IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications

Hugh W. Holbrook and David R. Cheriton
Department of Computer Science, Stanford University
{holbrook,cheriton}@dsg.stanford.edu

Abstract

In the IP multicast model, a set of hosts can be aggregated into a group of hosts with one address, to which any host can send. However, Internet TV, distance learning, file distribution and other emerging large-scale multicast applications strain the current realization of this model, which lacks a basis for charging, lacks access control, and is difficult to scale.

This paper proposes an extension to IP multicast to support the *channel* model of multicast and describes a specific realization called *EXPlicitly REquested Single-Source (EXPRESS)* multicast. In this model, a multicast *channel* has exactly one explicitly designated source, and zero or more channel *subscribers.* A single protocol supports both channel subscription and efficient collection of channel information such as subscriber count. We argue that EXPRESS addresses the aforementioned problems, justifying this multicast service model in the Internet.

1 Introduction

IP multicast [6, 7] is based on a *group* model of communication in which a set of hosts can be aggregated into a *group* with a single address. Any host can send to the group by sending to this address, the same as it can send to an individual host. Although this model of communication is attractive for multicast discovery and for small-scale meetings over the Internet, current realizations of it are strained in supporting very large-scale multicast applications such as Internet TV. For example, consider the "sports-tv.net" content provider who wishes to provide a live multicast video feed of the Super Bowl to 10 million subscribers. Based on costs of conventional media, like broadcast TV, such a large-scale delivery channel could have huge economic value, comparable to the millions of dollars paid in the current broadcast TV world. However, supporting this application is difficult with the IP multicast model and its current implementations.

As a first problem, IP multicast, as deployed today, violates common ISP billing models in which the input data rate is the basis for ISP charging, assuming that a customer input data rate of R imposes a delivery data rate of R. The delivery cost to the ISP of a large-scale multicast stream is much higher than that of a unicast stream of the same rate, yet both streams are equally charged in the input-rate-based billing model. In the case of sports-tv.net with its 10 million subscriber, 4 megabit per second MPEG-2 Super Bowl feed, the aggregate bandwidth delivered is 40 terabits per second, an enormous burden for the ISP. The problem is less severe for smaller sessions but it still may make multicast disadvantageous for an ISP to provide. As such, an ISP may decide to put off providing multicast, forcing a source wanting to reach k sites at rate R to simulate multicast with unicast and thus pay for $k{*}R$ bandwidth. Tackling this charging problem for multicast exposes a number of other difficulties with the group model.

As a second problem, the current IP multicast facility provides no indication of the group size, making it infeasible for an ISP to charge based on this value. In conventional media, a ten million subscriber base is much more valuable than a ten thousand one, and similar economics arise in the Internet. However, IP multicast does not currently help the ISP to distinguish between even very broad ranges of group size.

As a third problem, current implementations of IP multicast do not provide a means to restrict the allowed senders — any host can send to any IP multicast address. As a result, an unauthorized sender can send traffic to a multicast group, compromising its use, and thus its value. For example, returning to the Super Bowl example, a third party can maliciously or carelessly send its own high-rate data stream to the Super Bowl multicast address, say at the moment of the crucial touchdown, interfering with reception and creating great dissatisfaction among the customers of "sports-tv.net."[1] A content provider is unlikely to pay for a channel without assurance that it can control the content, *i.e.* the source(s). Thus, this Super Bowl application and many others are simply not feasible without source access control. Manual configuration might be used to restrict access to a small number of such large-scale groups, but this approach does not scale to support the thousands of Internet radio stations and TV channels that could be deployed world-wide.

As a fourth problem, the group model requires allocating a world-wide unique multicast address to each application to avoid extraneous cross traffic because the current IP multicast address space is globally shared among all hosts and applications.[2] With just 256 million multicast addresses for the whole world, a global address allocation mechanism such as IMAAA [11] is required, with all its deployment and operational issues. IPv6, once deployed, would eliminate the scarcity of multicast addresses, but there is still the challenge of ensuring global uniqueness. The slow deployment of IPv6 prompts examining solutions for IPv4.

Fifth, and lastly, scaling IP multicast routing for conventional group semantics remains a research challenge after many years of effort [15].

This paper proposes extending the IP multicast service model to support multicast *channels*, providing explicit support for these large-scale multicast applications. Section 2 defines the channel model and describes a realization of it for IPv4, called EXPRESS, and shows how it addresses the problems of these applications. Section 3 describes the EXPRESS protocol, ECMP, which supports channel subscriptions, receiver authorization and counting on the channel. Section 4 describes how EXPRESS can be extended at the application or middleware level to support multi-source applications including in particular large-scale *almost-single-source* applications, like distance learning, where audience members are allowed to gain the floor and address the subscribers. Section 5 demonstrates that per-channel costs are small and that EXPRESS memory and bandwidth usage scales linearly with the number of channels, suggesting that the larger number of channels used by some multi-source applications is not a problem in practice. Section 6 analyzes the cost of counting and describes *proactive counting*, a technique for collecting accurate and timely counts at lower cost than polling in some cases. In Sections 7 and 8, we present related work and our conclusions.

2 IP Multicast Channels

A multicast *channel* is a datagram delivery service identified by a tuple (S,E) where S is the sender's source address and E is a *channel destination address*. Only the source host S may send to (S,E).

A *subscriber* host requests reception of data sent to a channel by explicitly specifying both S and E to the network in a request. The source S sends to a channel by simply transmitting a datagram addressed to E. The network layer guarantees that all datagrams sent by S to destination E are delivered to all subscribers of channel (S,E) with delay and reliability of similar quality to that of a unicast datagram from S.

As illustrated in Figure 1, in the channel model, packets are addressed to the channel, whereas in the group model, packets are addressed to a host or a host group. Thus, contrary to the group model, two channels (S,E) and (S',E) are unrelated, despite the common destination address. A subscriber to (S,E) does not automatically receive packets sent to (S',E).

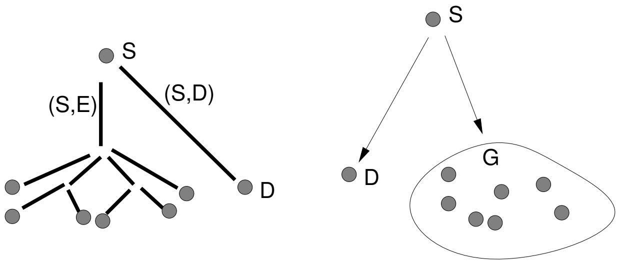

Figure 1: Channel vs. group addressing.

2^{24} class D addresses are allocated by IANA for experimental use by the single-source multicast model [14], shown in Figure 2. Thus, each host interface in the Internet can source up to 16 million channels. Routers identify a channel multicast datagram by its destination address.

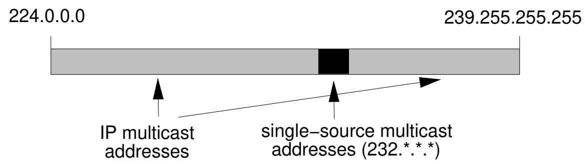

Figure 2: Single-source IP Multicast Addresses

Packet transmission to, and reception on, a channel use the same service interface as IP multicast, per [6].

