RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic

M.Grossglauser S.Keshav D.Tse
AT&T Bell Laboratories
600 Mountain Ave.
Murray Hill, NJ 07974
{grossgla,keshav,tse}@research.att.com

Abstract

Compressed video traffic is expected to be a significant component of the traffic mix in integrated services networks. This traffic is hard to manage, since it has strict delay and loss requirements, but at the same time, exhibits burstiness at multiple time-scales. In this paper, we observe that slow time-scale variations can cause sustained peaks in the source rate, substantially degrading performance. We use large deviation theory to study this problem and to motivate the design of Renegotiated Constant Bit Rate Service (RCBR), that adds renegotiation and buffer monitoring to traditional CBR service. We argue the the load placed on signalling by RCBR can be handled by current technology. We present a) an algorithm to compute the optimal renegotiation schedule for stored (off-line) traffic, and b) a heuristic to approximate the optimal schedule for online traffic. Simulation experiments show that RCBR is able to extract almost all of the statistical multiplexing gain available by exploiting slow time-scale variations in traffic. In more general terms, we believe that a clean system design must match control time-scales to the time scales over which the workload varies. RCBR works well because it makes intelligent use of this time-scale separation.

1 Introduction

Compressed video traffic is expected to be a significant component of the traffic mix in integrated services networks. One key characteristic of a compressed video source is its burstiness. That is, the source exhibits peak rates which can be significantly larger than the long term average rate. Recent research has pointed out another key characteristic: the presence of burstiness over multiple time-scales [20, 21, 11, 12]. There is a variation in the source rate not only over a period of milliseconds to seconds, corresponding to variations within a scene, but also over a period of tens of seconds to minutes, corresponding to scenes with differing information content. In this paper, we will argue that a *renegotiated service* best addresses the presence of burstiness over multiple time-scales. This motivates the design of Renegotiated Constant Bit Rate (RCBR) service, which is the simplest possible renegotiated service, for carrying compressed video traffic.

Our results indicate that even this simple service allows a network operator to extract almost all of the statistical multiplexing gain inherent in compressed video traffic. We describe analysis and simulations that indicate that RCBR is stable, efficient and has low overhead. For example, if an MPEG-1 compressed version of the "Star Wars" movie is transferred through our service, and if the average drain rate over the lifetime of the connection is 5% above the average source rate of 374Kbps, then 300Kbits worth of buffering at the end-system and an average renegotiation interval of about 12sec are sufficient for RCBR (cf. Fig.3). In contrast, a non-renegotiated service with the same drain rate would require about 100Mbit of buffering at the end-system (cf. Fig.5). While our focus is on compressed video traffic, our results are applicable to multiple time-scale traffic in general. We believe that our approach is successful because it correctly models compressed video traffic in terms of multiple time-scales. Schemes which do not take this into account perform relatively poorly [20, 21].

We will first discuss the effect of multiple time-scale burstiness on existing service designs in Section 2. We motivate the design of RCBR from the analytic viewpoint of large deviation theory in Section 3. The RCBR scheme is described in Section 4. Section 5 analyzes the scheme and presents experimental performance results. Section 6 presents related work. Finally, in Section 7, we discuss the effectiveness of our approach and relate it to other proposals for carrying compressed video traffic.

2 Performance Problems for Multiple Time-Scale Sources

It has been recently observed by several researchers [20, 21, 11, 12] that compressed video traffic exhibits burstiness over multiple time-scales. While the short term burstiness of MPEG sources due to the I,B, and P frame structure is well known, they have found fairly long durations, as long as 30 seconds, when the data rate of the video source is continuously near its peak rate. This is due to scenes with considerable motion or flashing lights, where, independent of the coding algorithm, the coder generates traffic near its peak rate. Unfortunately, these peak rates are much higher than the long term average rate. For example, we have found that for an MPEG-1 compressed version of the "Star Wars" movie, there are episodes where a sustained peak of five times the long term average rate lasts over 10 seconds.

In typical integrated services networks, variable bitrate traffic from a source is queued at a buffer at the end-system, and the network drains the buffer at a given drain rate. The drain rate is chosen based on a traffic descriptor supplied by the source. If sources exhibiting sustained bursts are allowed only a single (static) traffic descriptor to describe their behavior, they are faced with a series of poor choices. Assume, for the moment, that the drain rate is chosen close to the long term average rate in order to maximize the statistical multiplexing gain. Then, during sustained peaks, either the data buffer at the end-system has to be very large, or there will be many losses. If the loss rate is to be small and data buffers are made large, this leads to expensive buffering at the end-systems, and long delays for the sources. One could deal with this by admitting some bursts into the network, but to do so, intermediate switches and the receiver will need large data buffers to prevent cell loss during coinciding bursts. This is expensive and can lead to excessive queueing delays. Further, even a compliant source has considerable freedom to disrupt other sources by sending in data in very large bursts (on the order of tens of megabytes). We call this loss of *protection*.

Thus, burstiness at slow time-scales leads either to a) loss of statistical multiplexing gain, b) large data loss rate, c) large buffers in end systems or switches, leading to delays and expensive regulators or d) loss of protection. Given the current framework, there is no way to simultaneously avoid all four problems. This is a simple consequence of the fact that the sustained peaks in workload are not adequately captured by static descriptors. As we will argue later, these peaks are better captured by renegotiation of the drain rate at a slower time-scale.

3 Statistical Multiplexing of Multiple Time-Scale Sources

Recent work provides the theoretical basis for understanding the gain achievable by multiplexing traffic sources exhibiting the behavior described in the previous section [24]. In this work, each variable-rate stream is modeled as a process modulated by a multiple time-scale Markov chain: a chain which consists of several sub-chains between which the transitions have very small transition probability. The dynamics within each sub-chain model fast time-scale behavior (such as correlations between adjacent frames) while the transitions between the sub-chains model slow time-scale behavior (such as scene changes). The sustained peak observed by several researchers corresponds to remaining in a high-rate sub-chain for a long time in this multiple time-scale model.

There are several key results of this work. First, when one computes the *equivalent bandwidth* of an individual multiple time-scale stream, it is found that one has to allocate the maximum of the equivalent bandwidths of the fast-sub-chains. This essentially means that one has to allocate a rate near the sustained peak, and it underscores the fact that the statistical multiplexing gain due to smoothing using buffers is of limited use for traffics such as compressed video, because the slow time-scale of the correlation is significantly longer than the delay requirement. On the other hand, when a large number of independent multiple time-scale streams are multiplexed together, much more gain can be obtained looking beyond the equivalent bandwidth of each stream in isolation. This gain is due to the fact that with high probability, not too many sources can be in a high-rate sub-chain at any one time. Thus, the bulk of the gain is obtained through averaging between sources with respect to the slow time-scale dynamics rather than through smoothing by the buffer.

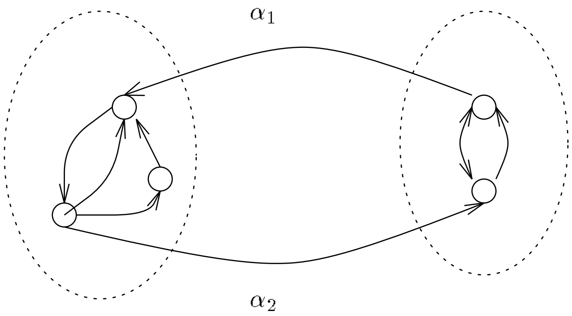

Figure 1: A multiple time-scale source with 2 sub-chains.

We now state these results more precisely. Consider a discrete time-slotted model and let X_t be the amount of data (measured in bits, bytes, cells etc.) generated per time-slot (duration of a frame,etc.). The process $\{X_t\}$ is *modulated* by a irreducible finite state Markov chain $\{H_t\}$ such that the distribution of X_t at time t depends only on the state H_t at time t. The state H_t can be thought of as modeling the burstiness of the stream at time t; the Markov structure models the correlation in the data generation rate over time. Let $\bar{\mu}$ be the mean data generation rate of the source. The state space $\mathcal{S}$ is decomposed into a union of disjoint subsets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_K$, $\mathcal{S}_k$ being the state space of the kth fast time-scale sub-chain. The kth fast time-scale sub-chain has a mean data generation rate of μ_k when considered in isolation, and the steady-state probability (long-term fraction of time) that the source is in sub-chain k is π_k. Let $\hat{\mu} \equiv \max_{1 \le k \le K} \mu_k$ be the maximum of the mean rates. Transitions between various fast time-scale sub-chains happen very rarely compared with the transitions inside each sub-chain; the former model the slow time-scale dynamics of the traffic stream. Let $\alpha_1, \alpha_2, \ldots, \alpha_m$ be the probabilities of these rare transitions; these are very small parameters. Thus, the stream would typically spend a long time in a sub-chain, and then occasionally jump to a different sub-chain. This multiple time-scale Markov-modulated model has been used in several video traffic studies [23, 19]. See Figure 1 for an example of a source with two sub-chains.

