Receiver-driven Bandwidth Adaptation for Light-weight Sessions

Elan Amir, Steven McCanne, and Randy Katz
University of California, Berkeley
{elan,mccanne,randy}@EECS.Berkeley.EDU

Abstract

Current Internet multicast conferencing tools treat all sources with equal importance in that they either statically allocate a fixed bandwidth to each source in a session, or they automatically adapt each source's transmission rate independently of all other sources. But not all sources are of equal interest to all receivers. We believe that to effectively support human to human communication, this disparity in receiver interest should be reflected in the rate-adaptation process. To this end, we propose a protocol called "SCUBA" that enables media sources to intelligently account for receiver interest in their rate-adjustment algorithms. SCUBA is orthogonal to and complements existing rate-adaptation schemes and can interoperate with either sender- or receiver-directed control systems. To scale the SCUBA protocol with multicast session size, we decouple the receiver-feedback process from the session size through sampling. This approach introduces a "tunable" tradeoff between convergence time and sampling accuracy that for large sessions is solely dependent on the control traffic bandwidth. In addition to its applicability in video conferencing, our control scheme can be combined with media transcoders to intelligently manage a bottleneck link at a well-known and fixed location in the network. We implemented SCUBA within our video conferencing tool *vic* and our media gateway *rtpgw* and feedback from their preliminary deployment indicates that the efficacy of the overall multimedia communication system has been greatly enhanced.

1 Introduction

Over the last several years, IP Multicast has been incrementally deployed in the Internet by building a virtual network, embedded in the unicast Internet, called the Multicast Backbone or "MBone". In parallel with the deployment of this new multicast network, the MBone research community developed a number of applications for multimedia conferencing

[23, 8, 19, 16, 13, 12] that each exploit the simple, efficient, and elegant IP multicast service model and its underlying routing mechanism. These tools, collectively known as the "MBone tools," encapsulate real-time, digital media streams into packets using the Real-time Transport Protocol (RTP) and multicast these packets on the network by simply sending them to a multicast group address. Receivers interested in a particular transmission simply "tune in" by subscribing to the multicast group in question. This loosely coupled, light-weight, real-time multimedia communication model is known as the *Light-weight Sessions* architecture [11].

Although the MBone tools can be used for a variety of communication styles (e.g., point-to-point internet phone calls, small group design collaborations or research meetings, or seminar distribution to large groups with tightly coupled feedback from the remote audience), a non-interactive, passive style of communication has emerged as the dominant model for MBone sessions. Here a single source "broadcasts" a signal to the MBone in a one-to-many distribution where interaction or feedback from the session participants rarely, if ever, occurs. For example, the regular, ongoing MBone broadcasts of university seminars, conference talks, and the NASA shuttle missions almost never entail multimedia flows from any but the primary source. While this style of transmission is adequate for much of the MBone content, it does not meet the requirements of close-knit collaboration, where the focus of interest may shift rapidly from site to site as the discussion moves about the distributed conference.

We believe this phenomenon — the rarity of large-scale interactive MBone sessions — is due not to the inherent nature of the MBone and the audio/video conferencing tools, but instead results from the lack of adequate tools and control protocols to make tightly knit collaborations effective across the wide area. In this paper, we focus on one particularly critical barrier to richer collaborations: because session bandwidth is currently allocated in a fixed and inflexible fashion, and because users must manually configure, enable, and disable transmission of their video signals, collaborations using the MBone tools tend to be comprised of a small number of video streams running at a constant transmission rate (typically the full amount of advertised session bandwidth). This is exactly the wrong model for effective human-to-human communication — people interact by shifting their focus from individual to individual not by simultaneously listening to and looking at a large number

of other individuals. We believe that applications should reflect this model by automatically and dynamically shifting the allocation of bandwidth among session participants as the focus in a distributed conference moves about. This dynamic model not only supports effective collaboration, but it also saves network resources since every source need not generate and transmit a continuous video stream.

In essence, we propose that media sources adapt their transmission rates to meet the collective preferences of the receivers in a multicast session. But this style of adaptation is only one piece of the rate adaptation problem; another important goal of adaptation is to accommodate and avoid network congestion by controlling the aggregate rate of traffic injected into the network across all senders¹. While several solutions for rate-adaptive audio/video have been proposed [3, 4, 17, 5, 24], none explicitly account for receiver interest in the adaptation process. In fact, existing algorithms adjust the rate of each media source independently and model the impact of competing flows on a given flow's control algorithm simply as background measurement noise.

We claim that these existing adaptation schemes can be greatly improved by augmenting them with receiver feedback to constrain the media adaptation process. To this end, we decompose the media rate control process as two complementary and orthogonal mechanisms:

- (a) the core adaptation process that adjusts the media delivery rate at each source, and
- (b) the receiver feedback protocol that disseminates receiver interest throughout the session to constrain the adaptation process.

A number of existing schemes address part (a) and each approach can be further classified as either sender- or receiver-based. Sender-based schemes like those in *ivs* [3] and *NeVit* [4] adjust the rate of a source using explicit receiver feedback, while receiver-based schemes like Receiver-driven Layered Multicast (RLM) [17], Destination Set Grouping [5], and Thin Streams [24], tune the delivery rate at each receiver individually. In this latter case, each receiver tunes its inbound reception rate by adjusting its "level of subscription" to a set of hierarchical signal layers that are striped across multiple multicast groups.

Although the research community has proposed a wide variety of rate-adaptation mechanisms, to our knowledge, no scheme for reflecting receiver interest has yet been proposed, and thus only part (a) of our two part problem has been solved. To address part (b), we have developed a scalable, light-weight, and tunable feedback protocol that reflects receiver interest back to the media sources. We deem the combination of rate-adaptation with receiver interest "intelligent adaptation" and we call our overall scheme Scalable Consensus-based Bandwidth Allocation or SCUBA because, roughly speaking, it employs receiver "consensus" to allocate session bandwidth among sources.

¹We assume the network provides only best effort service and that applications are thus adaptive. Furthermore, we view the development of real-time network services as a complementary optimization: our protocols will work effectively in concert with real-time services if and when they become widely available.