[1] Similar denial of service and resource consumption problems can occur with unicast, but the packet-amplifying property of multicast magnifies the problem, allowing one misbehaving sender to annoy and/or disrupt *all* group members at once.

[2] Administratively-scoped addresses reduce address space contention when the application's receiver population is known to be geographically localized, but this is not the case for the multicast applications motivating this work.

2.1 EXPRESS Service Interface Extensions

In the EXPRESS realization of the channel model, the IP multicast source host service interface is extended with the function

```
count = CountQuery(channel, countId, timeout)
```

where `channel` is an (S,E) pair identifying the channel. It efficiently collects a *best-efforts* count for the channel within the specified timeout. `CountId` identifies the type of count requested. One important count is the number of subscribers; other possibilities include application-defined votes, a measure of the number of links in the tree, or a weighted tree size measure.

A source uses this interface:

`channelKey(channel,` $K_{(S,E)}$ `)`

to inform the network that `channel` is authenticated. The network layer ensures that only hosts presenting $K_{(S,E)}$ can subscribe.

The source can also *subcast* a packet to a subset of the subscribers by relaying it through an internal node in the multicast distribution tree [16]. A router on the distribution tree for (S,E) decapsulates the packet received from S and forwards it toward all downstream channel receivers. This mechanism needs no additional interface – the source unicasts an encapsulated packet to an "on-channel" router, addressing the encapsulated packet to the channel.

The subscriber service interface is extended with

`result = newSubscription( channel [,` $K_{(S,E)}$`] ),`

where $K_{(S,E)}$ is an optional authenticator allowing access to a restricted channel. If a `newSubscription` fails due to a missing or improper key, the call returns a failure indication via the `result` parameter. A similar `deleteSubscription` interface is used to unsubscribe from a channel.

A subscriber replies to a `CountQuery` request with

```
count( channel, countId, count ).
```

2.2 Advantages

The EXPRESS channel model has advantages for the source, subscribers and ISP in large-scale multicast applications, when compared with the group model.

2.2.1 Source Advantages

EXPRESS provides 2^{24} channels *per source*, allowing each host to autonomously allocate channels. Duplicate allocation is an issue only at a single host, which the host operating system can avoid with a local database of allocated channels. With the large number of available addresses, channels need not be treated as a scarce resource. Furthermore, address management for large single-source applications is simplified relative to the group model by eliminating the need to request an address from an allocation service or return it to the global pool when the application is done using it.

Also, the source has exclusive transmission access to a channel and can, if desired, control other hosts' ability to subscribe to it. Authenticated subscriptions may be primarily important for small-to-medium sized channels. A large distribution channel's goal is often to maximize the receivership, and not to limit it, *e.g.*, an Internet TV channel paid for by advertising.

Finally, the source can use the `CountQuery` mechanism to efficiently determine the number of subscribers or to take a subscriber vote. This capability is useful for large-scale meetings or interactive television. Fusing inputs from the N subscribers to the source is the complement of delivering input from the source to N subscribers, and, as described later, naturally fits into the implementation.

Elaborating on this counting facility, an Internet TV station can conduct a poll of votes on some topical interest, getting a response from potentially millions of subscribers while only having to send and receive a small number of packets. A range of `countIds` is reserved to have application-defined semantics. For instance, a subscriber client could present an application-specific dialog box and message when such a `countId` query arrives.

This counting facility is also useful for reliable wide-area multicast. It can be used to efficiently collect positive acknowledgements or negative acknowledgments to determine how many subscribers missed a particular packet.

2.2.2 Subscriber Advantages

The subscriber of a large-scale channel is assured of only receiving traffic from the source it designates. It need not explicitly exclude other sources, as with IGMPv3. This is particularly important in applications where the bandwidth on the receiver's last-hop link (*e.g.*, a home ISDN link) is a valuable resource. The ability to provide feedback to the content provider with the count mechanism can also be considered a benefit to the subscriber in that it enables new application-level features.

2.2.3 ISP Advantages

EXPRESS channels are a potentially valuable service from which an ISP can receive new revenue. We argue that the new applications and traffic generated by this service would more than compensate the ISP for any loss of unicast revenue. The single source "ownership" of the channel gives a basis on which to charge and, of course, whom to charge, namely the source. By contrast, in the group model, although transmission costs can be billed to the sender, there is no easily identifiable entity to bill for the network costs of maintaining and operating the group. The ability, provided by the counting support, to determine the number of subscribers assists the ISP in charging for multicast channels based on different scales of use, differentiating among channels with 10s, 100s, 1000s, and millions of subscribers. The counting facility enhances multicast applications, benefits customers, and ultimately increases revenue for the ISP. Finally, EXPRESS is relatively simple to implement and manage, as described in the next section.

All of the above advantages come directly or indirectly from restricting the multicast channel to having a single source. We do not regard this as significantly restricting its applicability because even multi-source applications can be efficiently implemented in terms of single-source channels, as described in section 4, often, in our experience, with relatively simple modifications.

3 EXPRESS Count Management Protocol (ECMP)

EXPRESS is implemented using ECMP, a single common management protocol that both maintains the distribution tree and supports source-directed counting and voting. The protocol treats the routing and counting aspects similarly, and distribution tree construction for a single source is a restricted case of counting the subscribers in each subtree. ECMP generalizes the basic subscribe/unsubscribe function by allowing a count, rather than just a binary value per subtree, and by allowing other values to be counted as well.

The routing aspect of ECMP is simple because explicit source specification allows reverse-path forwarding (RPF) [5] to be used to route subscriptions and unsubscriptions toward the source. There is no need for an additional network-layer rendezvous protocol or other mechanism to determine how to deliver a packet from potentially any sender to all receivers, as necessary in the group model. The RPF routing component of ECMP relies on, and scales with, existing unicast topology information.

ECMP consists of three messages:

`CountQuery(channel, countId, timeout)`

`Count(channel, countId, count, [`$K_{(S,E)}$`])`

`CountResponse(channel, countId, status)`

$K_{(S,E)}$ is only supplied for authenticated channels.

The next section defines these messages and describes the protocol operation, first focusing on generic counting, assuming the channel distribution tree exists, then describing distribution tree set-up.

3.1 Generic Counting Operation

The source requests a count of a specified attribute by sending a `CountQuery` to the first-hop router for the channel, specifying a channel, the associated `countId` and a timeout.

The receiving router creates a record for this query for each downstream neighbor on the specified channel, decrements the timeout value by a small multiple of the measured round-trip time to its upstream neighbor and forwards the request to each downstream neighbor. By decrementing the timeout at each hop, the protocol tries to ensure that a router that fails to get a response from one of its children times out and sends a partial reply to its parent before the parent itself times out. A `CountQuery` is forwarded all the way to each subscriber host at the leaf of the tree.[3] Depending on the `countId`, the operating system either answers the query immediately, or forwards it to the subscribing application(s). An end-station host responds to the `CountQuery` with a `Count` message specifying the `countId` and the count value.

The value in the `Count` response is recorded locally. Once `Counts` are received from all neighbors, or after the timeout specified in the original query, the counts are summed and the total is sent upstream in a `Count` reply.

A router can either acknowledge or reject a `Count` message by sending a `CountResponse` indicating an unsupported count or an invalid authenticator.

