The Characteristics of WWW Traffic and the Relevance to ATM

Technical Report

Pär Karlsson* and Åke Arvidsson†
Department of Telecommunications and Mathematics,
University of Karlskrona/Ronneby,
S-371 79 Karlskrona, Sweden

Abstract

This document describes a study of the characteristics of recorded WWW traffic. Several parameters of the traffic are investigated, The results are used to investigate a scenario where ATM is used as the underlying transport mechanism. Problems with the deployment of ATM in the approach taken are considered and suggestions for improvements are made.

The high variability of the traffic implies that a fixed allocation of bandwidth between the mean and peak rate is an infeasible way to achieve a reasonable utilization of the system, since this results in tremendous buffering demands. This calls for a different view on the way to study the system under consideration. Different properties of the traffic must be taken care of by different methods. Variations over longer time-scales are dealt with by means of capacity allocation and fluctuations with shorter duration are buffered.

This realistic way of looking at the system might also put different tasks, such as traffic modelling, in a new light. For instance, does a model that is to be used for buffer dimensioning have to capture traffic behavior on time-scales that are longer than reasonably can be buffered anyway?

1 The WWW and Enabling Technologies

1.1 The World Wide Web

The World Wide Web (hereafter WWW) constitutes one of the most important and fastest growing applications on the Internet during the last couple of years and in the foreseeable future. WWW is basically a method for transfer of information between parties over the Internet. Information are located at WWW servers that are connected to the Internet and managed by companies, universities, organizations, or individuals. The servers are mostly built upon ordinary workstations running specialized server software. On the client side users are accessing information on the WWW servers using browsers that can present information in many formats (e.g. text, pictures, sound, and video). Browsers are now available for all popular operating systems. The activity of accessing information on the WWW by a browser is popularly denoted "surfing". The key technologies enabling the WWW explosion are HTML, HTTP and TCP/IP.

1.2 HTML

The Hypertext Markup Language (HTML) is the language used to create portable hypertext document for publications on the WWW. At the time of writing the most common version in use is HTML 2.0 defined in [BLC95]. The latest version is, however, HTML 3.2 [Rag97] which aims at replacing HTML 2.0.

*Email: pka@itm.hk-r.se, Telephone: +46 455 78063
†Email: akear@itm.hk-r.se, Telephone: +46 455 78053

HTML 3.2 is a recommendation from the World Wide Web Consortium (W3C), an international industry consortium founded in 1994 to develop protocols for the evolution of the World Wide Web.

1.3 HTTP

The Hypertext Transportation Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. HTTP is the protocol used for the exchange of information between clients and servers on the WWW. The most recent version is HTTP/1.1 [FIG+97], designed to take care of some performance problems in HTTP/1.0. A study of HTTP/1.1 performance can be found in [NGBS+97]. Since many different, sometimes incomplete, implementations of HTTP/1.0 exist, HTTP/1.1 also includes more stringent requirements than HTTP/1.0 to ensure reliable implementations. HTTP traffic is today usually transported over TCP/IP connections. The HTTP standard does not dictate that this must be the case, and only requires an underlying reliable transport service.

1.4 TCP

The Transmission Control Protocol (TCP) [Pos81] is a connection-oriented, end-to-end reliable protocol. TCP is designed to provide a means of reliable communication between processes on computers on different networks that are interconnected. TCP requires very little in terms of reliability from the underlying communication systems and only assumes that they provide a datagram service. The main functions of TCP are:

Connection management Takes care of initialization, maintenance and termination of logical connections between processes.

Basic Data Transfer Continuous streams of octets can be transported in both directions through a TCP connection.

Reliability The TCP handles the cases where data is lost, damaged, duplicated or delivered out of order by the underlying communication system.

Flow Control The amount of data fed into the network by a connection is controlled by means of a window mechanism.

TCP is implemented as a part in an operating system.

2 WWW Traffic Measurements

In the following section several characteristics of the traffic originating from a WWW server are investigated. The measurements were performed by Ericsson Utvecklings AB, Älvsjö. The server under study was connected to a 10Mbit/s Ethernet segment connected directly to a router. No other sources of traffic were present at the segment. A workstation connected to the segment running the program `snoop`, available in the operating system Solaris from SUN Microsystems, was used to observe the traffic. All IP packets to and from the WWW server during a period of approximately 10 hours were captured and selected fields from the header were logged to a file together with a time-stamp. According to the manual page for `snoop` the measurements have an accuracy of 4μs.

2.1 Log File Processing

The headers of the captured packets were processed and certain fields were logged to a file. The different fields logged included a time-stamp, the size of the Ethernet frame, source and destination hosts, source and destination port numbers, type of TCP packet (SYN, FIN, RST), sequence number, acknowledgment number, and data length. An example of the resulting log can be found in figure 1. The programming language Perl [WCS96] was found to be extremely well suited for the processing of the log file. Several

```
10.58840 WWW->cli2 s=58  D=4609 S=80 Syn Ack=2518282107 Seq=2792193918 Len=0
10.59039 cli2->WWW s=60  D=80 S=4609     Ack=2792193919 Seq=2518282107 Len=0
10.61223 cli2->WWW s=315 D=80 S=4609     Ack=2792193919 Seq=2518282107 Len=261
10.61460 WWW->cli2 s=281 D=4609 S=80     Ack=2518282368 Seq=2792193919 Len=227
```

Figure 1: Example of logged packet sequence

Perl scripts were produced to extract various subsets of information from the log file. The extracted data was later investigated further using MATLAB and various programs written in C++.