Consider now the situation when a single variable-rate traffic stream is buffered before entering a link of constant rate $\bar{c}$, $\bar{c}$ greater than the mean rate of the stream. Let the size of the buffer be B. We first want to estimate the probability of data loss due to buffer overflow and use the estimate to assign an equivalent bandwidth $e(p_{qos}, B)$ to the stream for a given quality-of-service loss probability requirement p_{qos}. For the asymptotic regime in which the statistics of the traffic stream is *fixed* and the buffer size B grows large, a *large deviations* estimate of the loss probability has been derived in [13, 9] and [18]. The estimate can be expressed in terms of a convex function $\Lambda(r)$ (called the log spectral radius function), which can be computed given the statistics of the traffic, such that for large buffer sizes B, the loss probability p is approximately:

$$p \approx \exp(-r^* B)$$

where r^* is the unique positive root of the equation $\Lambda(r) =$

$r\bar{c}$. For a given QOS requirement p_{qos}, one can use the above formula to derive the equivalent bandwidth $e(p_{qos}, B)$ of the stream. It is the minimum rate $\bar{c}$ of the link such that the loss probability requirement p_{qos} is satisfied, and is given by the formula:

$$e(p_{qos}, B) = \frac{\Lambda(r^*_{qos})}{r^*_{qos}}, \qquad \text{where } r^*_{qos} = \frac{-\log p_{qos}}{B} \tag{1}$$

Note that the equivalent bandwidth is between the mean and peak rates of the stream.

For multiple time-scale sources, one now has to look at the *joint* asymptotic regime when simultaneously the rare transition probabilities α_i's are close to zero and the buffer size B is large enough to absorb the fast time-scale fluctuations of the stream. It is shown in [24] that the loss probability in this asymptotic regime is

$$p \approx \exp(-\min_{1 \le k \le K} r^*_k \cdot B)$$

where r^*_k is the unique positive root of the equation $\Lambda_k(r) = r\bar{c}$ and Λ_k is the log spectral radius function of the kth fast time-scale sub-chain when considered in isolation (Theorem 4.2 of [24]). Hence the equivalent bandwidth $e(p_{qos}, B)$ of the multiple time-scale stream is given by

$$e(p_{qos}, B) = \max_{1 \le k \le K} e_k(p_{qos}, B), \tag{2}$$

where $e_k(p_{qos}, B)$ is the equivalent bandwidth of the kth fast time-scale sub-chain when considered in isolation. The intuition is that buffer overflows are due mainly to the effects of the most bursty sub-chain, and thus the bandwidth needed for the entire stream is just the bandwidth of that particular sub-chain.

Consider now the case when a small number n of independent multiple time-scale streams are multiplexed onto a constant rate link. It can be shown using the properties of the log spectral radius function that the equivalent bandwidth of the aggregate stream is simply the sum of the equivalent bandwidths of the individual streams, i.e.

$$e_{\text{agg}}(p_{qos}, B) = \sum_j e^{(j)}(p_{qos}, B).$$

Thus, the bandwidth needed for the aggregate stream is the sum of the bandwidths of the most bursty sub-chains in the individual streams. For the particular case when the streams have the same statistics and a common equivalent bandwidth $e(p_{qos}, B)$,

$$e_{\text{agg}}(p_{qos}, B) = ne(p_{qos}, B).$$

Here, the statistical multiplexing gain is reflected in a linear decrease in the buffering required per source, since in the multiplexing scenario the buffer of size B is shared among the n sources. This gain is due to the statistical multiplexing in the fast time-scale dynamics. On the other hand, by Eqn. (2), the capacity required per source is at least $\hat{\mu}$, the maximum of the mean rates of the fast time-scale sub-chains of each stream. This means that the multiplexing gain is still limited by the slow time-scale dynamics. In the case when $\hat{\mu}$ is near the peak rate of the source and far away from the overall mean rate $\bar{\mu}$ of the stream, we thus conclude that the gain from multiplexing a small number of streams is rather limited.

To get significant multiplexing gain, the limitation imposed by the slow time-scale dynamics can be overcome by multiplexing many independent and statistically similar streams. By a law of large number effect, the probability that many streams are simultaneously in a bursty sub-chain is small, so that a small loss probability can be guaranteed even if the capacity allocated per stream is less than $\hat{\mu}$. Specifically, let n independent and statistically identical multiple time-scale streams be multiplexed onto a link of rate $n\bar{c}$ and with a buffer of size nB (i.e. the link capacity and buffer space *per stream* is fixed in this scaling.) An estimate of the loss probability, in the regime of large n, can be obtained in terms of *only* slow time-scale statistics of the individual stream (with the fast time-scale dynamics averaged out.) Specifically, consider a random variable which takes on the value μ_k with probability π_k, where π_k is the steady-state probability that the stream is in sub-chain k and μ_k is the mean rate of sub-chain k. Let L be the log moment generating function of this random variable:

$$L(r) \equiv \log \sum_{k=1}^{K} \pi_k \exp(\mu_k r).$$

and define L^* by:

$$L^*(\mu) = \max_{r>0}[\mu r - L(r)],$$

the *Legendre* transform of L. Then the asymptotic estimate of the loss probability when there are many sources is

$$p \approx \exp(-L^*(\bar{c}) \cdot n) \tag{3}$$

Note that (3) is simply the Chernoff's estimate of the probability that the streams are in a combination of sub-chains whose total mean rate exceeds the channel capacity [27, 16], and does not depend on the fast time-scale statistics of the streams. The buffering essentially absorbs the fast time-scale variations of the streams but has little effect on the slow time-scale.

There has been some recent work suggesting that compressed video traffic can have a self-similar structure [12]. This model implies the presence of correlations in *many* different time-scales. While this is an issue worthy of further research, we would like to point out that the above results yield the key insight that, from the point of view of characterizing statistical multiplexing gain, the crucial matter is to distinguishing the time scales that are *faster* and those that are *slower* than that dictated by the buffering/delay requirements. For the former, the gain is obtained by smoothing using the buffer. For the latter, the gain is obtained by averaging between different sources, and moreover, the amount of such gain depends only on the *stationary* distribution of the slow time-scale process, and is independent of how many slow time-scales there are or how they are structured. (It can in fact be shown that even the Markovian structure of the slow process is not crucial in reaching this conclusion.) Thus, while the statistical studies of the "Star Wars" sequence in [12] indicate a self-similar structure in a spectrum of time-scales starting from tens of seconds to minutes to hours, this will not likely have a significant impact on the multiplexing gain since these time-scales are all slower than the buffering time-scale.

It is possible to achieve fully the statistical multiplexing gain characterized by Eqn. (3) when no restrictions are imposed on the traffic entering the network; however, such a scheme lacks robustness as there is no protection against malicious users. Moreover, this ideal scheme requires large buffering at each network node and also at each receiving node, the buffer size scaling linearly with the number of streams. The proposal to be presented in the next section is designed to extract the bulk of the statistical multiplexing gain while giving some protection to users and dispensing with buffering at the network node, at the expense of more signaling. Our scheme essentially focuses on the gain in the averaging of the slow time-scale dynamics rather than the averaging of the fast time-scale dynamics. The effect of a large number of sources on the statistical multiplexing gain has also been analysed recently in [10] and [2].

4 The RCBR Scheme

4.1 RCBR Service Description

In this section, we describe Renegotiated CBR (RCBR) service. This type of service has been proposed before for data service - the novelty of our scheme is to use renegotiations to deal with slow time-scale behavior in a compressed video workload. The basic idea for RCBR is to augment standard (static) CBR service with a renegotiation mechanism. In static CBR service, at the time of call setup, an end-system initiates a signalling message requesting a certain constant bandwidth from the network. In the forward pass, each switch performs an admission control test, and if this is successful, makes a tentative reservation and passes on the call setup message to the next switch along the path. On the reverse pass, if all the switches have admitted the call, the tentative reservation is confirmed, and the call is allocated a VCI. Since the traffic is described by a single number, the admission control test is trivial. Resources corresponding to the requested rate are reserved at each contention point. For CBR service, this would mean a small amount of buffers in addition to billing or other housekeeping information.

