Real-Time Cell Arrival Sequence Estimation and Simulation for IP-over-ATM Networks*

Hiroshi SAITO†, Toshiaki TSUCHIYA†, Daisuke SATOH†, Gyula MAROSI††, Gyorgy HORVATH††, Peter TATAI††, and Shoichiro ASANO†††, Regular Members

SUMMARY We have developed a new traffic measuring tool and applied it to the real-time simulation of a network. It monitors IP traffic on an ATM link and continuously transfers the length and timestamp of each IP packet to a post-processing system. The post-processing system receives the data, estimates the cell's arrival epoch at the transmission queue of the ATM link, and simulates the queueing behavior on-line if conditions differ from those of the actual system. The measuring tool and real-time simulation represent a new approach to traffic engineering. A new estimation problem, the arrival sequence estimation, is shown and some algorithms are proposed and evaluated. Also, a new dimensioning algorithm called the queue decay parameter method, which is expected to be robust and applicable to real-time control, is proposed and evaluated.

key words: real-time simulation, traffic engineering, ATM, IP

1. Introduction

Internet traffic is growing rapidly world-wide, and the proliferation of new applications is causing its characteristics to change. Meanwhile, Asynchronous Transfer Mode (ATM) has begun to be deployed in internet backbone networks, several types of WAN services, and fast broadband private networks for companies within a group. In such new telecommunication environments, traffic engineering faces two difficulties. One is the limited capability to monitor and measure traffic in network elements such as ATM switches and routers. While capturing traffic characteristics is one of the most important issues for economical development and evaluation of new technologies, extra functions in network elements to monitor and measure traffic would raise the cost of the network element and reduce the processing power for call control and/or packet forwarding. Thus, existing network elements do not give traffic engineers all the information about the traffic characteristics that they would like, while the traffic characteristics are becoming more and more complicated and the link and processor capacities are increasing. Even when we can monitor traffic, traffic engineering (including traffic control, management, and dimensioning) is still a challenge in an ATM backbone for Internet traffic. This is the second difficulty. In particular, empirical tests (with theoretical background) are more preferable than pure theoretical approaches in traffic engineering these days. Thus, modeling and characterizing traffic may not be enough to show the effectiveness of a proposed traffic engineering scheme. This paper shows how these two challenging issues can be overcome by developing traffic measuring tools outside network elements and passing its data to a real-time on-line simulator.

In addition to the measurement tool and simulator, this paper also proposes a promising link capacity dimensioning algorithm. The algorithm's accuracy is verified by the simulation tool using actual traffic measured by the developed measurement tool.

Overall, we propose traffic engineering steps using real-time on-line simulation: In the first step, the number of network resources such as link capacity or buffer size is dimensioned by a certain method. (The dimensioning algorithm may use traffic data measured at an ATM switching node and may be executed in a network management system. Actually, it is possible to apply a simple dimensioning algorithm that uses and keeps the target utilization. This is because, the third step checks whether the dimensioning was appropriate.) In the second step, the dimensioned number of network resources is set in the simulator using the actual traffic data continuously. Instead of the dimensioned number, the number planned in the following year's budget, for example, can be used. In the third step, the simulator checks whether the dimensioned or planned number of network resources is appropriate. The fourth step yields the number of network resources that was dimensioned/planned, after it has been checked and modified.
if necessary. (Normally, the dimensioned or planned number may need a certain margin to accommodate the traffic growth by the time the updated network resources are provided. If we can do these four steps in a short period of time and repeat them, we can implement the self-sizing network concept, which can adapt its network size to the ever-changing traffic [5]–[9].)

2. Measuring Tool

We developed a new traffic measuring tool called CapTie (capturing traffic while being tied to a post-processing computer) on the real-time OS called VxWorks [10]. The first version of CapTie [13] ran under MS-DOS because it was based on the header trace mode of OC3MON [1], [2]. OC3MON is an attractive tool that normally works in a statistical mode, which counts packets at a regular interval, e.g., five minutes. However, for our purpose, finer granularity is needed and multiple processes are convenient for remote operation. In addition, the original header trace mode in OC3MON writes data to a disk and stops working when the disk is full. This is inconvenient for us because we intended to simulate a network continuously. Therefore, we completely rewrote CapTie.

