HAP: A New Model for Packet Arrivals

Ying-Dar Jason Lin, Tzu-Chieh Tsai, San-Chiao Huang and Mario Gerla
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024

Abstract

Applications to be supported on broadband networks exhibit a wide range of traffic statistics and many of them are sensitive to delay and loss violations. To accurately estimate admissible workload and bandwidth requirement, a detailed traffic model, HAP (Hierarchical Arrival Process) is proposed in this paper. Packets generated from HAP are modulated by processes at user, application, and message levels. This model is a generalization of on-off traffic models and is shown to be equivalent to a special class of MMPP (Markov Modulated Poisson Process). Three algorithmic methods along with simulations are applied to evaluate the queueing performance under HAP traffic. Delay under HAP traffic can be well over tens of times higher than Poisson traffic, depending on parameters and load. 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

Ever since the early development of ARPANET [1], much efforts have been devoted into both measurements and modeling of computer network traffic and performance. In the measurement sector [2], the measurement environments range from wide area networks [1, 3] to local area networks [4, 5, 6]. In the modeling sector, the analytic traffic models fall mainly in the categories of Poisson and MMPP (Markov Modulated Poisson Process) [7, 8, 9].

A recent trace-driven simulation study [6] however 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, 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.

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 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 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 resource allocation. 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 [10, 11, 12] 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.

The paper is organized as follows. Section 2 formally presents the HAP model and its parameters. In section 3, we first show that HAPs are multi-dimension, infinite-state MMPPs. Three solutions are used to analyze HAP's queueing performance. Section 4 presents the numerical results regarding the accuracy of the approximate solutions and the queueing performance of HAP/M/1. Both long-term and short-term behaviors are studied. HAP parameters are adjusted in section 5 to study some topics. In section 6, we give implications for broadband network control. In-progress work on HAP is briefed in section 7.

2 The HAP Model

2.1 HAP with user, application, and message levels

In this section, we present the HAP model. 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 1 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
(In Fig 1, 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.

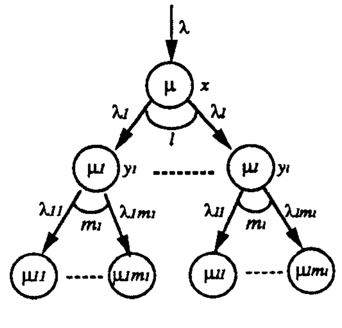

Figure 1: The HAP model

In the terminology of the object-oriented methodology, Figure 1 is a *containment hierarchy* of the *object class* "user". 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, $i = 1 \,..\, l$". Each "application i, $i = 1 \,..\, l$" 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 2 is an example of two instances of the "user" class where two nodes overlap if they belong to the same class/type.

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 3 where message instances of the same color are of the same message type.

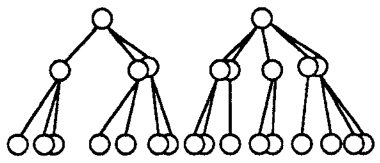
Figure 2: Two user instances of HAP class

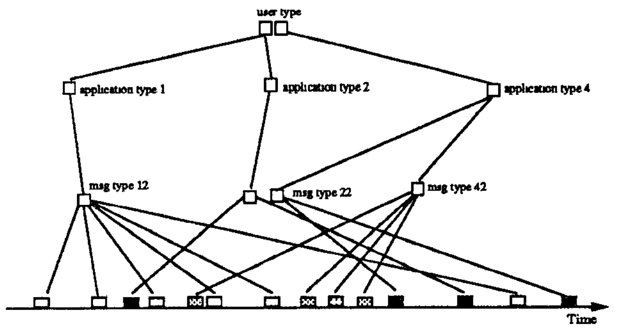

Figure 3: Decomposing a message stream into a HAP

2.2 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 and responses are sent back 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 4 with the additions of the parameters,

μ^c_{ij} : service time distribution for request message type j of application type i
μ^s_{ij} : service time distribution for response message type j of application type i
p^c_{ij} : probability that request message type j of application type i will trigger a response
p^s_{ij} : probability that response message type j of application type i will trigger next request