Users of RCBR service are given the option to renegotiate their service rate at any time. Renegotiation consists of sending a signalling message along the path, requesting an increase or decrease of the current service rate. If the request is feasible, the network allows the renegotiation, and upon completion of the request, the source is free to send data at the new CBR rate. During renegotiation, a switch controller does not need to recompute routing, allocate a VCI or acquire housekeeping records. This reduces the renegotiation overhead.

We anticipate that a rate increase or decrease would happen once every ten seconds or so (see Section 5.3). Thus, a renegotiation mechanism based on signalling, such as the one proposed as an ITU standard [1] would be adequate. A hardware implementation of signalling for this purpose is described in [3]; this is probably not necessary, given our time-scales of control. Note that in order to limit the renegotiation rate, it is likely that a user will be charged for each renegotiation, just as users are now charged per call setup.

Stored (off-line) and interactive (online) applications use RCBR services differently. Off-line sources can compute the desired series of CBR rates (the *renegotiation schedule*) in advance, and so renegotiation to increase the service rate can be carried out before actually increasing the data rate. For example, if a renegotiation takes 50ms because of speed of light propagation delays, a source could initiate renegotiation 50 ms before it needs the new rate. In practice, to ensure that data losses do not happen, reservations could span a few milliseconds longer than strictly necessary. If all systems in the network share a common time base, advance reservations could be done for some or all of the data stream [28]. Renegotiated decreases in the rate happen only after the source rate actually decreases. Otherwise, a switch might decrease its service rate before the source starts sending at the new rate. However, in both cases, the renegotiation and data transfer occur in parallel.

For interactive applications, the renegotiation schedule cannot be calculated in advance. Instead, we propose that an active component monitor the user-network buffer and initiate renegotiations based on the buffer occupancy level. This monitor could be part of the session layer in an ISO protocol stack, or reside in the Network Interface Unit (NIU) for "dumb" endpoints. As before, renegotiation and data transfer can happen in parallel. In Section 5.2.2, we describe some simple heuristics for initiating renegotiation that perform reasonably well.

What happens if a renegotiation fails? A trivial solution is that the source that failed renegotiation can try again. Of course, data will build up in the end-system data buffer while the second request proceeds, and there is the possibility of data loss. This may not be acceptable for some users. Such users might reserve resources at or close to the peak rate, so that the frequency of renegotiation is highly reduced, and so is the possibility of renegotiation failure. There is a clear tradeoff between buffer size, requested rate and the frequency of renegotiation. In any case, note that even if the renegotiation fails, the source can keep whatever bandwidth it already has.

Second, during admission control, a switch controller might reject an incoming call even if there is available capacity, if the resources used by the new call will make future renegotiations more likely to fail. This allows the network operator to trade off call blocking probability and renegotiation failure probability. We consider admission control in more detail in Section 5.4.

Finally, the signalling system could ask the user or application (perhaps out of band) to reduce its data rate. Since the network interface (i.e. the transport layer or NIU) is expected to be no more than a few milliseconds away from the end point, the control loop between the network interface and the user will be tight, so that responding to such signals should be easy, particularly for adaptive codecs [17]. Recent work suggests that even stored video can be dynamically requantized in order to respond to these signals [22, 8].

Thus, we believe that there are a number of techniques we can call upon to deal with renegotiation failures. With an appropriate combination, some users can choose to get few or no renegotiation failures. Other users might still see failures, but this may be quite acceptable, particularly if it is reflected in the pricing structure.

4.2 System Overhead

In this section, we consider the system overheads incurred by renegotiation. We argue that renegotiations impose only a small system overhead, and that an RCBR system scales well with network size and traffic intensity. We will also consider how to implement RCBR within existing ATM Forum traffic management proposals.

We first examine the overhead for renegotiation. Note that since renegotiations take place at the slow time-scale of the source, the overhead for RCBR at each source is inherently small. (Experiments in Section 5.3 show that for a source with a long term average rate of around 400Kbps and an end-system buffer of 300Kbits, the average renegotiation interval is on the order of 10 seconds.)

We now examine the overhead per renegotiation at an end-system and switch controller. For concreteness, we assume that the end-system implements an ISO-like protocol stack with an application layered above a session, transport and datalink layer. When an application performs a `write` system call to a session layer interface, application data is copied or mapped into a per-connection session layer buffer. To implement CBR service, this data would be metered out to the network at a constant rate either by the transport layer or the host adaptor, depending on the granularity with which the bitrate is defined. To add renegotiation to an existing CBR system, we add a buffer monitor routine that would be invoked every time data enters or leaves the buffer. This would need changes to the `write` system call and the task that drains the session layer buffer. The buffer monitor would initiate renegotiation (based either on the current buffer state or long-term statistics) by invoking the signalling subsystem. Thus, the overhead associated with the monitor is simply a procedure call plus whatever simple actions the monitor needs to take in order to determine whether renegotiations are necessary. In the common case, where no action is needed, the monitoring overhead is negligible.

Once the monitor determines that renegotiation is needed, composing and transmitting a signalling message is straightforward, since the data path for the renegotiation message is identical to that taken by the call setup message. If the signalling protocol is complex, the renegotiation message can be treated as a special case, and overheads can be reduced by using techniques such as pre-computed message headers and inlining of procedure calls. Thus, we believe that with careful coding, a mean renegotiation frequency of around ten seconds will not pose a significant overhead to the end-system.

When a renegotiation message arrives at a switch controller, the controller must check if sufficient bandwidth is available, and then pass on the renegotiation message to the next hop switch. Let us consider the overheads in performing this operation. For concreteness, we will assume a switch controller that shares a bus with the switch (such as in the FORE ASX 200 switch), so that the transmission delay from the switch fabric to the switch controller can be neglected. When a signalling message arrives at the switch controller, it must handle an interrupt. If renegotiation is special cased, which would be a reasonable thing to do, then bandwidth test can be done in the interrupt handler context, which does away with context-switching overheads. Since the network is internally CBR, the test reduces to checking if the rate asked for exceeds the available capacity, which only takes a few instructions. Next, the renegotiation message must be put into a cell and forwarded on to the next switch by introducing this cell into the switch fabric. Typical switch controllers have a special low latency path for this, and so we ignore the overhead in this path. If the onward message is pre-stored, with only the VCI and requested rate filled in on-line, this further reduces the latency.

By counting instructions for the device driver's interrupt routine, the bandwidth test, filling in a pre-computed cell, and introducing this into the switch fabric, we estimate that it is possible to reduce the CPU overhead per hop to about 200 instructions, or about 3 microseconds on a modern 60 MIPS processor. Even with a fairly conservative interrupt latency time of 7 microseconds, this comes to about 10 microseconds of CPU overhead per renegotiation per switch. If each renegotiation takes 10 microseconds at each switch controller, and renegotiations take up 40% of the switch controller load, then each controller can support 40,000 renegotiations per second. Conservatively assuming a mean renegotiation interval of 1 sec, this translates to capacity for handling 40,000 calls. More capacity can be easily added by using shared memory multiprocessors as switch controllers. We conclude that we can deal with renegotiation overheads with existing technology, and in the context of existing signalling systems.

We now consider how well RCBR scales with latency in the path, number of sources, and number of hops. As the propagation delay along a path increases, the performance of off-line RCBR does not decrease, since the end-system can compensate for latency when asking for a new rate. However, the performance of online RCBR would decrease with increase with latency, since prediction accuracy would decrease. This can be compensated for by increasing the end-system buffer, or by asking for more bandwidth than needed, thus reducing the statistical multiplexing gain.

As the number of RCBR sources in the network increases, the signalling load would also increase. As discussed above, we believe that current technology can already handle up to 40,000 RCBR sources, which seems adequate. Increases in CPU performance scale this number exponentially with time.

As the mean number of hops in the network increases, the probability of renegotiation failure would likely increase, since each hop is a possible point of failure. Moreover, the net signalling load on the network also increases. However, if there is a simultaneous increase in the number of alternate routes in the network, then load balancing at the call level might compensate for this increase. This is still an open area for research.

