ABR Service for Applications with Non-linear Bandwidth Utility Functions*

Zhiruo Cao Ellen W. Zegura

College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{zhiruo,ewz}@cc.gatech.edu

Abstract

An available bit rate (ABR) service allows applications to access a time-varying network capacity. In a basic ABR service, the available network capacity is divided "fairly" amongst active connections, without regard to the utility that each application derives from the bandwidth allocation. The goal of this paper is to improve both the global and individual utility obtained by applications using ABR, as compared to standard max-min bandwidth allocation. The key to our novel ABR algorithm is to relax the requirement for short-term fair bandwidth allocation, allowing an allocation that trades bandwidth between applications to increase the overall utility. To increase the utility of each individual application, we also re-allocate the bandwidth on the usual ABR time interval. This re-allocation increases the long-term average utility obtained by each individual application; it also ensures that the longer term average bandwidth allocated to each application is equal to the max-min allocation.

1 Introduction

An available bit rate (ABR) service allows applications to access a time-varying network capacity. Realization of such a service requires a control loop to allow information about available capacity to reach applications; further, applications must adjust their rate based on feedback information from the network. As an enhanced best-effort service, the advantages of ABR include efficient use of network resources, low loss rate, minimal *a priori* specification of application behavior, and a paradigm somewhat similar to the congestion control mechanisms in TCP over best-effort IP. Given these advantages, it seems likely that an ABR service will be important in both ATM and IP-based networks [7].

Consider a basic ABR service in which available capacity is divided "fairly" amongst active connections. The fair share of bandwidth is periodically conveyed to applications that adjust transmission rate accordingly. This specification of ABR makes certain assumptions about the applications using the service. Most obvious is the assumption that the applications have flexible bandwidth requirements; that is, they can adjust bandwidth usage in response to feedback, such that their usage does not exceed the allowed rate. More subtle is the assumption that the *utility* [9] or *benefit* to each application is a linear function of the allowed rate. The fair share mechanisms attempt to maximize the bandwidth allocated to each application; the implicit assumption is that applications will benefit in proportion to the allocated bandwidth.

The assumption of linear bandwidth utility is quite reasonable for traditional applications such as FTP, where doubling the available bandwidth corresponds to roughly halving the transfer time. More recently, however, best-effort service is being used for applications that do not fit the traditional model, for example real-time audio and video [3, 8], and interactive services; this suggests that an ABR service may be used by a wide range of applications.

Some of these applications will not have linear bandwidth utility functions, and thus may be inefficiently served by a standard ABR service. The non-linearity in the utility function can come from two sources: (1) the application may not be able to adapt to an arbitrary available bandwidth or (2) the benefit to the application may not be a linear function of the bandwidth. As an example of the first source, consider an application that has stored images at different quantization levels. It can adapt bandwidth usage, but only at the (discrete) rates that correspond to the set of stored images. The second source of non-linearity might arise, for ex-

*This work supported by Hitachi Telecom, USA.

ample, if there is a human in the loop. The human perception system has limited ability to detect changes in quality, thus the utility function may exhibit discrete changes corresponding to perception thresholds.

The goal of this paper is improve both the *global* and *individual* utility obtained by applications using ABR, as compared to standard max-min bandwidth allocation. Meeting this goal in a general network topology with arbitrary utility functions is ambitious. In this particular paper, we focus on the logical first step: a network with a single bottleneck link and utility functions that are stationary and memoryless. Some network environments and applications will meet these assumptions; further, the solution can be used as the basis for tackling the more complex and general problem.

The key to our Utility-Driven Allocation (UDA) algorithm is to relax the requirement for short-term fair bandwidth allocation, allowing an allocation that trades bandwidth between applications to increase the overall utility. This basic allocation is insufficient, however, to increase the individual utility. Thus, we also re-allocate the bandwidth on the usual ABR time interval. This re-allocation increases the long-term average utility obtained by each individual application (as compared to max-min allocation); it also ensures that the longer term average bandwidth allocated to each application is equal to the max-min allocation.

In the next section, we develop the necessary background in the basic ABR algorithms, standard fairness criteria and utility functions. We also motivate our scheme by considering the performance of standard ABR for a non-linear utility function. In Section 3 we present the UDA algorithm and explain how it addresses the limitations in standard ABR. Section 4 contains simulation-based results to quantify the improvement possible with UDA. In Section 5 we develop a new fairness definition and show how to achieve this fairness using bandwidth re-allocation. Finally, we conclude in Section 6.

2 Background and Motivation

2.1 Rate-based feedback

In the current Internet, network layer congestion control relies in large part of the end-to-end feedback mechanisms of TCP. Congestion is inferred due to timeouts, and the sending window is decreased in response. In a high speed network, this end-to-end feedback flow control mechanism may not be fast and effective enough for applications to promptly respond to changes in the network. Proper explicit feedback from the network about traffic load changes seems necessary to reduce congestion and improve application performance [7].