The statistics reported in the following are all derived from the traffic going out from the server. The reason for this is that this traffic is the most interesting since it is magnitudes larger in size than the incoming traffic. The number of incoming packets were 163716 and the outgoing were 156296, the latter is, however, in general much shorter.

2.2 IP Statistics

In figure 2 a histogram of the size of the outbound IP packets is shown. Size here means only the size of the IP packet (header and payload), the Ethernet overhead is not included. As could be expected, the distribution contains a lot of small packets that carry TCP control messages and HTTP requests and answers. A large peak can also be found at 1500 octets, this peak comes from the maximum possible Ethernet frame size which is 1514 octets (the Ethernet header constitutes 14 octets).

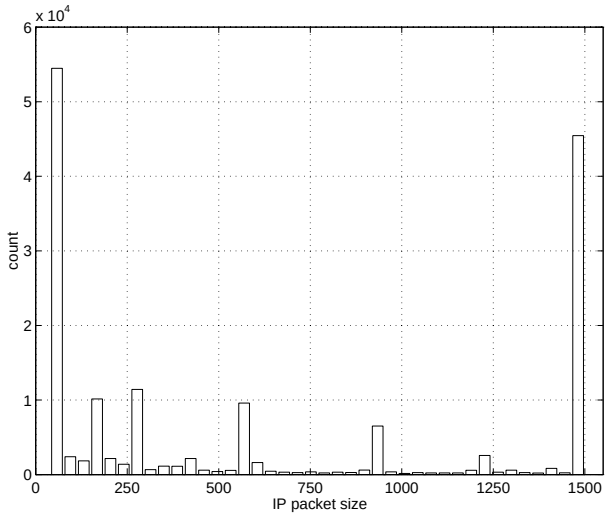

Figure 2: Histogram of IP packet sizes

In figure 3 a histogram of the total amount of data transported over the TCP connections is presented. Both IP and TCP overhead as well as possible retransmissions are included in figure 3. The HTTP/1.0 implementation used opens a separate TCP connection for every element of a HTML page that the clients request.

In figure 4 the duration of the TCP sessions is presented. The actual duration of each TCP connection had to be extracted from the IP-log with some care. By inspection of the log it was found that the TCP connections often were closed a long time after the actual data transfer had ceased. Since we are more interested in the actual time it takes to transfer the data we decided to consider the stop time of each connection to be the time at which the last packet containing data was logged.

During the work associated with the log file, it was discovered that clients sometimes tried to use TCP connections that had been closed several hours ago. This resulted immediately in a RST response from the

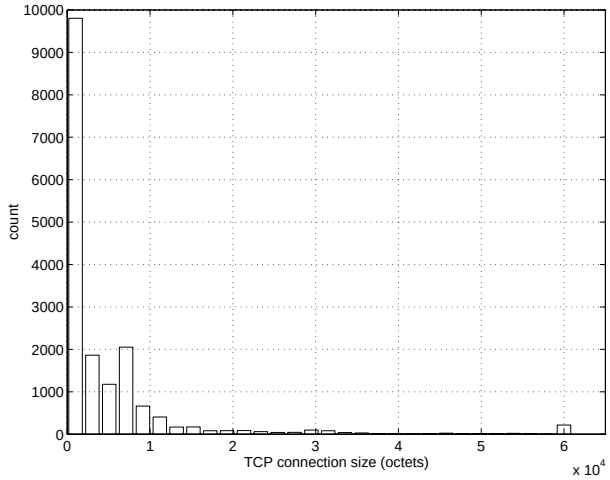

Figure 3: Histogram of the total amount of data transported over TCP connections

server. Clearly this is a bug in the TCP implementation of the clients. The operating system of the clients with this behavior is unknown.

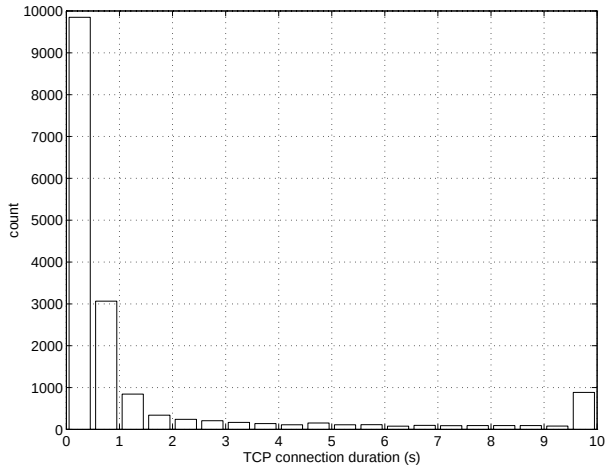

Figure 4: Histogram of the duration of TCP connections

Figure 5 presents the mean bit rate, obtained by dividing the total size with the duration of each TCP connection. As can be seen the bit rate enjoyed by connections is highly variable, from a few kbit/s to about 1 Mbit/s. The large differences in bit rate is probably mostly due to bottlenecks outside of the local network the server is connected to. Unfortunately the location of clients could not be extracted from the log file. Checking the dependence of users location and the "speed" they experience while communicating with the server remains to study.

It should be noted that the method used by HTTP/1.0 to open a separate TCP connection for every element to download is highly wasteful on several points. First of all it generates a high amount of unnecessary control traffic to set up and tear down connections. Since the TCP flow control mechanism also takes some time to find a good rate of inserting traffic into the network it also is disadvantageous to have short (in terms of transferred data as well as duration) connections. These shortcomings are (among other things) targeted in HTTP/1.1.