Finally, we consider how one might implement renegotiation in the existing ATM Forum traffic management framework. One way to do this would be to augment Q2931 signalling with renegotiation messages, and implement renegotiation along the lines discussed above [1]. Alternatively, Resource Management cells used for ABR service could be reinterpreted as renegotiation requests. In effect, we could reuse the ABR mechanisms, but change the direction of information flow to be from user to network, instead of network to user, as it currently stands. This implementation is not without problems, since the signalling system and the RM-cell manager need to maintain a consistent view of the system state. Given that the renegotiations occur infrequently for each source, the added complexity may be unwarranted.

5 Analysis

5.1 Stochastic Analysis using Multiple Time-Scale Model

Using a multiple time-scale Markov model for the traffic stream, we can characterize how much of the multiplexing gain described in Section 3 our proposed scheme can capture. Assume that the scheme does an ideal job in separating the slow and fast time-scales, such that it renegotiates a new CBR rate whenever the source jumps from a fast time-scale sub-chain to another. For a given endpoint buffer size B and a buffer overflow loss probability requirement p_{qos}, the new CBR rate it should renegotiate for is the equivalent bandwidth $e_k(p_{qos}, B)$ of the sub-chain k the source enters (the equivalent bandwidth being computed as in Section 3). Since π_k is the steady-state probability that the stream is in sub-chain k, the stream will demand a CBR rate of $e_k(p_{qos}, B)$ for π_k long-term fraction of time. When n independent and statistically identical streams share a single link of capacity $n\bar{c}$, the probability of renegotiation failure is roughly the probability that the total CBR bandwidth demand exceeds the available capacity; for large n, we can use Chernoff's estimate to approximate this as

$$\exp(-L_e^*(\bar{c}) \cdot n), \tag{4}$$

where

$$\begin{aligned} L_e(r) &= \log \sum_{k=1}^{K} \pi_k \exp(e_k(p_{qos}, B) \cdot r) \\ L_e^*(\mu) &= \max_{r>0}[\mu r - L_e(r)] \end{aligned}$$

Comparing this to the loss probability (3) when there is a shared buffer of size nB, we see that this renegotiation failure probability is larger since the equivalent bandwidth $e_k(p_{qos}, B)$ of every sub-chain is greater than its mean rate μ_k. Viewed in another way, the capacity per stream needed for the same level of performance is greater in our scheme. This discrepancy in bandwidth requirement is due to the fact that our scheme does not take advantage of a large shared buffer to effectively absorb all fast time-scale variations through statistical multiplexing. However, for sources with small fast time-scale fluctuations superimposed on larger slow time-scale variations, the equivalent bandwidths of the sub-chains will be close to the mean-rates for reasonably sized buffers, and the discrepancy will be small. This is further substantiated by the experimental results presented in Section 5.3.

5.2 Computing Renegotiation Schedules

In the previous section, we have analyzed the performance of RCBR when it has perfect knowledge of the stochastic model of the source. This section presents algorithms to compute a renegotiation schedule that can take advantage of the time-scale separation, but without explicit knowledge of which sub-chain the source is in at any given time. We present two algorithms that transform a given data rate function into a stepwise CBR data rate function. The first algorithm determines an optimal schedule for a playback application based on total knowledge of the user's data rate function and a pricing model discussed below. The second algorithm is a causal heuristic that could be used for interactive users, where the rate function is not known in advance.

5.2.1 Optimal Renegotiation Schedule

We model the problem with a slotted time queue. For video, a time slot would typically be the duration of a frame. Renegotiations occur on the boundary between slots. Let $r_i, i = 0, 1, \ldots, N-1$ denote the amount of data entering the queue during time slot i, and let s_i denote the service rate during time slot i. The session duration is N time slots. We assume the service rate during any time slot is in a given set $\mathcal{C} = \{c_0, c_1, \ldots, c_{K-1}\}$.

We have assumed a constant cost per renegotiation ϕ and a cost γ per *allocated* bandwidth and time unit. Therefore, the total cost is given by

$$\phi \cdot \sum_{i=1}^{N-1} (1 - \delta(s_{i-1}, s_i)) + \gamma \cdot \sum_{i=0}^{N-1} s_i \tag{5}$$

with

$$\delta(x, y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{otherwise} \end{cases}$$

For a given r_i, the optimal allocation minimizing the total cost, has to be found, subject to the buffer constraint

$$0 \leq b_i \leq B \text{ for } i = 0, 1, \ldots, N-1 \tag{6}$$

where b_i is the queue size at the end of time slot i, with

$$b_i = \begin{cases} 0 & \text{if } i < 0 \\ \max\{b_{i-1} + r_i - s_i, 0\} & i = 0, 1, \ldots, N-1 \end{cases} \tag{7}$$

We solve this optimization problem with a Viterbi-like algorithm [26]. Let us first introduce some notation (cf. Fig. 2). A *node* is a 4-tuple (i, k, b, w), where i denotes (discrete) time, $k \in \{0, \ldots, K-1\}$ denotes a bandwidth allocation $c_k \in \mathcal{C}$, $b \in \{0, \ldots, B\}$ denotes a buffer occupancy, and w denotes the weight, which equals the partial cost of the best path to this node. A *branch* connects a node (i, k, b, w) to another node $(i+1, k', b', w')$ if $b' = \max\{b_i + r_{i+1} - c_{k'}, 0\}$. It has an associated weight of $\gamma \cdot s_{i+1} + \phi \cdot (1 - \delta(s_i, s_{i+1}))$. A branch represents one step in the evolution of the system state, given a choice of the new rate allocation $c_{k'}$. A *path* is a sequence of branches. The cost of a path is the sum of the cost of its branches. All possible paths form the *trellis*. A *full path* is a path connecting a node with $i = 0$ with a node with $i = N-1$, and corresponds to a feasible renegotiation schedule.

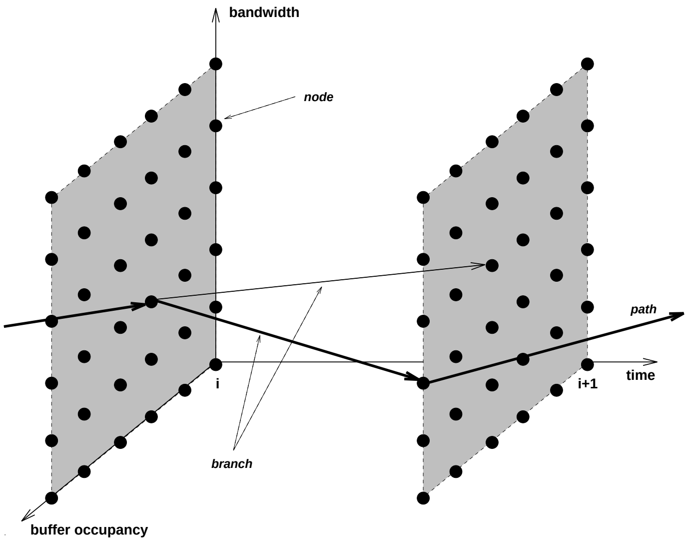

Figure 2: An illustration of the trellis to be used for the Viterbi-like algorithm.

Now we can formulate the optimization problem as follows: find the shortest path from some node at time zero to some node at time $N-1$. The algorithm to do this is presented below.

1. Set $i = 0$. Create the initial set of nodes $(0, k, 0, 0)$ for $k \in \{0, \ldots, K-1\}$.
2. Create all the branches between nodes of slot i and nodes of slot $i+1$. Set the weight according to (5) for the nodes of slot $i+1$.
3. Prune paths according to Lemma 1 given below.
4. Increment i and repeat steps 2 and 3 as long as $i < N$.

5. Choose one of the paths with the minimum weight as the solution.

We now present a lemma that governs the pruning of paths.