As defined by ATM Forum, Available Bit Rate (ABR) service is an ATM layer service category as well as a flow control mechanism using several types of feedback to control a source's data transmission in response to the changing network state [4]. A rate-based feedback flow control has been adopted and standardized. In the ATM Forum ABR scheme, the feedback is conveyed to the source through specific control cells called Resource Management (RM) cells. RM cells can be generated by the source, traverse the network and turned back to the source by the destination. RM cells can also be generated by switches in the network during severe congestion and sent to the sources. When a source receives RM cells, it must adjust its transmission as indicated by values carried in RM cells. Through this RM cell feedback mechanism, congestion can be limited while maintaining a high network utilization. Two types of rate-based flow control approaches have been proposed, namely *binary feedback* and *explicit rate marking*. Our work in this paper focuses on an explicit rate ABR service: the RM cells convey an explicit rate to the source.

2.2 Fairness

One of the major advantages of ABR service over a pure best-effort service (e.g., IP or Unspecified Bit Rate) is that fairness policies can be implemented in resource sharing. A number of policies have been proposed for fairness of a flow control algorithm. For example, one might define fairness weakly to mean that all users get a nonzero share of the resources [1]. Alternatively, one might give preference to traffic that has traveled more hops [6]. A commonly used fairness criterion in a rate-based flow control system is max-min fairness, introduced in [5]. This fairness definition has been specified by the ATM Forum as a major goal of an ABR flow control algorithm.

Mathematically, max-min allocation can be defined as follows. A feasible bandwidth allocation vector $B = (b_0, b_1, \ldots, b_{n-1})$ assigns bandwidth b_i to source i such that no links in the network are congested:

$$\sum_{sources\ k\ using\ link\ j} b_k \leq C_j,$$

where C_j is the capacity of link j. The max-min allocation is the feasible bandwidth allocation vector that is the "largest", in the following sense. First, assume that the components in each feasible bandwidth allocation vector B are sorted so that $b_i \leq b_{i+1}$ for $i = 0 \ldots n-2$. We define an ordering over the feasible bandwidth allocation vectors where $A > B$ if and only if there exists some j such that $a_i = b_i$ for $0 \leq i < j$ and $a_j > b_j$. The max-min allocation is a largest feasible bandwidth allocation vector, under the ordering defined.

The max-min fairness criterion attempts to equally allocate available bandwidth amongst all connections bottle-necked at a link. This principle is fair since all connections that share a link obtain an equal share of link bandwidth, provided they can all use that fair share.

2.3 Utility functions

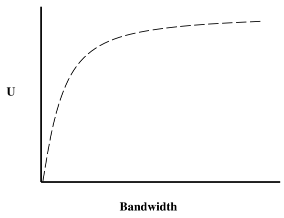
The new ABR scheme we are going to propose is motivated by Shenker's observations on the utility function of applications [9]. Shenker formalizes the goal of network design as being to maximize the sum of the utility of user applications, defined as $V \equiv \sum_i U_i(s_i)$. In this formalism, the utility function U_i maps the service delivered by the network into the performance of application i. The quantity s_i contains all measures of service delivered to application i, e.g., bandwidth, delay, delay jitter and loss ratio.

Shenker observes that traditional data applications such as electronic mail, remote terminal access, and file transfer are rather tolerant of end-to-end delays and throughput. This class of applications are called *elastic* applications. A convex bandwidth utility function (Figure 1(a)) can be used to describe the utility as a function of available bandwidth to this class of applications.

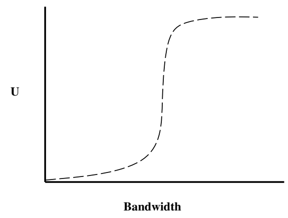
Real-time applications are generally delay sensitive and may have hard real-time requirements. For this class of applications degradation in bandwidth may severely degrade the performance. A reasonable model for this class of applications is a mostly concave utility-bandwidth function (See Figure 1(b)). Rate-adaptive real-time applications are especially worth studying for use with ABR service. Applications in this class adjust their transmission rate in response to network congestion. At high and low bandwidths the marginal utility of additional bandwidth is very small. The bandwidth utility curves have the general shape shown in Figure 1(c).

The nonlinearity in application utility also come from the fact that an application may not be able to adjust to an arbitrary bandwidth offered by the network. Thus, the application utility functions may not continuously increase or decrease as the available bandwidth changes. Such an application may have a utility that can be characterized by a stepwise function, e.g., as shown in Figure 1(d). As an example, a connection carrying a video stream stored at several different quantization levels may be considered to have a stepwise utility function, because data is transmitted at discrete rates for each different quantization level.