ECMP also allows any router on the channel distribution tree to initiate a query without source cooperation, enabling the network layer to count network-layer resources. For example, in a large-scale channel that spans many administrative domains, the ingress router for transit domain D might initiate a query to count the number of links used within D. This information could be used to make inter-domain settlements or for resource planning. A sub-range of `CountIds` is designated for locally-defined use.

This generic mechanism is also used to maintain the channel distribution tree.

3.2 Distribution Tree Maintenance

A reserved `countId`, *subscriberId*, designates the number of subscribers in a subtree. A router must record the per-channel subscriber count for each interface.[4]

A `newSubscription` causes an unsolicited *subscriberId* `Count` message to be propagated toward the channel source, according to the RPF algorithm, the same technique used to route source-specific joins in PIM-SM and joins to the core in CBT. As illustrated in Figure 3, the *subscriberId* message propagates hop-by-hop until it reaches the source or a router already on the

[3] `CountIds` corresponding to some network-layer resources are not propagated all the way to leaf hosts. These counts use a separate range of the `CountId` space.

[4] At a minimum, it must record whether the count is zero or nonzero.

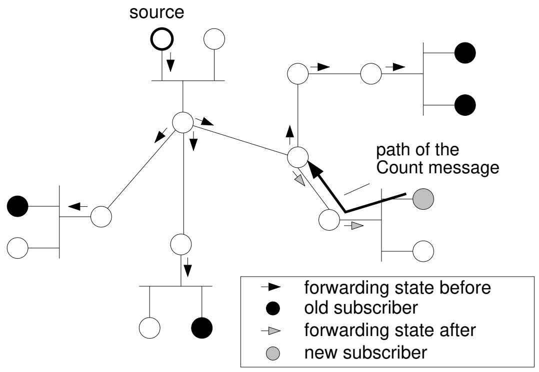

Figure 3: A host subscribing to an EXPRESS channel.

distribution tree. A host unsubscribes by sending a zero `Count` message upstream. A router receiving an authenticated subscription passes $K_{(S,E)}$ upstream for validation. The subscription is eventually validated or denied by a `CountResponse` from the upstream router, and a valid key is cached so that further authenticated requests can be denied or accepted locally. Key distribution is not part of ECMP – hosts must learn $K_{(S,E)}$ with an out-of-band mechanism.

A router can select either TCP or UDP mode for ECMP on each interface, or both. TCP is provided for core routers with few neighbors and many channels, whereas UDP is intended for use in edge routers, with many neighboring end hosts but fewer channels. In TCP mode, the router maintains a TCP connection to each neighbor and a per-channel subscriber count for each neighbor. The associated count is subtracted from the sum provided upstream if the connection fails. On connection establishment, the downstream neighbor sends an unsolicited `Count` message for each channel it has going upstream to this node. With TCP operation, a periodic refresh of each long-lived channel is unnecessary — a single per-neighbor keepalive is sufficient to detect a connection failure.

For UDP operation, the upstream router periodically multicasts a `CountQuery` request, analogous to an IGMP query, causing all the UDP neighbors to respond with `Count` messages for the specified channel. The router maintains a per-channel, per-interface count of UDP neighbors. A UDP neighbor unsubscribes by sending a zero `Count` message, causing the upstream router to decrement its sum and re-issue a `CountQuery` on that interface (like IGMPv2). Unlike IGMPv2, but like the proposed IGMPv3, there is no report suppression. All multicast ECMP datagrams are sent to a well-known ECMP address.[5]

When a topology change causes a router to select a different upstream router for a channel, it sends a current `Count` message to the new upstream router and a zero `Count` message to the old upstream router, unsubscribing it there. Hysteresis is applied to prevent route oscillation.

3.3 Neighbor Discovery

A reserved *neighbors* `countId` designates neighboring EXPRESS routers. Each router periodically multicasts such a UDP `CountQuery` message. The responses received allow the router to establish connections as above.

Another `countId` indicates *all channels*, and a `CountQuery` with this value is periodically sent to a LAN-local group multicast address. This message solicits `Count` retransmissions from all hosts for all channels, analogous to an IGMP general query.

3.4 EXPRESS Packet Forwarding

ECMP sets up the *Forwarding Information Base (FIB)* entries at each router, just as with conventional multicast. The EXPRESS forwarding procedure is nearly identical to that of conventional IP multicast. In particular, when a router receives an EXPRESS packet, it looks up (S,E) in the FIB and forwards the packet to the set of outgoing network interfaces, if the incoming interface matches the FIB entry's, dropping or forwarding to the CPU if not.[6] An EXPRESS multicast packet that does not match an exact (S,E) entry in the FIB is simply counted and dropped, as opposed to being forwarded to a rendezvous point as in PIM-SM, or broadcast, as with PIM-DM and DVMRP.

The currently deployed router forwarding mechanisms, including hardware-accelerated multicast engines, support EXPRESS without modification, reducing the cost, risk and time to revenue for ISPs and for router vendors.

3.5 Authenticated ECMP vs. End-to-End Encryption

The authenticated access mode of ECMP is primarily intended to protect the source and the network from incurring unwanted costs due to unauthorized subscriptions. Authenticated subscriptions do not subsume and are not subsumed by end-to-end encryption. Encryption provides confidentiality of encrypted data, but does nothing to protect access to network resources or to prevent a malicious user from getting access to the data in the first place. On the other hand, while authenticated access may be sufficient for some applications, in general we expect conventional end-to-end encryption to be used when strong data confidentiality is a requirement.

[5]This restricted local use of multicast can fit the EXPRESS model by using a well-known *localHost* value as the source. This is all that ECMP and most applications need.

[6]The incoming interface check is well-known in reverse-path multicast algorithms, and it is used to prevent data loops, as described in [5, 7].

In our authentication scheme, routers are trusted to securely cache and manage $K_{(S,E)}$. The security of $K_{(S,E)}$ during router-to-router communications and in the router is sufficient in most cases, because routers must already fundamentally be trusted (at the network layer) to only forward data to properly authorized destinations. Router-to-router links internal to the network are usually secured from snoopers to protect the regular data traffic in any case. On untrusted shared-media links, point-to-point encryption of the key can be used to prevent snooping.

3.6 ECMP Advantages

ECMP provides a simple integrated protocol that supports subscription and multicast channel maintenance and counting. It eliminates the need to implement, support and understand different protocols for host-to-router and router-to-router as well as separate protocols for access control and accounting, yet is simpler than other multicast routing protocols because of the single-source restriction.

ECMP is implemented on top of UDP and TCP, and so can be deployed on an end system host that supports IP multicast without changing the host operating system. Hosts can continue to use IGMP for the rest of the class D address space. ECMP largely builds on techniques proven in IGMP, PIM-SM [9] and CBT [2], reducing the risk associated with completely new protocol mechanisms. However, implementing EXPRESS using ECMP, a new protocol, seems warranted by the opportunity to both simplify the protocols required for multicast as well as to provide extended features.