CapTie is a program running on a PC tapped at an ATM link. It monitors traffic on the link by using an ATM network interface card (NIC). CapTie can monitor the IPv4 traffic on ATM adaptation layer type 5 (AAL5). When the ATM NIC receives a cell and processes the physical layer, it judges whether it is the first cell of a IP datagram (packet) by checking that the VPI/VCI of the cell is in the last-cell table in CapTie. (A VPI/VCI of a cell is added in this table if the cell is judged as the last cell of a packet by checking the payload type field in the cell header. If the VPI/VCI of the cell is registered in this table and the cell is not the last cell, the VPI/VCI is removed from the table.)

If a cell is judged as the last cell of a packet, CapTie removes the LLC/SNAP encapsulation header and reads the packet length field in the packet header in the cell and generates a record consisting of the following four items: the CapTie counter value, the ATM NIC slot number, and the packet’s timestamp and length. The record generated by CapTie is transferred to another PC that simulates (part of) a network using the record, and CapTie continues working (Fig. 1). As shown later, the first target of the simulated network is the output buffers at the switching node in which the monitored link is accommodated.

The record items are used as follows: (1) CapTie and the post-processing PC are connected via telnet over Ethernet. In our experience, the data rate between them is a few percent of the traffic on the monitored ATM link. Thus, when 100-Mbps traffic on the link is monitored, 1 Mbps or more of data traffic is transferred between the two PCs. Here, there is a danger of some data being lost. The value of CapTie’s counter increases one by one when a packet is monitored. Thus, the post-processing PC can check whether any traffic records are lost. (2) CapTie can monitor more than one link simultaneously. To distinguish the record for the ATM link being monitored, an ATM NIC slot number is used. This allows network simulation to be performed if the PC running CapTie has many slots accommodating ATM NICs. (3) The timestamp is a key item of data for simulation. The timestamp uses 64 bits: 32 bits show the number of seconds and another 32 bits express the fraction of a second; this achieves nano-second granularity. (4) Packet length is another key item of data in the simulation. It determines the amount of traffic.

3. Estimation of an Arrival Sequence

The first goal of our traffic measuring tool is to reproduce the cell/packet arrival sequence at the transmission queue (output buffer) of the output link. (Note that CapTie monitors the cell/packet carried sequence of the link.) If this reproduction succeeds, we can analyze the traffic offered to the transmission queue accurately, simulate the transmission queue behavior, and can evaluate some interesting metrics. Otherwise, the result of the traffic analysis is misleading even when the timestamp has fine granularity. (For example, the peak
traffic never exceeds the link capacity when we observe the traffic transmitted at the link.) However, the accurate reproduction of the cell/packet arrival sequence is not a straightforward task. Since our measurement device is tapped at a link, what we observe is not the cell arrival sequence or the packet arrival sequence but the cell transmission sequence or the packet transmission sequence. Therefore, our measuring device incorporates an algorithm for estimating the arrival sequence from the transmission sequence.

In this section, we investigate algorithms for estimating an arrival sequence from a transmission sequence. Here, the arrival sequence is defined by the series of cell/packet arrival epochs observed just before the output transmission queue in the ATM switching node that has an ingress end point of the monitored link in Fig. 1. The output buffer in the ATM switching node is sometimes called the reference model when we would like to emphasize the comparison between the reference model and the virtual model, which is a model used in the simulation (Fig. 1). In the remainder of this section, UBR (unspecified bit rate) on the monitored ATM link is implicitly assumed because UBR is used for data transmission on ATM in most cases. As a result, cells of a packet are assumed to be offered to the network at the line speed, while packet transmission from an end system may be controlled by an upper layer.

3.1 Simple Cell Arrival Sequence Estimation Algorithm

Assume that CapTie observes the sequence of cells passing a tapping point in the middle of a link and judges whether a cell is the first cell of a packet based on the payload type indicator of the cell header. If it is the first cell of a packet, CapTie reads the packet length field of the IP packet header in the cell and generates a record consisting of the packet length and the timestamp showing when the cell was detected by CapTie. Actually we can observe the time instance when the first cell of each packet passes a certain point of a link and then observe how many cells belong to the same packet and will pass the point. Based on this information, the simple cell arrival sequence estimation (S-CASE) algorithm estimates the arrival instance of each cell at the arrival observation point.