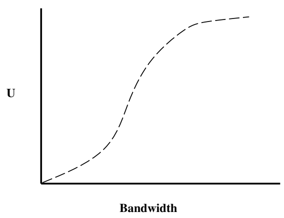
To illustrate that the max-min fairness allocation may be poorly suited for non-linear bandwidth utility functions, consider two applications sharing a link



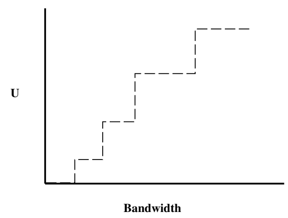
(a) Elastic Utility



(b) Real-time Utility



(c) Rate-adaptive Real-time Utility



(d) Stepwise Utility

Figure 1: Utility as a function of bandwidth for different application classes

with capacity C. Assume the applications have the same (concave) utility function, given by b^2, where b is the bandwidth allocated to the application. A max-min allocation scheme will allocate bandwidth of $C/2$ to each application, for individual utility of $C^2/4$ and total utility $C^2/2$. This allocation will remain the same over time.

Alternatively, the bandwidth could be allocated unfairly, so that one application receives bandwidth C (utility C^2), while the other receives bandwidth 0 (utility 0). The total utility is then C^2, which improves upon the $C^2/2$ value above. To improve the individual utility and achieve longer term fairness, we can *reallocate* the bandwidth during the next allocation time interval, swapping the allocation between the two connections. The average utility seen by each application is then $C^2/2$, while the average bandwidth allocation is $C/2$. Thus, we have improved the total and individual utility, while maintaining fair bandwidth allocation over the longer term.

From the simple example shown above, we have the following important observations. First, standard ABR algorithms do not, in general, maximize the overall utility of user applications. Second, considerable gains in overall utility are possible with non-equal bandwidth allocation, but fairness issues have to be addressed. Third, bandwidth re-allocation can improve the performance seen by individual applications.

Next, we propose an alternative flow control scheme

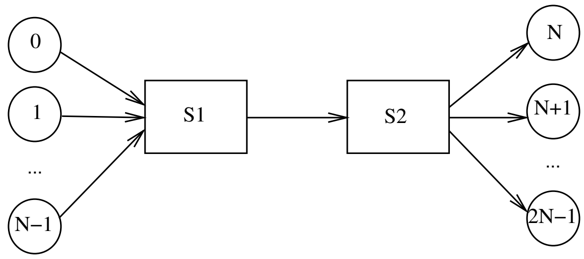


Figure 2: A simple ATM network model

for ABR service that allocates bandwidth to connections with nonlinear bandwidth-utility functions, and achieves the optimal total utility. The utility optimization/bandwidth allocation algorithm takes the available bandwidth C, the utility functions f_i for each connection i and the number of connections N as input, and computes the bandwidth allocation vector B. We address the problem of reallocating a assigned bandwidth in Section 5.

3 Utility-Driven Allocation

3.1 Single class utility optimization

Consider a single bottle-neck network configuration as shown in Figure 2. There are N ABR connections $0, 1, 2, \ldots, N-1$ going through Switch S1 from end stations in the left hand side of the network, and these connections terminate at end stations $N, N+1, N+2, \ldots, 2N-1$. These N connections are bottle-necked at Switch S1, and sharing the available bandwidth C. For simplicity, we assume that all connections have the same utility function; we will address the situation connections have different utility functions later in this paper. We also assume that the utility of a connection in this environment is a function of the bandwidth the connection is utilizing at the current time, expressed as $U = f(b)$ where U is the utility and b is the bandwidth. In other words, we assume the utility function is stationary and memoryless.

Consider an arbitrary bandwidth allocation algorithm which allocates bandwidth b_i to connection i. To be feasible, the allocation must satisfy $\sum_i b_i \leq C$. The overall utility for the allocation is given by $U = \sum_i f(b_i)$. Given the number of connections N, the total available bandwidth C at the bottleneck, and the utility-bandwidth function f, the total utility U can be maximized by finding a set of b_i.

When $U = f(b)$ is a step function, there are a fixed number of possible values for the b_i's. In this case, the maximization problem can be re-written as a variation of the Knapsack problem and solved using Integer Linear Programming (ILP) [2]. We must transform the problem formulation to fit the integer programming model. Specifically, let s_j denote the bandwidth at step j of the utility function, $0 \leq j < K$, where K is the number of steps. Clearly, each $b_i \in \{s_0, \ldots, s_{K-1}\}$. Assume that a total of n_j out of N connections are each allocated a bandwidth s_j. The overall utility U is computed as:

$$U = f(s_0) \times n_0 + f(s_1) \times n_1 + \ldots + f(s_{K-1}) \times n_{K-1}$$

where n_j's are constrained by the following two equations:

$$n_0 + n_1 + \ldots + n_{K-1} = N$$

$$s_0 \times n_0 + s_1 \times n_1 + \ldots + s_{K-1} \times n_{K-1} \leq C$$

By finding a set of n_j's that maximizes U, a set of optimal b_i's can be determined.