We created two variants of SCUBA — a "flat delivery" variant to complement sender-based adaptation and a "layered delivery" variant to complement receiver-based adaptation. In the flat delivery model, we use receiver feedback to constrain the rate chosen by the sender-based adaptation algorithm, while in the layered delivery model, we use the feedback to control the manner in which source signal layers are mapped onto network channels. Since the layers in a hierarchical distribution are ordered according to their relative importance, we can prioritize the ensemble of sources by controlling where each source falls within the hierarchy. In both the flat and layered delivery models, the feedback signal is identical and thus a single protocol can be shared across the two schemes. Even in the simple case where applications do not adapt (as is commonly the case for Mbone broadcasts), SCUBA is still useful — its flat-delivery variant will effectively manage the session's fixed bandwidth by dynamically adjusting each source's rate to reflect receiver interest.

One of the key advantages of receiver-based multicast over its sender-based counterpart is that the burden of adaptation is moved from the source to the receivers thus enhancing the scalability of the system. One could argue that we could apply this same technique to the receiver-interest problem and eliminate SCUBA altogether. That is, receivers could explicitly tell the network which sources they are interested in (via multiple multicast groups or source-based prunes²) and the network would automatically eliminate unwanted flows from the corresponding regions of the network. If no receiver expressed interest in a given source, the flow would be pruned back all the way to the originating host. While simple and elegant, this approach suffers from two drawbacks. First, the level of interest must be binary (i.e., a receiver must say either they want a source at full quality or not at all). And second, the scheme would share bandwidth inefficiently because it could not account for the constraints of shared paths. SCUBA instead enables sources to share bandwidth intelligently because the receivers convey their interest back to the source and thus effectively agree upon the best way to share the limited bottleneck capacities. We therefore believe a protocol like SCUBA is a necessary and important mechanism for rate-adaptive multimedia applications.

The rest of this paper describes SCUBA in detail, presents analysis to show that the protocol is viable and scalable, and demonstrates the utility of SCUBA in the context of several real applications. In the next section, we introduce and describe the basic operation of the SCUBA protocol. We then analyze the scalability of SCUBA and, in particular, derive confidence bounds on the algorithm's convergence time. Next we describe the deployment of SCUBA in the context of several applications. Finally, we summarize the status of SCUBA, present our plans for future work, and conclude.

²Source-based pruning is not part of the current IP Multicast specification but is included in the next version, IGMP-3, which is under review by the IETF.

Figure 1: Static bandwidth allocation: sources send at a fixed rate and apportion the link bandwidth equally.

Figure 2: Dynamic bandwidth allocation: sources dynamically adjust their transmission rate in response to receiver interest and apportion the link bandwidth accordingly.

2 The SCUBA Protocol

The basic premise of SCUBA is to reflect receiver interest back to the sources in a multicast session using a scalable control protocol. Figures 1 and 2 illustrate the core problem and its high-level solution. In current Mbone sessions, bandwidth is allocated in a fixed fashion as shown in Figure 1. Each sender transmits at some fixed rate, where the rates are chosen either manually or through sender-based adaptation. In either case, an equal amount of bandwidth is typically allocated to each source. Clearly, this outcome is undesirable if the sources are not equally important across all the receivers. But, by integrating receiver feedback into the adaptation process, we can weight each source's transmission rate in proportion to receiver interest. This approach is illustrated in Figure 2, where receivers generate feedback that controls each source's sending rate. In this case, source S_0 transmits at a higher rate than source S_1 because more receivers express interest in receiving the S_0 flow. Likewise, because there is no interest in source S_2 , its transmission is disabled entirely.

However, in a large multicast session composed of heterogeneous receivers with different capabilities, configurations, and preferences, how can we determine a single bandwidth partition that simultaneously satisfies all receivers? As above, one receiver might deem a source S_0 most impor-

tant while another prefers source S_1 . In light of conflicting feedback like this, we adopt a simple solution: participants reach consensus through voting. If half of the receivers wish to view source A and half wish to view source B, then a reasonable policy is to partition the available session bandwidth equally across the two sources. Now the problem becomes how to poll the receivers, assess the vote, and convey the outcome to each individual source.

Unfortunately, receivers cannot immediately generate notification messages when there is a state change or event in the session that would cause a shift in focus since uncontrolled control traffic like this could cause large, transient traffic loads. Instead, we scale back the rate at which receivers vote based on the multicast session size, much as the RTP [20] control protocol, RTCP, adjusts the rate of its control messages to maintain a constant control traffic load independent of session size. However, a lower messaging rate increases the response time and circumvents our fundamental goal of supporting effective human-to-human interaction (i.e., because slow convergence will prevent users from shifting their focus on a natural human time scale).

To alleviate this problem, we model SCUBA on the "exit poll" principle. In a large vote we can very accurately and with high confidence predict the outcome from only a small (unbiased) sample of the population. In SCUBA, therefore, we derive the source bandwidth partition from a relatively small sample of the receiver population.

2.1 Feedback Algorithm

To conduct the distributed polling process, each receiver associates a weight with every active media source. Qualitatively, the weights define the relative priorities of each source with respect to a given receiver. These weights might be explicitly specified by the user, but more likely (and more naturally), they would be inferred through the user-interface (e.g., when the user manipulates the application to display a given source, the application would automatically instruct SCUBA to give maximum weight to that source).

Each receiver periodically advertises a *source weight report*, which is simply a vector of sources and their corresponding weights with respect to that receiver. Report intervals are dynamically sized to limit the control bandwidth to a fixed percentage of the session bandwidth. This scaling mechanism is employed by several *announce/listen* control protocols [18], including RTCP control traffic in RTP, Scalable Reliable Multicast (SRM) [7] session messages, and the Session Announcement Protocol (SAP) [9]. We analyze the scalability of the *announce/listen* metaphor with respect to SCUBA in Section 3.

As each receiver *announces* its source weight report by multicasting it to the session, each source *listens* for these reports, collects them up, and combines them into an aggregate metric, which we call the *average source weight*. Ideally, this metric would be the arithmetic mean across all reported values for that source, but in practice, we can only approximate the exact mean because maintaining a globally consistent view of every receiver's preference for every source would entail costly if not outright unrealistic overhead. Consequently, we approximate the average source weight as the

mean of a *sample* of reported values taken from the last M reports received by the source. Let $S_0 \dots S_{N-1}$ be the set of senders, and $R_0 \dots R_{M-1}$ be the set of receivers. The average source weight for S_k , or w_k , is then computed as follows:

$$w_k = \frac{1}{M} \sum_{i=0}^{M-1} \frac{w_{k,i}}{\sum_{j=0}^{N-1} w_{j,i}}$$

where M is the size of the sample of reporting receivers, $w_{j,i}$ is the weight for source S_j reported by receiver R_i , and N is the number of sources weighted in each report.