Let $t(i)$ be the observed timestamp of the first cell of the $i$-th packet, and let $v(i)$ be the number of cells belonging to the $i$-th packet. Let $H$ be the capacity of the input link to the ATM switching node that has the ingress end point of the monitored link (Fig. 1) where we assume for simplicity that each input link has the same link capacity. Let $L$ be the length of a cell. The following estimated arrival epoch of the $j$-th cell belonging to the $i$-th packet, $t(i, j)$, is provided by the S-CASE algorithm.

\[ t(i, j) = t(i) + (j - 1)^*L/H \]  

(1)

The S-CASE algorithm assumes implicitly that the first cell does not wait for any time in the transmission queue and that the remaining cells arrive at the input link speed without interruption and do not wait in the transmission queue.

3.2 Busy Period Sensing Cell Arrival Sequence Estimation Algorithm

Assume that we can observe whether or not the output link is busy as well as observing the timestamp of each cell of a packet and the number of cells belonging to the packet. Let $b$ be a busy flag, which is 1 when the output link is busy and 0 when it is idle. This flag can be obtained directly observing the status of the output link or by checking the timestamp of each cell.

The busy period sensing cell arrival sequence estimation (BPS-CASE) algorithm estimates the arrival epoch of the $j$-th cell belonging to the $i$-th packet as follows. It maintains $x$, the number of active flows on the output link, and $T$, the reference time.

(i) The number of active flows, $x$, increases by one when the first cell of a packet is detected and decreases by one when the last cell of a packet is detected. That is, the definition of the number of active flows in this paper is the number of simultaneously existing packets on the link, and it is not defined by using the source (destination) address or source (destination) port number.

(ii) If the link is busy at start time $s$, the reference time $T$ is updated to be $s$. When the first cell of a packet is detected and if the timestamp is $t$, the reference time $T$ is updated to be $t$.

The BPS-CASE algorithm stores the values of $b$, $x$, and $T$ for estimating the arrival epoch of each cell of the $i$-th packet when the first cell of the packet is detected. Let $b(i)$, $x(i)$, and $T(i)$ be their values just before the detection of the first cell of the $i$-th packet. By using them, the BPS-CASE algorithm estimates the arrival epoch of each cell of the $i$-th packet when the last cell of the packet is detected.

Consider the case in which the first cell of the $i$-th packet is detected when the link is idle (i.e., $b(i) = 0$). In this case, this cell does not wait for transmission. Therefore,

\[ t(i, 1) = t(i), \]

where $t(i, j)$ is the estimated arrival epoch of the $j$-th cell of the $i$-th packet and $t(i)$ is the timestamp of the first cell of the $i$-th packet.

Next, consider the case in which the first cell of the $i$-th packet is transmitted during a busy period of the link and assume that the link is kept busy while cells from the first one of the $(i - 1)$-th packet through the first one of the $i$-th packet are detected (Fig. 2). In
Fig. 2 Example of the event sequence (1).

Fig. 3 Example of the event sequence (2).

In this case, $T(i)$ is the transmission time (equivalently, the time stamp) of the first cell of the $(i-1)$-th packet. Because the link is busy between $T(i)$ and $t(i)$, the number of cells between the first cell of the $(i-1)$-th packet and the first cell of the $i$-th packet is given by $(t(i) - T(i))C/L$, where $C$ is the output link capacity (Fig. 3). Since the number of active flows is $x(i)$, the mean time taken for this number of cells to arrive at the transmission queue of the reference model is $(t(i) - T(i))/C$ if each active flow offers cells at the input link capacity $H$ and the number of active flows does not change between $T(i)$ and $t(i)$. Therefore, if the estimate $t(i-1, 1)$ of the arrival epoch of the first cell of the $(i-1)$-th packet is accurate, the arrival epoch of the first cell of the $i$-th packet can be estimated as

$$t(i, 1) = t(i-1, 1) + (t(i) - T(i))C/(Hx(i)).$$

Here we should note that the estimated arrival epoch $t(i, 1)$ of the first cell of the $i$-th packet should be less than or equal to the transmission time $t(i)$ of this cell. To keep this constraint $t(i, 1) \leq t(i)$ for any $T(i) = t(i-1) \geq t(i-1, 1)$, we use $\max(C, Hx(i))$ instead of $Hx(i)$. Consequently, we obtain the following estimation rule that is applicable to both $C > Hx(i)$ and $C \leq Hx(i)$.