Lemma 1 *A path X going through a node $x = (i, k_x, b_x, w_x)$ is not optimal if there exists a path Y through a node $y = (i, k_y, b_y, w_y)$ such that* [1]

$$b_y \leq b_x \text{ and } w_y \leq w_x + \begin{cases} \phi & \text{if } k_y \neq k_x \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Proof: Assume the condition is true. First, if $k_x = k_y$, then path Y has smaller or equal buffer occupancy and smaller or equal weight than path X. Due to the buffer constraint, for all future time slots, the best full path containing X must have a bandwidth allocation that is at least the bandwidth allocation of the best full path containing path Y. Therefore, it cannot have a lower weight than the best full path containing Y. Second, if $k_x \neq k_y$ then for any $k \in \{0, \ldots, K-1\}$ such that a branch from x to a node $x' = (i+1, k, b_{x'}, w_{x'})$ exists, there exists a branch from y to a node $y' = (i+1, k, b_{y'}, w_{y'})$ such that $b_{y'} \leq b_{x'}$, as the service rate in interval $i+1$ is the same and by assumption, $b_y \leq b_x$. As the difference in cost of the branch connecting y to y' and the branch connecting x to x' cannot be larger than ϕ, the first part of the proof applies to x' and y'. □

Instead of the buffer bound (6), it is also possible to enforce a delay bound. This might be desirable in real-time applications, if sufficient buffer space is available, but the Quality of Service still requires to keep delays low. The condition for all data entering during time slot $i - D$ to have left at the end of time slot i is

$$b_{i-D} \leq \sum_{j=-D+1}^{0} s_{i+j} \quad i = D, \ldots, N-1 \tag{9}$$

The runtime complexity of the optimization algorithm very much depends on the cost ratio ϕ/γ, the buffer size B and above all the number of bandwidth levels K. Also, the user rate function $\{r_i\}$ has has an impact on how many candidates remain valid at each time slot. We have found that if we restrict K to about 20, optimizations can be done in reasonable time, even for long traces like the Star Wars movie (approx. 174000 samples) [11]. For larger K, e.g. 100, it quickly becomes impracticable.

We call *bandwidth efficiency* the ratio of the original stream's average rate to the average of the piecewise constant service rate, i.e.

$$\frac{\sum_{i=0}^{N-1} r_i}{\sum_{i=0}^{N-1} s_i}.$$

It is clear from Fig. 3 that there exists a tradeoff between bandwidth efficiency and renegotiation frequency. This tradeoff depends on the cost ratio ϕ/γ: raising the price for renegotiation results in a lower renegotiation frequency and a lower bandwidth efficiency, and vice versa. The network operator can announce these prices to the user, and the user optimizes his network usage accordingly. Note how close the bandwidth efficiency gets to one with very reasonable renegotiation frequencies; for example, with one renegotiation every 177 frames, which corresponds to slightly more than 7 seconds, we achieve over 99% of bandwidth efficiency! This is a clear manifestation of the slow time-scale behavior of compressed video streams.

[1] Note that this allows us to do more than the "standard Viterbi" pruning, i.e. among paths terminating in a common node, keep only the one with the lowest weight. We can also prune across nodes.

5.2.2 Causal Renegotiation Schedule

For interactive sources, the optimization algorithm described above cannot be used to determine optimal renegotiation points. For such sources, causal heuristics have to be used to make decisions about requesting new rates. Such heuristics predict the future bandwidth requirement based on some statistics collected in the past. The goal of this section is to show that heuristics resulting in satisfactory performance do indeed exist, although their derivation is somewhat *ad hoc*.

The heuristic we present is based on a AR(1) bandwidth estimator and on buffer thresholds. Three parameters have to be tuned: a high and a low buffer threshold B_h and B_l, respectively, and a time constant T, which should reflect the long-term rate of change of the rate function. The rate predictor we have used is

$$\widehat{r}_{i+1} = (1 - T^{-1})\widehat{r}_i + T^{-1}(r_i + \max\{b_i - B_h, 0\}) \tag{10}$$

where r_i is the actual incoming rate during slot i, and b_i is the buffer size at the end of slot i. The additional term $T^{-1} \cdot \max\{b_i - B_h, 0\}$ in the estimator adds the bandwidth necessary to flush the current buffer content within T. This is necessary to have a sufficiently fast reaction to sudden large buffer buildups.

The algorithm is very simple. Let

$$s_{new} = \lceil \frac{\widehat{r}_{i+1}}{\Delta} \rceil \Delta \tag{11}$$

with Δ the bandwidth allocation granularity. A new bandwidth s_{new} is then requested if

$$(b_i > B_h \text{ and } s_{new} > s) \text{ or } (b_i < B_l \text{ and } s_{new} < s) \tag{12}$$

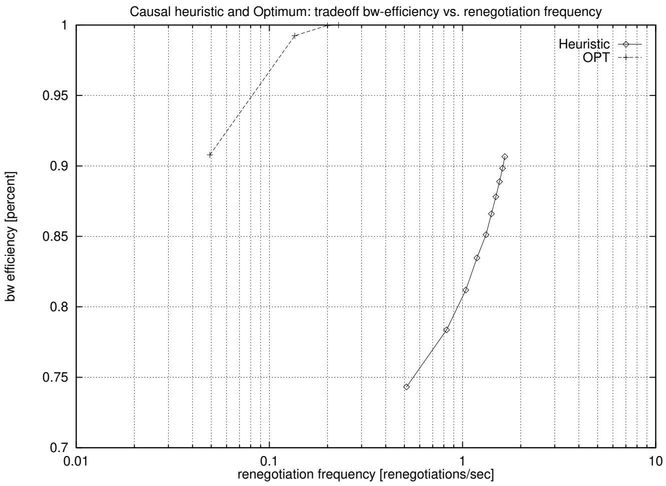

Figure 3: The tradeoff between bandwidth efficiency and renegotiation frequency for the AR(1)-based heuristic, compared to the optimum, for the "Star Wars" trace.

It can be seen in Fig. 3 that using the heuristic, we need about one renegotiation a second to achieve 80% of

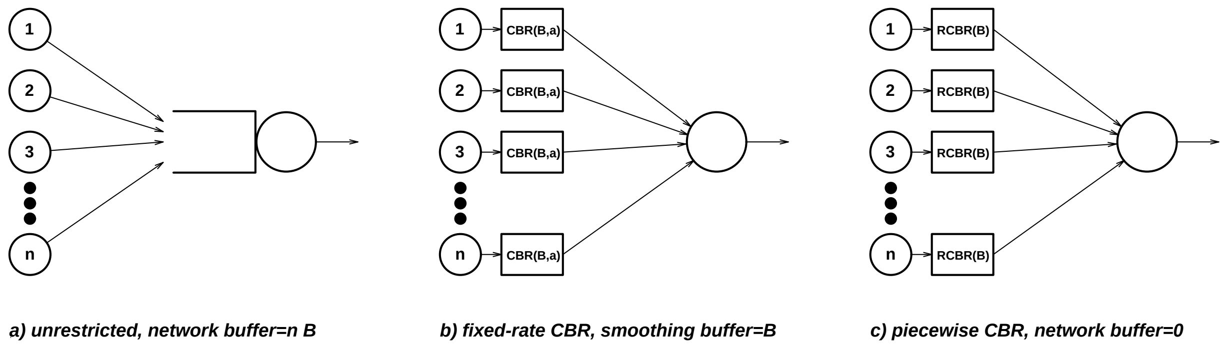

Figure 4: The three experiments to assess statistical multiplexing gain (SMG) of our proposed service.

bandwidth efficiency. Although this is considerably less than what can be achieved with the optimal allocation, it still represents a relatively small load on the signalling system. However, this gap suggests that better heuristics can be found, and we hope to address this problem in future research. For example, the prediction quality could be improved by taking into account the inherent frame structure of MPEG encoded video.

5.3 Experimental results

We have compared the statistical multiplexing gain (SMG) achievable through our scheme with two other scenarios (cf. Fig. 4), in order to see how much of the SMG inherent in video traffic the RCBR service can extract.

The first scenario (a) multiplexes n streams without any restriction on a server with rate c and buffer size nB. This is used to estimate the maximum achievable SMG for the given sources. The second scenario (b) represents traditional CBR service, with a smoothing buffer of size B at the network entry and a fixed CBR rate a for each source. The third scenario (c) represents our approach. Each source is smoothed by a dedicated buffer of size B and transformed into a stepwise CBR stream, which is then transported without further buffering in the network (except some cell level buffering). The total service rate is c and the total amount of buffering is fixed at nB in all three scenarios[2].

The streams we have used is the MPEG-1 encoded trace of the Star Wars movie [11]. The n sources are randomly shifted versions of this trace. The buffer size B was chosen as 300Kbit, slightly more than the maximum size of three consecutive frames in the trace. This approximately corresponds to the buffering of current video codecs. In the optimization for RCBR described in Section 5.2.1, we have used a bandwidth granularity of 1Kbps, with $K = 50$.

