The Breadcrumb Forwarding Service: A Synthesis of PGM and EXPRESS to Improve and Simplify Global IP Multicast

Koichi Yano*
University of California, Berkeley
yano@cs.berkeley.edu

Steven McCanne
University of California, Berkeley
mccanne@cs.berkeley.edu

ABSTRACT

If ubiquitously deployed, IP Multicast promises to provide an efficient datagram service for an arbitrary sending host to reach an arbitrary and dynamic set of destination hosts anywhere in the Internet. Unfortunately, two very difficult problems —interdomain multicast routing and viable end-to-end multicast transport— have yet to be solved and deployed satisfactorily.

This paper proposes that two existing but independent network mechanisms— the EXPRESS service model and the network component of the Pragmatic Multicast protocol (PGM)— be synthesized in a scheme we call the *Breadcrumb Forwarding Service* (BCFS) to simultaneously tackle the problems of interdomain multicast routing and end-to-end reliable multicast. Like EXPRESS, BCFS utilizes explicit-source group join and like PGM, enhances the network forwarding architecture with finer-grained group control. In this paper, we detail BCFS service model and router mechanisms to support the service. To demonstrate the flexibility and efficiency of BCFS, we describe the application examples built on this service model, which can accommodate not only PGM and also a novel reliable multicast transport protocol.

1. INTRODUCTION

The cornerstone of the Internet's resounding success is arguably the end-to-end design principle [27], which says that a given system function should operate at the lowest communication layer in which it can be wholly and correctly realized. When applied to network design, an end-to-end philosophy naturally leads to an architecture where few constraints are placed upon the network itself — e.g., the network can drop, delay, replicate, and corrupt packets — and richer services like reliable, sequenced delivery are defined and implemented at the edge of the network in an "end to end" fashion. In the Internet architecture, the IP network layer offers a *best effort* delivery service and richer transport services like TCP are built on this best-effort IP service.

This split between a deliberately simple network layer and a rich transport layer naturally leads to a robust and scalable system. Because so few presumptions are placed on the network, not only is such a network relatively easy to engineer and deploy at large scale, but end-to-end protocol and application designers must conscientiously account for the indeterminacies of the underlying network. In effect, the best-effort service model calibrates the designer's expectations for an environment like the Internet, where consistent, homogeneous, and high-performance communication is often the exception rather than the rule. Thus, the end-system software that results tends to be *robust*, and as a consequence, the Internet as a whole has continued to scale gracefully despite an onslaught of new and evolving constituent technologies that are decentrally managed, heterogeneous, and imbued with mixed levels of reliability.

*On leave from Information Technology Lab., Canon Inc..

Quite naturally, then, when Deering proposed IP Multicast [8] — an enhancement of the traditional Internet architecture for efficient multipoint packet delivery — he very deliberately appealed to the end-to-end design principle. Like unicast, Deering's multicast service model is best effort and richer services like reliability must be implemented in the end-hosts. Unfortunately, whereas the end-to-end approach has enjoyed tremendous success as the bedrock of the unicast Internet, its adaptation to the multicast has created two very difficult design problems that have yet to be satisfactorily solved: First, because multicast routing is so more complex than its unicast counterpart, a viable interdomain multicast routing protocol has yet to be developed; and second, transport services like reliable multicast are confounded by the best-effort network model where packet drops can impact indeterminate subsets of the receiver group. Despite more than a decade's worth of research, a viable interdomain multicast routing has yet to materialize and a reliable multicast transport protocol that offers congestion control and robust and scalable behavior remains a research problem.

The failure of many and varied research efforts to bear truly viable end-to-end multicast transport protocols [10, 33, 17] or truly viable wide-area, interdomain multicast routing protocols [9, 1, 15] brings into question whether the proposed multicast service model is in fact the appropriate core building block. None of the proposals for reliable multicast have satisfied the requirements for "safe" deployment on the public Internet [19], e.g., is scalable, robust, congestion controlled, accommodates heterogeneity, and so forth. Nor have the routing protocols provided the degree of control, stability, flexibility, robustness, and scalability required by service providers to deploy them flexibly in the complex peering relationships that will be required for universal deployment.