A typical ILP problem is NP-Complete, thus the running time is potentially a concern. Two factors mitigate this. First, when K, the number of steps in the utility function, is relatively small, the problem can be solved quite efficiently [2]. Second, the bandwidth allocation will be computed as part of the connection setup process, and thus is not on the more time-critical data path. Re-allocation will be done during transmission, however this can be accomplished using more efficient algorithms.

3.2 Extensions

The utility optimization algorithm above assumes a single utility function. The algorithm can easily be extended to a set of connections with different utility functions. Assume at a bottle-neck switch, there are N connections. Let f_i denote the utility function for connection i; let b_i denote the bandwidth allocated to connection i. To be feasible, the allocation must satisfy $\sum_i b_i \leq C$. The overall utility for the allocation is given by $U = \sum_i f_i(b_i)$.

This is a somewhat similar problem as we presented earlier in this section; with some modifications, the same techniques can be applied to optimize U, subject to the constraint of longer term fair bandwidth allocation. The basic idea is to group connections with the same utility function into a class. Next, each connection class d is allocated bandwidth $N_d \frac{C}{N}$, where N_d is the number of class d connections and N is the total number of ABR connections. Within a class, the single-class utility optimization described above can be applied. This approach will maintain longer term fairness, while improving individual and overall utility.

In a network with arbitrary topology, a more general utility-optimization problem can be defined as follows. For a feasible allocation vector $B = (b_0, b_1, \ldots, b_{n-1})$, the total utility for the allocation is $\sum f_i(b_i)$, where

f_i denotes the utility function for source i. Using integer linear programming, a centralized algorithm can be easily found to compute an allocation that maximizes the utility, but it appears difficult to find a distributed version. Our analysis in this paper will focus on the utility optimization in a single bottle-neck network.

4 Performance Analysis

Before considering the issue of reallocating the bandwidth, we examine the gains in total utility that are possible using the algorithms from the previous section [1]. We provide performance analysis comparing the total utility for the proposed Utility-Driven Allocation (UDA) and the max-min allocation for a variety of utility functions operating over the single bottle-neck link configure of Figure 2. As a preview, the goal of the reallocation schemes will be to achieve individual utility equal (over the longer term) to $1/N$ of the total utility, where N is the number of connections.

We first consider a utility function whose envelope is linear, but with discrete steps. In this case, the total utility expected from the max-min approach can match the utility-optimization approach relatively closely over some regions. Assuming we have a utility function as shown in Figure 3(a), we measure the total utility from the two schemes for a set of connections passing through the bottle-neck. The result is depicted in Figure 3(b), where we show the total utility as a function of the number of bottle-necked connections. The total available bandwidth at the bottleneck is set at 9Mb/s. Note that in max-min allocation, when the number of ABR connections increases, the average bandwidth allocated to each connection decreases. When this average allocation crosses the edge of a step in the bandwidth-utility curve, the total utility drops sharply, and then increases again. When the average bandwidth falls below 0.6, each individual connection has utility of zero. In contrast the utility driven bandwidth allocation scheme can maintain the total utility at a high level using unequal bandwidth allocation.

In Figure 4(a), we show a stepwise bandwidth utility curve for an elastic application, which may model traditional applications such as FTP. In Figure 4(b), we show the total utility using the nonlinear optimization algorithm when the number of ABR connections increase from one to 50, in comparison with the utility obtained under the standard max-min bandwidth allocation scheme. The total available bandwidth at the bottleneck is set at 2Mb/s. Notice that the max-min scheme has comparable total utility over most of the range. This is because for a convex utility function, UDA and max-min schemes tend to allocate the same bandwidth to connections. With fixed total bandwidth and number of connections, trying to allocate bandwidth away from the average will cause the utility for one connection to increase only slightly while causing the utility for the other to decrease substantially. As a result, the total utility decreases.

In Figure 5, we show the utility gains for real-time rate-adaptive applications, which may have concave utility functions. A stepwise utility function for this class of applications is shown in Figure 5(a), whose corresponding continuous curve is shown in Figure 1(c). In Figure 5(b), we show the total utility using UDA algorithm when number of ABR connections increase from one to 50 with bottleneck bandwidth of 9Mb/s, in comparison with the utility achieved under standard max-min bandwidth allocation scheme. Notice that the max-min scheme performs poorly in comparison to the non-linear scheme over most of the range. This is because for a concave utility function, allocating bandwidth higher than the average to one connection while correspondingly allocating lower to another will increase the total utility. Therefore, the max-min bandwidth allocation yields the total utility much less than the optimal.