The single-source restriction makes ECMP simpler to manage at the network layer than group multicast protocols. First, there is no need to select and manage rendezvous points, as in PIM-SM or configure when traffic should split off into source-specific trees. Relatedly, with EXPRESS channels, multicast traffic only travels along paths from the source to the subscribers. In contrast, with group multicast protocols, packets can traverse routes that are distant from the expected direct path from source to receiver, either detouring via the rendezvous point or broadcasting throughout a domain, depending on the protocol in use. The relative simplicity of the RPF routing algorithm used by EXPRESS eases the task of verifying that multicast is, or is not, flowing as expected.

4 Multi-source Multicast Applications

Multi-source applications can be built on top of EXPRESS channels by using multiple channels, one per source, or by allowing several sources to share a channel using higher-level *relaying* through the channel's source host. We first consider large-scale multicast applications that are *almost single-source*.

A good example of an almost single-source application is distance learning. In this application, the primary source is the lecturer or teacher multicasting over a channel to a large number of student listeners (subscribers). However, any one of these students may become a (secondary) source for a short period of time when asking a question. A networked conference, like SIGCOMM via the Internet, has similar characteristics.

These *almost single-source* multicast applications can be built using EXPRESS channels and the relaying approach supported by a middleware layer as follows.

4.1 The Session Relay Approach

Each SR-based application, *e.g.*, conference or lecture, has an associated *session relay (SR)* on an application-selected host SR that acts as the source for the EXPRESS channel (SR,E) to which each participant in the lecture subscribes. The SR coordinates access to the session. The primary lecturer or speaker either resides on the SR or relays its packets to it and onto the multicast channel by unicasting an encapsulated packet to the SR, as in Figure 4. Students ask questions which

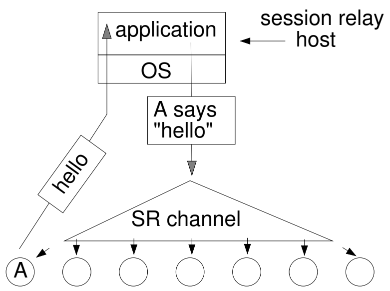

Figure 4: The session relay approach.

the other students can hear by relaying their transmissions through the session relay to the multicast channel (SR,E). The SR can use an application-layer relay protocol or an IP-in-IP-like encapsulation with application-level access control on the encapsulated forwarding. In either case, the application can strictly monitor and control the traffic over the multicast channel. In contrast, the rendezvous or core mechanisms of existing multicast protocols support neither.

The session relay channel address (SR,E) can be provided along with publishing or advertising the time, date and topic of the event so is not a major additional burden. Event advertisement can use web page, a "push" EXPRESS channel from one or more directory services, email, or other means. Secure applications [18] require the sources to be known in advance if source-specific keys are used, providing another opportunity to inform the sources of the session relay channel. In any case, the session relay channel only needs to be known to the session participants, not all routers in the Internet, as arises with network-layer rendezvous points in PIM-SM [9].

An alternative to the pure session relay approach is for a secondary sender to create a new channel for which it is the source and use the SR to ask all other session participants to subscribe to the new channel. The space cost for a reasonable number of such channels is modest, as argued in Section 5. However, creating a new channel incurs additional subscription overhead for the subscribers. Therefore this technique is primarily applicable when the new source is going to transmit for an extended period of time and when there is considerable delay benefit to using the direct channel over relaying.

4.2 Advantages

The session relay structure provides a number of benefits to large-scale multicast applications.

First, the application can select the placement of SRs to minimize communication. For example, an enterprise multicast video conference with participants scattered throughout the various branch offices can select an SR located near the topological center of the enterprise WAN, whereas a training lecture that targets employees at just one site can select an SR local to that site, avoiding the overhead of sending traffic off-site. In contrast, with network-layer approaches as in PIM-SM [9], the network administration selects the RPs as part of network configuration independent of applications, and applications have no control over the RPs they end up using.

Second, an application can select to use additional backup SRs for fault-tolerance, controlling their number, placement, and switch-over policy. It can also choose between pre-subscribing participants to the backup multicast channel for faster fail-over, or only setting up the backup channel when the primary one fails, saving on expected channel charging, options we refer to as "hot" and "cold" standby. In contrast, with network-selected rendezvous points, their location, degree of replication and criteria for fail-over are independent of these application considerations.

Finally, the SR can provide application-specific functionality beyond simply relaying data and transmitting notifications of new sources. For example, the SR can supply "floor control" when relaying data to the session, effectively acting as an intelligent "audience microphone", accepting unicast input from authorized audience members, assigning the floor to the next speaker, and then forwarding its traffic to this session. In particular, in a lecture, the SR can ensure that one question is transmitted to the audience at a time, that the answer immediately follows the question, and that no member disrupts the session with excessive questions. This application-specific access and content control is important for large-scale lectures and conferences, our particular interest.

As another example, the SR can add sequence numbers to relayed packets, as required in reliable multicast protocols [22, 16, 13]. The SR establishes this reliable communication with all receivers, allowing a secondary (relaying) source to take advantage of this shared reliable channel and avoid the cost of setting up its own. This secondary source simply needs a reliable unicast connection (*e.g.*, TCP) to the SR. In contrast, if the rendezvous point is provided at the Internet level where it is oblivious to these transport layer issues, each source needs to set up its own reliable multicast session with the attendant delays and costs, even though it may be forwarding through a central rendezvous point, just like with the session relay host.

4.3 Session Relaying as an ISP Service

An ISP can provide one or more well-positioned session relay servers as a value-added service for customers. The application then contracts for an SR channel for a given period of time, similar to the way that conventional satellite time is reserved or purchased or proprietary audio and video conferencing facilities are provided by the phone companies. Such a service would allow smaller-scale applications to use multicast without requiring the ISP to enable native multicast transmission from all customers' hosts.

This approach requires standardized relaying and access control protocols, an effort we are pursuing. For simple packet relaying, this functionality can be provided as an application-controlled operating system extension, avoiding the overhead of a transit through the application layer.

4.4 Other Applications

Going beyond almost single-source multicast applications, multi-source video conferencing or small multiplayer games can be implemented using either a separate channel for each source, or the SR approach if the extra latency is not an issue.

This approach is more competitive with existing multicast implementations than one might expect without some consideration. PIM-SM encounters the same delay-state tradeoff as EXPRESS, providing a choice between the higher delay of a shared multicast tree rooted at the rendezvous point and the extra state cost of source-specific trees. The key difference is that EXPRESS provides application control of the policy for switching between shared tree and source-specific tree, and for locating the shared tree root.

With CBT, the transmission through the core is similar in behavior and cost to relaying via the SR but without the application-level control. Moreover, there is no option of using a source-specific tree (short of setting up a new group) if the core introduces excessive delay. The use of a bi-directional shared tree can provide faster delivery to subscribers on the path from the sender to the SR or core and a slight reduction in bandwidth along this path, but neither of these considerations seem compelling in the situations and applications we have encountered.

For a multi-participant conference, the number of channels necessary is intrinsically small because it is simply not productive to have meetings with large numbers of active speakers or sources. Therefore, the setup and maintenance overhead of N channels is not significant and, with the millions of channels per host, there is no address-space consumption issue with using N channels.

Other multi-source applications can be similarly structured using a combination of channels, some logically shared and some not, depending on data and delay requirements. Using these techniques, EXPRESS appears applicable to all multicast applications we are aware of, except multicast discovery, which we believe is fundamentally not scalable Internet-wide.