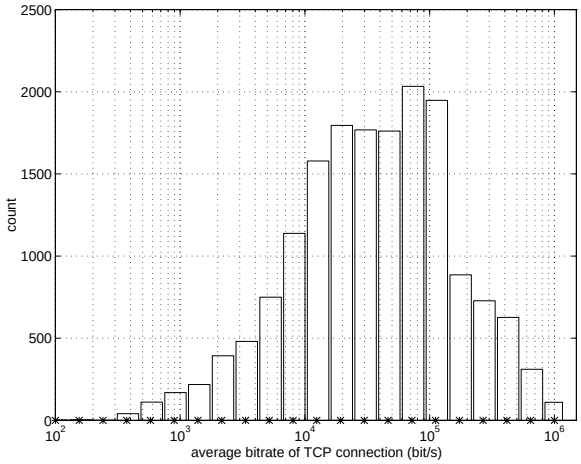

Figure 5: Histogram of average bit rate of TCP connections

2.3 Session Arrival Statistics

Due to the fact mentioned above that HTTP/1.0 opens a separate TCP connection for every element on a HTML page there exists a high correlation between the arrivals of TCP connections from a client. It would be interesting to instead observe the behavior of human users when "surfing". The strong correlation between TCP connections do not reflect the real behavior of human users, it is created by the protocols involved. We now introduce the concept of a session. A new session is considered to start when a human user selects to download a new page.

To capture the session arrival process one should preferably look directly at the contents of the actual traffic conveyed over the TCP connections. Another alternative might be to get this information from the server logs, an issue which remains for further study.

Since this information was not available in our trace we developed a decision mechanism to separate different sessions from each other. By introducing a limit of how long time there should be silence (no new TCP connections) from a certain client before considering the download of a page to be finished, we were able to separate sessions from each other. Around 4 seconds was experimentally found to be a good limit. From the 17347 TCP connections in our trace 6022 could be considered as indicating a new session with the rule above and a limit of 4 seconds.

The decision to use 4 seconds as a limit is of course somewhat arbitrary. This length does, however, also make sense as a minimal separation between user activities. Figure 6 shows the number of detected sessions resulting from different limits.

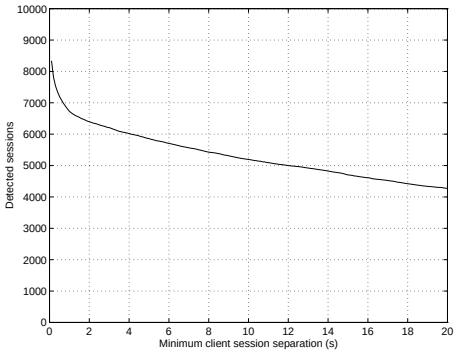

Figure 6: Number of sessions detected for different session separation limits

2.3.1 Distribution of Session Arrival Times

The start time of each of the TCP connections in our log is presented in figure 7. As can be seen the intensity at which new TCP connections occur decreases slightly at the end of the trace. In the investigations below the last 2.14 hours of the trace were rejected not to include this non-stationarity in the results.

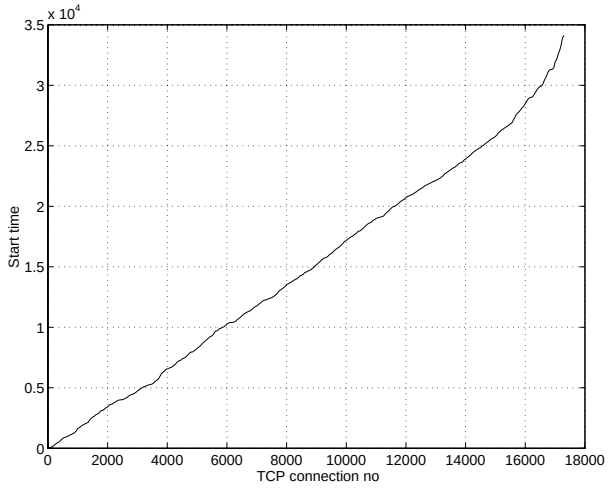

Figure 7: Start time for TCP connections

Figure 8 shows the distribution of the interarrival times between TCP connections. The use of several TCP connections to retrieve a page is here manifested in the large count of short interarrival times. Mainly this large amount of short interarrival times argues against the times being exponentially distributed.

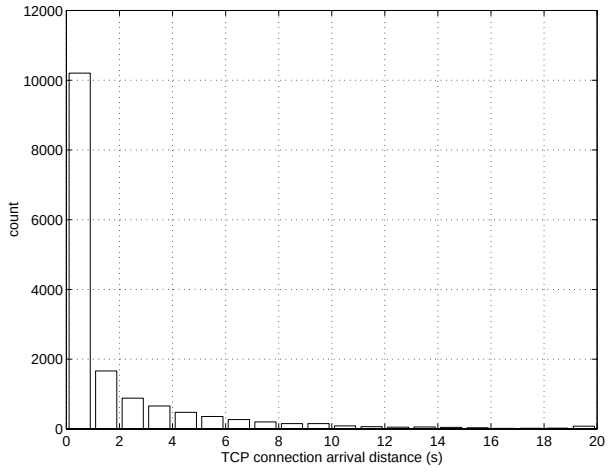

Figure 8: Histogram of interarrival times between TCP connections

In figure 9 the interarrival times between the detected sessions are presented. The limit used in the detection of sessions was 4 seconds.

