A complete framework for on-line Compression of ATM streams

Ioannis Papaefstathiou *
Ioannis.Papaefstathiou@cl.cam.ac.uk
University of Cambridge, Computer Laboratory

Abstract

It has been proved that the link layer compression is very cost-effective when applied to ATM networks. In this paper we present a complete framework for link-layer compression. The framework consists of two different compression schemes. The choice of which scheme to use depends on the special characteristics of the targeted network. We also present the guidelines of a Connection Admission Control Algorithm, which we claim is very effective when used with any of the two proposed compression schemes. We finally support that the Framework can be easily adopted by network operators that would like to have an inexpensive -and without any side effects- scheme to increase the effective bandwidth of their network.

1 Introduction

An important application of ATM is in the interconnections of private LANs, through WAN links. A typical such a network is shown in Figure 1. ATM is used as the backbone infrastructure in such networks, due to its scalability, speed and potential fault tolerance. One of the main characteristics of these networks is that the cost of the WAN links is comparatively high. By increasing the network traffic we can send over these links we can increase the effectiveness of the whole network. This can be done by compressing the data transmitted. In particular, ATM cells can be compressed and then encapsulated over newly formed standard ATM cells. In this paper we show the effectiveness (especially in this kind of interconnection networks) of this simple idea.

The compression algorithm proposed in this paper works as follows : In the gateway of the LAN (where the WAN link is attached to) we try to collect all the cells that have the same header, we remove all the headers and form a long packet. Then and before we actually transmit this long packet we apply our compression algorithm on it and then encapsulate the compressed stream into ATM cells. The major advantage of this approach in comparison with our older approach described in [1] ,[2] is that the WAN network involved can be a typical one (no compression unit needed).

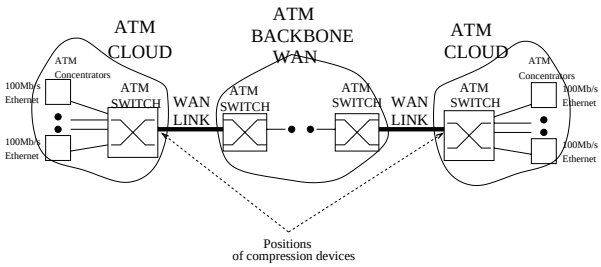

Figure 1: Typical application of our compression scheme

Since these newly formed cells are ordinary cells in terms of their format a standard ATM Call Admission Control(CAC) algorithm can be used for determining whether or not to accept a new connection to the network.

Because the behaviour of the compressed traffic is much less predictable than the one of the original data (since it depends heavily on the very hardly determined compression factor and the cross-talk between virtual circuits) we argue that a measurement based approach should be used in a network that uses our compression scheme. In particular we argue that an algorithm presented by Crosby et al.[3] based on recent theoretical developments in the field of large deviations [8, 9] produces near optimal results.

*Supported by a Marie Curie Research Training Grant under TMR activity 3

2 Compression Algorithm

The algorithm we use for compression is a variation of the LZ(Lempel Ziv) algorithm [7], which is probably the most commonly used class of algorithms.

The actual compression process consists of examining the data that are ready for transmission, to identify any sequences of strings of data bytes which already exist in the encoder history. If an identical such history is available to a decoder, this matching string can be encoded and output as a 2 element string, containing a byte count and a history location. It is then possible for a decoder to reproduce this string exactly by copying it from the given location in its own history. If an incoming byte of data does not form part of a matching string, a one element string, containing this embedded value, is encoded and then transmitted to explicitly represent this byte. In other words it is a byte-oriented compression scheme.

A decoder performs the inverse operation by first parsing a compressed data stream in the two element strings (encoded string of more than one byte) and one element strings(encoding for just one character).

