# Real-Time Cell Arrival Sequence Estimation and Simulator for IP/ATM Networks [⋆]

Hiroshi Saito[1] Toshiaki Tsuchiya[1], Gyula Marosi[2], Gyorgy Horvath[2], Peter Tatai[2], and Shoichiro Asano[3]

[1] NTT Service Integration Laboratories
[2] Technical University of Budapest
[3] NACSIS (National Center for Science and Information Systems)

**Abstract.** 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 rep resent a new approach to traffic engineering. A new estimation problem, the arrival sequence estimation, is shown and some algorithms are proposed and evaluated.

## 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. So, 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

[⋆] Hiroshi Saito, NTT Service Integration Labs., 3-9-11, Midori-cho, Musashino-shi, Tokyo, 180-8585, Japan. E-mail: saito.hiroshi@lab.ntt.co.jp, URL: http://www.hslab.tnl.ntt.co.jp 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.

This kind of approach has received attention recently. DARPA (Defense Advanced Research Projects Agency), through its NGI (Next Generation Internet) project, aims at real-time network simulations as part of network engineering [11]. Hewlett Packard has also announced that NetMetrix, one of its products, can be combined with a simulation tool [12]. In practice, it seems to be different from our tool because we continuously monitor traffic and give every packet a timestamp, and the simulator uses the timestamp sequence continuously.

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 roughly dimensioned by a certain method. 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 reso urces 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],[6], [7], [8], [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].)

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, 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 (Figure 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.

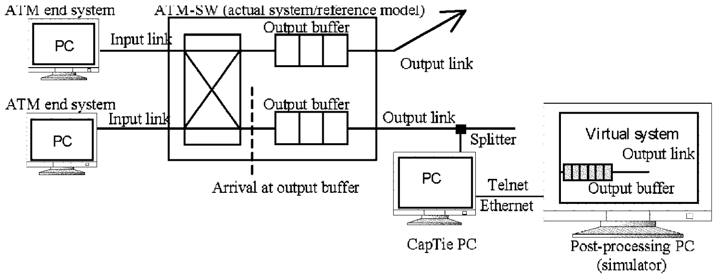


**Fig. 1.** ATM network with our system.

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, so 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 bi ts 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 fi ne 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. Our measurement device is tapped at a link, so 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 Figure 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 mod el, which is a model used in the simulation (Figure 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 p acket 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 $n(i)$ 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 (Figure 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 th e S-CASE algorithm.

$$t(i, j) = t(i) + (j - 1) * L/H \tag{1}$$

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 obtained by 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 $x$ of active flows 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 (Figure 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 (Figure 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 $(t(i) - T(i))C/(Hx(i))$ 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

$$t(i, 1) = t(i - 1, 1) + (t(i) - T(i))C/(Hx(i)). \tag{2}$$

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 $t(i, 1) <= t(i)$ for any $T(i) = t(i-1) >= t(i-1, 1)$, 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 $C <= Hx(i)$.

$$t(i, 1) = t(i - 1, 1) + (t(i) - T(i))C/max(C, Hx(i)). \tag{3}$$

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 (Figure 4). In this case, $T(i)$ is the transmission epoch of

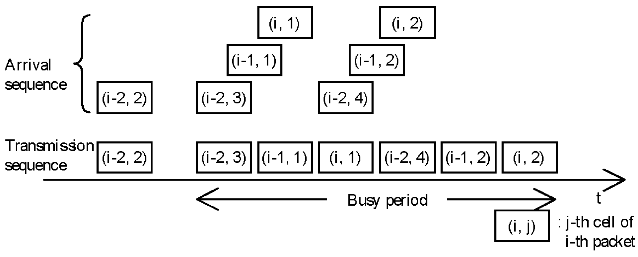


**Fig. 2.** Example of the event sequence (1).

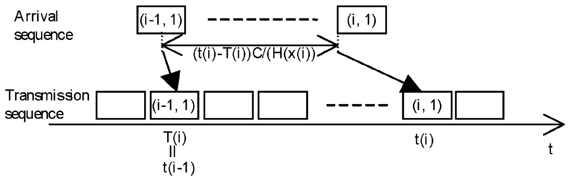


**Fig. 3.** Example of the event sequence (2).

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,

$$t(i,1) = T(i) + (t(i) - T(i))C/max(C, Hx(i)). \tag{4}$$

The arrival epoch of the $j$-th cell of the $i$-th packet is estimated as follows for any of the cases mentioned above.

$$t(i,j) = t(i,1) + (j-1)L/H \tag{5}$$

## 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 (Figure 5). In this computer simulation, packets were generated according to a Poisson process and