4.5 Session Relay Cost/Performance

Using a session relay, session management is handled by a middleware class library that is reusable across multiple applications. (We are developing such middleware, with particular focus on large-scale conferencing.) Thus, application writers need not be burdened with a large development cost as a result of using session relaying and EXPRESS.

Applications also appear relatively easy to modify to match this model. For instance, RTCP [21], a session management protocol, is used by many existing applications to measure group reception quality and other session-wide attributes, and it depends on multi-sender multicast to limit the overall rate of RTCP traffic. Biersack and Nonnenmacher [19] describe how an RTCP-like protocol can be adapted to use single-sender multicast. Furthermore, many uses of RTCP, such as measuring group size and average loss rate, are readily implemented with the `CountQuery` mechanism. If desired, the SR can also perform application-specific summarization of reports to inform receivers of session-wide values (like loss rates) that are carried in RTCP.

A given network can add relay points as necessary to scale the "SR capacity" of an enterprise network and to prevent it from becoming a bottleneck. Each low-cost PC today is capable of forwarding data at a rate in excess of 100 Mbps, fast enough to serve dozens of compressed broadcast-quality video streams (3-6 Mbps) or thousands of CD-quality audio streams (100 Kbps) on one session relay, with more in the future, so this technique can be quite cost-effective.

For an application structured to use the basic SR model, the use of the channel model imposes no additional state cost over a state-efficient shared-tree implementation of multi-sender multicast, like CBT. The use of a hot standby SR/channel adds additional state (approximately twice as much), but this cost is small relative to the benefit for applications requiring fault tolerance, as argued in Section 5.

Finally, the maximum relayed delay from a sender to the most distant subscriber is at most twice the distance from the most distant subscriber to the session relay itself, assuming symmetric paths. Wide-area applications need to tolerate highly variable delays in the Internet, so adding relaying does not make the application's problem significantly harder. Moreover, if ISPs provide session relaying as a service, they can be situated so as to reduce delay and to maximize throughput, as discussed previously. Furthermore, the SR delay is only a problem for highly delay-sensitive applications that cannot tolerate the added delay of the unicast transit to the SR. However, it is questionable whether a shared-tree implementation can provide sufficiently low delay for these applications; if not, per-source channels or source-specific trees must be used anyway.

5 Cost and Scalability

This section estimates the router memory and CPU costs of EXPRESS routing. Using current memory and CPU prices, we show that these costs are small when compared to the value of a multicast service and when compared to the fixed operational costs of running a network service. These operational costs are largely independent of the multicast model provided, and they include the cost of personnel, physical space, long-haul fiber links, and uninterrupted 24-by-7 operation. Based on our analysis, it appears feasible for a router to support millions of multicast channels without extraordinary investment in processing power or memory.

The key cost and scalability issues for EXPRESS are:

1. cost of router FIB memory for channels,

2. cost of management-level router state,

3. cost of maintaining this state.

5.1 The Cost of Fast-path Router Memory

One FIB entry is created for each EXPRESS channel in each router on the distribution tree. The FIB entry specifies the set of outgoing network interfaces to which packets should be forwarded, and it must be consulted for every multicast packet. Because of this, FIB memory is generally the most expensive memory in a high-performance router.

An EXPRESS FIB entry can be represented in 12 bytes, as shown in Figure 5, assuming 32 interfaces per router. Using 4 nanosecond SRAMs that deliver about

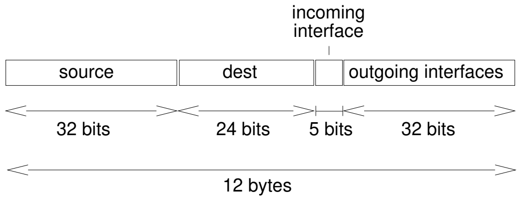

Figure 5: EXPRESS FIB entry format

100 million lookups per second, each 12 byte FIB entry uses 0.066 cents of memory (based on a price of \$55 per megabyte, as of early 1998 [17]).

Figure 6 presents a cost model that apportions the purchase cost of the FIB entries at a single router to the applications that use the FIB state. The model assigns costs in proportion to the session length. The $1/u$ term

m = FIB memory purchase cost per byte
e = bytes per FIB entry
t_s = session s duration
t_r = router lifetime
u = FIB utilization
p_{sr} = The FIB cost of session s at router r.

$$\text{Then } p_{sr} = me\frac{t_s}{t_r}\frac{1}{u}.$$

Figure 6: A cost model for FIB memory.

accounts for the fact that the FIB must, on average, have unused entries to accomodate the peak demand without running out of entries. The model assigns a fraction of the cost of the unused entries to each active session, proportional to the session's duration.

Consider a ten subscriber channel, as would arise in the multicast video conference described in the previous section. Assuming an average 25-hop path from source to each subscriber, the FIB memory cost of the channel would be approximately \$0.0075, less than eight cents for the whole conference. The derivation follows.

In the worst case, the channel multicast distribution tree has a "star topology" with no fanout in the network, except at the root. If each participant is h hops away from the source, then an n-receiver channel occupies at most nh total FIB entries, across the network. nh is an upper bound; the number of FIB entries will be lower if there is sharing in the multicast tree. A multi-sender application using k channels requires at most knh total FIB entries, for the duration of the application session. With p_{sr} as the per-entry cost, the total FIB memory cost of the session is given by c_s:

$$c_s \leq knhp_{sr} = knh\frac{met_s}{t_r u}.$$

Assuming a session duration of 20 minutes, a 1% average FIB utilization, a one year (31,536,000 seconds) router lifetime, a network diameter of 25, and the memory prices cited above, the FIB memory cost of the fully-meshed 10-way conference with 10 channels is:

$$c_s \leq 10 \times 10 \times 25 \times \frac{\$.00066 \times 12 \times 1200}{31536000 \times .01} = \$.075,$$

or less than eight cents for the FIB memory cost of the twenty minute conference, or about one cent per participant.

The costs per subscriber are even lower with large-scale applications. For instance, consider a long-running stock ticker application with 100,000 subscribers. If each receiver is twenty-five hops from the source, then the multicast tree contains approximately 200,000 links (assuming a fanout of 1 or 2 everywhere in the tree). According to the cost model above, the yearly memory cost of the FIB state for this application is still only $200000 \times \$.00066/.01 = \$18200$, or 0.18 cents per subscriber per *year*.

In comparison, a small community cable TV channel can lease for approximately \$1.00 per *potential* viewer per month and TV channels have been sold recently for \$25.00 per *potential* viewer. (*Potential* here means reachable as opposed to average actual number of viewers.)

Even with the conservative parameter estimates above, the FIB memory costs are small relative to the expected economic benefit of the applications supported.

This analysis suggests that the FIB memory costs do not justify the need to support shared multicast distribution trees, especially given that the shared tree approaches offer no state savings in the single-source or relay structure.