Clearly figure 9 looks more like an exponential distribution than figure 8. The estimated mean session distance is 5.10 and the estimated variance is 26.0. This also seems to agree well with an exponential distribution whose mean is $1/\lambda$ and variance is $1/\lambda^2$. To test the hypothesis that the session distances are exponentially distributed a chi-square test was performed. The result of the test is reported in table 1. The threshold values are based on a significance level of 5%. The test was performed with equiprobable

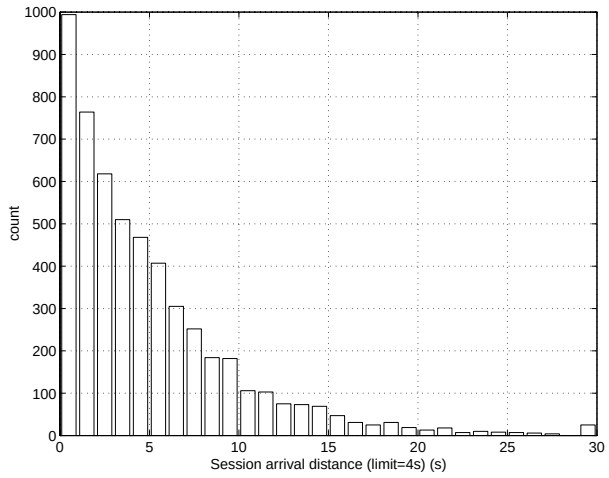

Figure 9: Histogram of interarrival times between sessions

Intervals	10	20	30	40	50
Degrees of freedom	8	18	28	38	48
χ^2 test statistic	13.36	18.60	38.48	43.69	44.79
χ^2 threshold value	15.51	28.87	41.34	53.38	65.17

Table 1: Chi-square test for exponentially distributed interarrival times

intervals. Since the parameter of the exponential distribution was estimated from the data, the degrees of freedom for the χ^2 distribution was decreased with one as recommended in the literature [Leo93]. The conclusion of the test is that we can not say that the interarrival times not are exponentially distributed.

2.3.2 Independence of Arrival Times

Still to determine are to what degree successive interarrival times are correlated. Figure 10 shows an unbiased estimation of the autocorrelation at different lags for the intervals. Clearly the correlation have decreased significantly already at lag 1. There does, however, seem to exist some correlation up to lag 20. Since the mean arrival distance here is 5.10 seconds lag 20 corresponds to about 100 seconds.

For successive interarrival times $X_n,\ n = 1,2,3,\ldots,N$ the unbiased estimation of the autocorrelation at different lags k takes the following form.

$$R_X(k) = \frac{1}{N-k}\sum_{n=1}^{N-k} X_n X_{n+k}, \qquad k \geq 0 \tag{1}$$

Now assuming a hypothesis as follows; the X_ns are independent identically exponentially distributed with parameter λ gives

$$\mathrm{E}[X] = \mathrm{E}[X_n] = 1/\lambda, \quad \mathrm{E}[X^2] = \mathrm{E}[X_n^2] = 2/\lambda^2, \quad \mathrm{V}[X] = \mathrm{V}[X_n] = 1/\lambda^2. \tag{2}$$

Letting $Z_n = X_n X_{n+k}$ it is easily verified that

$$\mathrm{E}[Z] = \mathrm{E}[Z_n] = 1/\lambda^2 \tag{3}$$

$$\mathrm{V}[Z] = \mathrm{V}[Z_n] = 3/\lambda^4. \tag{4}$$

Now the mean of the autocorrelation estimator is

$$\mathrm{E}[R_X(k)] = \frac{1}{N-k}\mathrm{E}\left[\sum_{n=1}^{N-k} Z_n\right] = 1/\lambda^2. \tag{5}$$

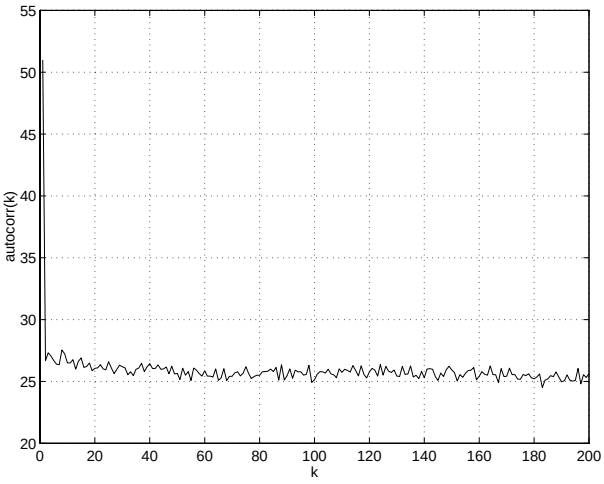

Figure 10: Autocorrelation of interarrival times

To find the variance of $R_X(k)$ we first need the covariance between different Z_n.

$$\mathrm{Cov}[Z_i, Z_j] = \mathrm{E}[Z_i Z_j] - \mathrm{E}[Z_i]\mathrm{E}[Z_j] \tag{6}$$

Expanding this reveals

$$\mathrm{Cov}[Z_i, Z_j] = \begin{cases} \mathrm{V}[Z] & i = j \\ \mathrm{E}^2[X]\mathrm{V}[X] & i \pm k = j \\ 0 & \text{otherwise} \end{cases}. \tag{7}$$

Returning to the variance of $R_X(k)$ we find

$$\mathrm{V}[R_X(k)] = \frac{1}{(N-k)^2}\mathrm{V}\left[\sum_{n=1}^{N-k} Z_n\right] = \frac{1}{(N-k)^2}\sum_{i=1}^{N-k}\sum_{j=1}^{N-k}\mathrm{Cov}[Z_i, Z_j]. \tag{8}$$