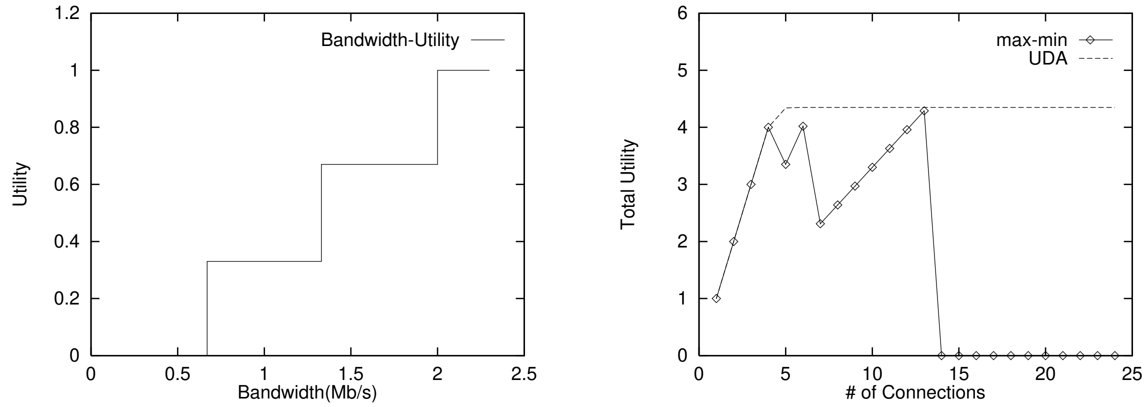
We observe that the shape of the envelope for the step function is more significant than the number and spacing of steps. For example, Figure 6 has the same envelope and similar performance to Figure 5, though with more steps. Generally, for connections with convex utility functions, max-min allocation performs well, but for connections with concave utility functions, max-min allocation yields low total utility.

The performance of the proposed utility driven bandwidth allocation (UDA) can also be evaluated under varying total available bandwidth at the bottleneck. A comparison of overall utility for 10 connections under the two bandwidth allocation approaches is shown in Figure 7. The utility function for the connections is that shown in Figure 5(a). Notice that the standard algorithm is only competitive as the total available bandwidth becomes larger.

It can be observed from our analysis that the total utility obtained using the max-min bandwidth allocation algorithm can be extremely low as the number of ABR connections at the bottleneck increases or as the total available bandwidth decreases. Further, our modified algorithm is able to maintain high total utility over the full range of connections and available bandwidth at the bottleneck.

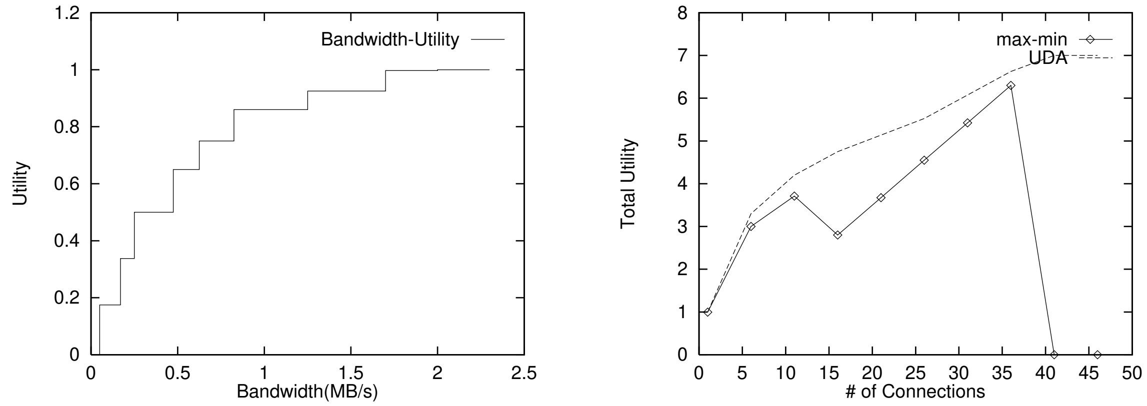
Essentially, the utility gain comes from the unequal bandwidth allocation to connections with nonlinear bandwidth utility function. Some connections may be

[1] The Linear Programming package *lp_solve* we use for the analysis is written by Michel Berkelaar, and is available at ftp://ftp.es.ele.tue.nl/pub/lp_solve.