$$t(i, 1) = t(i-1, 1) + (t(i) - T(i))C/\max(C, Hx(i)).$$

Finally, consider the case in which the first cell of the $i$-th packet is transmitted during a busy period and the first cell of the $(i-1)$-th packet is not included in this busy period (Fig. 4). In this case, $T(i)$ is the transmission epoch of the first cell forming this busy period. Note that the arrival epoch of this cell is $T(i)$ because this cell does not wait for transmission. Therefore, similarly to the discussion above,

$$t(i, j) = t(i, 1) + (j-1)L/H$$

3.3 Numerical Examples

We investigated the accuracy of the S-CASE and BPS-CASE algorithms through a computer simulation, which simulated the whole system including the reference model and the point where the timestamp is applied (Fig. 5). In this computer simulation, packets were generated according to a Poisson process and their length distribution was assumed to be geometric. They were offered to the transmission queue of the reference model (a FIFO queue). Through observation of the cell transmission sequence, the cell arrival sequence to the transmission queue of the reference model was estimated by the S-CASE and BPS-CASE algorithms. Since we knew the true cell arrival sequence in this computer simulation, we could compare the true and estimated cell arrival sequences. The accuracy was compared in terms of the difference in cell loss ratios (CLR).
Table 1  Simulation conditions and virtual systems.

<table>
<thead>
<tr>
<th>Simulation condition</th>
<th>Average packet length (Cell)</th>
<th>Offered load source (Mbps)</th>
<th>Number of sources</th>
<th>Input speed (Mbps)</th>
<th>Virtual system (buffer size, output link capacity (Mbps))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>30</td>
<td>50</td>
<td>(128, 50), (128, 37), (128, 30)</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>30</td>
<td>150</td>
<td>(128, 50), (128, 37), (128, 50)</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>1</td>
<td>30</td>
<td>50</td>
<td>(128, 50), (128, 37), (128, 30)</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
<td>1</td>
<td>30</td>
<td>15</td>
<td>(128, 50), (128, 37), (128, 30)</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>1</td>
<td>30</td>
<td>150</td>
<td>(128, 50), (128, 37), (128, 30)</td>
</tr>
<tr>
<td>6</td>
<td>50</td>
<td>1</td>
<td>30</td>
<td>50</td>
<td>(256, 50), (256, 37), (256, 30)</td>
</tr>
<tr>
<td>7</td>
<td>50</td>
<td>1</td>
<td>30</td>
<td>50</td>
<td>(256, 50), (128, 50), (384, 50)</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>1</td>
<td>30</td>
<td>50</td>
<td>(128, 50), (128, 75), (128, 37)</td>
</tr>
</tbody>
</table>

that occurred when the true and estimated cell arrival sequences were offered to the virtual system and the reference model. This is explained below. One reason for measuring the accuracy in terms of the CLR is that the final target of the cell arrival sequence estimation is to use it for performance analysis.

By using the actual and estimated cell arrival sequences, the CLR is the reference model and virtual systems in the computer simulation were evaluated, where the virtual systems were defined as a FIFO queue with a different buffer size or a different output link capacity from the reference model. In the following figures, \((k, m, n)\) denotes the \(k\)-th simulation condition, the \(m\)-th virtual system, and the \(n\)-th estimation method. Simulation conditions and virtual systems (their buffer size and output link capacity) are summarized in Table 1. (The 0-th virtual system means the reference model.) The estimation method is 0 when the estimated cell arrival sequence is the true cell arrival sequence; 1 when the estimation method is the S-CASE algorithm; and 2 when it is the BPS-CASE algorithm.

