OVERLOAD PERFORMANCE OF AN ADAPTIVE, BUFFER-WINDOW ALLOCATION SCHEME

FOR A CLASS OF HIGH SPEED NETWORKS

Bharat Doshi
HO 3M-335
AT&T Bell Laboratories
Holmdel, NJ 07733
USA

Harry Heffes
Department of EE and CS
Stevens Institute of Technology
Hoboken, NJ 07030
USA

Main Contribution.
We discuss an adaptive buffer-window allocation scheme to be used at the burst level for large file transfers over wide area high speed packet networks. The scheme is analyzed using matrix analytic techniques and is shown to have many desirable self-adaptation properties.

1. INTRODUCTION

Packet switched networks supporting bursty data sources are typically engineered and operated to provide traffic concentration for economical operation. This concentration necessitates real time congestion avoidance and congestion control mechanisms in addition to call set up controls and network management functions. The primary objectives of these real time controls are: avoid or minimize congestion related packet losses, and retransmissions in the event of a packet loss; and to treat the active users fairly. One approach used successfully in narrowband packet switched networks with virtual circuit based operations is to dedicate a buffer of size equal to the working window for the call holding time of a virtual circuit and to serve the active virtual circuits in a round robin (RR) manner with small quantum size [11]. As the speeds of the network and applications increase by orders of magnitude, two factors play important roles: on the one hand, the working windows increase dramatically requiring enormous buffers to support the above strategy; on the other hand, with the usual go-back-N protocol in end systems, the bandwidth loss in the event of a packet loss increases dramatically making buffer dedication even more desirable. A possible solution for resolving these conflicting objectives is to reserve the working window's worth of buffers only when a virtual circuit is active and to adapt the working window (and buffer allocation) based on the network status. This approach, which avoids congestion induced packet losses, involves two levels of call set up.

The routing and virtual circuit set up functions are performed at the first level. The second level involves a reservation request from a virtual circuit ready to transmit bulk data. The network, based on its current status, determines the appropriate working window size and communicates this to the end system. Simultaneously, each of the network nodes allocates buffers equal to this working window size.

The allocation is kept during the activity burst and is removed when the virtual circuit becomes idle. Since the working window required for each virtual circuit to keep the trunk utilization high is a decreasing function of the number of active users sharing a trunk, the working window and buffer allocated is made a decreasing function of the total buffer currently allocated, thus providing a self-adaptation mechanism. Active VCs are still served using an RR discipline. A similar scheme is discussed in [8].

The paper is organized as follows: The congestion avoidance and allocation schemes are more precisely defined in Section 2, and in Section 3 we present a quasi-birth-death model for the basic allocation scheme. The model is analyzed in Section 4, using matrix analytic methods which exploit the structure of the underlying model. In addition to obtaining steady state results, we present a first passage time analysis which is useful in evaluating the transient congestion control behavior. Numerical results are presented in Section 5 for a variety of performance measures, including reservation blocking and the carried allocation process. The first passage time results of Section 5 provide both quantitative and qualitative insight into the transient adaptation properties of the control mechanism. Concluding remarks, implementation considerations and further work are discussed in Section 6.

2. CONGESTION AVOIDANCE AND BASIC ALLOCATION SCHEMES

Suppose a virtual circuit, about to initiate a large file transfer, sends a request for a large window size \( W_f \) for use during the actual transmission of the file. As the reservation packet traverses its path in the network, each node calculates the maximum buffer size it can assign to this virtual circuit, based on the status of its buffers. If the calculated value is smaller than the value indicated in the reservation packet, the node modifies the reservation packet to include the smaller window-buffer size. The receiver thus obtains the reservation packet with the minimum allocation specified. The reservation packet acknowledgement can carry this information back to the transmitter, which will begin transmission with the specified window. The network nodes may read this information and adjust their allocations accordingly.

In this basic scheme, we assume that the allocation is not changed during the period of reservation. Of course, if a virtual circuit, for which buffers are reserved, goes idle, then its reservation is cancelled. If it goes active again, the new reservation request will be granted based on the network status at that time. Thus, the congestion avoidance scheme has some inherent self adaptation...
properties. Full windows, \( W_1 \), are granted during periods of light activity and smaller windows are granted during periods of heavy activity.