To assess the SMG for all three scenarios, we have determined the channel service rate per stream c/n, as a function of n, needed to guarantee a desired bit loss probability. In scenario (a) and (b), bits are lost due to buffer overflow. In scenario (c), bits are lost due to failure in renegotiating for a higher CBR rate (in which case we assume the source has to temporarily settle for whatever bandwidth remaining in the link until more bandwidth becomes available). Determining c is straightforward for scenario (b). For scenarios

[2]Note that this comparison is pessimistic, because shared buffers in a switch are more costly than endpoint buffers, due to more severe timing constraints. Therefore, given a total buffer size nB, scenarios (b) and (c) are actually preferable.

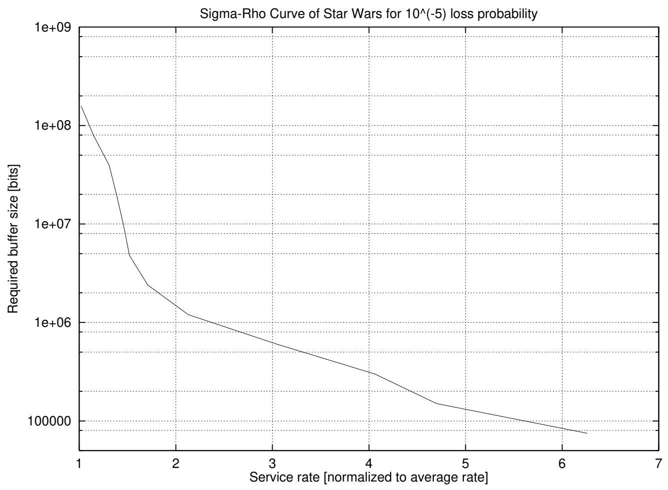

Figure 5: The (σ, ρ)-curve of the video trace for 10^{-5} loss.

(a) and (c), we find for each n the *minimum* c that guarantees the desired loss probability: for each n, we do a binary search on c; for each step in the search, we do many simulations, where each simulation has a randomized phasing of the sources, and compute the average fraction of bits lost as an estimate of the loss probability. At each step, we repeat the simulations until the sample standard deviation of the estimate is less than 20% of the estimate. Results for 10^{-5} loss probability requirement are depicted in Fig. 6.

We observe that case (a) represents a lower bound on the bandwidth per source for a given loss probability, and therefore an upper bound on the SMG. This is because the buffer is shared among all n streams. In the CBR case (b), the bandwidth per stream is a, of course, regardless of the number of streams n. Note that a can be determined from the corresponding (σ, ρ)-curve of this trace in Fig. 5. (For any given service rate ρ, this curve gives the minimum buffering σ such that the fraction of bits lost is less than 10^{-5}.) As has been previously observed in the literature, this is close to the peak rate [21, 12]. For the given buffer size, a is 4.06 times the trace's average rate of 374Kbps.

Our scheme achieves slightly less SMG than the unrestricted case because buffers are not shared and the fast time-scale multiplexing gain is not exploited, as explained

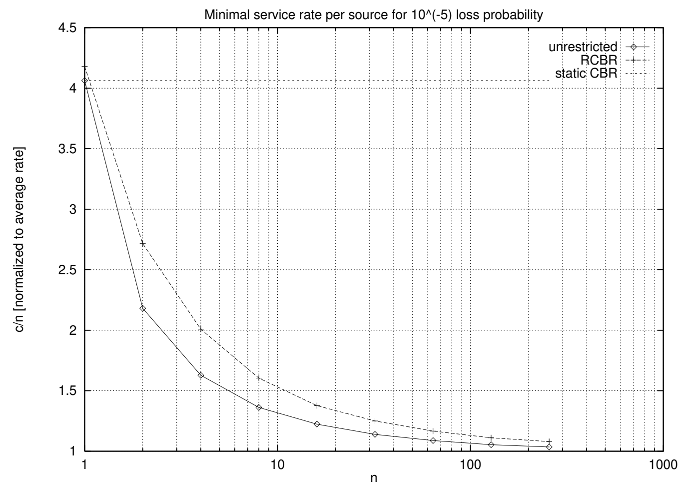

Figure 6: Statistical multiplexing gain (SMG) achievable for 10^{-5} loss probability.

in Section 5.1. Nevertheless, we are able to extract most of the SMG, especially for high service rates. For example, at at service rate of 100 times the average rate, we require about three times less bandwidth than the static CBR approach. Asymptotically, the value for c/n for the stepwise CBR function approaches the inverse of the bandwidth efficiency obtained in the optimization algorithm.

5.4 Admission Control

In this section we present some insight into an admission control scheme suitable for RCBR. Due to the inherent uncertainty in the bandwidth requirements of calls over their lifetime, there is always a possibility that a renegotiation for a higher bandwidth can fail due to unavailable capacity. The role of call admission control is to maintain the load of the network at a reasonable level so that the probability of such renegotiation failure is small. Note that there is a tradeoff between call blocking and renegotiation failure probabilities. Also, users who want a hard guarantee on always getting the bandwidth they need can reserve a bandwidth near their peak rate and never renegotiate for a lower rate (though they may have to pay a higher price for their calls). Such a hard guarantee is possible because users never have to give up the bandwidth they have successfully negotiated for. On the other hand, users who want to pay less can negotiate for higher rates only when they need more bandwidth, but they do so at a risk that they may not be able to obtain it. Thus, admission control allows the network to extract good multiplexing gain, and the users get charged lower prices in return for the small risk they take.

The basis for admission control is the ability to estimate the renegotiation failure probability. This is possible in a large system because there is a lot of statistical regularity in the aggregate traffic, due to the law of large numbers (particularly if there is peak rate constraint on each user). We explain this further below.

While renegotiations take place at a slower time-scale than the fast time-scale variations of the traffic streams, there is another separation of time-scale between the renegotiations and call arrivals and departures. While renegotiations occur every few seconds, calls last for minutes or even hours. Thus, for the purpose of call admission, a reasonable criterion is to keep the probability of renegotiation failure low given the fixed number of calls admitted (i.e. no need to worry about call level dynamics.) A given desired threshold on the renegotiation failure probability determines the maximum number of calls that can be admitted into the system at any one time.

For stored video applications, the renegotiation failure probability can be estimated reliably. Given an optimal renegotiation schedule, we compute the empirical distribution (histogram) of bandwidth requirements throughout the lifetime of a call, i.e. the fraction of time π_k that a bandwidth level c_k is needed during the call, $k = 1 \ldots K$. This distribution can be viewed as the traffic descriptor of the call. When there are n such calls sharing a link of total capacity c, the renegotiation failure probability p_f can be estimated by Chernoff's approximation (a similar approximation was used in Section 5.1). We use the following refinement due to Bahadur-Rao [6]:

$$p_f \approx \frac{1}{\eta\sqrt{2\pi n L''(\eta)}} \exp(-L^*(\frac{c}{n}) \cdot n) \tag{13}$$

where

$$\begin{aligned} L(r) &= \log \sum_{k=1}^{K} \pi_k \exp(\mu_k r) \\ L^*(\mu) &= \max_{r>0} [\mu r - L(r)] \end{aligned}$$

and η satisfies $L'(\eta) = \frac{c}{n}$. Note that this formula gives an $O(\frac{1}{\sqrt{n}})$ correction term to the basic exponential estimate.

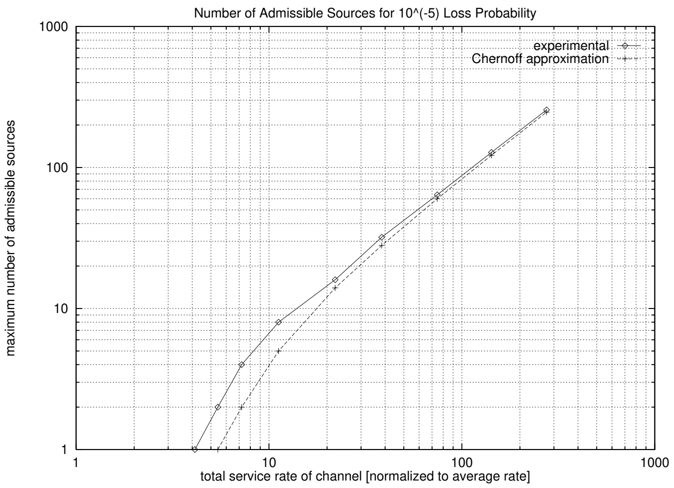

Figure 7: Performance of Chernoff's Approximation

Using this formula, the maximum number of calls the system can carry for a given threshold on the renegotiation failure probability can be computed, and new calls will be rejected when this number is exceeded. Note that the system can deny new calls even when there is available capacity, so as to safeguard against fluctuations of bandwidth requirements of the calls already admitted. Thus, Chernoff's approximation quantifies the amount of slack needed in the available capacity. In the presence of ABR traffic, this slack could be fairly large without underutilizing the network.