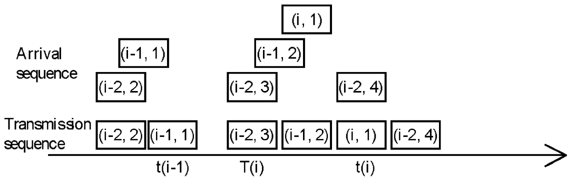


**Fig. 4.** Example of the event sequence (3).

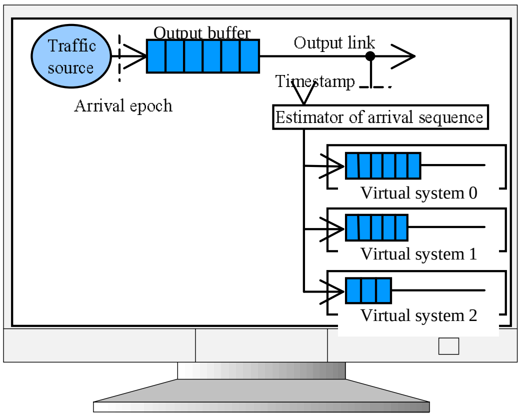


**Fig. 5.** Computer simulation.

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 (CLRs) that occurred when the true and estimated cell arrival sequences were offered to the virtual system and the r eference 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 CLRs of 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. Under the following numerical examples, we assume that the average offer load from a source is 1 Mbps and that the number of sources is 30. (Thus, the total offered load is fixed.)

The results for Condition 1 are plotted in Figure 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 CLRs for the true cell arrival sequence were almost the same for (3, 0, 0) and (1, 2, 0),

**Table 1.** Simulation conditions and virtual systems

| Simulation condition | Average packet length (Cell) | Input speed (Mbps) | Virtual system (buffer size, output link capacity) (Mbps) | | |
|---|---|---|---|---|---|
| | | | 0 | 1 | 2 |
| 1 | 10 | 50 | (128, 50) | (128, 37) | (128, 30) |
| 2 | 10 | 150 | (128, 30) | (128 ,37) | (128, 50) |
| 3 | 50 | 50 | (128, 50) | (128, 37) | (128, 30) |
| 4 | 50 | 15 | (128, 50) | (128, 37) | (128, 30) |
| 5 | 10 | 150 | (128, 50) | (128, 37) | (128, 30) |
| 6 | 50 | 50 | (256, 50) | (256, 37) | (256, 30) |
| 7 | 50 | 50 | (256, 50) | (128, 50) | (384, 50) |
| 8 | 50 | 50 | (128, 50) | (128, 75) | (128, 37) |

the estimation accuracies were different. Using either estimation algorithm, the CLRs in (1, 2, 1) and (1, 2, 2) were similar to the true CLR of (1, 2, 0). This means that the estimation in (1, 2, *) was easy. On the other hand, the CLRs in (3, 0, 1) and (3, 0, 2) were different from the true CLR of (3, 0, 0). This means that the estimation in (3, 0, *) 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 CLRs 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 larger 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))).