Expanding this using the results above gives

$$\mathrm{V}[R_X(k)] = \frac{1}{(N-k)^2}\left((N-k)\mathrm{V}[Z] + 2(N-2k)\mathrm{E}^2[X]\mathrm{V}[X]\right) = \frac{5N-7k}{(N-k)^2\lambda^4}. \tag{9}$$

We now can form a normalized estimate of the autocorrelation that should be normally distributed with zero mean and unit variance if our hypothesis is correct.

$$N(k) = \frac{R_X(k) - 1/\lambda^2}{\frac{\sqrt{5N-7k}}{(N-k)\lambda^2}} \tag{10}$$

In figure 11 this normalized estimation of the autocorrelation is plotted from lag 1 to lag 100. The dotted line is the upper bound of a 95% confidence interval based on an normal distribution with zero mean and unit variance. Up to lag 15 there seems to be a higher correlation present than could be expected based on the assumptions above. The conclusion is therefore that some noticeable correlation exists between session arrivals. This correlation is, however, not very strong and it can also be expected to be less significant in an environment with more users present, at least when networking and server resources are expanded accordingly and thus not constitutes a limitation. It can be noted that the correlation remains above zero for all but a few of the lags in figure 11, but since all these values fit within the 95% interval we refer from drawing any conclusions from this fact. An expansion of this investigation should preferably be to test the distribution and independence for shorter intervals. The arrivals in each interval could then be tested against parameters specific to the interval under study. An even better fit can be expected from such a study.

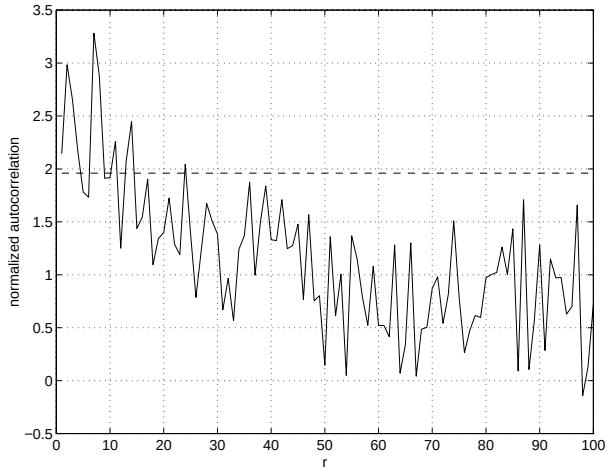

Figure 11: Normalized autocorrelation

3 ATM

3.1 IP over ATM

To investigate the impact of this type of traffic when ATM is used as the underlying transport mechanism a series of simulations were performed. A simulation environment in C++ was built to test various ideas. For the conversion of the captured IP traffic the method presented in figure 12 was used. The approach

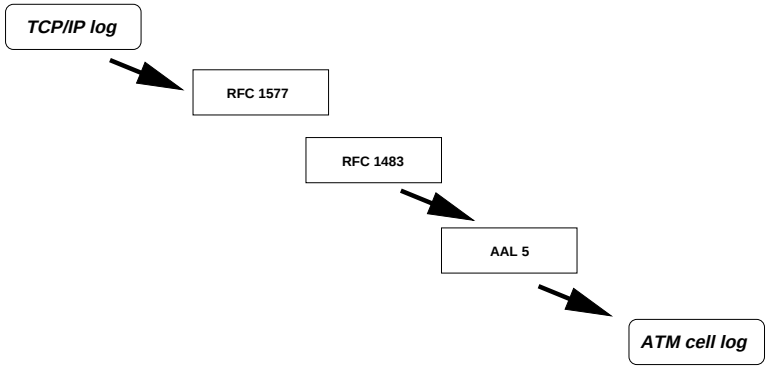

Figure 12: IP over ATM

in figure 12 is denoted “Classical IP over ATM”. RFC 1577, “Classical IP and ARP over ATM” [Lau94] defines how to use ATM as a replacement for standard network technology when using IP. Large parts of RFC 1577 are concerned with resolving of IP addresses to ATM addresses, these parts are not included in our simulations.

The actual mapping of IP packets into ATM cells are defined in RFC 1483 [Hei93]. RFC 1483 describes two different methods for carrying connectionless traffic (*e.g.* IP) over ATM AAL5 PDUs. The first method multiplexes several protocols over one ATM Virtual Circuit. This is accomplished by prefixing the AAL5 PDUs by an IEEE 802.2 Logical Link Control (LLC) header. The second method separates different protocols by using separate VCs for different protocols. Clearly the second method should be the preferred one since the first method duplicates functions found in ATM. The second method, also denoted “VC Based Multiplexing” also have the nice property of allowing separation of different traffics on different VCs.

The mapping of IP packets into AAL5 PDUs is rather straightforward and the number of cells an IP

packet generates can be found from the following formula.

$$N_{cells} = \left\lceil \frac{IP_{size} + 8}{48} \right\rceil \tag{11}$$

Where N_{cells} is the number of ATM cells, IP_{size} is the size of the IP packet in octets. (8 is the size of the AAL5 PDU trailer.) Since TCP and IP are designed to work over non-reliable communication systems with varying characteristics, it may be a good candidate for the ABR and UBR ATM service classes. This is not, however, the scenario used in the simulations below.

3.2 Simulations

Below the method outlined above are used in simulations. It should of course be noted that the following simulations can be questioned from the fact that they just uses a snapshot of collected traffic. What is disregarded in the simulations are the interactions between different protocol layers. For example the possible influence at the TCP layer of delays or losses at the ATM layer is not included. It should also be noted that these interactions not only are present between protocol layers, but there also exists a connection between user behavior and the experienced performance, see figure 13. Simulations like these can, however, give valuable information and insights into the problems involved.