(a) Utility function (b) Total utility

Figure 3: Performance with linear stepwise utility function



(a) Utility function (b) Total utility

Figure 4: Performance with elastic utility function

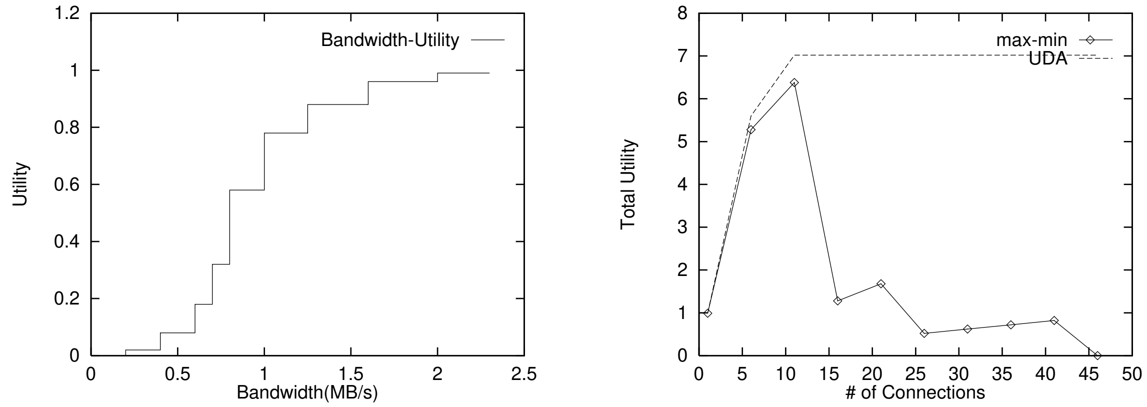
allocated less bandwidth than others in our algorithm. But it is important to note that allocating low bandwidth to a connection in our algorithm is different from blocking a connection in admission control. First, it is presumed that applications using ABR services can tolerate a bandwidth as low as its MCR. Second, through bandwidth re-allocation, which we will discuss in the next section, all connections will be treated fairly over a longer term.

Finally, we consider one example of multi-class optimization. Because the total bandwidth can be always partitioned between classes and utility optimization can then be done within classes, we can expect similar utility improvement by this multi-class optimization over the max-min allocation as we see in the single class utility analysis. Figure 8 shows the total utility for two classes of connections. In the experiment, 20 connections have an utility function shown in Figure 4(a), and other 20 connections have an utility function as shown in Figure 5(a).

5 A New Fairness Definition

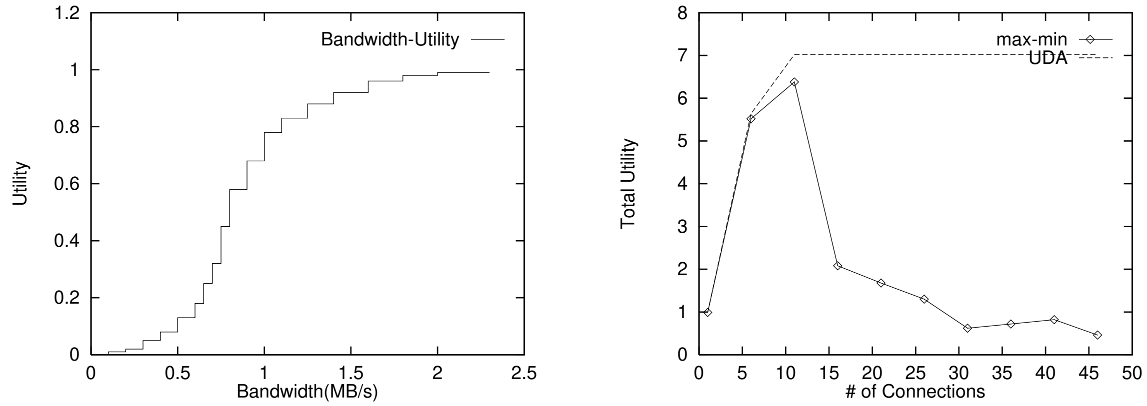
The algorithm we described in the previous section can achieve optimal utility, but it inevitably allocates more bandwidth to some connections than to others, and this fails to meet the fairness requirement. An ideal solution would be to achieve the maximum utility while still allowing connections at a bottleneck to be treated fairly. We have designed ways to realize this by re-allocating the bandwidth at regular time intervals. By re-allocating bandwidth at certain times, connections are ensured to receive fair treatment over a longer term, even though they may receive unfair treatment over the short term. Further, the long term overall performance of each application can be improved under the UDA scheme.

We argue that fairness requires a notion of time. If the bandwidth allocated to applications over a given time period is equal, the allocation is fair with respect to that time period. We further argue that applications may often be insensitive to fairness on an extremely



(a) Utility function (b) Total utility

Figure 5: Performance with rate-adaptive utility function



(a) Utility function (b) Total utility

Figure 6: Performance with fine-grained step utility function

short time scale, and rather will be satisfied with fairness over the longer term. This observation gives us the flexibility to allocate the resources in a more efficient way to achieve greater application utility.

Two assumptions have to be made in order to make the above arguments valid. We assume that the utility function is stationary (i.e., independent of time) and memoryless (i.e., independent of past allocations). In addition, we assume that fluctuations in available bandwidth do not adversely affect the application. It can be argued that an application using ABR service will experience fluctuation in any case, due to variations in available load. The ABR service does not make any guarantees except that the minimum rate will be supported. Further, the aggregated rate of connections at the bottleneck is still well controlled, so the fluctuation of bandwidth allocated to an individual connection should not affect the stability of the network.

A variant of max-min fairness, *T-fairness*, can be defined as follows: At bottleneck switch S, during any time period T, ABR connections are treated fairly if the *accumulated* bandwidth allocated to each of them is equal. Thus the definition contains time T as a parameter, allowing a range of fairness definitions.