5.2 Cost of Management-Level State

ECMP also requires that the router maintain state for the process or management layer as part of maintaining the channel distribution tree and supporting count activity. The state required for each count activity is roughly 16 bytes, namely:

```
[channel, countId, count].
```

plus various implementation fields. If we further double this size to 32 bytes to allow for implementation fields, assume an average fan-out of 2 (so three records including the upstream record) and assume 2 counts outstanding at any time on a channel, the DRAM memory cost per channel is 192 bytes (this does not need to be fast-path memory as it is not part of the packet forwarding decision). Adding another eight bytes to store $K_{(S,E)}$, the total size is 200 bytes. At $1.00 per megabyte, each channel costs less than 1/50-th of a cent in incremental cost over the assumed one year lifetime of the router, making it a negligible cost, even if our cost model is off by several orders of magnitude. Again, our focus here is on the *incremental* costs of EXPRESS over shared tree approaches, not all the costs of the router or running the multicast service.

5.3 The Cost of State Maintenance

Maintaining the FIB requires sending, receiving, and processing `Count` and `CountQuery` messages. This cost is potentially significant with a very large number of EXPRESS channels.

Consider a router with one million active channels, where each channel's active lifetime is 20 minutes. Further assume that the average fanout of a channel is two (recall that a multicast tree 20 hops deep with a fanout of two has 2^{20} or one million members). In this scenario, the router receives four million Count messages every 20 minutes, and sends two million. This means processing 3,333 requests per second and generating half as many, for a total of approximately 5000 Count events per second. [7]

With TCP operation, it is not necessary to send a periodic refresh for long-lived channels – a single periodic per-connection keepalive detects TCP failures. This aspect allows the TCP-based protocol to efficiently support very large numbers of channels, as only one message is required to initiate subscription and one to end it, and per-channel timers are eliminated.

Without authentication, approximately 92 16-byte `Count` messages fit in a 1480-byte maximum-sized TCP segment on Ethernet. In our example, using ECMP over TCP, a router would receive 36 (3333/92) data segments, or 424 kilobits per second of control traffic, and send half as much.

We implemented TCP-based ECMP as a user-level process on a workstation and measured the costs of channel maintenance. In the measured scenario, the router had eight active Ethernet neighbors continuously sending subscribe and unsubscribe events. The core router processed approximately 4,500 incoming events per second (subscribe and unsubscribe), roughly corresponding to the event rate of the million-channel scenario above. Event processing at this rate used four percent of the CPU[8] on a 400 megahertz Pentium-II machine, or approximately 3500 cycles per event. In another run, a sustained rate of 33,000 events per second was reached using 43% of the CPU, or 5200 cycles per event. The per-event cost is thus approximately 5,000 cycles per event. (We suspect that the measured discrepancy in cycle counts is caused by increased cache pressure at the higher event rate.)

Using the free-running cycle counter in the Pentium-II CPU, we profiled the time spent in various parts of the code. The median event processing time was approximately 2700 cycles per subscribe and 3300 cycles per unsubscribe. This time includes a hashed lookup of the channel data structure, allocating a new channel data structure when needed, determining the physical interface of the request, computing the necessary FIB manipulation, looking up and sending a message to the next-hop upstream neighbor, and recording the unicast route used. Incoming and outgoing buffer management took about 995 more cycles per event, leaving about 1000 cycles per event unaccounted for in our measurements. We hypothesize that most of the missing time was spent on TCP processing in the operating system.

Our implementation simulated an RPF neighbor calculation of approximately 400 cycles, somewhat slower than the best published algorithms for software-based forwarding, *e.g.,* [23]. The actual FIB manipulation was not done in our measurements, and we anticipate such costs to be implementation-dependent. However, allowing for a 2000 cycle FIB manipulation penalty, the total CPU utilization would climb to only to six percent in the measured scenario.

Our experiments suggest that, with proper code structuring and protocol design, the CPU and communication cost required to maintain state for large numbers of multicast channels can be kept small relative to available processor power in modern CPUs. This experience and analysis argues against the need for complex protocol mechanisms to further reduce multicast state. The cost per channel is low and the overall cost to the ISP to support a large number of channels is also relatively modest and growing linearly with the number of channels.

6 Counting Overhead and Proactive Counting

The counting use of ECMP increases the message level proportionately with the rate at which it is used. With many large-scale multicast applications, counts are only meaningful as approximations over long time periods, like minutes. For instance, to charge for the transmission of a video over the Internet, one might look at the average number of subscribers over the 90 minutes or so of the movie, perhaps sampling the count every 5 or 10 minutes. For this use on important large-scale channels, the counting overhead is small and should not be prob-

[7] In this section, we analyze the expected costs in routers near the backbone of the Internet. Higher fanout might occur near the edges, but many fewer channels cross such an edge router because it serves many fewer clients. As a result, core or backbone routers are the demanding case.

[8] As reported by `time`.

lematic for the ISP or source. Excessive use of counting has the potential to significantly burden router CPUs, and so it may be necessary to limit or charge for its use.

For large-scale channels that have high economic value, a greater level of accuracy may be desired than what can be achieved efficiently by polling. In particular, with large, mostly-quiescent channels, the cost of periodically polling all routers can be high. In this case, the network layer can proactively maintain the count rather than requiring the source to continually poll it.[9]

We simulated an algorithm in which receivers and routers proactively send `Count` message upstream without requiring a `CountQuery` solicitation; we call this *proactive counting*.

A source can request that proactive counting be used for any `countId`, and this request is propagated to all routers in the multicast tree. In our simulations, the source additionally specifies two parameters, described below, that define an *error tolerance curve* as shown in Figure 7. A point on the error tolerance curve indicates

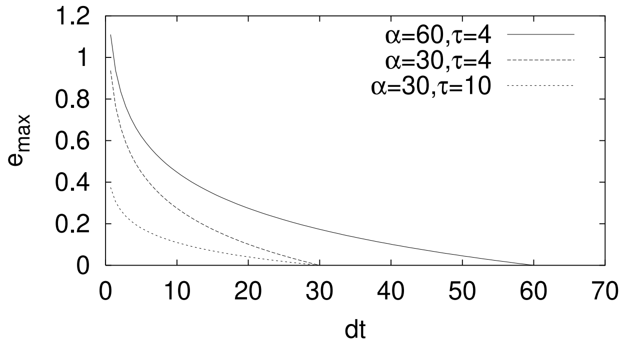

Figure 7: Error tolerance curves used in our proactive counting simulations.

the maximum error (the y-axis) that a router tolerates before sending a proactive `Count` message upstream, as a function of the time since the last `Count` was sent (the x-axis). At each node, the error is computed as

$$e_{rel} = \max\left(\frac{c_{adv}}{c_{cur}}, \frac{c_{cur}}{c_{adv}}\right).$$

c_{adv} is the last count advertised upstream and c_{cur} is the current sum of counts received from downstream.

We simulated curves of the form

$$e_{max} = \frac{-\log(dt/\tau)}{\alpha},$$

where e_{max} is the maximum tolerated error, and dt represents the time since the last update. τ controls the x-intercept – the maximum delay until any change is transmitted upstream. α controls the rate of decay without changing the maximum allowed error tolerance. This curve was chosen to allow fast convergence during periods of large change while using little bandwidth during periods of little change, and to allow the convergence/bandwidth tradeoff to be adjusted with few parameters.