The results for Condition 1 are plotted in Fig. 6. For high CLR or small output link capacity, both algorithms were accurate. As the CLR became lower or the output link capacity became larger, the S-CASE underestimated the CLR.

Figure 7 shows our investigation of Conditions 1 and 3. While the CLR is the true cell arrival sequence were almost the same for \((3, 0, 0)\) and \((1, 2, 0)\), the estimation accuracies were different. Using either estimation algorithm, the CLR is the true CLR of \((1, 2, 0)\). This means that the estimation in \((1, 2, \ast)\) was easy. On the other hand, the CLR is the true CLR of \((3, 0, 0)\). This means that the estimation in \((3, 0, \ast)\) was difficult. This seems to be mainly because it was easier to estimate the CLR of a small output link capacity (with short packets) than that of a large one (with long packets). (Item (v) below shows that long packets made estimation easier. Thus, a small output link capacity seems to be one of the main reasons for this result.)

Figure 8 compares the CLR derived by the estimated and true cell arrival sequences. In addition, the CLR evaluated by the upper bound formula [4], [5] using the mean and peak cell rates of each flow is also shown for comparison. Here, the mean cell rate was
the true mean cell rate and the peak cell rate was the cell rate transmitted by the input link capacity. In this figure, the y-axis denotes the log of the ratio of an estimated CLR to the true CLR. For example, for each simulation condition, S-CASE ((∗,0,1)/(∗,0,0)) means the log of the ratio of the CLR estimated by S-CASE for the virtual system 0 to the true CLR for the virtual system 0. That is, the y-axis denotes a metric of CLR estimation error.

(i) We should note that the estimation was accurate when the capacity of the output link of the virtual system was smaller than that of the reference model, but may be inaccurate when it is larger. For example, for simulation condition 1, the estimation error for virtual system 0 when S-CASE was used (that is, S-CASE ((∗,0,1)/(∗,0,0))) was a large absolute value than the estimation error for virtual system 1 when S-CASE was used (S-CASE ((∗,1,1)/(∗,1,0))). Here, virtual system 1 had less capacity than virtual system 0 (the reference model).

(ii) BPS-CASE is normally more accurate than S-CASE. For example, for simulation condition 1, S-CASE’s estimation error for virtual system 1 (S-CASE ((∗,1,1)/(∗,1,0))) had a larger absolute value than BPS-CASE’s (BPS-CASE ((∗,1,2)/(∗,1,0))).

(iii) Simulation conditions 1 and 5 for virtual system 0 and simulation condition 2 for virtual system 1 produced low CLRs and the estimation was inaccurate with S-CASE for simulation conditions 1 and 5 and with BPS-CASE for simulation condition 2. See S-CASE ((∗,0,1)/(∗,0,0)), S-CASE ((∗,1,1)/(∗,1,0)) and BPS-CASE ((∗,1,2)/(∗,1,0)). It was difficult to estimate a low CLR accurately.

(iv) The accuracy of BPS-CASE deteriorated when the input link capacity was much larger than the output link capacity of the reference model. See BPS-CASE ((∗,1,2)/(∗,1,0)) for simulation condition 2 and compare it with that for simulation condition 1.

(v) BPS-CASE was accurate for simulation conditions 3, 4, and 6. There, the input link capacity was less than or equal to the output link capacity of the reference model and the packet size was long. It was more accurate for simulation condition 6 than for simulation condition 3, while the CLR for simulation condition 6 was lower than for simulation condition 3. (Normally, the estimation error becomes large when the CLR is lower.) This seems to be because the virtual system for simulation condition 6 had a longer buffer than that for simulation condition 3.

(vi) The absolute value of the estimation error obtained by BPS-CASE was always smaller (i.e., more accurate) than that by the upper bound formula.

(vii) We also compared the CLR of each virtual system for each simulation condition derived using the Poisson arrival assumption instead of the arrival sequence estimated by S-CASE or BPS-CASE. Here, the Poisson arrival assumption means that the cells were generated according to a Poisson process with mean equal to the mean observed in the simulation. Since the CLR obtained under the Poisson arrival assumption was smaller than −10 in this figure for all simulation conditions, it was not plotted.