More precisely, each node (trunk module) calculates the buffer size and window allowed for a new request as follows: Let \( B \) be the total buffer size available for reservation, and let \( \gamma \) denote the currently reserved portion of the buffer (i.e., \( B - \gamma \) is available for allocation). Consider threshold levels \( K_1 < K_2 < \cdots < K_m = B \) and feasible window-buffer allocations \( W_1 > W_2 > \cdots > W_m \). Let \( n = (n_1, n_2, \ldots, n_m) \) denote the allocation state, where \( n_i \) is the number of virtual circuits presently allocated a window of size \( W_i \). Note that the allocation state, in general, is a much more slowly varying quantity than the actual occupancy of the buffers (i.e., during the time a given virtual circuit has an allocation \( W_i \) the actual number of queued packets for this virtual circuit can vary considerably). A new reservation request which finds the allocation in state \( n \) receives an allocation of size \( W_i \) kilobytes (KB) for this node if

\[
(n + \epsilon)W^T \leq K_i \quad \text{and} \quad (n + \epsilon_{i-1})W^T > K_{i-1},
\]

where

\[
W = (W_1, W_2, \ldots, W_m)
\]

and \( \epsilon_i \) is a unit row vector in direction \( i \). Note that, since \( \gamma = nW^T \), we allocate \( W_i \) if

\[
W_i + \gamma \leq K_i \quad \text{and} \quad W_{i-1} + \gamma > K_{i-1}.
\]

Finally, if

\[
(n + \epsilon_m)W^T > K_m = B
\]

the reservation request is denied (blocked). We note that the threshold levels and window sizes should be chosen in such a way as to keep the trunk busy and provide a new reservation request its fair share of the trunk bandwidth. Furthermore choosing window sizes \( W_i \) with a power of two convention may simplify implementation.

The above scheme allows different virtual circuits to have different buffer allocations and window sizes at a given time. This is not necessarily unfair, from the throughput point of view, if the virtual circuits are served in a round robin fashion by the trunk module, if the thresholds are appropriately chosen and if the smallest window, \( W_m \), is large enough. The reader is referred to [4,6,7] for a discussion of fairness in data communication networks. It is, however, possible (but should be engineered to be improbable) that a given virtual circuit has a small window in the absence of virtual circuits with large windows. This can occur, for example, if many large window virtual circuits become idle during the residency time of the given virtual circuit. Of course, one can design in the option of renegotiating the window size under such occurrences.

3. THE BUFFER-WINDOW ALLOCATION MODEL

In this section we present a simple model for a single buffer (corresponding, possibly, to a bottleneck trunk) operating with the above policy. The basic assumptions are that reservation requests for full windows, \( W_1 \), arrive as a Poisson process with rate \( \lambda \). We further assume that, in the absence of contention from other virtual circuits, the reservation duration is exponentially distributed with mean \( \mu^{-1} \). Finally, if a total of \( N \) virtual circuits have reservations, we assume that each receives \( \frac{1}{N} \) of the trunk bandwidth (i.e., the trunk is processor shared amongst virtual circuits with reservations and each virtual circuit queue is non empty). As will be seen, the methodology allows for generalization of the last assumption.

Under the above stated conditions, the model corresponds to a continuous-time Markov chain. If we denote \( n, W \) and \( \epsilon_i \) as above, let \( l = [1, 1, \ldots, 1] \), and define the allocation indicator function by

\[
\delta_l(n) = \begin{cases} 
1 & \text{if } (n + \epsilon_l)W^T \leq K_l \text{ and } (n + \epsilon_{l-1})W^T > K_{l-1}, \\
0 & \text{otherwise},
\end{cases}
\]

then the steady state probabilities, \( p(n) \), satisfy the following balance equations. For \( n \) a non-blocking state, i.e., \( nW^T + W_m \leq K_m = B \), and \( n \neq 0 \), we have

\[
(\lambda + \mu)p(n) = \lambda \sum_{i=1}^{m} p(n - \epsilon_i)\delta_l(n - \epsilon_i) + \mu \sum_{i=1}^{m} \frac{n_i + 1}{n} p(n + \epsilon_i).
\]

(2a)

For \( n \) a blocking state, we have

\[
\lambda p(n) = \mu \sum_{i=1}^{m} p(n - \epsilon_i).
\]

(2b)

and for \( n = 0 \), we have

\[
\lambda p(n) = \mu \sum_{i=1}^{m} p(n - \epsilon_i).
\]

(2c)

Figure 1 shows an illustration of the state transition rate diagram corresponding to the case of two window sizes, \( W_1 = 256 \text{ KB} \) and \( W_2 = 128 \text{ KB} \), with thresholds \( K_1 = 512 \text{ KB} \) and \( K_2 = 1024 \text{ KB} \). We have defined the level of the process to be the number of virtual circuits with small windows allocated (i.e., \( n_2 \)). Note that there are three blocking states correspond to levels 4, 6 and 8. The use of the processor sharing discipline should also be evident from the figure.