In our particular implementation different dictionaries for the different ATM flows are used. Each flow consists of all the cells with the same VPI/VCI bits. So according to the VPI/VCI bits, the corresponding dictionary for the compression of the payload is accessed. At the other end by looking at the VPI/VCI bits a decision of what dictionary to use for the payload is made. The reason for using different dictionaries for different flows is that, as it is known, the more interelated the data stored in a dictionary are, the better the compression ratio. With very high probability all the cells belonging to the same VP/VC, are parts of the same large unit of data. This is not the case for the current IP over ATM protocols like LANE and CLIP but we believe we head towards the multiple VC/VP solutions (like AREQUIPA [4]) since the QoS issues can be supported more efficiently in the latter solutions. However even when used together with a protocol like LANE our compression scheme causes a smaller but still significant improvement on the efficiency of an ATM network [11].

3 CAC Algorithm

The CAC algorithm used, named Measure, uses estimates of the traffic entropy, obtained from on-line measurements, to perform CAC. The behaviour of Measure is shown graphically in Figure 2. Given a current multiplex of calls, with a particular (bursty) traffic activity (comprised of both compressed and uncompressed traffic), the system makes an estimate of the bandwidth requirement of the multiplex. This is obtained from the entropy estimator (depicted by a thermometer). A new call attempt declares its peak rate P when requesting admission. The CAC algorithm essentially sums the peak rate P, and the current estimate of the effective bandwidth; if the total is less than the link capacity, then the connection can be accepted without violating the QoS of any of the currently multiplexed calls. As soon as the new call commences, the estimator will begin to revise its idea of the current resource requirement, producing in turn a new estimate of the bandwidth requirement of the sources. When the next call attempt arrives the procedure is repeated, as shown. If a new call attempt arrives before the estimator has developed an accurate estimate of the new effective bandwidth, as shown in Figure 3, the algorithm acts *conservatively*. It uses the most recent stable estimate of the effective bandwidth of the multiplex, plus the sum of the peak rates of all subsequent calls. Thus, in Figure 3, the second call will be rejected, because the sum of the two peaks $P + P'$ and the first effective bandwidth estimate exceeds the link capacity.

4 Simulation of the Compression Scheme

Our compression scheme was simulated using an adapted version of the NIST ATM Network Simulator[6]. The most important advantage of this simulator is that it has some specific components for dealing with IP traffic transmitted over ATM. These components were used for loading the simulator with real IP traffic.

The data traces used in the simulator were collected using tcpdump from an FDDI ring, which connects four 8-port 10Mbps Ethernet Switches in the LAN at the Cambridge University Computer Laboratory, and from an SDDI ring which is the backbone which interconnects all the different sites at the University of Cambridge network. Since the latter network is the

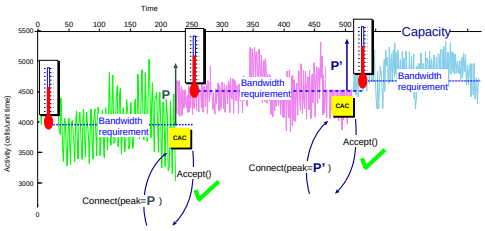

Figure 2: Operation of the Measure CAC algorithm

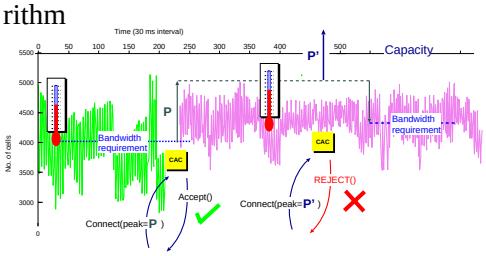

Figure 3: Conservative Call Acceptance using the Measure CAC algorithm

IP		TCP	
TRACE SIZE	COMP RATIO	TRACE SIZE	COMP RATIO
362543480	0.2742	332884548	0.2622
418068578	0.2843	250395541	0.2796
364950357	0.2995	296691959	0.3127
610230148	0.2791	396224640	0.2955

Table 1: LAN scenario : Simulation Results, $Compression\,ratio = \frac{compressed\,size}{raw\,size}$.