To simplify the calculation and reduce the amount of state maintained at the source, we constrain the sum of weights in a report to be 1, i.e.,

$$\sum_{j=0}^{N-1} w_{j,i} = 1 \quad \forall i$$

Now the expression for w_k is simply:

$$w_k = \frac{1}{M} \sum_{i=0}^{M-1} w_{k,i}$$

To enhance SCUBA's robustness, we deliberately designed the protocol using only *soft state* [6]. Sources and receivers can join and leave the session at will without impacting other session members. None of the state maintained at any source alone is critical to the correct execution of the algorithm since all state eventually times out and is refreshed by receiver reports. Hence, failure recovery is built into the protocol; we need no further mechanism to handle network partitions, host failures, and so forth. Finally, SCUBA control messages are idempotent — each message supersedes all previously sent messages — further enhancing the protocol's scalability and its resilience to packet loss.

Not only is the SCUBA protocol robust and scalable, but it is also quite simple. The algorithm run at each source can be implemented in just a few lines of code. The following pseudo-code implements the weight calculation algorithm:

```
foreach src in active_source_list {
  w = 0
  foreach rcvr in sample_receiver_list {
    w = w + last_reported_weight(rcvr,src)
  }
  avg_weights[src] = w/nsamples
}
update_rate_controller(avg_weights)
```

The first stage here computes a weight for each reported source, including the local source. In the second stage, the rate controller is invoked to carry out the consequent action. While the method of computing the source weight is identical for both the flat and layered delivery models, the rate-control action in the second stage is not. In the following two subsections, we detail how the average source weight or weights are translated into the sender's rate-control decision in each of the two delivery models.

2.2 Flat Delivery

In the flat delivery model, all sources transmit their media over a single network channel that has a uniform aggregate rate. We assume either that the multicast session has a fixed bandwidth allocation or that a sender-based rate-adaptation algorithm computes a rate limit for a given source. In either case, we call this overall bandwidth limit the *session bandwidth*.

The flat delivery adjustment decision maps the transmission weight directly to the source transmission bandwidth, i.e., if B_k is the transmission rate for source S_k , then we set

$$B_k = w_k B$$

where B is the session bandwidth, and w_k is the average source weight for S_k .

More generally, the mechanism that maps the receiver interest vector into a source-rate control decision is arbitrary and can be customized for the application or environment at hand. Having recognized this, we adopt the following rule (assumed for the remainder of the paper): 95% of the session bandwidth is allocated across sources that each have at least one interested receiver (i.e., those whose average weight is non-zero) while the remaining 5% is equally apportioned over the remaining sources (i.e., those whose average weight is zero). As a result, each active source multicasts its signal continuously, though perhaps at a very low rate. Each receiver can then build a thumbnail display of all the active sources, thereby enhancing the "sense of presence" of a multimedia collaboration. The following pseudo-code segment implements this policy:

```
if (avg_weight > 0) {
  share = 0.95 * avg_weight
} else {
  share = 0.05 / zero_weight_src_num
}
tx_rate = session_bandwidth * share
```

To further illustrate the flat delivery variant of SCUBA, we present the following example. Let S_0 and S_1 be senders and let R_0, R_1 and R_2 be receivers. Assume that at a certain time, the source weights of $\{S_0, S_1\}$ advertised by each receiver are:

- $R_0 : \{0.8, 0.2\}$
- $R_1 : \{0.8, 0.2\}$
- $R_2 : \{0.5, 0.5\}$

Upon receipt of these reports (for simplicity we assume the sample set includes all three receivers), the average source weights computed by S_0 and S_1 become 0.7 and 0.3, respectively. Assuming the session bandwidth, B , is 128 kb/s, proportional bandwidth allocation results in S_0 transmitting at $128 \times 0.95 \times 0.7 = 85.12$ kb/s and S_1 at 36.48 kb/s.

To better illustrate the dynamics of this distributed algorithm, we implemented it in the *ns* network simulator [15] and ran a simple experiment consisting of four sources and three receivers across a single bottleneck link. All sources were placed on one side of the link while the receivers

Figure 3: Simulation of the flat-delivery variant of SCUBA

were placed on the other. The session bandwidth was set to 400 kb/s.

Figure 3 shows the resulting sample paths. The sources join the session at times 1.4, 1.7, 2.0 and 2.3, respectively. Before time 3, no receivers report interest, and thus the sources equally share the 5% reserve. Then, at times 3.0, 3.2, and 3.3, receivers R_0 , R_2 , and R_3 each respectively announce interest in source S_1 . In response, S_1 transmits at the full 95% allocation (i.e., 380 kb/s). Finally, at time 3.4, R_2 shifts its interest from S_1 to S_3 and from that point on S_1 and S_3 share the 95% allocation in proper proportion.

2.3 Layered Delivery

Although SCUBA and sender-based rate adaptation work in concert to effectively incorporate receiver interest into the adaptation process, the sender-based approach is fundamentally limited because it cannot accommodate receiver heterogeneity. More precisely, a particular source's data rate is the same at all receivers, but in a session with heterogeneous qualities of receiver connectivity, congestion-free operation requires the source to set its bandwidth to that of the most constrained link. Running each source at the rate of the slowest link is clearly undesirable and prevents well-connected receivers from fully exploiting their available bandwidth.

This limitation can be overcome with a *layered* transmission scheme [1, 17, 21, 22]. In this model each source generates a layered media stream that is striped across multiple network channels. The layers are typically cumulative, i.e., if a receiver wishes to decode layer n , it must also receive layer 1 through layer $n - 1$, and each additional layer enhances the quality of the reconstructed signal. Thus, to maximize quality, a receiver subscribes to as many network channels as is possible without congesting the network. One approach, known as Receiver-driven Layered Multicast (RLM), is detailed in [17].

