 1986.

[Ste90] Stevens, W. R., *UNIX Network Programming,* p.204-209, Prentice Hall, 1990.

[TP91] Terek, R. and J. Pasquale, *Experiences with Audio Conferencing Using the X Window System, UNIX, and TCP/IP,* USENIX, Nashville, TN, Summer 1991.

[VR89] Virtamo, J. T. and J. W. Roberts, *Evaluating Buffer Requirements in an ATM Multiplexer,* Proceedings of GLOBECOM '89, Dallas, November 1989.

[VPV88] Verbiest, W., L. Pinnoo, and B. Voeten, *Statistical Multiplexing of Variable Bit Rate Video Sources in Asynchronous Transfer Mode Networks,* Proceedings of GLOBECOM '88, p.208-213, Hollywood, FL, November 1988.