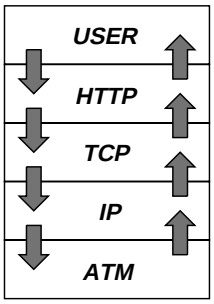

Figure 13: Layer interaction to be considered

The queuing system to be simulated is presented in figure 14. Some preprocessing of the log file was performed to determine the start and end of the TCP connections. This information is used under the simulation to schedule the creation and activation of each TCP connection. After creation each TCP connection transmits packets based on the trace. The packets are converted into cells and passed to the buffer at a rate that matches the speed of the 10Mbit/s Ethernet LAN the packets were collected from. The queuing system that the following statistics are derived from is the second one in figure 14. The first buffer in figure 14 was

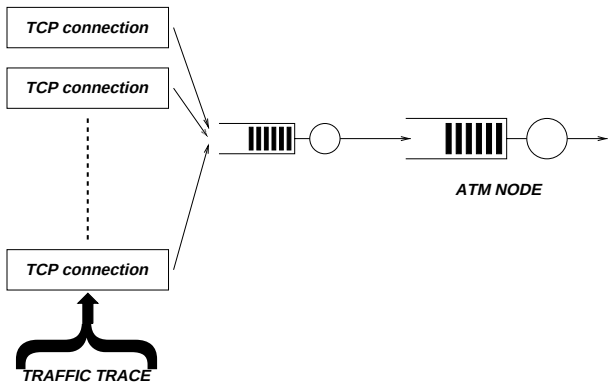

Figure 14: Simulation scenario

introduced after the discovery that the time-stamps in the log in some cases apparently lacked accuracy. This caused some packets to overlap in time slightly and cause a queue buildup even when the capacity of the server was set to the link rate at which cells were injected into the buffer by the TCP connections. By introducing the first buffer these irregularities are smoothed out.

3.3 Cell Arrival Process

Figure 15 shows the cell arrival process at different time-scales. It shows the number of cells arriving in 1000 successive slots with the slot size varying from 10^{-2} to 10^{1} seconds. The uppermost plot thus

represents an elapsed time of 10^4 seconds and the plot at the bottom covers 10 seconds. The arrival process clearly exhibits high variability over all the time-scales. Figure 16 shows the index of dispersion for counts

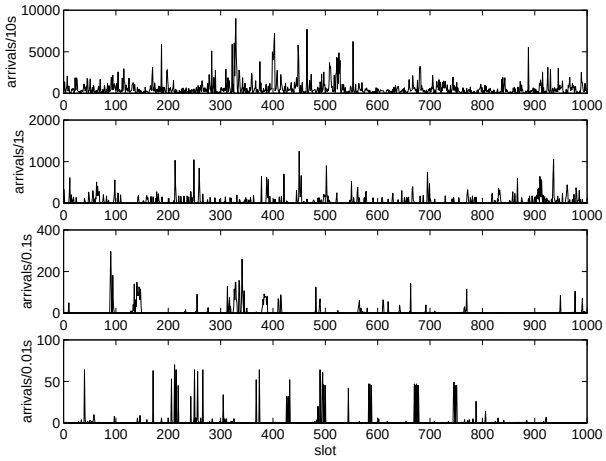

Figure 15: The cell arrival process at different time-scales

of the cell arrival process. The index of dispersion, defined as $\mathrm{IDC}(t) = \mathrm{V}[N_t]/\mathrm{E}[N_t]$, where N_t is the number of arrivals in an interval of length t, is a measure of the variability of the process over different time-scales. What is interesting in figure 16 is that the curve doesn't seem to level out as t increases. The high amount of noise at the end of the plot is due to the fact that the IDC estimations in that area are based on very few samples. There is, however, an increasing trend visible in the curve. An estimation of the

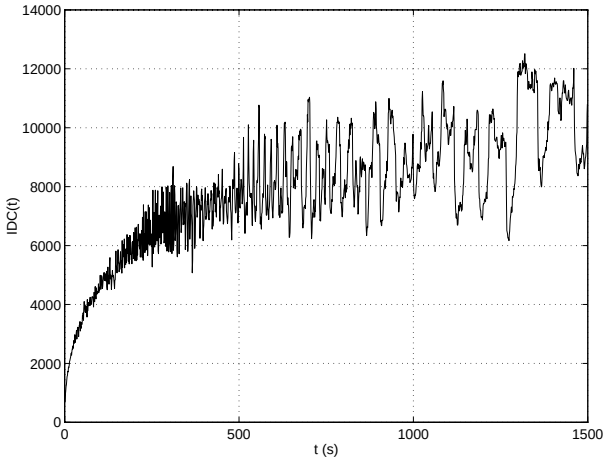

Figure 16: Index of dispersion for counts for the cell arrival process

Hurst parameter for the cell arrival process via the rescaled range statistic revealed a value of 0.7.