Figure 9 shows our investigation of conditions with different network resources (buffer and link capacity). When there were more network resources in the virtual system than in the actual system, the estimation errors were larger than when there were fewer network resources in the virtual system. BPS-CASE was more accurate and better than S-CASE. In particular, BPS-CASE was accurate even when the buffer size of the virtual system was different from that of the actual sys-
tem.

4. Real-Time Simulation

The post-processing PC, which receives the data from CapTie via Ethernet, uses one of the arrival sequence algorithms mentioned in the previous section. As a result, we can (approximately) reproduce the cell arrival sequence at the multiplexing point (the transmission queue) of a switching node. Therefore, if we provide a simulator in the post-processing PC, provide a virtual system or a model of the transmission queue in the simulator, and give it the reproduced arrival sequence, we can simulate the performance behavior of new controls and new network resource conditions as if they were applied at the transmission queue (Fig. 1).

In this study, we used a simulator to evaluate the accuracy of an output link capacity dimensioning method called the queue decay parameter method, which is described in the following section. This dimensioning method is expected to be applicable to various arrival processes without explicit traffic models and is appropriate even for real-time computation. Therefore, if this dimensioning method is accurate, it will be a good candidate for an algorithm that can be used for the self-sizing network referred in Sect. 1. The simulator in the post-processing PC simulates an output link and its transmission queue in the switching node that has an ingress end-point of the monitored link. We assumed that the transmission queue could be modeled as a single-server FIFO queue with a finite-size waiting buffer where the service was cell transmission.

As a first step, the output link capacity was dimensioned by the proposed dimensioning algorithm, which used the traffic measurement data obtained by the switching node. The dimensioned output link capacity was used for the output link capacity (that is, the service rate of the server) in the simulator and the actual buffer size of the simulated transmission queue was used as the buffer size of the simulator. The estimated and reproduced cell arrival sequences were offered to the simulator in the post-processing PC, and the CLR of the transmission queue (output buffer) of the output link was evaluated in the simulator. In the numerical example below, S-CASE was used in the simulation for simplicity.

5. Queue Decay Parameter Method

Here, we describe a simple and heuristic link capacity dimensioning method based on the buffer occupancy measurement. We call this method the queue decay parameter method. Consider a single server queueing model as a model of the transmission queue in an ATM switching node. From the result of the large deviation principle and related works (e.g., see [6]), the asymptotic behavior of the queue length $Q$ is expressed as

$$p(k) \equiv P\{Q \geq k\} = \alpha \exp(-\beta \nu(k)),$$

where $\alpha$ and $\beta$ are constants (independent of $k$) and $\nu(k)$ is a function of $k$. These parameters depend entirely on the nature of the input traffic. In order to study the input traffic, we made cell-level measurements in a real network in advance and captured the real-time sequence of cell arrivals. At each measurement, we obtained a sample sequence of 1.3 million cell arrivals. The total number of available samples during five days of measurement was 150. With those data, we made a simulation of a FIFO single server queueing model, where the service time was the cell transmission time. (Note that this was not an on-line simulation but one using the recorded cell arrival data repeatedly.) Figure 10 shows the distribution $p(k)$ obtained by the simulation with 44 samples (on the same day). (The $x$-axis denotes $k$, the queue length, normalized by the buffer size.) In this case, the arrival rate of each sample was between 6 and 12 Mbps, and the link capacity was set to 15 Mbps. From the figure, we can observe that the decay parameter $\alpha$ is nearly equal to 1. Moreover, the decay of the queue length seems exponential. Therefore, $\nu(k) = k$ and the traffic is considered to have short-term dependence. At least, in this range of queue length, we can dimension the traffic under the assumption that it has short-term dependence. This argument remained valid for other sample sequences, though we omit similar figures. As a result, we have a simple approximation formula

$$p(k) \cong \exp(-\beta k).$$

Equation (7) is similar to the M/M/1 queueing model with different values of $\beta$. In the M/M/1 case, $\beta = -\log(\rho)$ where $\rho$ denotes the utilization. In Fig. 10, each sample gives the ratio of $-\log(\rho)$ to $\beta$ as being from 6 to 11, which is greater than in the M/M/1 case (because of the burstiness of the input traffic). With the same samples, we tested the values of $-\log(\rho)/\beta$ for different link capacities, varying from 15 to 40 Mbps, in Fig. 11. For each sequence of data, the values of

![Fig. 10](image-url)
−log(ρ)/β were almost constant, where the link capacity was in the range of 15–40 Mbps. From those observations, we finally obtained a simple link capacity dimensioning method using a probability p(k) that the queue length exceeds a certain k as follows.