In [17], each source stripes its data in the most straightforward manner possible — a signal layer k is mapped to the corresponding network layer k as depicted in Figure 4. But this arrangement treats each source equally; in SCUBA,

Figure 4: Simple Layered Delivery

we want to distinguish more important sources from less important ones and reflect this into the transmission algorithm. Fortunately, we can achieve this goal with a novel assignment of signal layers to network channels based on the source weights computed by SCUBA. Given a weight, each source determines its *signal layer mapping*, i.e., the assignment of each of its signal layers to the corresponding network channel. The source then transmits each of its signal layers on the network channel defined by the mapping.

Figure 5: Prioritized Layered Delivery

Figure 5 illustrates how signal layers are manipulated by SCUBA. The figure shows how two sources each with four signal layers are mapped onto eight possible network channels. The left-hand side of the figure shows a layer mapping that corresponds to a prioritization of source S_1 over S_0 , while the right-hand configuration corresponds to a prioritization of S_0 over S_1 . Thus, if SCUBA control messages indicate that source S_1 has a larger average weight (i.e., higher priority) over source S_0 , then the layer mapping on the left would be collectively enforced by the sources. Likewise, in the reverse case, the right-hand configuration would be automatically selected.

We can further illustrate the natural fit between SCUBA and receiver-driven bandwidth adaptation by considering the process whereby an individual receiver incrementally adds network channels to its overall subscription. Suppose a receiver in the context of Figure 5(a) has initially subscribed to just one network channel. At this point, that receiver not

B	the session bandwidth (e.g., as computed by RLM)
N	the number of active sources
S_k	media source k
w_k	the average weight for S_k computed across the session via SCUBA
M	the number of signal layers
R_k	the bit-rate of signal layer k
C	number of network channels
L_k	target bandwidth limit for the k 'th network channel
$\mu_{j,n}$	the network channel assigned to signal layer n at source S_j

Table 1: Summary of notation.

only receives a stream of adequately low bandwidth, but this particular stream is guaranteed to originate from *the session's highest priority source*, in this case, S_1 . As the receiver subsequently increases its channel subscription, it not only increases the quality of its current flows, but also receives additional sources *in priority order*. Thus, when the receiver increases its subscription to two network channels, it additionally receives the low quality layer of source S_1 . Upon adding the third channel, it receives equivalent qualities of both sources. The fourth channel then increases the quality of the high priority stream, S_1 . In this way we complement a raw bandwidth adaptation mechanism like RLM with a mechanism that explicitly accounts for the relative importance of media flows across the session.

We now formalize the algorithm for determining the signal layer mappings from the set of average weights. In addition to the notation defined earlier, we define the following quantities:

- Let R_j be the transmission rate for signal layer j . Note that we assume that the rate hierarchies induced by all the media sources are homogeneous and further that each layer generates a constant bit-rate flow. Modifying our algorithm to account for heterogeneous and dynamic sources is a topic of future research.
- Let C be the total number of network channels available for transmission and let L_k be the bandwidth limit on the k 'th network channel. We assume that:

$$\sum_{k=0}^{C-1} L_k = B$$

- Let $\mu_{j,n}$ be the network channel assigned to signal layer n at source S_j .

Our notation is summarized in Table 1.

The goal of the algorithm is to compute $\mu_{j,n}$ — i.e., to map signal layers onto network channels — according to the SCUBA weights ($w_1 \dots w_N$) while adhering to the bit-rate constraints ($L_1 \dots L_C$). In addition, the algorithm should have the property that a network channel number assigned to

```

 $k \leftarrow 0$  (1)
while ( $k < N$ ) { (2)
     $\hat{w}_k \leftarrow w_k$  (3)
     $t_k \leftarrow 0$  (4)
     $\hat{L}_k \leftarrow L_k$  (5)
     $k \leftarrow k + 1$  (6)
} (7)
while ( $\exists k$  s.t.  $\hat{w}_k \neq 0$ ) { (8)
     $l \leftarrow \arg \max_k \{\hat{w}_k\}_{k=0}^{N-1}$  (9)
     $c \leftarrow \mu_{l,t_l-1}$  (10)
     $r \leftarrow R_{t_l}$  (11)
    if ( $r > \hat{w}_l B$  or  $t_l \geq M$ ) { (12)
         $\hat{w}_l \leftarrow 0$  (13)
    } else { (14)
        while ( $c < C$  and  $r \geq \hat{L}_c$ ) { (15)
             $c \leftarrow c + 1$  (16)
        } (17)
        if ( $c < C$ ) { (18)
             $\mu_{l,t_l} \leftarrow c$  (19)
             $t_l \leftarrow t_l + 1$  (20)
             $\hat{w}_l \leftarrow \hat{w}_l - \frac{r}{B}$  (21)
             $\hat{L}_c \leftarrow \hat{L}_c - r$  (22)
        } else { (23)
             $\hat{w}_k \leftarrow 0$  (24)
        } (25)
    } (26)
} (27)

```

Figure 6: Signal Layer Mapping Algorithm

layer k for some source is less than or equal to any channel assigned to a layer k of any other source with equal or lower priority. In summary, we impose the following constraints on our algorithm:

1. The network channel limits are observed.
2. No source S_k transmits more than its total share, $w_k B$.
3. $w_k \geq w_j \Rightarrow \mu_{k,n} \leq \mu_{j,n} \quad \forall n$

One algorithm that obeys these constraints is given in Figure 6. We assume that all values for $\mu_{j,n}$ are initially undefined and that a reference to an undefined value (line 10) is interpreted as 0 (including negative index values). t_k is a “temporary variable” that represents the next unprocessed signal layer of source S_k . Upon completion, $\mu_{j,n}$ contains a feasible layer mapping. A proof that our algorithm adheres to the stated constraints is given in the appendix.

3 Scaling Analysis

Any multicast protocol that makes decisions at a source on behalf of all the receivers in the network is fundamentally prone to scaling problems. There is simply no scalable mechanism to keep a tightly consistent view of every receiver in the network at each source. We explicitly address this scaling problem in SCUBA by sacrificing absolute consistency in favor of scalability through the use of sampling. In this section, we examine this tradeoff in detail, outline the parameters that we can adjust to tune SCUBA's scalability, and identify the fundamental limitations of our scheme.

There are two principal protocol parameters that limit the scalability of SCUBA with respect to session size:

- the control message size, and
- the expected source weight convergence time.