3.4 Queuing Process

3.4.1 Static Capacity Allocation

To study the impacts of the high variability of the traffic, we used the simulation scenario outlined above and dimensioned the capacity of the server to achieve different levels of utilization of the server. Table 2 reports some statistics of the buffer occupancy at different levels of utilization. In figure 17 the buffer

Utilization	75%	50%	25%	10%	1%
Max queue	98597	71823	34144	10050	3197
Mean queue	$2.64 \cdot 10^4$	$7.14 \cdot 10^3$	$2.28 \cdot 10^3$	$6.70 \cdot 10^2$	74.4

Table 2: Buffer occupancy statistics at different server utilization levels

occupancy for the case with a server load of 50% is shown. As could be expected, table 2 and figure 17

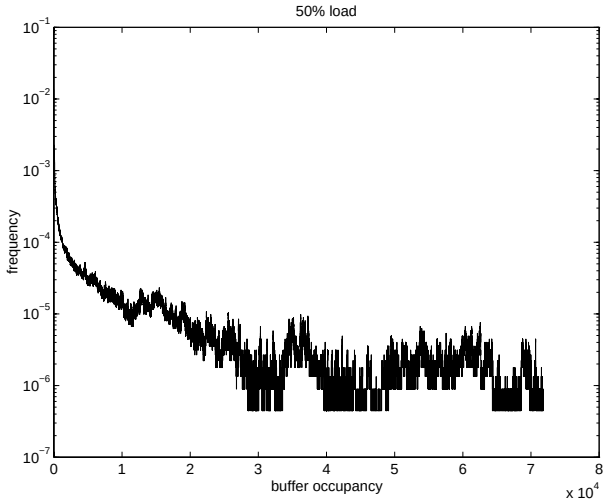

Figure 17: Buffer occupancy with a server utilization of 50%

tell us that a dimensioning based on the mean of the whole trace not is a realistic approach. Even in the case with 1% server load a not negligible demand for buffer capacity exist. The long queue lengths are undesirable both from an implementation point of view and the delays they cause. The only feasible way of transporting this traffic with a static allocation of capacity seems to be a peak rate allocation.

The conclusion is that the high variability over long time-scales not can be taken care of by buffering to achieve a high utilization. Other control mechanisms must be present to deal with it if a high utilization is needed. In the next section we further explore this scenario.

3.4.2 Dynamic Capacity Allocation

To investigate the more realistic scenario when the capacity allocation in the server not is fixed over the whole trace, we included a simple adaptive capacity allocation scheme based on the characteristics of each TCP connection in the simulation. The principle is outlined in figure 18. When a new TCP connection is opened, the capacity of the server is increased with an amount proportional to the mean rate in cells/s of the TCP connection. When the connection is terminated this capacity is released. This is still an allocation based on the mean, but on a much shorter time-scale (seconds instead of hours). This information could be thought of as being signaled at connection setup time if known at that time.

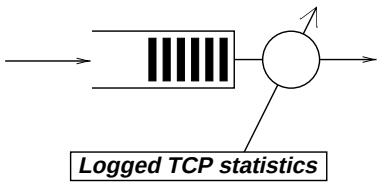

Figure 18: Capacity allocation based on TCP characteristics

Using the allocation scheme with allocation of capacity based on the characteristics of the TCP connection, the demands on buffer capacity were much more modest. An allocation of capacity that integrated over time equaled the capacity allocated in the case of a static allocation resulted in buffer demands about 20 times lower than in the static case. An example is shown in figure 19 where the buffer occupancy is

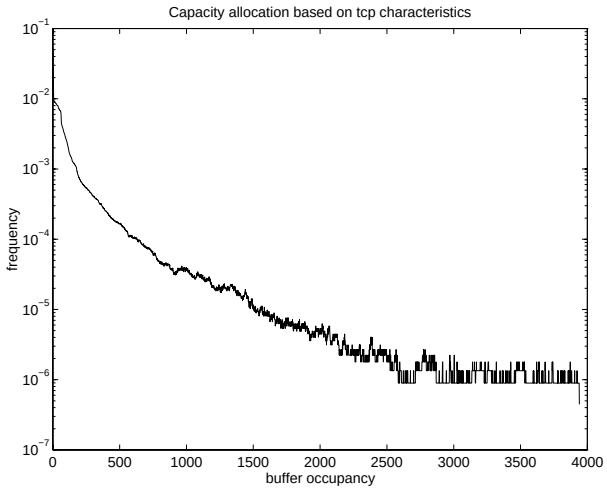

Figure 19: Buffer occupancy with capacity allocation based on mean bit rate of each TCP connection

shown for a case when the extra capacity allocated for each TCP connection is twice its mean bit rate. Integrated over time this means that totally the same amount of capacity is allocated as in the static allocation case in figure 17. The resulting buffer demand is in this case close to the case when a static allocation required a utilization of the server as low as 1%.

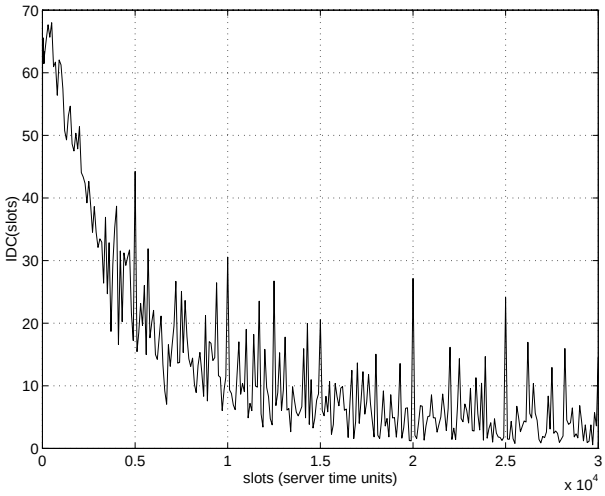

Figure 20: Index of dispersion based on arrivals during service time

In figure 20 a variant of the index of dispersion for the arrival process with adaptive allocation is presented. The IDC estimation is now based on the arrivals during one service time as the basic time-period. The main feature of figure 20 is that we now have a stationary behavior. After a first peak the curve decays quickly and the increasing trend from figure 16 is gone. What this tells us is that the variations over longer time-scales now not are seen in the queuing system, since they are taken care of by the capacity allocation function.