4. STEADY STATE AND FIRST PASSAGE TIME ANALYSIS

For simplicity of presentation, we present the results for two window sizes and note that the more general case \( (m \geq 2) \) can be handled similarly [10]. To analyze this model we define the level of the Markov process as \( n_2 \) and denote the steady state probability vector corresponding to level \( i \) as

\[
\pi_i = [p_{0i} \cdots p_{Ni(0,i)}]
\]

(4-1)

where \( N_i(i) \) represents the maximum value \( n_i \) can assume in level \( i \). For the illustration in Figure 1, \( N_i(i) = 2, i = 0, \ldots, 4 \); \( N_i(i) = 1, i = 5, 6 \); and \( N_i(i) = 0, i = 7, 8 \). The steady state probability vector for the overall process is then given by

\[
\pi = [\pi_0, \pi_1, \cdots, \pi_{N_2}],
\]

(4-2)

where \( N_2 \) is maximum level attainable. With these definitions, \( \pi \) satisfies

\[
\pi Q = 0,
\]

(4-3)

and

\[
\pi L^T = 1,
\]

(4-4)

with the block tridiagonal infinitesimal generator of the form
The infinitesimal generator $Q$, which corresponds to a quasi-birth-death (QBD) process, is shown in Figure 2 for the illustration of Figure 1. Note the form of the infinitesimal generator $Q$ and the normalization condition.

The above QBD process falls in the class of finite birth and death models in randomly changing environments treated in [5], from which the following algorithm for evaluating the steady state probabilities is obtained.

(i) Compute the matrices $\overline{C}_n$ ($n = 0, \ldots, N_2$) from the backward recursion

$$\overline{C}_n = A_n + L_n(-\overline{C}_{n+1}) M_{n+1},$$

and final condition

$$\overline{C}_{N_2} = A_{N_2}.$$  

(ii) Evaluate $\pi_n$ from

$$\pi_0 \overline{C}_0 = 0,$$

the forward recursion

$$\pi_n = \pi_{n-1} L_{n-1} (-\overline{C}_{n-1}), \quad 1 \leq n \leq N_2,$$

and the normalization condition,

$$\sum_{n=0}^{N_2} \pi_n \overline{C}(\pi_n) + 1 = 1,$$

where $\overline{C}$ is a $j$-vector of all ones.

We note that the $(k, j)$th element of the matrix $-\overline{C}_{n-1}$ represents the average time the QBD process spends in the state $(n, j)$ before reaching level $n-1$, given that the process started in state $(n, k)$ [5]. Thus,

$$[-\overline{C}_{n-1}]_{kj} \geq 0,$$

and (4-8) provides a useful check on the computations.

To evaluate some of the transient properties of the congestion avoidance scheme, we turn our attention to a first passage time analysis of the QBD process. In addition to providing quantitative results, we will see that the first passage time perspective can provide qualitative insight into the self-adaptation properties of the control mechanism.

We are particularly interested in the average first passage time to level $j$, given that the system starts in state $(n_1, n_2) = (0, 0)$. Using the notation of [5], we define the first passage time from level $n-1$ to level $m \geq n$,

$$\tau_{n,m} = \inf \{t > 0 : n_2(t) = m \mid n_2(0) = n-1\}.$$  

We further define the matrix $G^{(n,m)}(s)$, with elements $G^{(n,m)}_{ij}(s)$ corresponding to the Laplace-Stieltjes transform of the probability distribution function

$$g^{(n,m)}(s) = P[\tau_{n,m} \leq s, n_1(\tau_{n,m}) = j \mid n_1(0) = i].$$  

Finally, we define the vector $u^{(n,m)}$ with components $u^{(n,m)} = E \{\tau_{n,m} \mid n_1(0) = i\},$

for $i = 0, \ldots, N_1(0)$. Note that we are interested in $u^{(n,0)}$.