The size of each control message depends on the number of reported sources. If each receiver includes a weight for every active source, then the report size scales linearly with the number of active sources. In a large session with large numbers of active sources, the resulting message size will quickly exceed a reasonable limit, but we can avoid this problem by simply imposing a fixed limit on the message size. If we limit the number of sources for which each receiver reports to a small constant, then the control packet size is bounded. We claim that this limit is reasonable since human attention can realistically encompass only a handful of sources simultaneously.

The scalability analysis of the average source weight calculation time is more involved and we present it in the following section.

3.1 Convergence Time

As control messages are received, each source incrementally calculates its average source weight and adapts accordingly. But because reports arrive only intermittently (not immediately upon a state change) and because the source uses a fixed sample of the receiver population, some amount of time passes before a (perhaps abrupt) change in receiver interest is reflected back to the sources. For example, the active "speaker" in a conferencing application might change and thus cause each receiver to shift its interest to the new source; only after some amount of time of monitoring SCUBA control messages do the active sources learn of the change in interest. We call this latency the *average source weight convergence time* (ASCT). More precisely, the ASCT is the expected time between a change in receiver interest and the point at which a source learns an accurate estimate of its average source weight within the session.

As described in Section 1, the aggregate set of source weights can be viewed as a "consensus" on the allocation of bandwidth to sources across the session. The ASCT — i.e., the time to reach this consensus — should be small. If too large, the computed source weight can become stale. The "usability threshold" is the point where fluctuations in average source weights occur faster than the sources can calculate them. In this case, the weights become invalid even

before they have been established. Thus, given that the time-scale of fluctuations in receiver interests is determined by actual users, the average source weights must be computed on the time scale of seconds. In addition, we require that this time bound remain fixed as the number of receivers and/or sources increases.

To limit the report bandwidth and decouple it from the session size, we allocate a fixed share of the overall session bandwidth for this purpose. But to constrain the bandwidth in this fashion, each receiver's individual messaging rate must decrease in proportion to the session population. Consequently, the convergence time scales linearly with the number of receivers. Assume we have n receivers, and for simplicity assume a fixed report size S . Then, if B is the bandwidth of the control channel, the ASCT, t_{conv} , is given by $t_{conv} = nS/B$. This $O(n)$ growth implies that in conferences with large numbers of receivers the ASCT will exceed the "usability threshold."

In order to combat the linear time growth of ASCT, we turn to sampling. Rather than relying upon a control message from every receiver to correctly compute receiver interest, we instead use a sample of receiver reports to determine the average source weight. Specifically, we construct a sample of size N from the last N reports received by the source. Our analysis below indicates that a source can compute its average weight within a small confidence interval with high probability using a relatively small number of receiver report samples. This means that a source can calculate its average source weight, within tight probabilistic bounds, in a relatively short amount of time, in particular, within the "usability threshold."

We now derive the number of sample receiver reports, N , that a source requires to calculate its average weight within error ϵ and confidence $1 - \alpha$. ϵ is known as the *confidence interval* while $1 - \alpha$ is known as the *confidence level* where $0 \leq \alpha, \epsilon \leq 1$. More formally, we want to find the smallest N that satisfies:

$$P(|\mu_N - \mu| < \epsilon) \geq 1 - \alpha$$

where μ is the distribution mean, i.e., the actual average source weight, and μ_N is the sample mean, i.e.,

$$\mu_N = \frac{1}{N} \sum_{k=1}^N X_k$$

where X_k is a random variable whose value is the weight for this source reported by the k 'th sampled receiver report. Since the receivers generate reports from independently randomized timers, we can assume that $X_1, X_2, \dots, X_N$ are independent and identically distributed (when receiver interest is stationary).

A consequence of the Central Limit Theorem is that

$$N = \left(\frac{Z_{1-\alpha/2}\sigma}{\epsilon} \right)^2 \quad (1)$$

where σ is the standard deviation of the weight distribution, and Z_p is the p -quantile of the unit normal variate [14]. Using this result, we can compute the number of samples required to establish the desired confidence level for a given

(a) Number of samples vs. Confidence interval for $\sigma^2 = 0.25$.

(b) Number of samples vs. σ^2 for $\epsilon = 0.07$.

Figure 7: Number of SCUBA reports required to determine the average weight within given confidence bounds. The time values assume a control traffic bandwidth of 6.4 kb/s which is 5% of a 128 kb/s session bandwidth.

confidence interval. Equation 1 shows that for given confidence bounds the number of samples is a linear function of the variance of the weight distribution which *does not depend on the size of the sample space*. This is a key result: we have decoupled the convergence time of the algorithm from the session size and thus SCUBA can scale to arbitrarily large sessions. Moreover, we can fine tune the accuracy of the sample mean by trading off convergence time — larger sample sizes are more accurate but converge more slowly, while small sample sizes converge quickly by sacrificing accuracy.

Figure 7(a) shows a plot of the number of samples required to determine the average weight versus the confidence interval, ϵ , for the highest possible variance in a $[0,1]$ distribution $\sigma^2 = 0.25$ which occurs in a strictly bimodal $\{0,1\}$ -valued distribution. The plot shows a family of curves, each of which corresponds to a different confidence level, varying from 0.9 to 0.99. As a particular example, Figure 7(b) shows the number of samples required to determine the average weight for a confidence interval of 7%, i.e., $\epsilon = 0.07$. Once again, each line in the figure shows the number of samples required for different confidence intervals $1 - \alpha$ ranging from 0.9 to 0.99.

To convert these values to an actual convergence time, we must assume an average report packet size. If we assume that each receiver source weight report is 6 bytes (4 bytes for the source id + 2 bytes for the weight) and constrain the number of sources per report to some number, say 5, then including the 28 byte UDP header and an 8 byte SCUBA header we get an average report size of 66 bytes.

The right-hand y-axes in Figures 7(a) and (b) show the amount of time needed to accumulate the corresponding number of samples assuming a session bandwidth of 128 kb/s with 5% of this bandwidth allocated to the SCUBA control traffic.