From the discussion above, maximizing overall utility can be solved using Integer Linear Programming, and the result is a bandwidth allocation vector of $(b_1, b_2, \ldots, b_N)$, where b_i is the bandwidth allocated to connection i. Note that the difference between bandwidth allocated to different connections can be very large. Currently we are considering three ways to enforce approximate T-fairness in the bandwidth allocation scheme.

5.1 Rotating allocation

The simplest way to enforce fair allocation over time period $T = N \times t$ is to allocate $b_{(i+j)\,mod\,N}$ at time $j \times t$ to connection i. If the bandwidth is allocated in such a way, after time period $N \times t$, every connection is allocated $b_1 + b_2 + \ldots + b_N$. We can think of

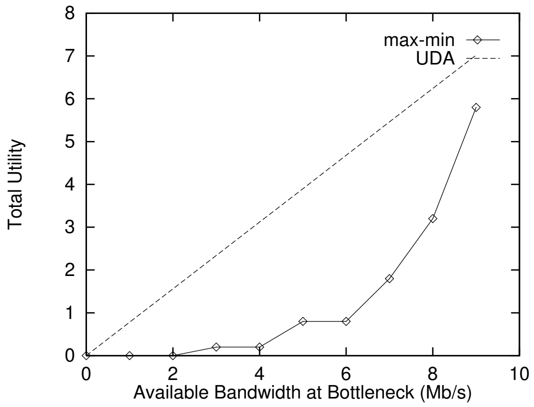


Figure 7: Total utility with varying available bandwidth

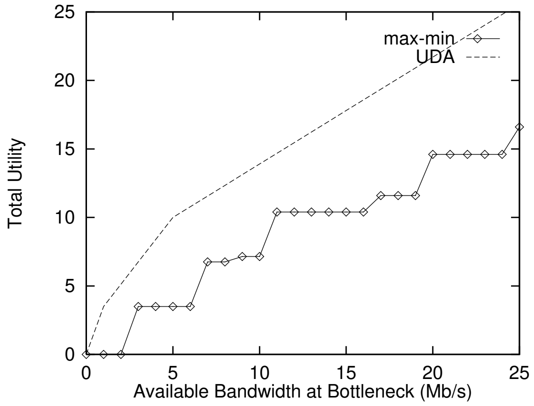


Figure 8: Total utility of two classes of connections

this method as rotating the elements of the allocation vector among connections. After a complete cycle, all connection receive the sum of the elements of the vector $(b_1, b_2, \ldots, b_N)$.

5.2 Minimal rotating allocation

The above algorithm guarantees that after $N \times t$, the bandwidth allocated to applications is equal. Note, however, that it may not be necessary to use N rounds to achieve equal allocation. A modification can be made to find a number $m \leq N$ that, after $m \times t$, connections receive equal bandwidth. Mathematically, m can be minimized by solving an Integer Programming problem which is described as follows:

$$\sum_{k=0}^{N-1} x_{lN+k,j} = 1$$

for $l = 0, 1, \ldots, m-1$, and $j = 0, 1, \ldots, N-1$,

$$\sum_{l=0}^{m-1} \sum_{j=0}^{N-1} b_j x_{lN+k,j} = \frac{m}{N} \sum_{j=0}^{N-1} b_j,$$

for $k = 0, 1, \ldots, N-1$.

x_{ij} is a 0/1 variable with value 1 indicating that bandwidth a_j is allocated to connection $i \, mod \, N$ at time step i/N. If there is a 0/1 solution to x_{ij} in the above equations to a minimum m, we can schedule the allocation in m cycles, until all connections receive the same amount of bandwidth.

The new allocation will be fair, but it has limitations. Even if we can solve the scheduling problem so that $m \leq N$, the worst case is still N. Thus the scheme is not guaranteed to scale well when the number of connections grows very large.

5.3 Statistical allocation

Both methods described above require a reasonable amount of local processing in calculating the fair share for each connection at the bottleneck. In order achieve fairness, previous allocations have to be recorded, and used for later allocation. Keeping this state between steps of allocations makes the schemes somewhat complicated and time consuming. An alternative *statistical allocation* approach may be considered. In the statistical fairness allocation approach, the bandwidths in the allocation vector are randomly assigned to connections. Using probabilistic analysis, we can determine the probability that a given connection has been treated fairly over time. We consider this to be a promising approach since the implementation is more straightforward. However, the assurance of fairness will be statistical, not deterministic.