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

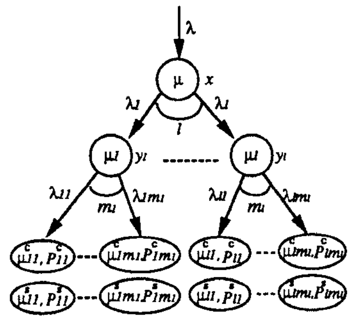

Figure 4: HAP with client-server interaction

2.3 Example HAPs

Figure 5(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.

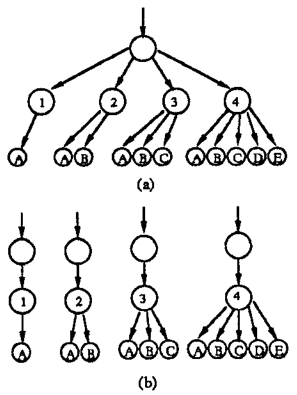

Figure 5: Example HAPs

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 5(b), we divide the original HAP into four HAPs representing four heterogeneous user types.

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. 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.

3.1 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 [14, 15]. Recently, 2-state MMPP has been used extensively to approximate a superposition of various arrival processes like data, voice, and video traffic [7, 8, 9]. A complicated algorithmic solution is applied to a 2-state-MMPP/G/1 queue [8]. 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 6 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 7, 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).

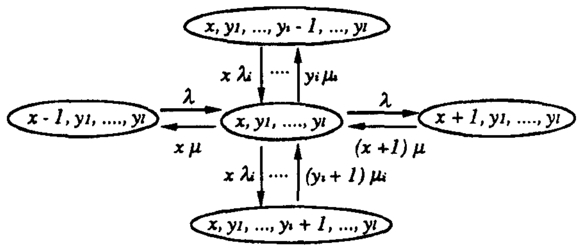

Figure 6: Embedded Markov chain for HAP

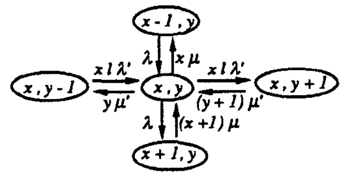

Figure 7: Simplified embedded Markov chain for HAP

3.2 Solutions to HAP/M/1

3.2.1 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

$$P(x, y_1, ..., y_l, z)$$

$$\times[\lambda + x\mu + \sum_{i=1}^{l}(x\lambda_i + y_i\mu_i) + \mu'' + \sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}]$$

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

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

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

$$+(y_i+1)\mu_i P(x, y_1, .., y_i + 1, .., y_l, z)]$$

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

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.

3.2.2 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 6. 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

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

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

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

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 [16]; 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 $W(y) = 1 - \sigma^* e^{-\mu''(1-\sigma^*)y}$.

3.2.3 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.

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/∞ [16], 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 8 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).

Mean numbers of users and applications

As pointed out, the arrivals and departures of user and application instances can be modeled as M/M/∞.

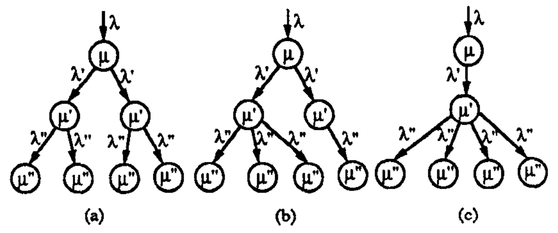

Figure 8: Three HAPs with equivalent mean arrival rate

Thus, the mean number of user instances, $\overline{x}$, is $\frac{\lambda}{\mu}$ and the mean number of application instances, $\overline{y}$, is

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

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, $i = 1 \ldots l$. When HAP is in state $(x, y_1, \ldots, y_l)$, 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