The algorithm for evaluating expected first passage times is in terms of the matrices $C_n$, $n = 0, 1, \ldots, N_2$. The reader is referred to [5] for their probabilistic interpretation, including the fact that the elements of $-C_n^i$ are all non-negative. The matrices $C_n$ satisfy the recursion

$$C_n = A_n + M_n(-C_{n-1}), \quad 1 \leq n \leq N_2$$

with initial condition

$$C_0 = A_0.$$

In terms of the above quantities, we can now specify the recursions [5] for calculating $u^{(n,0)}$:

$$u^{(n,0)} = -C_{n-1}^{-1} (1^T M_n A_{n-1} u^{(n-1,0)})$$

$$u^{(1,0)} = u^{(0,0)} + G^{(0,0)}(0) u^{(0,0)}$$

$$G^{(1,0)}(0) = G^{(0,0)}(0) (-C_{n-1}^{-1} M_{n-1})$$

with initial conditions

$$u^{(1,0)} = -C_{n-1}^{-1} 1^T.$$

Recall that the expected first passage time from the origin to level $n$ is given by

$$[1, 0, \ldots, 0] u^{(n,0)}.$$  

In addition to expected first passage times, [5] contains results for the second moments of first passage times.

5. NUMERICAL RESULTS

We first present results for the steady state, window allocation probabilities for the illustrative example of Figures 1 and 2, and note the self-adaptive behavior with the offered load $\lambda/\mu$. Figure 3 shows that, for low load ($\lambda/\mu = 0.6$), the probability mass is concentrated on areas of the state space containing large windows whereas, in overload ($\lambda/\mu = 1.6$), Figure 4 shows a shift in the probability mass to the small window region of the state space. The self-adaptive behavior will be further demonstrated when we study "the carried allocation characteristic" for a larger problem.

It should be noted that there are regions of the state space which are "undesirable". For the illustrative example, $n_1 = 0, n_2 = 1$ is such a state, since the single active virtual circuit will have a reduced throughput due to an insufficient window. We first note that this reduced throughput can be incorporated into the model by reducing the death rate, in Figure 1, from this undesirable state. We further note that the probability of being in an undesirable ("bad") state can be made sufficiently small by the appropriate choice of buffer size and thresholds. Of course, in overload, the undesirable state, probabilities become smaller. For example, for $\lambda/\mu = 1.6$ the probability associated with $n_1 = 0, n_2 = 1$ is a half of one percent.
If, on the other hand, one finds oneself in such a state, a larger window can always be negotiated.

Let us now turn our attention to a larger problem in order to illustrate the transient performance, in terms of the first passage times, and the self-adaptive behavior, in terms of the carried allocation process. In particular, we consider a buffer of size $B = 3,072$ KB, with two window sizes of 256 KB and 64 KB. With thresholds of $K_1 = 1536$ and $K_2 = B$, we can have a maximum of 6 active virtual circuits with large windows and of maximum of 48 active virtual circuits with small windows. For this example, the shift in the probability mass from large windows to small windows, as the load is increased, is even more pronounced than that shown in Figures 3 and 4. In Figure 5 we have chosen to demonstrate this dramatic self-adaptive behavior by observing the "carried allocation processes", $E(n_1)$ and $E(n_2)$ as a function of the offered load $(\lambda/\mu)$. Note the different scales and the sharpness of the characteristics. Figure 6 shows that the blocking probability as a function of the offered load exhibits an excellent behavior with about 2% loss at offered load equal to 1. Figure 7 shows the first passage time behavior of this scheme. In particular, it shows the expected time for the system to move from the state (0,0) to a state in which $n_2$ virtual circuits are allocated the smaller window, when the offered load is 2.0. It also shows this first passage time behavior for a scheme in which only small windows are allocated. The latter represents the 'ideal' from an adaptation speed perspective. While the average first passage time tracks the 'ideal' very well up to $n_2 = 30$, we see that the adaptation beyond that is somewhat sluggish. In particular, the system stays in, or below, states corresponding to $n_2 = 44$, $n_2 = 40$, etc. for relatively long periods of time. This, of course, corresponds to having 1 and 2 virtual circuits with large windows, respectively. With a large number of other virtual circuits to contend with, they slow down and occupy the buffers for a long time and block new requests from getting the smaller window. We will discuss a method to speed up the adaptation in Section 6. Figure 8 shows the marginal probability distributions of $n_1$ = number of virtual circuits with the smaller window, for an offered load equal to 2.0. Figures 7 and 8 show the importance of a transient congestion control analysis to uncover important phenomena not observable in the steady state results.