We see that for given confidence bounds the amount of time required by a source to calculate its average weight varies widely depending on several factors, most significantly the variance of the weight distribution and amount of control traffic. In the figures we assumed a relatively conservative session bandwidth and within it a relatively conservative allocation to report messages. Increasing the control

Figure 8: Time to accumulate N samples vs. Control traffic bandwidth

traffic bandwidth leads to a corresponding linear decrease in the convergence time. We illustrate this dependency in Figure 8 which shows a plot of the amount of time needed to accumulate N samples vs. control bandwidth for various values of N . The horizontal line marks a somewhat arbitrary "usability threshold" (10 seconds) that we would like to achieve. Thus, according to the desired confidence bounds, we must choose an appropriate control bandwidth. Comparing these results to Figures 7(a) and (b) we see that allocating 20–25 kb/s to control traffic suffices for very tight confidence bounds on the average weight convergence within the 10 second threshold.

These results imply that it is advantageous to modulate the control bandwidth based on event changes to speed up convergence and then slow down the report announcements when the system is idle. For example, a receiver could increase the control bandwidth for 15 seconds after it changes its source weight report allowing this information to be more quickly assimilated by the sources, thereby lowering the convergence time while maintaining an acceptable level of control traffic.

An alternative interpretation of this bandwidth/delay tradeoff is that the delay represents the width of a moving average filter applied to the incoming samples that is required to achieve the desired confidence bounds. In this interpretation, the window size represents a design tradeoff. As the above analysis has shown, setting the window too small reduces our confidence bounds. On the other hand too large a window causes samples to become widely separated in time decreasing the accuracy of the source weight estimation.

In summary, our analysis shows that the use of report sampling combats the linear growth of the ASCT. The set of feasible confidence bounds that conform to our “usability threshold” of 10 seconds are independent of the session size but are directly dependent on the amount of control traffic that is allocated to the SCUBA reports. Fortunately, we found that very tight confidence bounds are attained with only 20–25kb/s of control traffic. While this level of control traffic is acceptable in some environments, it is unrealistic for environments like the current Mbone where the entire session bandwidth often comprises only 128 kb/s. A more reasonable control bandwidth would be 5 kb/s or so, and in this case, SCUBA is limited to session sizes on the order of 200 to 250 receivers, which in practice, accounts for almost all of the current Mbone content. But to scale beyond this, we must resort to an alternative mechanism. Section 5 describes one promising scheme, currently under development, that utilizes hierarchy to enhance SCUBA’s scalability.

Figure 9: Cross-media bandwidth allocation using SCUBA. Data and control messages are designated with solid and dashed arrow, respectively.

4 Applications

Though SCUBA defines a generic mechanism that complements existing rate-adaptation protocols, our original design goal was much more application specific. We initially set out to develop a robust, distributed protocol for the control of application-level media gateways [2], but as the protocol was refined and tuned for this specific application, it became apparent that SCUBA constituted a robust and useful building block in its own right.

This section details three applications that show how side-information generated by SCUBA can improve their perfor-

mance. We first describe the integration of SCUBA in our video conferencing tool *vic* [16]. We then detail the application of SCUBA to media gateway control. Finally, we show how a floor control application can leverage SCUBA in its design as well as how SCUBA can exploit floor control knowledge to determine its weight allocation.

4.1 Video Conferencing

One important application of SCUBA is to control a simple video conference. In this case, receivers can determine the source weights required by SCUBA via the user interface. In our implementation embedded in *vic*, we infer the set of sources that the user is interested in from the disposition of the user interface, and weight them accordingly with some simple heuristics.

In addition to the disposition of the *vic* user interface, SCUBA can utilize cues from cross-media activity. For example, video windows can be configured to automatically switch to the remote user that is currently active on the audio channel. Thus, SCUBA can utilize activity on the audio channel to configure its receiver-interest report on the video channel. Similarly, pen input to a shared whiteboard tool can be used to raise priority of media streams originating from the same source.

Some multimedia conferencing tools, e.g., *vic* [16] and *vat* [13], use a common control channel (called a “coordination bus”) to exchange cross-media control information. For example, when *vat* receives the first packet of a “talk spurt” it announces the event over the coordination bus. Any tool listening to the bus can then react accordingly. For example, *vic* could use the audio channel activity notification to allocate more computational resources to decoding and displaying the corresponding participant’s video stream as opposed to those from other inactive participants.

While *receivers* can utilize cross-media cues to control their behavior, the coordination bus model does not directly extend across the network from receivers to *sources*. Not only do we want the receiver to allocate more computational resources to decoding the active participant’s media, but we also want those media to be *transmitted* at higher quality. Rather than build an *ad hoc* protocol for this purpose, SCUBA provides all the requisite functionality. In effect, SCUBA provides a level of indirection that captures cross-media events at the receivers and reflects the desired dependencies back to the sources via receiver interest reports.

Figure 9 illustrates an example sequence of events between a source and a receiver, each of which has an audio and video component. When the audio receiver detects data from the source, it notifies the video receiver via a coordination bus message. In response, the video receiver expresses interest in that source via SCUBA control messages. In turn, the video source increases its transmission rate precisely as desired. Thus, there is no need to build a special-purpose voiced-switched video network protocol since SCUBA completely subsumes this functionality with a clean and uniform mechanism.

Figure 10: SCUBA in the "media gateway architecture."

4.2 Media Gateway Control

A common configuration for multimedia conferences entails one or more users participating across a slow, bottleneck link with the primary set of users participating across a higher performance "well connected" region of the network. For example, a user might tap into an MBone conference via ISDN or a high-speed modem; alternatively, multiple receivers might share a low-speed wireless link.

Media gateways [2] are one approach for managing network bandwidth in these environments. In this model, a gateway sits on the "well-connected" portion of the network and manages the bottleneck link by limiting the rate of incoming media streams through transcoding. In earlier work, we designed a gateway "application programming interface" (API) that enables fine-grained control of the rate of transmission on a per-source basis. We are now developing a comprehensive architecture for launching, configuring, and controlling gateways in arbitrary environments, including the constrained-link example mentioned above.

One of the difficult challenges in realizing our media gateway architecture was the design of the control protocol that runs between the low-bandwidth receivers and a gateway. The receivers must somehow interact with the gateway API to establish the bandwidth allocations across active sources. One obvious approach is to simply centralize the mechanism and force all network agents to rendezvous at a well known location. But this would introduce a central point of failure. A much better approach is to decentralize control through a distributed mechanism like SCUBA.