Figure 8 shows a simulated short event with about

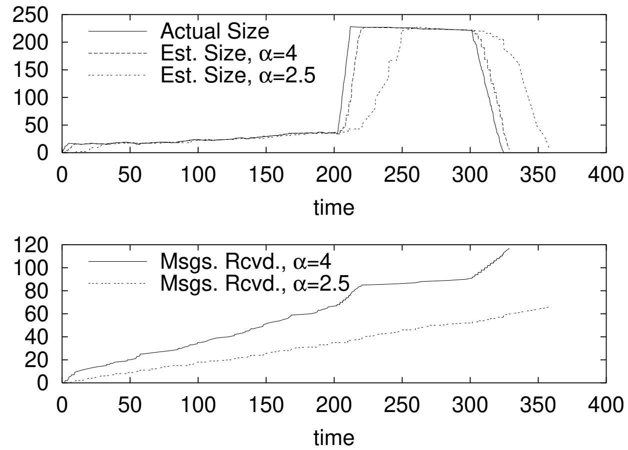

Figure 8: Error convergence and bandwidth used by proactive counting.

250 subscribers and a 3 minute duration. The scenario has an initial burst of subscriptions at time 0, followed by slow subscriptions until time 200, a burst of subscriptions at time 200, then no activity until time 300, when all hosts unsubscribe quickly.

The upper graph compares the actual group size to the estimated group size (c_{cur}), as measured at the root of the tree, for two different values of α. In both scenarios, τ was 120. The lower graph shows the number of `Count` messages delivered to the source for the two different values of α. A steep slope of the bandwidth curve indicates a high bandwidth utilization at that point in time, and a flat curve indicates no bandwidth utilization.

The figure illustrates the tradeoff between the accuracy of the count presented to the source and the number of messages needed to maintain it – a more accurate count is more costly. When $\alpha = 4$, the estimated size tracks the actual size very closely. When $\alpha = 2.5$, the estimated size lags behind the actual size after the large burst of subscriptions, but the total bandwidth used is approximately 2/3 that of the $\alpha = 4$ case. In practice, most applications are expected to tolerate some degree of error in exchange for reduced bandwidth usage.

With the current scheme, the convergence time of the algorithm grows approximately linearly with the depth of the tree. However, the depth of a tree grows logarithmically with the group size. Our experiments

[9] The standard tradeoffs between interrupt-driven and polling operation applies here.

to date indicate that this is a promising technique and that reasonable parameter choices give a useful level of accuracy at modest network cost.

7 Related Work

We consider three major areas of related work:

7.1 Service Models and Routing

The basic model of single-source multicast used here is not new. ECMP is similar in operation to the tree-building parts of CBT [2] and the source-specific joins of PIM-SM [9], both deriving from reverse-path forwarding [5]. Our contribution consists primarily of specifying this model as an extension of IP multicast, demonstrating its advantages for large-scale multicast applications, and adding the accounting support.

The Simple Multicast Routing Protocol (SMRP), developed as an extension to the AppleTalk protocol supports a similar model of multicast. The channel model is also somewhat similar to the multicast service provided by connection-oriented network protocols such as ST-II [24]. We are not aware of prior integration of this model into IPv4, however.

Support for authenticated subscriptions was proposed as far back as 1985, in one of the earliest descriptions of IP multicast (RFC 966) [8]. This mechanism was eliminated later [6], although other proposals have revived this idea, most notably Ballardie's work (RFC 1949) [1, 2]. RFC 1949 proposes a general security model encompassing joiner authentication, session key management, and per-host access controls. EXPRESS provides the subset of this functionality that is adequate for applications and easy to deploy and manage.

The *inclusion/exclusion lists* of IGMPv3 [4] allow a receiver to enumerate the set of sources that it wishes to hear from or exclude. They are far more general than the access control in EXPRESS, but this generality adds protocol complexity and their integration into multicast routing protocols remains incomplete.

BGMP [15], in combination with a global address allocation scheme like [11], improves the scalability of IP multicast routing, but it does not address the issues of access control, resource accountability and multicast management, although some of our techniques might be applicable to BGMP.

The session relay approach, including the switchover of a source from the shared relay to a direct channel is similar in some ways to the PIM-SM rendezvous point. However, session relays are selected and controlled at the application layer, not the network layer.

The subcasting capability of EXPRESS is similar to the `SUBTREE_MCAST` proposed by RMTP [16] except that with EXPRESS, only the channel source can subcast on a channel, preserving the single-source property.

The recently proposed Simple Multicast [20] carries a similar goal of simplifying multicast routing and improving the scalability of address allocation in the Internet. However, it supports multiple sources per multicast tree with bi-directional shared trees and appears to require changes to the multicast forwarding fast path. In the Simple model, a packet from a source other than the root of the shared tree travels to the root and is multicast out from there. This is similar to the operation of a session relay, but with Simple, subscribers on the source's branch of the tree receive a packet on its way up the tree rather than on its way down. We are not aware of an application where the resulting modest reduction in bandwidth, delay or router state is compelling, and we see the change to the fast path as an impediment to its adoption. To date, the Simple proposal does not address accounting or access control, two key elements of our work.

7.2 Accounting

RSVP [25] focuses on resource allocation, rather than resource accounting, as provided by EXPRESS. Herzog, Shenker, and Estrin's "Axiomatic Analysis" [12] comes closest to our work in the area of multicast accounting. They examine the theory and application of how the network can measure the per-link costs of a single-source group and fairly apportion them to the group's receivers. Our work is fundamentally different, however, in that we focus on how to bill the *source* for the cost of the multicast channel, rather than the receivers. We argue that billing the source is simpler for the ISP. The source can then choose to generate revenue at the application layer based on subscriptions, pay-per-view or advertising.

7.3 Counting

Pure application-layer algorithms for scalable counting in multicast groups have been studied [3, 10, 19]. [19] adapts these schemes for single-source multicast. They achieve scalability through probabilistic polling and either suppression or multiple rounds of polling. In contrast, EXPRESS provides router support for counting because it is a simple extension of the subscription protocol, service providers need counting support in the network itself rather than relying on the clients, and, for large channels, the ECMP approach avoids some risks inherent to application-layer counting.

Elaborating on this last point, the application-layer schemes depend on end node clients to compute and respond with the correct probability to avoid an implosion of responses. However, consider the Super Bowl multicast channel with 10 million subscribers again. With this large number of hosts, there is a risk of serious feedback implosion and congestion if the suppressing reply or replies from the source [19] is lost on any large branch of the tree or if misbehaving clients respond when they should not. With these problems, we conjecture that ISPs would not rely on these pure application-layer schemes and large-scale content providers would be reluctant to use them for fear of disrupting their channel. Multi-round schemes like [3] avoid the implosion risk, but are slower than suppression-based approaches. The ECMP approach also appears to provide similar accuracy with lower risk using relatively simple mechanisms, and we suggest that, for very large scale groups, the control message overhead is outweighed by these benefits. Of course, applications are still free to use the application-layer schemes in addition to the EXPRESS support.

8 Conclusions

EXPlicitly REquested Single-Source (EXPRESS) multicast channels are a straightforward extension to the conventional IP multicast model, supporting large-scale multicast applications, such as Internet TV, distance learning and wide-area multicast file updates. The single-source restriction dramatically simplifies implementation while facilitating important additional capabilities including access control, accounting and local-to-host multicast address allocation. While the single-source model of multicast is not new, this work makes several contributions to its use in the Internet.