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

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

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

$$\times\sum_{i=1}^{l}y_i\sum_{j=1}^{m_i}\lambda_{ij}\}.$$

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

$$1-\sum_{x=0}^{\infty}\sum_{y_l=0}^{\infty}\cdots\sum_{y_2=0}^{\infty}[\frac{x\lambda_1}{\mu_1}e^{-\sum_{j=1}^{m_1}\lambda_{1j}t}\sum_{j=1}^{m_1}\lambda_{1j}e^{\frac{x\lambda_1}{\mu_1}e^{-\sum_{j=1}^{m_1}\lambda_{1j}t}}$$

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

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

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

$$1 - \sum_{x=0}^{\infty} e^{x \sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t}}$$

$$\times x\left(\sum_{i=1}^{l} \frac{\lambda_i}{\mu_i} e^{-\sum_{j=1}^{m_i} \lambda_{ij} t} \sum_{j=1}^{m_i} \lambda_{ij}\right)$$

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

which is reorganized to

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

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

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

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

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

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

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

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

$$\times [L(t)N(t) + L(t)M^2(t) + \frac{\lambda}{\mu} L^2(t) M^2(t)], \textit{where} \tag{10}$$

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

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 [17]. 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.

4 Numerical Results

We now present some results on HAP. 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 other. 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.

4.1 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, $\sum_{i=1}^{l} y_i \sum_{j=1}^{m_i} \lambda_{ij}$, is large compared to the state transition rate of the Markov chain in Figure 6 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 dut 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%. When utilization is over 30%, the approximations start to drift far away from the exact results by Solution 0 and simulations. 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.

4.2 Message interarrival time

For the same traffic level, HAP exhibits a higher degree of burstiness than Poisson. In Figure 9, 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 $\frac{1}{\overline{\lambda}}$, 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 10.

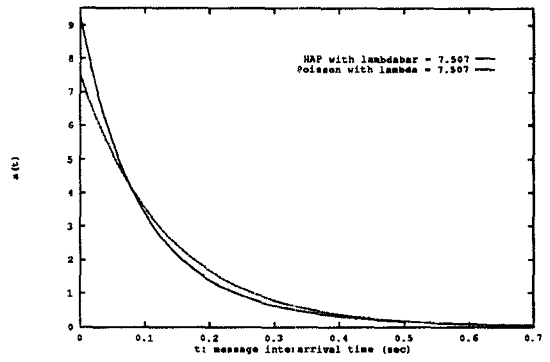

Figure 9: Message interarrival time distribution

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

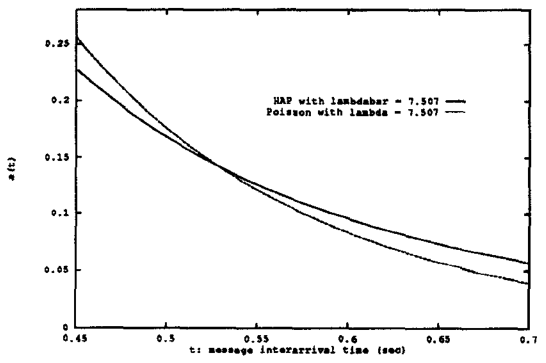

Figure 10: Tail of message interarrival time distribution

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.

4.3 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 11 and 12, 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 11, when μ'', 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 12, we adjust the load, by changing λ, 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, $\bar{\lambda} = 8.25$

Figure 11: Average Delay versus Server Capacity

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 12: Average Delay versus Message Arrival Rate

4.4 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 13 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.

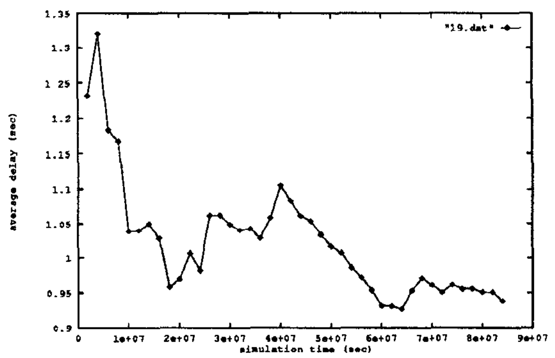

Figure 13: Fluctuation of HAP Simulation

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 14 (in a one-hour period) where the number of messages in the queue is traced. In the extreme case as shown in Figure 15, 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 16 and 17, 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.