We test the accuracy of this approximation by comparing it with the simulation results for the Star Wars video trace in Section 5.3, assuming the optimal renegotiation schedule used there. For a wide range of total link bandwidth and a desired renegotiation failure probability of 10^{-5}, we compare the maximum number of Star War sources admissible as predicted by formula (13) and the actual number obtained in the simulation results (Fig. 7). We see that the approximation is in general conservative and quite accurate when there are more than 15 sources. For example, with a total bandwidth 75 times the average rate of the source, Chernoff's approximation predicts 60 admissible sources, while the actual number is 64.

For interactive applications, accurate knowledge of the bandwidth requirements of a call is usually not available *a priori*. Here, we propose to use the bandwidth reservation histories of the current calls in the system to estimate their future behavior for the purpose of admission control. More specifically, at each time when a new call arrives, we compute for each k ($k = 1, \ldots, K$), the total amount of time, summed over all current calls, that bandwidth level c_k has been reserved in the past. This yields an empirical distribution $\{\hat{\pi}_k\}$ of bandwidth requirements for a typical call. The idea is to use $\{\hat{\pi}_k\}$ to estimate the distribution $\{\pi_k\}$ of the bandwidth requirements throughout the entire lifetime of a call. Making the conservative assumption that the new call will always reserve at the peak rate P, the bandwidth remaining for the other calls after the admission of the new call is $c - P$. Using Chernoff's approximation again, we can estimate the renegotiation failure probability, if this call is accepted, to be

$$p_f \approx \frac{1}{\eta\sqrt{2\pi n \hat{L}''(\eta)}} \exp(-\hat{L}^*(\frac{c-P}{n}) \cdot n)$$

where

$$\begin{aligned} \hat{L}(r) &= \log \sum_{k=1}^{K} \hat{\pi}_k \exp(\mu_k r) \\ \hat{L}^*(\mu) &= \max_{r>0}[\mu r - \hat{L}(r)] \end{aligned}$$

and η satisfies $\hat{L}'(\eta) = \frac{c-P}{n}$. The new call is accepted if this failure probability is less than the desired threshold. Since calls arrive at a time-scale much slower than that of the renegotiations of bandwidth requirements of calls and there are a large number of independent calls in the system, there should be sufficient observations to accurately track the renegotiation failure probability as the number of calls in the system varies. Moreover, in a large system, this technique should be robust to the time variations of the statistics of individual calls. However, more theoretical and experimental work is needed to validate this approach.

6 Related Work

The key contributions of our paper are to note that compressed video traffic has significant burstiness in the slow time-scale, and show that renegotiation allows us to extract almost all the SMG available from exploiting this variation. Recently Chong et al [4] and Zhang and Knightly [29] have independently published work that comes to the same conclusions. Zhang and Knightly present a renegotiated VBR service. Chong et al have concentrated on the online prediction problem using artificial neural networks. Our work differs from theirs in some important aspects. First, our work is based on theoretical foundation of large deviation analysis of multiple time-scale sources, which gives us deeper insight into the nature of the multiplexing gain, leading to an analysis of the renegotiation failure probability for ensembles of renegotiating sources. Second, we have obtained the optimal off-line renegotiation algorithm. Finally, we have considered the system aspects of the problem in more detail. Nevertheless, we feel that their work complements ours and reinforces the importance of renegotiation for multiple time-scale sources.

The two core mechanisms for RCBR are renegotiation and rate prediction. In-call renegotiation has been proposed for bursty data traffic by Hui [16], Turner [25], Doshi and Dravida [7], and Boyer and Tranchier [3]. In their work, a traffic source sets up a burst level reservation before sending, or in some cases, during, a burst. However, since data traffic bursts can occur every tens of milliseconds the reservation process has to be fast. This speed is not essential for RCBR, where renegotiations happen once every tens of seconds. In addition, we believe that renegotiation is effective mainly as a mechanism to extract SMG from slow time-scale variations in source traffic. Data traffic exhibits burstiness in the fast time-scale, and thus renegotiation for data traffic is not likely to be economical in practice. Nevertheless, the mechanisms for renegotiation proposed in the literature can be used for RCBR with minor changes.

De Veciana and Walrand have proposed a *periodic averaging of rate* scheme to smooth traffic at the network edge [5]. Like RCBR, the output of their traffic shaper is also a piecewise CBR stream. The basic difference, however, is that they do not model the multiple time-scale nature of the traffic stream and their scheme is not designed to capture the statistical gain from multiplexing many sources with slow time-scale dynamics.

Current proposals in the ATM Forum for dealing with ABR traffic are similar in spirit to RCBR in that a source obtains a stepwise-CBR rate allocation from the network. However, in the ABR framework, there is an assumption that the source has an intrinsically infinite data rate that is modulated by the fair share of the available network capacity. Thus, the data rate from a source is dynamically adapted to the available capacity in the network. This is the opposite of our situation, where the source has an intrinsic data rate that the network tries to accommodate. In other words, in the ABR case the rate information flows from the network to user, but in the RCBR case, the information flows from the user to the network.

The rate prediction problem has been extensively studied from several different perspectives in the past. The problem can be viewed in terms of linear (Kalman) prediction [14]. Other promising methods are described in [15]. Chong et al [4] have proposed a Artificial Neural Network based approach for prediction, and have shown that it compares well with more traditional alternatives.

7 Discussion

We believe that the performance tradeoff space for traffic management looks something like Figure 8. Starting from the right and moving to the left, we have the Synchronous Digital Hierarchy for telephony, Static CBR, Static VBR, Renegotiated CBR, Renegotiated VBR (RVBR), ABR, and finally, datagram service. As we move from right to left,

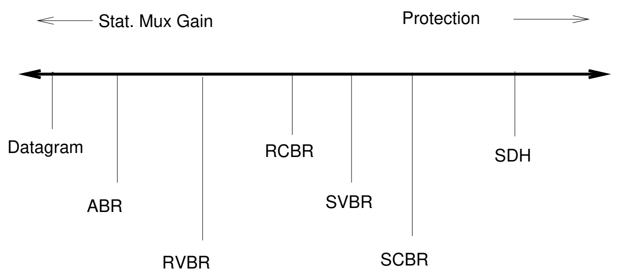

Figure 8: Design space for traffic management.

the statistical multiplexing gain (SMG) achievable increases, but, if the network resources allocated to a stream are kept the same, the protection between streams decreases, that is, one stream can more adversely affect another's performance. For example, as one moves from Static CBR to Static VBR, more SMG is possible, but there is a loss of protection, since, given a fixed amount of buffering, a VBR source could experience packet loss due to a coincident burst from another source. Note that in moving from Static CBR to Static VBR similar protection can be bought, but only at the cost of increased buffering. Similarly, as we move from Static VBR to RCBR, we incur renegotiation overheads, but can potentially exploit slow time-scale variations in the source rate to get increased SMG. RVBR allows more SMG, since both slow and fast time-scale variations are exploited. However, there is more overhead for renegotiation, per-stream regulation, and larger buffers at each switch. The next step along the spectrum is to ABR, where a certain bandwidth is guaranteed at every instant of time, but the network informs the user dynamically as this allocation changes. With ABR service, there is much less protection between streams, since each user's bandwidth depends on the demand of the others. Moreover, considerable effort is needed to share bandwidth fairly. However, even more SMG is possible, since SMG is extracted at the burst level. Finally, with datagram service, the most SMG is available, since call level, burst level and cell level statistical multiplexing is possible. Unfortunately, datagram service also has the least protection - a single burst from a malicious or ill-behaved source can affect all the others.