Integrating SCUBA with the media gateway architecture is relatively straightforward because each active transcoder embedded in the gateway can be modeled simply as a SCUBA source. From a receiver's point of view, there is no difference between a rate-controlled output on the origi-

nal source or a rate-controlled output of a transcoder within a gateway. Therefore, we can run SCUBA locally among the receivers and the transcoders in the media gateways over a managed subset of the session. Again, the SCUBA protocol subsumes the functionality that would otherwise be implemented by a special-purpose protocol for managing the bandwidth of a locally-administered bottleneck link.

Figure 10 illustrates how SCUBA operates in tandem with a media gateway. Media streams from MBone sources S_0, S_1 , and S_2 are received by the gateway and are transcoded by per-source transcoders T_0, T_1 , and T_2 . Each transcoder performs format and/or rate conversion on the input media stream and transmits the resulting output stream across the bottleneck link where it is received on the other side of the link by the receivers ($R_0, \dots, R_3$). The original high-rate streams are not forwarded across the link.

By running SCUBA between the low bandwidth linked receivers and the gateway, scarce bottleneck bandwidth can be dynamically apportioned in an intelligent manner among the transcoders. In this way SCUBA provides a robust and distributed control mechanism for the gateway free from the vulnerabilities of centralized control.

4.3 Floor Control

Traditionally, floor control protocols orchestrate access to the constituent media channels of a distributed conference. A potential benefit of the exclusivity of channel access is efficient network utilization because only the sources that are "granted the floor" are allowed to transmit media and consume network bandwidth. SCUBA can further enhance existing floor control protocols by providing the mechanism for throttling media sources in response to floor control activity. For example, if the floor control protocol determines that some source has been granted the floor, then all the receivers could simply register interest in that source via SCUBA. As a result, the active source would increase its transmission rate as is desired. This approach obviates the need to intertwine the floor control protocol directly with source's rate adjustment algorithm. In addition, the method by which the floor configuration is mapped onto sending rates is not fixed at each source, but rather can be tuned at each individual receiver.

Alternatively, a floor control tool might directly control the source's sending rate. For example, once the floor control tool "grants the floor" to a particular source, that source could begin transmitting at the rate appropriate to the "holder of the floor" and all other sources would immediately limit their rate. Once this transition has occurred, the sources continue adjusting their transmission rates according to the SCUBA reports. Since the holder of the floor will in general be the source of interest, SCUBA reports will only confirm what has already occurred in the session: that the floor holder should receive the highest bandwidth share. Thus, through the use of the floor protocol, we have decreased the convergence time by using the floor control directive to "guess" the next bandwidth partition. Between floor control actions, SCUBA could continue to adapt dynamically to fine-grained changes in receiver interest. In short, SCUBA can effectively augment floor control proto-

cols through the integration of receiver interest.

5 Status and Future Work

To demonstrate a proof of concept of our design work, we implemented SCUBA in both our video conferencing tool, *vic* [16], and our media gateway, *rtpgw* [2]. These implementations follow the algorithms detailed in Section 2.2 and illustrated in Figure 3. In conjunction with our SCUBA-enhanced *vic*, we deployed and controlled these gateways in the manner described in Section 4.2. In future work we plan to extend our *vic* and *rtpgw* implementations for layered transmission and RLM [17] using the signal mapping algorithm detailed in Section 2.3.

Section 3 detailed the limitations of the sampling approach to calculating the average source weight in low-bandwidth sessions with large numbers of receivers. We are currently developing an alternative approach to sampling that employs a spatial network hierarchical to enhance the scalability of the announce/listen receiver interest protocol. Each level of the hierarchy corresponds to a “report region” and within each region, a representative is elected. The representative aggregates the reports within that region, summarizes the aggregate information into a single report, and forwards that report up the hierarchy. In this way we greatly enhance the scalability of the protocol since the average weight convergence time is approximately proportional to the number of sources in local regions rather than in the entire session.

6 Summary

In this paper we presented a novel protocol called SCUBA that complements existing rate-adaptation algorithms with a mechanism for discovering and exploiting receiver interest. As with RTCP and SRM, SCUBA embraces the announce/listen metaphor and uses periodic and robust receiver reports to explicitly rank active sources. When combined with layered transmission, SCUBA enables a source to infer an arrangement of signal layers onto network channels that best suits a heterogeneous set of layered receivers. We formalized a model for mapping signal layers onto network channels and presented an algorithm that satisfies a number of important and natural constraints.

By exploiting sampling, we overcame the linear growth in the convergence time of a naive version of the distributed SCUBA algorithm. Each source uses sampling to determine its share of the session bandwidth in a reasonable time with a high level of confidence independent of the session size. We pointed out the shortcomings of this approach and are currently developing new versions of the protocol that we believe will surmount these problems.

Finally, we outlined how SCUBA could be integrated into and exploited by several important applications such as traditional video conferencing, media gateways, and floor control.

We believe that reflecting receiver interest back to the sources in a multimedia conference is a key mechanism

for future collaboration technologies. SCUBA achieves this goal in a light-weight, robust, and scalable manner. By dynamically accounting for receiver interest in the allocation policy, sources can share the available session bandwidth in a way that best reflects the natural model for human-to-human communication.

7 Acknowledgements

SCUBA was inspired by Jacobson’s “video silence suppression” concept, which uses receiver interest to enable and disable video sources [10]. We are grateful to Hari Balakrishnan and Venkat Padmanabhan for providing constructive comments and lively discussions throughout the writing of this paper. Anindo Banerjee and Tom Henderson provided helpful comments on and corrected oversights in the paper. We thank Deana McCanne for suggesting several improvements to the paper’s presentation. A host of people provided constructive criticism on our work when it was first presented at the UCB Daedalus retreat; in particular, we thank Jim Gemmel, Kevin Mills, Sridhar Seshan, Mark Handley, and Lixia Zhang for their helpful comments. Finally, the anonymous reviewers provided valuable feedback on our work and its presentation.

Appendix

We present a proof of correctness of the signal mapping algorithm detailed in Section 2.3. We use the same notation introduced in Section 2.3.

Claim: This signal layer mapping meets the following constraints:

1. The network channel limits are observed.
2. No source S_k transmits more than its total share, $w_k B$.
3. $w_k \geq w_j \Rightarrow \mu_{k,n} \leq \mu_{j,n} \quad \forall n$

Proof: In the proof we refer to line numbers in the algorithm detailed in Figure 6.