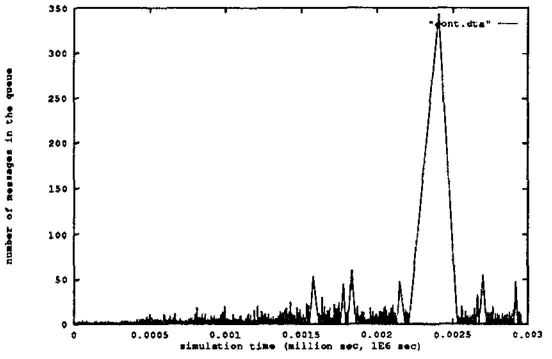

Figure 14: Arrivals and Departures in a One-hour Interval

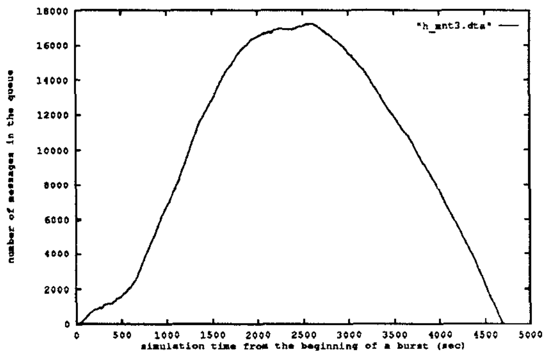

Figure 15: The Peak Busy Period

Figure 18 lists the statistics of busy/idle period and height of the mountains (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.

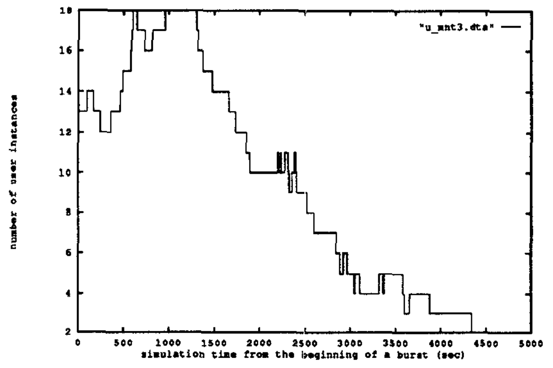

Figure 16: User Arrivals and Departures in the Peak Busy Period

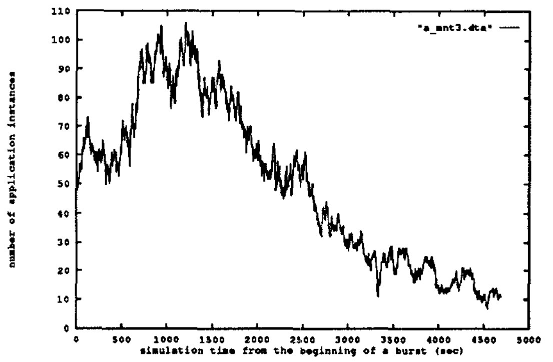

Figure 17: Application Arrivals and Departures in the Peak Busy Period

Heights of the mountains affect the *waiting times* of new arrivals, while *lengths* of *busy periods* determine the *numbers* of arrivals that will have waiting times of these orders. 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, 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.

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

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

Figure 18: Busy and Idle Periods: HAP versus Poisson

5 Adjusting HAP Parameters

We now use Solution 2 to adjust HAP parameters to study several topics: levels of modulating processes, arrival versus departure processes, and bounding the numbers of users and applications. Since Solution 1 and 2 are not good approximations when the utilization is over 30%, we are interested in observing the *trend* of the results by adjusting HAP parameters, not the quantitative differences. We use the set of parameters described at the beginning of Section 4 as a starting point to adjust different parameters.