6. CONCLUSIONS AND RELATED WORK

We have provided a methodology to study the performance of an adaptive, buffer-window allocation scheme and presented results for the special case which uses two threshold values and two window sizes. The file lengths are assumed to be exponentially distributed. The results show that this semi-dynamic scheme exhibits very desirable self adaptation behavior allowing relatively small buffers without additional signaling needed for more dynamic adaptive schemes. A few issues, not addressed here, are presented elsewhere:

(i) Recall that the balance equations presented in Section 3 are valid for an arbitrary number of window sizes (and thresholds) and thus can be solved for this general case. We have presented the analysis of the case with two window sizes here. A recursive scheme for the case with more window sizes (and threshold values) has been developed [10].

(ii) Another interesting performance measure (besides the blocking probability, probability of a 'bad' state and adaptation speed) is the sojourn time distribution. The sojourn time here, refers to the time it takes to complete a file transfer. The distribution of the sojourn time, coupled with the blocking probability performance, is useful in deciding the maximum number of virtual circuits to be supported (and hence the buffer size) to achieve the right tradeoff between utilization and file transfer delay. The analysis of the sojourn time distribution is presented in [9,10].

(iii) The results presented here correspond to exponential file lengths (and hence service times). In addition, it is assumed that the reservation and cancellation procedures are instantaneous. A more variable service time (say, hyperexponential) is likely to affect the adaptation speed and the 'bad' state probability. Also, the time to complete the reservation procedure is at least one round trip delay. The same is true of the time to complete cancellation. During these periods, buffers are allocated but not used. Inclusion of these delays would make the efficacy of the adaptive scheme depend on the relative values of the service time and round trip propagation delay. An approximate analysis and a detailed simulation model of the case including general service times and propagation delay are presented in [1] along with the results for relatively long file transfers (average service time equals 20 to 40 round trip propagation delays).

(iv) When the file transfers are relatively short (average service time smaller than the round trip propagation delay), the effect of propagation delay is very significant. Also, such short file transfers typically occur in bursts: During 'active' periods, a number of short files are transferred by a virtual circuit but with short idle periods between file transfers. Between such 'active' periods, there are relatively long idle periods. Two alternatives exist: a) reserve the buffer/window for each file transfer; b) reserve the buffer/window for each 'active' period. Analysis methodology and representative results for utilization-delay tradeoffs and buffer sizing are presented in [2].

(v) We saw that the transient behavior of the adaptive scheme studied here could exhibit some undesirable behavior although its overall impact in steady state is minimal. On the other hand, the results in [1] show that even the steady state probability of a 'bad' state may not be ideal when the service times are highly variable. A more dynamic scheme is needed in this case. A simple enhancement to the scheme discussed here could grant the reservation for a fixed period (say, T seconds) of time. If a virtual circuit needs to transmit beyond that, it will go through another reservation request (a renewal request) at which time it may get a smaller, larger or the same window/buffer allocation. The results for such an adaptive scheme with reservation renewal will be presented in future.

(vi) The schemes studied here and in [1,2] reserve buffer resources and adapt windows to match the allocation. Such reservations can also be made for bandwidth parameters. Analysis of some reservation schemes for bandwidth parameters and comparison of these schemes with schemes involving buffer reservations can be found in [3].

REFERENCES


Figure 1: State Transition Rate Diagram - Illustration

Figure 2: Infinitesimal Generator - Illustration: $\beta = \lambda + \mu$

Figure 3: Steady State Window Allocation Probabilities

(Offered Load = 0.6)

$p(n_1, n_2)\quad n_2$

Figure 4: Steady State Window Allocation Probabilities

(Offered Load = 1.6)
Average First Passage Time from (0,0)

\[
\rho_0 = 2 \quad (K_1 = 1536)
\]

\[
\rho_0 = 2 \quad (K_1 = 0)
\]

Only Small Windows Allocated

Level - \( n_2 \)

\( E (n_2) \)

<table>
<thead>
<tr>
<th>Level - ( n_2 )</th>
<th>( E (n_2) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

Figure 5: Carried Allocation Processes

Figure 7: Transient Behavior - Adaptation Speed

Figure 6: Blocking Characteristic

Figure 8: Marginal Probabilities