HTTP (non-local)		FTP (non-local)	
TRACE SIZE	COMP RATIO	TRACE SIZE	COMP RATIO
470465052	0.6023	226539198	0.8239
278311624	0.5286	148709094	0.8277
273319843	0.5224	178827100	0.8692
151159980	0.6623	144921233	0.8479
186316184	0.6529	108130352	0.8602

Table 2: WAN scenario : Simulation Results, $Compression\,ratio = \frac{compressed\,size}{raw\,size}$. These traces consist of flows with either source or destination address outside the University of Cambridge.

backbone which interconnects different LANs and it is used by 10000 people with a variety of backgrounds and ages, its data can probably be described as representative. Moreover because a lot of these users are computer educated, perhaps they represent the next generation of Internet users (the one that will use the Net for everything from banking services to electing its political leader).

These data were suitably parsed and loaded to the ATM network simulator. A particular characteristic of this parsing is that each TCP flow is associated with a different VP/VC flow[1]. In order to have as general results as possible, large amounts of a wide range of data types were collected. These results can be found in Table 1 and Table 2.

As it can be derived from these tables the kind of networks that would be perfect for applying this compression scheme to, will be, as we have already mentioned, the LAN interconnection networks over WANs (See Figure 1). In these networks the high compression ratio of the LAN traffic[2], and the high cost of the WAN links are combined. So our compression scheme will be much more effective there since someone will be able to send 155Mbps of data over a 50Mbps leased WAN link, at the additional cost of a couple of our inexpensive devices.

A remark for the data of Tables 1 and 2 is that there is a variation in the compression ratios of even the same kind of traffic. The reasons for that are mainly the following :

- In some cases we have more pre-compressed traffic than is some others.
- The size of the TCP flows in some cases is smaller than in some others. Since the smaller the size the lower the compression ratio, the size of the flows can probably result in this variation.

[1] As it is know currently LANE doesn't do this but there are plenty of other schemes that they do it.

[2] Consider that what we define as LAN traffic here is mainly this interconnection LAN traffic since the network from which we collected our traces is such a network

5 CAC Simulation

This section presents the results of simulation experiments using the CAC algorithm of Sections 3 . The aim of the experiments was to evaluate the performance of our approach mainly with respect to the number of calls our network can service without violating the QoS requirement.

Simulation Model. In each of our simulations we model a single output buffer and transmission link of an ATM switch. The link speed used is 100 Mb/s, which corresponds to the TAXI transmission rate for the Fairisle ATM network at Cambridge.

The results presented are based on experiments using the compressed and the uncompressed http traces, and the compressed and uncompressed IP traffic ones. The compressed traces were produced by the simulation of Section 4. Consider again that these traces were derived from *real* networks. The buffer size was 512 cells and the CLR constraint was 10^{-4}. The Peak rate for both types of traffic is 10Mb/sec whereas the mean rates are:

- 1.34/sec for the uncompressed http one
- 0.81Mb/sec for the http compressed one
- 1.46Mb/sec for the IP uncompressed one
- 0.7 for the compressed IP one.

Note that, the compressed rate is equal to the uncompressed rate times the compression ratio.

Call Model. We study a scenario in which calls of a particular traffic type arrive according to an exponential inter-arrival time distribution, an assumption which appears to be well founded [10]. In the absence of real-world data we have used call lengths which are exponentially distributed. We present results for *long* calls. In both cases calls arrive at a high (Poisson) process with mean 5 calls/s. Blocked calls are lost, but the high arrival rate means that the system is continually faced with new call attempts. We thus expect the system to remain close to maximum utilisation. Calls have an exponentially distributed length with a mean of 60 seconds. Each accepted call transmits an independent *trace*. This trace is derived by randomly selecting a start point in the traffic trace. Calls are not correlated in any way.

For the results which compare the performance of the different algorithms, the *same random seed* was used with each different algorithm, resulting in precisely the same call arrivals process in each case. This allows us to compare not only the statistical properties of the algorithms, but also their dynamic behaviour.