1. Measure the utilization ρm and probability p(k) that the queue length exceeds a certain k.
2. Estimate the decay rate βm by βm = −log(p(k))/k (see Eq. (7)).
3. With objective cell loss ratio CLRo and buffer size K, estimate the desirable decay rate β∗ that satisfies CLRo = p(K) = exp(−βk) by β∗ = −log(CLRo)/K.
4. Assume that −log(ρ)/β is constant and calculate the utilization ρ∗ when p(K) = CLRo by solving the formula

   −log(ρ∗)/β∗ = −log(ρm)/βm. (8)

5. Dimension the link capacity so that the utilization becomes ρ∗.

When the ratio −log(ρm)/βm in Eq. (8) changes in some range, conservative management is possible with the greatest value, because a larger value will lead to lower utilization ρ∗.

6. Numerical Example of Real-Time Simulation

6.1 Example 1

The developed system was applied to a bi-directional link in SINET (the Science Information Network) [3], which is a nationwide large IP network for research organizations in Japan, whose core network is implemented on an ATM network. The monitored link was between the University of Tokyo and the network office of NACSIS (National Center for Science and Information Systems). As mentioned in the previous section, the capacity of each link was dimensioned using the queue decay parameter method based on data measured in the week before this trial by an ATM switching node accommodating the link. The CLR objective was 10−6.

Figure 12 [13] plots the CLR of each link simulated by the real-time simulator for one hour during a busy hour. The CLR obtained by the simulator agreed closely with the objective. (In our experience, the CLR is a very difficult parameter to manage. Thus, the agreement was actually far better than we expected.)

The direction from the NACSIS network office to the University of Tokyo has a large amount of traffic with long packets because it includes a lot of traffic downloaded from the U.S. The opposite direction has less traffic with shorter packets. The link utilization obtained in the simulation was higher in the direction from the University of Tokyo to the NACSIS than in the opposite direction (Fig. 13). That is, the smaller link had higher utilization than the larger one whereas a larger link can usually achieve higher utilization due to the statistical multiplexing gain. The reason for this unexpected result is that short packets are dominant in the direction from the University of Tokyo to the NACSIS network office (Fig. 14). However, the CLR of this direction had a larger fluctuation than that of the opposite direction because the offered traffic was small (Fig. 12).

Comparing Figs. 12 and 13, we see that high average utilization during ten minutes does not mean a high CLR. This is because traffic fluctuations over a period much shorter than ten minutes strongly affect the CLR. Thus, we cannot estimate the CLR based only on a ten-minute average utilization.
6.2 Example 2

We monitored the carried traffic between the open computer network (OCN) and the network office of NACSIS, which provides a nation-wide internet among research organizations. The OCN is an IP network service provided by NTT Communications. By using the queue simulation implemented in the developed system, we evaluated the VP bandwidth estimated by using the $H_2/D_1/K + 1$ queue model for the output queue of the VP, where $K$ is the buffer size and the queue has the first-in-first-out discipline.

Our system monitored the traffic and calculated the average and the variance of the number of bytes transferred during each minute. Based on this average and this variance, we calculated the loss probability of the $H_2/D_1/K + 1$ for a fixed service rate and found a service rate for which the loss probability is equal to the cell loss ratio (CLR) objective. (To simplify the analysis, we used the large deviation approximation for $H_2/D_1/K + 1$.) In the simulator, we set the VP bandwidth to the newly calculated value for every fifteen minutes and evaluated the CLR of the simulated model. Figure 15 shows the CLR of the simulated model in the simulator when the objective CLR is $1.0 \times 10^{-3}$. (Thus, if the $H_2/D_1/K + 1$ model is accurate, this CLR should be $1.0 \times 10^{-3}$.) The approximated $H_2/D_1/K + 1$ queue analysis gave more bandwidth than the bandwidth which met the objective CLR because the simulated CLR was lower than the objective CLR.