1. $\hat{L}_c$ represents the unused bandwidth available in network channel c . Thus, the check in line 15 guarantees that no source will cause the rate limit of a network channel to be exceeded
2. $\hat{w}_l B$ represents the portion of the bandwidth share of source S_l that has not been utilized by the source. Thus, the decrementing of $\hat{w}_l$ in line 21, along with the check in line 12, guarantees that no source will transmit more than its total share, which is $w_l B$.
3. The algorithm iterates through the average weight list, A , looking for the largest remaining weight, $\hat{w}_l$. Now assume that for the two sources in question S_k , and S_j , all signal layers $1, \dots, n-1$ have been mapped to network channels. Next assume that the result of the algorithm’s output for layer n is such that

$$\mu_{k,n} > \mu_{j,n}$$

This along with line 9 imply that

$$\hat{w}_k < \hat{w}_j \quad (1)$$

Line 21 of the algorithm implies that in order for (1) to hold the following must also be true:

$$w_k - \sum_{i=1}^{n-1} \frac{R_i}{B} < w_l - \sum_{i=1}^{n-1} \frac{R_i}{B}$$

$$\Downarrow$$

$$w_k < w_l$$

Finally, the argmax directive in line 9 furnishes us with the bootstrap for the induction in the first iteration which completes the proof of statement 3 of the claim. $\square$

References

- [1] AMIR, E., MCCANNE, S., AND VETTERLI, M. A layered DCT coder for Internet video. *IEEE International Conference on Image Processing* (Sept. 1996).
- [2] AMIR, E., MCCANNE, S., AND ZHANG, H. An application-level video gateway. In *Proceedings of ACM Multimedia '95* (Nov. 1995), ACM.
- [3] BOLOT, J.-C., TURLETTI, T., AND WAKEMAN, I. Scalable feedback control for multicast video distribution in the Internet. In *Proceedings of SIGCOMM '94* (University College London, London, U.K., Sept. 1994), ACM.
- [4] BUSSE, I., DEFFNER, B., AND SCHULZRINNE, H. Dynamic QoS control of multimedia applications based on RTP. In *Proceedings of the First International Workshop on High Speed Networks and Open Distributed Platforms* (St. Petersburg, Russia, June 1995).
- [5] CHEUNG, S. Y., AMMAR, M. H., AND LI, X. On the use of destination set grouping to improve fairness in multicast video distribution. In *Proceedings IEEE Infocom '96* (San Francisco, CA, Mar. 1996), pp. 553–560.
- [6] CLARK, D. D. The design philosophy of the DARPA Internet protocols. In *Proceedings of SIGCOMM '88* (Stanford, CA, Aug. 1988), ACM.
- [7] FLOYD, S., JACOBSON, V., MCCANNE, S., LIU, C.-G., AND ZHANG, L. A reliable multicast framework for lightweight sessions and application level framing. In *Proceedings of SIGCOMM '95* (Boston, MA, Sept. 1995), ACM.
- [8] FREDERICK, R. *Network Video (nv)*. Xerox Palo Alto Research Center. Software on-line³.
- [9] HANDLEY, M. *SAP: Session Announcement Protocol*. Internet Draft, Nov 19, 1996.
- [10] JACOBSON, V. DARTNET Planning and Review Meeting, Dec. 1991.
- [11] JACOBSON, V. SIGCOMM '94 Tutorial: Multimedia conferencing on the Internet, Aug. 1994.
- [12] JACOBSON, V., AND MCCANNE, S. *LBL Whiteboard*. Lawrence Berkeley Laboratory. Software on-line⁴.
- [13] JACOBSON, V., AND MCCANNE, S. *Visual Audio Tool*. Lawrence Berkeley Laboratory. Software on-line⁵.
- [14] JAIN, R. *The Art of Computer Systems Performance Analysis*. John Wiley & Sons Inc., 1991.
- [15] MCCANNE, S., AND FLOYD, S. *The UCB/LBNL Network Simulator*. University of California, Berkeley. Software on-line⁶.
- [16] MCCANNE, S., AND JACOBSON, V. *vic: a flexible framework for packet video*. In *Proceedings of ACM Multimedia '95* (Nov. 1995), ACM.
- [17] MCCANNE, S., JACOBSON, V., AND VETTERLI, M. Receiver-driven layered multicast. In *Proceedings of SIGCOMM '96* (Stanford, CA, Aug. 1996), ACM.
- [18] SCHOOLER, E. M. A multicast user directory service for synchronous rendezvous. Computer science department, California Institute of Technology, Sept. 1996.
- [19] SCHULZRINNE, H. Voice communication across the Internet: A network voice terminal. Technical Report TR 92-50, Dept. of Computer Science, University of Massachusetts, Amherst, Massachusetts, July 1992.
- [20] SCHULZRINNE, H., CASNER, S., FREDERICK, R., AND JACOBSON, V. *RTP: A Transport Protocol for Real-Time Applications*. Internet Engineering Task Force, Audio-Video Transport Working Group, Nov. 1991. Internet Draft expires 3/1/96.
- [21] SHACHAM, N. Multipoint communication by hierarchically encoded data. In *Proceedings IEEE Infocom '92* (1992), ACM, pp. 2107–2114.
- [22] TAUBMAN, D., AND ZAKHOR, A. Multi-rate 3-D subband coding of video. *IEEE Transactions on Image Processing* 3, 5 (Sept. 1994), 572–588.
- [23] TURLETTI, T. *INRIA Video Conferencing System (ivs)*. Institut National de Recherche en Informatique et en Automatique. Software on-line⁷.
- [24] WU, L., SHARMA, R., AND SMITH, B. Thin streams: An architecture for multicasting layered video. In *Proceedings of the Seventh International Workshop on Network and OS Support for Digital Audio and Video* (St. Louis, Missouri, May 1997), ACM, pp. 183–192.

³<http://ftp.parc.xerox.com/net-research>

⁴<http://ftp.ee.lbl.gov/conferencing/wb>

⁵<http://ftp.ee.lbl.gov/conferencing/vat>

⁶<http://www-mash.cs.berkeley.edu/ns/>

⁷<http://www.inria.fr/rodeo/ivs.html>