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 19, 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 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 upper-level arrival processes has more impact on $\overline{\lambda}$ while lower-level arrival processes has more impact on burstiness. The explanation is that adjusting λ'' directly affects the message arrival process. The same observation also applies to the 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. 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, users or applications, that come frequently but go quickly generate shorter bursts than the arrivals, with equivalent $\overline{\lambda}$, that come infrequently but stay longer.

Effect of admission control

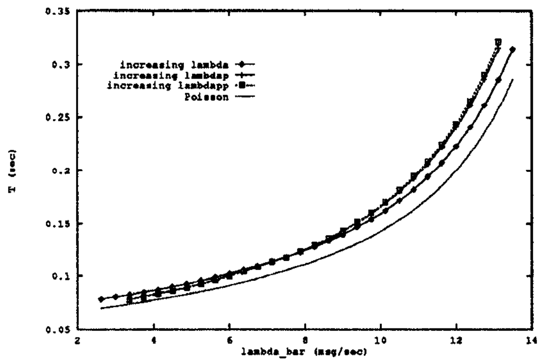

Figure 19: Levels of arrival processes

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 20 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. However, this simple admission control can not bound the number of messages. That is, we still have no control at the message level.

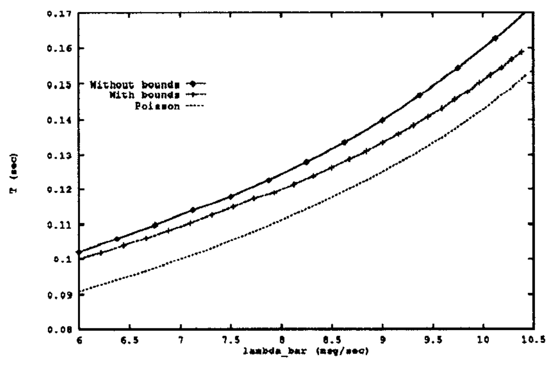

Figure 20: Effect of bounding users and applications

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. It is the only formal model, so far, that exhibits the same congestion phenomena in the real network. 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 [6]. 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. To reduce burstiness at the message level, we can design the end-to-end protocol, window flow control for example, to reduce the message arrival rate, which reduces the burst length at message level, and block operations, by fragmenting messages into blocks along with window flow control, to reduce the burst length.

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. The less bursty applications will suffer a lot. By the same argument, due to HAP's extreme burstiness, multiplexing HAP traffic with non-HAP traffic should be avoided, especially when the non-HAP traffic is some real-time application. More numerical results are required to justify this implication.

7 Conclusion and Future Work

In this paper, we introduce and formalize a new class of traffic model – HAP. HAPs have both short-term and long-term burstiness and correlation. We analyze its queueing delay with three solutions and simulations. Basic numerical results on message interarrival time and queueing behavior are reported. HAP's behavior matches the real network performance. With th results on delay patterns, we give implications on broadband network control.

There are several promising directions for future research. We are currently studying the effects of adjusting HAP parameters. This includes the effects by dimensioning HAP (changing its structure) and by mutiplexing HAPs with non-HAP traffic. 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.

Finally, we discuss 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 [10] can be used to compute the admissible call region. If we store this admissible call region in an 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. Interconnection of LANs/MANs to ATM networks is one of the services to be offered in B-ISDN. CCITT Recommendations [18, 19] 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 complicated as it involves the dynamic interaction between CO and CL traffic. HAP can serve as a good model for either CL or CO traffic. Further study is on-going.

References

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

[2] 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.

[3] Heimlich, S. A., *Traffic Characterization of the NSFNET National Backbone,* Proceedings of the 1990 Usenix Conference, January 1990.

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

[5] Gusella, R., *A Measurement Study of Diskless Workstation Traffic on an Ethernet,* IEEE Transactions on Communications, vol. 38, No. 9, pp.1557-1568, September 1990.

[6] 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, 1139-1149, September 1991.

[7] 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.

[8] 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.

[9] 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.

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

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

[12] 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.

[13] 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, pp.986-995 September 1986.

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

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

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

[17] 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.

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

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