7. Conclusions

We have developed a traffic monitoring tool called CapTie and a real-time simulator that uses data from CapTie. The cell arrival sequence estimation methods used in it were evaluated. Using these two systems, the proposed dimensioning method was evaluated for real IP traffic data on an ATM network and shown to be accurate. The developed system enables us to implement a new approach to network engineering. In the near future, we will apply the system to a self-sizing network.

Acknowledgments

We would like to express our thanks to Joel Apisdorf, who developed OC3MON. Without his help, we would not have started this study. We also thank engineers in NTT Advanced Technology and the staff of NACSIS, who helped us to apply our system to the network. To monitor the traffic between OCN and NACSIS, Kawanahara Ryoichi helped us. We thank him.

References


Hiroshi Saito graduated from the Univ. of Tokyo with a B.E. degree in mathematical Engineering in 1981, and an M.E. degree in Control engineering in 1983 and received the Dr.Eng. in Teletraffic Engineering in 1992. He joined NTT in 1983. Currently he is working on teletraffic issues and network architecture in high-speed networks in NTT Service Integration Labs as a Senior Research Engineer, Supervisor. He received the EIICE young engineer award in 1990, the telecommunication Advance ment Institute award in 1995 and the excellent paper award of the Operations Society in Japan in 1998. Dr. Saito is a senior member of IEEE, and a member of IFIP WG 7.3, and the Operations Society in Japan. He is the author of the book “Teletraffic technologies in ATM networks” (Artech House) and a co-author of the book “Basis of Teletraffic Theory and Multimedia telecommunication” (IEICE), etc.

Toshiaki Tsuchiya received B.S. and M.S. degrees in information science from Tokyo Institute of Technology in 1990 and 1992 respectively. He joined NTT laboratories in 1992. He is currently working for NTT East. His main research interests are in queuing theory and performance evaluations of communications systems.

Daisuke Satoh received the B.E. and M.E. degrees in electronics and communication engineering from Waseda University, Tokyo, Japan, in 1992 and 1994, respectively. He joined NTT in 1994. His research interests include teletraffic issues, software reliability models and integrable systems. Mr. Satoh is a member of the Operations Research Society of Japan, the Japan Society for Industrial and Applied Mathematics, and the Physical Society of Japan.

Gyula Marosi graduated in 1993 at the Faculty of Electrical Engineering, Technical University of Budapest (TUB). Then he joined the Department of Telecommunications and Telematics there. From 1993 to 1997 he was Ph.D. student and now works as a lecturer. He holds courses on programming, computer systems, information and multimedia systems, and participates in many R&D projects. His present research interest includes multimedia systems, telecommunication protocols, and application software development in C.

Gyorgy Horvath graduated in 1988 at the Faculty of Electrical Engineering, Technical University of Budapest (TUB). Since that time he is employed by the Department of Telecommunications and Telematics at TUB. He manages the Department’s network, organizes laboratory training for students, and participates in many R&D projects. His present research interest includes hardware development, FPGA technics, low level programming, and application software development in C.

Peter Tatai graduated in 1964 at the Faculty of Electrical Engineering, Technical University of Budapest (TUB). From 1964 to 1986 he was employed by the Research Institute for Telecommunications, Budapest, from 1976 as the Head of Code Modulation Systems Department. Since 1986 he is with the Department of Telecommunication and Telematics, TUB, where he lectures on telecommunications, measurement technology, and signal processing. He also manages many R&D projects of the Telecommunication and Signal Processing Laboratory there. His present research interest includes digital speech processing and measurements in telecommunications.

Shoichiro Asano had graduated at Electronic Engineering, Faculty of Engineering, The University of Tokyo, in 1970. He received M.E. and D.E., both form the University of Tokyo, in 1972 and 1975 respectively. Dr. Asano is a professor at the National Institute of Informatics (NII) and a Professor at the University of Tokyo. His researches are mainly focused on digital integrated network architecture and development of Science Information Network in Japan. He is a vice chairman of Information, Computer, Communication Policy (ICCP) at OECD.