For the statistical fairness allocation, we quantitatively define the *unfairness* UF:

$$UF = \frac{\sqrt{\frac{\sum_i (U_i - \overline{U})^2}{N}}}{\overline{U}}$$

where U_i is the accumulated utility of connection i, and $\overline{U}$ is the average accumulated utility of connections. The accumulated utility U_i over time t is defined as $U_i = \int_0^t u_i \, dt$, where u_i is the instant utility for connection i at time t. In a single bottle-neck network with eight connections passing through the bottle-neck, each connection has an utility function as shown in Figure 5(a). When the available bandwidth for ABR connections at the bottleneck is 4.5Mb/s, an allocation which optimizes the total utility to the eight connections is $(0, 0, 0, 0.8, 0.8, 1.0, 1.0, 1.0)$Mb/s. Figure 9 shows that with random bandwidth allocation to achieve fairness, the unfairness in the utility received by connections drops significantly after five cycles of bandwidth reallocations at the switch. Note that the N-fairness approach would achieve perfect fairness after eight cycles.

To illustrate the effect on an individual connection, we plot the accumulated utility for one connection over

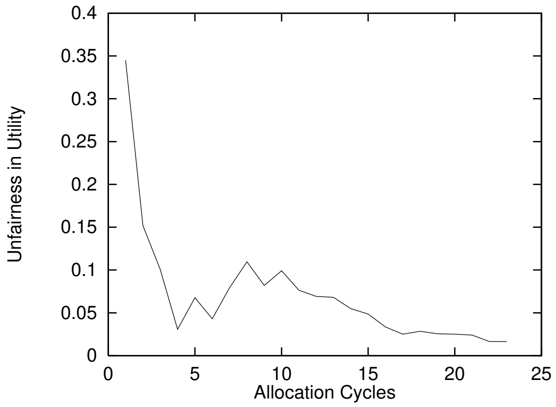


Figure 9: Statistical allocation

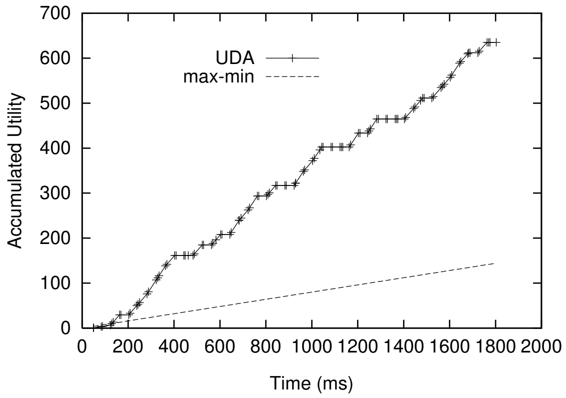


Figure 10: Accumulated individual utility over time

time, in Figure 10. As expected, the utility is greatly improved over the max-min allocation.

Some work remains to be done in this area. For example, efficient algorithms have to be designed to make sure the bandwidth is not over allocated by the statistical approach. Further, schemes combining N-fairness and statistical fairness may be desirable in that N-fairness can be used when N (the number of ABR connections) is small while statistical fairness is more suitable when N is large.

6 Conclusions

In this paper, we have proposed a new ABR algorithm (UDA) that improves both the global and individual utility obtained by applications, while maintaining longer term fair allocation of bandwidth. To develop the algorithm we introduce a stepwise utility function to model the bandwidth-utility relationship. For stepwise utility functions, we have shown that a switch can maximize the utility using a straightforward method. We have investigated the improvement possible in the UDA algorithm, as compared to max-min allocation. We have found significant improvement under a variety of utility functions, number of connections and available bandwidth.

Improving the individual utility and maintaining longer term fair allocation is achieved by reallocating the bandwidth on standard ABR time intervals. To address the fairness issue, we propose fairness definition that explicitly includes the notion of time. Further, we have designed methods to achieve longer-term fairness under the new definition. Our analysis and simulation show that this new fairness definition can be achieved in the nonlinear ABR algorithm.

References

[1] K. Bharat-Kumar and J. M. Jaffe. A New Approach to Performance-Oriented Flow Control. *IEEE Trans. on Commun.*, 1981.

[2] Vašek Chvátal. *Linear Programming*. W. H. Freeman and Company, NY, 1983.

[3] S. Floyd, V. Jacobson, S. McCanne, et al. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *Sigcomm 95*, 25(5):342–356, August 1995.

[4] ATM Forum. ATM Forum Traffic Management Specification 4.0. 1996.

[5] J. M. Jaffe. Bottleneck Flow Control. *IEEE Transactions on Communications*, 29:954–962, July 1981.

[6] Raj Jain. Congestion Control in Computer networks: Issues and Trends. *IEEE Network Magazine*, 1990.

[7] C. Lefelhocz, B. Lyles, S. Shenker, and L. Zhang. Congestion Control for Best-Effort Service: Why We Need a New Paradigm. *IEEE Networks*, pages 10–19, January 1996.

[8] M. Macedonia and D. Brutzman. MBone Provides Audio and Video Across the Internet. *IEEE Computer*, 27(4):30–36, April 1994.

[9] Scott Shenker. Fundamental Design Issues for the Future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, September 1995.