First, we showed how EXPRESS channels can be provided as a simple modification to the IP multicast service model using a small portion of the class D address space, yet providing orders of magnitude more multicast channels per host than available in the group model. Moreover, channel addresses can be allocated by each host and without global coordination.

Second, we showed that EXPRESS channels can be implemented using a single simple protocol that requires no change to the current fast-path mechanisms in routers. The single-source restriction makes the combination of the existing unicast routing protocol and the subscription information adequate for building and maintaining the multicast distribution trees. In particular, this restriction eliminates the need for non-scalable broadcast-and-prune behavior, bi-directional shared trees, and network-layer maintenance of rendezvous points with complex transitions between shared and non-shared tree routing.

Third, we showed how the counting support in EXPRESS can be provided as a simple but powerful extension. Just as multicast supports efficient one-to-many communication, counting provides efficient many-to-one communication, useful to both the network provider and large-scale multicast applications.

Fourth, we showed how large-scale multicast applications that are *almost single-source*, such as conferences where audience members are allowed to speak, can be implemented on top of EXPRESS channels with a modest amount of middleware using application-selected and controlled session relay nodes. We also argue that session relays are a potentially attractive value-added service for ISPs to provide for smaller scale multicast uses such as multi-player games.

Finally, we argue that all scalable multicast applications that we are aware of can be implemented efficiently and robustly using the EXPRESS service model. Besides single-source and almost single-source applications, truly multi-source applications such as a video meeting can be implemented using an EXPRESS channel per source, leveraging the fact that the number of sources is limited by human dynamics. Our analysis of router state and state maintenance costs indicates that, with a careful implementation, the cost of the multiple channels is small for practical numbers of active sources. EXPRESS does not support multicast-based discovery and advertisement protocols except on the LAN, but these techniques are fundamentally not scalable to the wide area.

Given our belief that multicast is most compelling for large-scale applications, and that the group model appears difficult to implement Internet-wide, we believe there is a strong case that the channel model could largely supplant the group model. We plan on-going work to explore EXPRESS multicast further and to promote making it a standard Internet service.

Acknowledgments

This paper greatly benefitted from the comments of Christophe Diot, Dino Farinacci, Michael Greenwald, Craig Partridge, and Jonathan Stone. Special thanks to Steve Deering, who provided many valuable discussions, suggestions, and ideas. We also want to acknowledge the support of Jon Postel for IANA's allocation of a range of class D address space for single-source multicast shortly before his untimely death.

References

[1] BALLARDIE, A. Scalable multicast key distribution. In *Internet Requests for Comments (RFC1949)* (May 1996), no. 1949.

[2] BALLARDIE, A. Core based trees (CBT) multicast routing architecture. In *Internet Requests for Comments* (Sept. 1997), no. 2201.

[3] BOLOT, J.-C., TURLETTI, T., AND WAKEMAN, I. Scalable feedback control for multicast video distribution in the internet. In *Proceedings of SIGCOMM 1994* (London, England, Aug. 1994), ACM SIGCOMM, pp. 139–146.

[4] CAIN, B., DEERING, S., AND THYAGARAJAN, A. Internet group management protocol, version 3. In *IETF Internet Drafts (Work in Progress)* (February 1999), IETF. draft-ietf-idmr-igmp-v3-01.txt.

[5] Dalal, Y., and Metcalfe, R. Reverse path forwarding of broadcast packets. *Communications of the ACM 21*, 12 (Dec 1978), 1040–1048.

[6] Deering, S. Host extensions for IP multicasting. In *Internet Requests for Comments (RFC1112)*. Aug 1989.

[7] Deering, S. *Multicast Routing in a Datagram Internetwork.* PhD thesis, Stanford University, 1991.

[8] Deering, S., and Cheriton, D. Host groups: A multicast extension to the internet protocol. In *Internet Requests for Comments (RFC966)*. Dec 1985.

[9] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and Wei., L. Protocol independent multicast-sparse mode (PIM-SM): Protocol specification. In *Internet Requests for Comments (RFC 2117)* (June 1997).

[10] Friedman, T., and Towsley, D. Multicast session membership size estimation. Tech. rep., UMass Amherst Computer Science Department, 1998.

[11] Handley, M. Session directories and internet multicast address allocation. In *Proceedings of SIGCOMM 1998* (Vancouver, BC, Aug. 1998), ACM SIGCOMM.

[12] Herzog, S., Shenker, S., and Estrin, D. Sharing the "cost" of multicast trees: An axiomatic analysis. In *Proceedings of SIGCOMM 1995* (Cambridge, MA, Aug. 1995), ACM SIGCOMM, pp. 315–327.

[13] Holbrook, H., Singhal, S., and Cheriton, D. Log-based receiver-reliable multicast for distributed interactive simulation. In *Proceedings of SIGCOMM 1995* (Cambridge, MA, Aug. 1995), ACM SIGCOMM, pp. 328–341.

[14] Internet Assigned Numbers Authority. Internet multicast addresses. http://www.isi.edu/in-notes/iana/assignments/multicast-addresses.

[15] Kumar, S., Radoslavov, P., Thaler, D., Alaettinoglu, C., Estrin, D., and Handley, M. The MASC/BGMP architecture for inter-domain multicast routing. In *Proceedings of SIGCOMM 1998* (Vancouver, BC, Aug. 1998), ACM SIGCOMM.

[16] Lin, J., and Paul, S. A reliable multicast transport protocol. In *Proceedings of IEEE Infocom, 1996* (1996), IEEE Infocom.

[17] Lincoln, J. Market-driven price quote on 4Mbit BurstRam. personal communication, Feb 1998. Motorola FSRAM Marketing division.

[18] Mittra, S. Iolus: A framework for scalable secure multicasting. In *Proceedings of SIGCOMM 1997* (Cannes, France, Aug. 1997), ACM SIGCOMM.

[19] Nonnenmacher, J., and Biersack, E. W. Scalable feedback for large groups. *IEEE/ACM Transactions on Networking* (1999). To be published.

[20] Perlman, R., Lee, C.-Y., Ballardie, A., Crowcroft, J., Wang, Z., Maufer, T., Diot, C., and Green, M. Simple multicast: A design for simple, low-overhead multicast. In *IETF Internet Drafts (Work in Progress)* (February 1999), IETF. draft-perlman-simple-multicast-02.txt.

[21] Schulzrinne, H., Casner, S., Frederick, R., and Jacobson, V. Rtp: A transport protocol for real-time applications. In *Internet Requests for Comments (RFC1889)*. Jan. 1996.

[22] Speakman, T., Farinacci, D., Lin, S., and Tweedly, A. PGM reliable transport protocol specification. In *IETF Internet Drafts (Work in Progress)* (Aug. 1998), IETF. draft-speakman-pgm-spec-02.txt.

[23] Srinivasan, V., and Varghese, G. Fast address lookups using controlled prefix expansion. *ACM Transactions on Computer Systems 17*, 1 (Feb 1999), 1–40.

[24] Topolcic, C., et al. Experimental Internet stream protocol, version 2 (ST-II). In *Internet Requests for Comments (RFC 1190)* (Oct. 1990).

[25] Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D. RSVP: A new resource reservation protocol. *IEEE Networks Magazine* (September 1993).