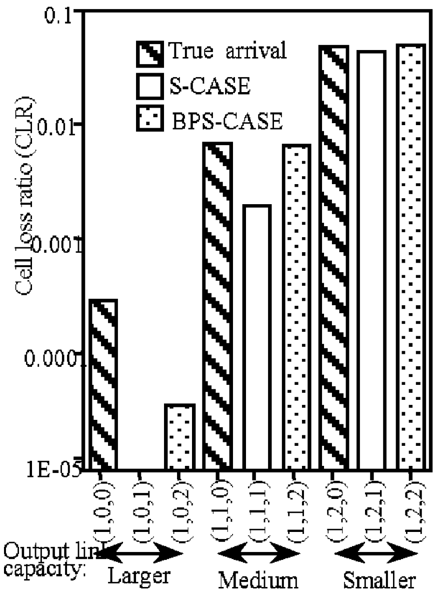


**Fig. 6.** Cell loss ratio for condition 1.

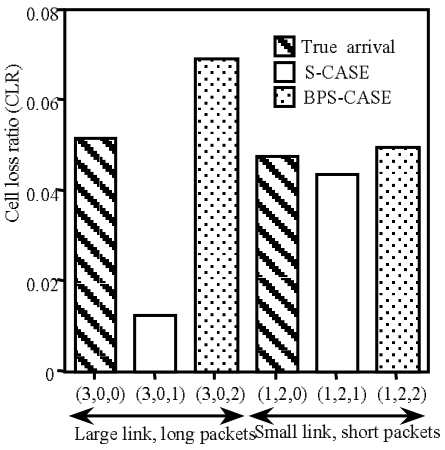


**Fig. 7.** Errors in estimation for similar CLR.

(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

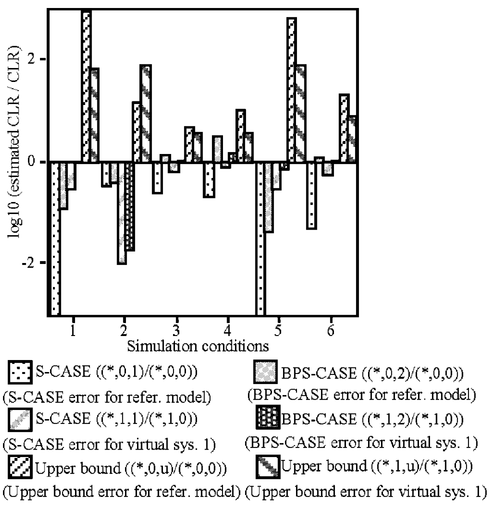


**Fig. 8.** Errors in estimation.

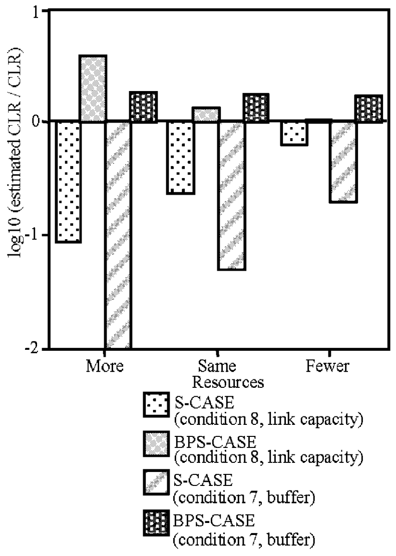


**Fig. 9.** Error in estimation for different network resources.

with mean equal to the mean observed in the simulation. The CLR obtained under the Poisson arrival assumption was smaller than -10 in this figure for all simulation conditions, so 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 system.

## 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 (Figure 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 [13]. 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.

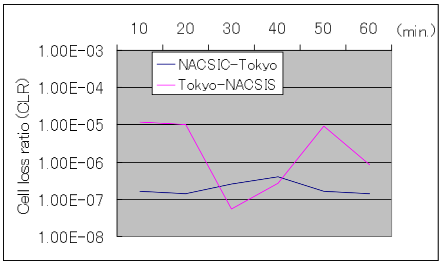
As a first step, the output link capacity was dimensioned by the dimensioning algorithm, which was propsoed in [13] and used the traffic measurement data obtained by the switching node. The dimensioned output link capacity was used for the output link capacity (that is, the service rate of the server) 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 si mulator 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 Numerical Example of Real-Time Simulation

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). The capacity of each link was dimensioned using the queue decay parameter method [13] based on data measured in the we ek before this trial by an ATM switching node accommodating the link. The CLR objective was 10-6.

Figure 10 [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, so 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 (Figure 11). 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 (Figure 12). However, the CLR of this direction had a larger fluctuation than that of the opposite direction because the offered traffic was small (Figure 10).



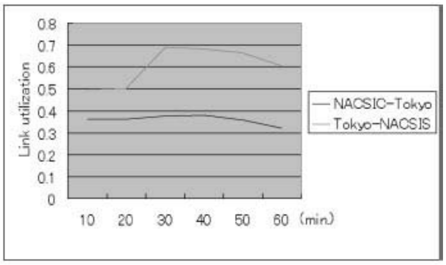


**Fig. 10.** Cell loss ratio in simulated output link.

**Fig. 11.** Link utilization in simulated output link.

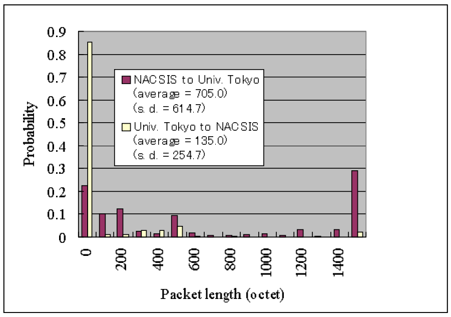


**Fig. 12.** Packet length distribution.

Comparing Figures 10 and 11, 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 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.

## References

1. K. Thompson, et al., IEEE Network, pp. 10-23, Nov./Dec. (1997).
2. http://www.nlanr.net/NA/Oc3mon

3. S. Asano, IEICE Journal, 81, 4, pp. 402-406 (1998).
4. H. Saito, Teletraffic Technologies in ATM Networks, Artech House, Boston (1994).
5. H. Saito, IEEE Trans. Communications, 40, 9, pp. 1512-1521 (1992).
6. N. G. Duffield, et al., Proc. Cam. Phil. Soc., 118:363-374 (1994).
7. H. Saito, et al., NTT Review, 8, 1, pp. 56-65 (1996).
8. S. Nakagawa, et al., NOMS'96, Kyoto (1996).
9. H. Saito, IEEE Communication Magazine, 35, 5, pp. 146-153 (1997).
10. http://www.wrs.com/products/html/vxworks.html
11. M. Maeda, IEEE ATM workshop '99, Kochi (1999).
12. http://www.tmo.hp.com/tmo/ntd/products/products.html
13. H. Saito, et al., ICCCN'99, Boston (1999).