Reacting to this mixed success, several researchers have proposed alternative multicast service models [11, 23]. The EXPRESS service model [11], for example, simplifies multicast service model through advocating a model where a multicast tree is rooted at a single source and receivers explicitly indicate that source when subscribing to a multicast channel. In contrast, pragmatic multicast (PGM) [29] and generic router assist (GRA) [5] attempt to tame the complexity of multicast transport by moving richer abstractions into the routing layer. This paper posits that perhaps the best of both worlds could be had by both simplifying the multicast service model with an explicit-sender constraint and by simultaneously augmenting the router's core mechanisms with the minimal functionality necessary to accomplish a wide range of scalable transport abstractions.

Instead of introducing transport-aware mechanisms in the network layer as suggested by PGM and GRA, we believe these services are so fundamental to a successful multicast service that the essence of these transport mechanisms should be integrated directly into the network-layer. At the same time, the merits of single-source multicast should be adopted to tame the complexity of multicast routing. To do this, a network multicast layer can be developed that is based on an explicit-source subscription model that is optimized with fast-join/leave and a label priority structure that can be effectively exploited by transport abstractions to build scalable end-to-end multicast protocols. Our service model, which we call the *Breadcrumb Forwarding Service* (BCFS), is a straightforward synthesis of the EXPRESS service model and the network-layer piece of PGM. The novelty here is not so much the definition of this new multicast service abstraction, but rather the supposition that this combination could provide adequate richness for the development of scalable end-to-end multicast transports, while simultaneously retaining enough simplicity to allow the deployment of such a service in the global Internet. Though we obviously have not demonstrated this claim definitively herein, we believe that the approach is promising and warrants further investigations based on the preliminary results and proofs of concept developed thus far.

To support our hypothesis, we outline a novel reliable multicast transport protocol for bulk data transfer, which we call *Rainbow*. Rather than present complete details of Rainbow, which are published elsewhere [32], we simply sketch the overall framework and describe how Rainbow builds on BCFS lending evidence to this paper's thesis. Unlike most all other reliable multicasts including PGM, which transmit data using the normal IP multicast delivery service then recover from losses using some other mechanisms, our protocol is built exclusively upon the BCFS service model, demonstrating the viability of the service model as a generic replacement for IP multicast. Also unlike previous works, Rainbow includes a congestion control algorithm that is modeled after that in TCP and thus provides a viable solution for congestion-controlled reliable multicast. In this approach, each receiver maintains its own congestion window and runs slow-start and congestion avoidance [13] individually by driving the equivalent of the TCP "ack clock" with breadcrumb requests. To enhance Rainbow's scalability and support asynchronous receiver subscriptions, Rainbow utilizes a Digital Fountain [4] at the source to temporally decorrelate what data to send from when it must be sent. This approach allows receivers to exercise asynchronous and autonomous behavior while simultaneously enjoying the performance benefit of synchronous multicast communication.

Our core contribution lies not in the particular protocol described herein, which we continue to investigate and refine, but rather in the overall architecture and general direction of the approach. We readily admit that several practical engineering issues and details for how BCFS would be implemented are omitted from this first generation design. We have not implemented this in a router, nor have details of state maintenance at a routers and requirement of memory and processor been discussed yet. Justification of Rainbow's efficiency also remains future work. In a nutshell, BCFS brings together several novel protocol idioms described elsewhere into a new framework for multicast communication that not only represents a viable method for deployment — as evidence by Cisco's efforts in developing and deploying PGM — but provides a service that can be successfully used across a wide variety of multicast applications and protocols. In addition to supporting Rainbow, BCFS can effectively support a variant of PGM and, as described later, provides a superset of the EXPRESS multicast service model and thus shares many of its attractive properties. While we believe this approach shows promise, we are not arguing that it is the definitive answer to multicast routing and transport; rather, we acknowledge the difficulty and richness of these problems and simply claim that the mechanisms described herein form the kernel of a number of interesting research investigations to be conducted as future work.

In the remainder of this paper, we discuss previous works in the next section. In section 3, we detail the BCFS service model and router mechanisms to support the model. In section 4, several application examples including Rainbow are described. We conclude the paper in section 5.