Clearly a mechanism such as this must be present to convey this type of traffic in the light of the results above. How to obtain the information needed for correct allocation decisions is still a question for further research. Since the statistics of the TCP connections used here not are known in real life at the time needed, they are of course unavailable until the connection is closed, some other mechanism must be present. One

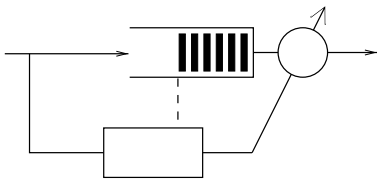

Figure 21: Online estimation of required capacity

alternative can be to make use of the flow control at the TCP layer and make it available at the ATM layer. Another alternative is an online estimation of required capacity as in figure 21. An approach like this is studied in [LCHZ] and [CLG] for VBR video traffic.

3.5 Traffic Modelling

The results above should be noted when searching for suitable traffic models. Since the capacity allocation not can be static if a high utilization is called for, the capacity allocation scheme must be included in the model. In view of the result that such dynamic capacity allocation hides the variations over longer time-scales for the buffer, the question arises if it is really necessary to incorporate these in the model. At least for the purpose of buffer dimensioning this should not be the case.

From a modelling point of view this is advantageous since it generally should be easier to construct analytically tractable models, *e.g.* Markov models, with limited variability over different time-scales.

Another approach is to take a top down view, starting from user behavior and modelling the impacts of it throughout the layers. This has the attractive feature of separating the demands of the user from the way they are fulfilled by the technology. Together with the indications found above that the WWW traffic at the user level appears to be close to a Poisson process, this implies that a system like this could be modeled as an M/G/m system, perhaps with heavy-tailed distributions of file sizes and session durations.

In all cases it is clear that a traffic model, due to the close interaction between functions at different layers, not can be developed alone, but must be closely matched to a model of the system the traffic interacts with.

3.6 A Closer Look at WWW and ATM

It is here also appropriate to make a reflection about the approach taken. It deserves to be noticed what the different layers of our system really do. The HTTP layer deals with exchange of information between parties, this is a connection-oriented activity. At the TCP layer connections are established over the connectionless functions at the IP layer. At the ATM layer we have a connection-oriented service, but this is now used to emulate the connectionless behavior of IP. Clearly we are losing/hiding the benefits of ATM with this approach![1]

In a scenario with large-scale deployment of ATM equipment between all parties participating in the communication it would be better to map the HTTP layer functions more directly to ATM functions. One way could be to adapt the TCP functionality (*e.g.* keeping the API), but realizing its services over ATM directly. If ABR is used on the ATM layer its rate control could then be used directly on this “new” TCP layer (possibly with some additional end-to-end flow control). This would also make the sources (WWW servers) to behave like truly greedy sources since they always will have data to send until the connection is closed. This can not be considered to be the case when TCP/IP is mapped over ABR since the TCP flow control not can be expected to behave like a greedy source.

Another way of dealing with the TCP/IP to ATM interfacing problems is to shape the ATM traffic more aggressively at the source. This does not necessarily imply huge buffers at the network interface cards. Instead it is a question of with what rate TCP releases traffic to the card (into the net), a task similar to the TCP flow control mechanism today. The buffer is in that case present at some other location (in the workstation memory or on a hard disk), where the data resides.

[1] IP over ATM is of course better motivated in situations when the penetration of ATM equipment is not total.

4 Further Work

In the future we would like to further investigate the impacts of WWW traffic in the light of ATM technology. We will expand the traffic characterization study with more extensive and longer measurements. Based on lessons learned in this work the measurement method should be adapted and more targeted at the statistics we wish to derive. Several ways of refining the measurements are under consideration.

Other areas to deal with are how WWW traffic characteristics change when HTTP/1.1 becomes commonly deployed and to further develop how the WWW technology preferably should be adapted to use ATM. This among, other things, should include development of suitable traffic and system models.

References

[BLC95] T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0 (RFC 1866), September 1995.

[CLG] Song Chong, San-qi Li, and Joydeep Ghosh. Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM. Proc. Infocom 94, Toronto, Ontario, Canada, pages 1c.2.1-1c.2.10.

[FIG+97] R. Fielding, UC Irvine, J. Gettys, J. Mogul, DEC, H. Frystyck, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1 (RFC 2068), January 1997.

[Hei93] J. Heinanen. Multiprotocol Encapsulation over ATM Adaptation Layer 5 (RFC 1483), July 1993.

[Lau94] M. Laubach. Classical IP and ARP over ATM (RFC 1577), January 1994.

[LCHZ] San-qi Li, Song Chong, Chia-lin Hwang, and Xirong Zhao. Link Capacity Allocation and Network Control by Filtered Input Rate in High Speed Networks. Proc. Globecom '93, Houston, Texas USA, vol. 2, pages 744-750.

[Leo93] Alberto Leon-Garcia. *Probability and random Processes for Electrical Engineering.* Addison Wesley, second edition, 1993.

[NGBS+97] H. F. Nielsen, J. Gettys, A. Baird-Smith, H. Wium Lie, and C. Lilley. Network Performance Effects of HTTP/1.1, CSS1, and PNG. W3C Note, February 1997. Submitted to ACM SIGCOMM '97.

[Pos81] J. Postel. TRANSMISSION CONTROL PROTOCOL (RFC 793), September 1981.

[Rag97] D. Ragget. HTML 3.2 Reference Specification. W3C Recommendation, January 1997.

[WCS96] Larry Wall, Tom Christiansen, and Randal L. Schwartz. *Programming Perl, 2nd Edition.* O'Reilly, 1996.