The point here is that RCBR is not a panacea. It is one choice in a spectrum of possible network services, and is best suited to traffic whose variation is not confined to the fast time-scale. This seems to match at least the subset of the compressed video traffic workload that has been measured in the literature. Other services could also be used to carry compressed video traffic: ABR, Static VBR, RVBR and Static CBR have all been proposed in the past. Ultimately, a network provider and user must choose a service based on their relative costs, efficiencies and afforded qualities of service.

Nevertheless, we feel that RCBR service has some clear benefits. First, it is easy to implement, since we are merely adding a renegotiation component to the well-understood Static CBR service. *Neither complex scheduling disciplines nor large buffers are required in the network switches.* We can keep the network core fast, cheap and dumb, and put intelligence in the edges to extract the SMG from slow time-scale variations.

Second, an RCBR network is always stable. Each admitted call or burst moves the system from a stable configuration to another stable configuration. Thus, the network operator can easily guarantee zero loss and small queueing delays within the network.

RCBR gives us an advantage over unrestricted sharing since a source retains its allocated bandwidth even if a renegotiation to further increase this bandwidth fails. Besides, if a renegotiation fails, this is explicitly known, so that the source can take corrective measures. This makes it easier to integrate RCBR with techniques such as dynamic requantization of stored video, adaptive coding and multilevel scalable coding.

We have already shown that RCBR gets more SMG than a static service. There is another significant advantage. Users of a static service get only one chance to provide the network with a traffic descriptor. If they guess wrong, they either get poor SMG, or suffer from large delays, which might be unacceptable. With RCBR, a source has the option to modify its traffic descriptor as this evolves in time. The danger is that the network might admit too many ill-described users, so that at some future time, the renegotiation failure rate may be too high. This is because there really is no free lunch. If a user is admitted into a network before its traffic is characterized, then there is always the possibility that mistakes will be made by admitting too many users. However, Section 5.4 indicates that we might be able exploit the law of large numbers to make this risk acceptably small.

To conclude, we have shown that a source with slow time-scale variations would suffer performance problems when carried over a static service. Large deviation analysis provides theoretical insight into this problem and motivates the design of RCBR service. We have considered the system aspects of implementing RCBR and have carried out several experiments to measure its performance. The results in Section 5.3 show that RCBR obtains most of the slow time-scale SMG with a fairly small load on the signalling system. Further, it is possible to compute the optimal renegotiation schedule for a real traffic source in a reasonable amount of time. Finally, we have studied the call admission problem and come up with admission control tests based on a large deviation analysis. Thus, our analysis and experiments show that RCBR service is simple, efficient, and well suited for multiple time-scale traffic.

8 Acknowledgments

The authors would like to thank Ken Clarkson and Carsten Lund for many helpful suggestions and discussions on the renegotiation optimization problem. We are indebted to Mark Garrett for providing the traces of MPEG compressed Star Wars.

References

[1] ITU-T Draft Standard Q.2963. *Preliminary Draft, ITU-T*, 1995.

[2] D.D. Botvich and N.G. Duffield. Large Deviations, the Shape of the Loss Curve, and Economies of Scale in Large Multiplexers. *Preprint.*

[3] P.E. Boyer and D.P. Tranchier. A reservation principle with applications to the ATM traffic. *Computer Networks and ISDN Systems*, 24:321–334, 1992.

[4] S. Chong, S.Q. Li, and J. Ghosh. Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM. *IEEE Journal on Selected Areas of Communications*, 13:12–23, January 1995.

[5] G. de Veciana and J. Walrand. Traffic Shaping for ATM Networks: Asymptotic Analysis and Simulations. *submitted to IEEE/ACM Transactions on Networking*, 1992.

[6] A. Dembo and O. Zeitouni. *Large Deviation Techniques and Applications*. Jones and Bartlett Publishers, 1992.

[7] B. Doshi and S. Dravida. Congestion Controls for Bursty Data Traffic in Wide Area High Speed Networks: In-Call Negotiations. *Proc. ITC Specialist Seminar 7, Morristown, NJ*, 1990.

[8] A. Eleftheriadis and D. Anastassiou. Meeting Arbitrary QoS Constraints Using Dynamic Rate Shaping of Coded Digital Video. *Proc. 5th Workshop on Networking and Operating System Support for Digital Audio and Video*, pages 95–106, April 1995.

[9] A. Elwalid and D. Mitra. Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High-Speed Networks. *IEEE/ACM Transactions on Networking*, 1:329–343, June 1993.

[10] A.I. Elwalid, D. Heyman, T.V. Lakshman, D. Mitra, and A. Weiss. Fundamental Bounds and Approximations for ATM Multiplexers with Applications to Video Teleconferencing. *to appear in IEEE JSAC, special issue on Advances in the Fundamentals of Networking*, 1995.

[11] M. W. Garrett. *Contributions Toward Real-Time Services on Packet Switched Networks*. PhD thesis, Columbia University, 1993. Chapter IV.

[12] M. W. Garrett and Walter Willinger. Analysis, Modeling and Generation of Self-Similar VBR Video Traffic. In *ACM Sigcomm '94*, pages 269–280, University College London, London, UK, August 1994.

[13] R.J. Gibbens and P.J. Hunt. Effective Bandwidths for the Multi-type UAS Channel. *Queueing Systems*, 9:17–27, 1991.

[14] G.C. Goodwin and K.S. Sin. *Adaptive Filtering Prediction and Control*. Prentice Hall, 1984.

[15] I. Hsu and J. Walrand. Quick Detection of Changes in Traffic Statistics: Application to Variable Rate Compression. In *Proceedings of the 32nd Allerton Conference on Communications, Control and Computing, Monticello, IL*, 1993.

[16] J.Y. Hui. Resource Allocation for Broadband Networks. *IEEE Journal on Selected Areas in Communications*, 6(9), December 1988.

[17] H. Kanakia, P.P. Mishra, and A. Reibman. An Adaptive Congestion Control Scheme for Real-Time Packet Video Transport. *Proc. ACM SigComm*, 1993.

[18] G. Kesidis, J. Walrand, and C.S. Chang. Effective Bandwidths for Multiclass Markov Fluids and Other ATM Sources. *IEEE/ACM Transactions on Networking*, 1(4):424–428, August 1993.

[19] M. Nomura, T. Fujii, and N. Ohta. Basic Characteristics of Variable Rate Video Coding in ATM Environment. *IEEE Journal on Selected Areas of Communications*, 7(5), June 1989.

[20] E. P. Rathgeb. Modeling and Performance Comparison of Policing Mechanisms for ATM Network. *IEEE Journal on Selected Areas in Communications*, 9(3):325–334, April 1991.

[21] E. P. Rathgeb. Policing of Realistic VBR Video Traffic in an ATM Network. *International Journal of Digital and Analog Communications Systems*, 6:213–226, 1993.

[22] R. Safranek, C. Kalmanek, and R. Garg. Methods for Matching Compressed Video to ATM Networks. *Proc. of IEEE IT Workshop on Information Theory, Multiple Access and Queueing Theory, St. Louis*, page 6, April 1995.

[23] P. Sen, B. Maglaris, N. Rikli, and D. Anastassiou. Models for Packet Switching of Variable-Bit-Rate Video Sources. *IEEE Journal on Selected Areas of Communications*, 7(5), June 1989.

[24] D. Tse, R. Gallager, and J. Tsitsiklis. Statistical Multiplexing of Multiple Time-Scale Markov Streams. *to appear in IEEE JSAC, special issue on Advances in the Fundamentals of Networking*, 1995.

[25] J.S. Turner. Managing Bandwidth in ATM Networks with Bursty Traffic. *IEEE Network Magazine*, September 1992.

[26] A.J. Viterbi and J.K. Omura. *Principles of Digital Communication and Coding*. McGraw-Hill, 1979.

[27] A. Weiss. A New Technique for Analyzing Large Traffic Systems. *Advances in Applied Probability*, 18:506–532, 1986.

[28] L.C. Wolf, L. Delgrossi, R. Steinmetz, S. Schaller, and H. Wuttig. Issues in Reserving Resources in Advance. *Proc. 5th Workshop on Networking and Operating System Support for Digital Audio and Video*, pages 27–37, April 1995.

[29] H. Zhang and E.W. Knightly. A New Approach to Support Delay-Sensitive VBR Video in Packet-Switched Networks. *Proc. 5th Workshop on Networking and Operating System Support for Digit al Audio and Video*, pages 275–286, April 1995.