Results

Figures 4 and 5 show histograms of the average number of connections in progress during an experiment run, for both the Measure algorithm and the Peak-Rate one. In each figure the left-most histogram in the plot was made using peak-rate admission control; this in all the cases allows for an average of approximately just 10 connections in the system, since the peak rate of all the calls is 10Mb/sec. The central histogram was made using the Measure algorithm applied to the uncompressed traces. The advantage gained by exploiting statistical multiplexing is clearly apparent as the average number of calls in progress is higher in every case. The right-most histogram shows the effectiveness of the compression scheme when used together with the Measure Algorithm. The average number of calls in progress is much higher than in the uncompressed case and it is almost equal to the number of the uncompressed calls admitted times the reciprocal of the compression ratio. In other words the difference in the admitted calls is equal to the speedup of the traces which clearly shows the proposed CAC algorithm can be used very effectively in conjunction with the compression scheme.

References

[1] I. Papaefstathiou, *Accelerating ATM : Online compression of ATM streams*, 18th IEEE IPCCC'99, Phoenix, Arizona, 10-12 February 1999.

[2] I. Papaefstathiou, *Compressing ATM streams*, IEEE Data Compression Conference 1999 (DCC'99), Utah, 29-31 March 1999 .

[3] Simon Crosby and Ian Leslie and John Lewis and Raymond Russell and Fergal Toomey and

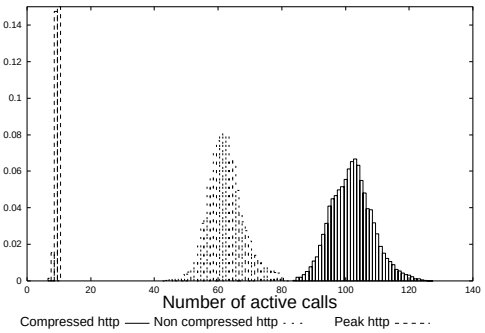

Figure 4: Histograms of the average number of calls in progress over an experiment run, for the http trace.

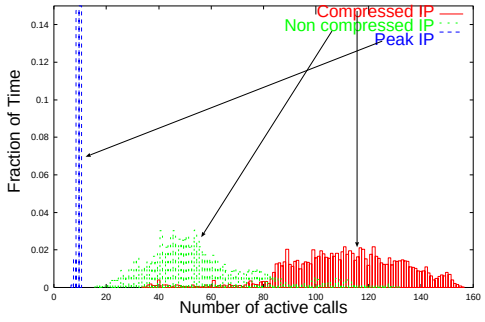

Figure 5: Histograms of the average number of calls in progress over an experiment run, for the IP trace

Brian McGurk, *Practical Connection Admission Control for ATM Networks Based on On-line Measurements*, Computer Communications, January 1998

[4] Ecole Polytechnique Federale de Lausanne, *Application REQUested IP over ATM, Technical Report*, http://lrcwww.epfl.ch/arequipa/, July 1997

[5] E. Fiala and D. Greeve, *Data Compression with Finite Windows*, Communications of the ACM, Vol 32, No 4, April 1989, pp 490-505

[6] National Institute of Standards and Technology, *NIST ATM/HFC Network Simulator: Operation and Programming Guide* , March 1995

[7] J. Ziv and A. Lempel, *A Universal Algorithm for Sequential Data Compression*, IEEE Trans. Information Theory, Vol IT-23, No 3, May 1978, pp 337-343

[8] N. G. Duffield and J. T. Lewis and Neil O'Connell and Raymond Russell and Fergal Toomey, *The Entropy of an Arrivals Process: a Tool for Estimating QoS Parameters of ATM Traffic*, Proceedings of the 11th UK Teletraffic Symposium, Cambridge, March 1994

[9] N.G. Duffield and J.T. Lewis and N. O'Connell and R. Russell and F. Toomey *Entropy of ATM Traffic Streams*, IEEE Journal on Selected Areas in Communications, Special issue on advances in the fundamentals of networking - part 1, 13(6), August 1995

[10] V. Paxson and S. Floyd, *Wide-Area Traffic: The Failure of Poisson Modeling*, Proceedings ACM SIGCOMM 94, London, UK, August 1994.

[11] I. Papaefstathiou, *Measurement based Connection Admission Control algorithm for ATM networks that use low level compression*, 7th International Conference on Intelligence in Service and Networks", IS&N 2000, February 25-28, Athens, Greece.