2. PREVIOUS WORK

A fundamental challenge inherent in the original IP Multicast service is the level of indirection afforded by the group addressing model, which required routers anywhere in the Internet to build a tree that interconnected members of a group anywhere in the Internet. To avoid this complexity and allow the multicast system to reuse the unicast routing system, the EXPRESS service model abandons the anonymity of the class D group address [11]. Instead, EXPRESS advocates a model where a multicast tree is rooted at a single source and receivers explicitly indicate that source when subscribing to a multicast channel. Similarly, the *simple multicast* architecture [23] proposes that the group addressing architecture be extended to explicitly embed the IP address of a "rendezvous point" in the multicast group. In either model, the normal unicast routing infrastructure can be used to route multicast data and/or control traffic since unicast addresses are explicit in the revised addressing architecture. The premise is that this approach simplifies the so-called *rendezvous problem*, and because of several other attractive properties, purportedly induces a more viable multicast architecture.

Whereas EXPRESS and simple multicast re-examine the multicast routing architecture from first principles, other work advocates the strategic placement of intelligence within the network infrastructure to solve multicast transport problems. For example, RMTP proposes that *designated receivers* be placed within the network infrastructure to carry out localized retransmissions using subtree multicasts to enhance scalability [18]; LRMP proposes the deployment of logging receivers that provide a similar function [12]; and the Reliable Multicast proXy (RMX) architecture [6] relies on proxy agents within the network to carry out format and protocol conversion to accommodate network heterogeneity and effect congestion control.

Rather than rely on service node deployment within and across the network, other works address the problem of how one might jointly optimize the design of a new multicast transport service with complementary end-to-end transport protocols, thereby retaining many of the merits of the end-to-end approach. The Lightweight Multicast Services (LMS) architecture [22] pioneered this basic approach. In LMS, multicast routers conspire to arrange the receivers into a tree-based hierarchy that is congruent with the underlying network topology. This hierarchy is exposed to the end clients through a service model extension that allows a host to send a packet to its logical parent in the tree. This extension in turn enables an end-to-end multicast transport protocol that implicitly exploits the network topology to optimize its performance. Similarly, the AIM architecture [16] provides a rich addressing structure that from within a single framework offers many different forwarding services, e.g., subtree multicast, standard group multicast, anycast, and so forth. On top of AIM, several multicast transports have been fashioned, including a reliable multicast transport. Finally, the randomcast forwarding service was proposed as an alternative to LMS to enhance robustness by breaking the hierarchy of parent/child relationships with randomized forwarding thus eliminating single points of failure. Reliable multicast protocols can then be layered on top of the randomcast forwarding service, e.g., Search Party and Rumor Mill [7].

A similar, though less modular, approach has been undertaken in the PraGmatic Multicast (PGM) protocol [29]. Here, routers are enhanced with transport-level knowledge and an end-to-end protocol is built on top of this transport-aware network infrastructure yielding a monolithic solution for reliable multicast loss recovery. In PGM, receivers generate NACKs to repair missing data, and PGM-aware routers coalesce NACKs by maintaining per-packet sequencing state in the routers. This state, in effect, forms a "trail of breadcrumbs" from the receivers missing a given piece of data back to the source of that data. When the source receives notification of new breadcrumb state, it generates a retransmission that in turn follows the breadcrumbs to each requesting receiver and simultaneously tears down the breadcrumb state that represents that retransmission request. This loss recovery scheme is optimal in the sense that retransmissions are sent to only those receivers that are in need of that data.

Unfortunately, in PGM, the network service is not clearly defined as a separable and reusable component network forwarding service. Instead the network component is a PGM-specific router optimization rather than a general extension to the multicast forwarding service. We believe that this codependence between the network and transport layers is unnecessary. In fact, the PGM router-assist component can instead be cast as an explicit-source multicast service that is optimized for fast group establishment and teardown. In this interpretation, PGM NACKs represent explicit-source group joins, as in the EXPRESS service model, and the source-generated PGM retransmissions represent special data packets that simultaneously induce group teardown. Thus, we can recast PGM as a EXPRESS-like Breadcrumb Forwarding Service (BCFS) and an end-to-end transport protocol layered on top of BCFS, where PGM is but one of many possible transport protocols. When viewed in this light, each PGM NACK request corresponds to a multicast subscription interaction in a framework in which multiple multicast groups provide multicast loss recovery [14].

3. SERVICE MODEL AND ROUTER MECHANISM

3.1 Service Model

The BCFS unifies the EXPRESS service model and the network component of PGM into a single, flexible multicast service model. To this end, BCFS provides a single-source, request-based multicast service, where groups can be efficiently set up and torn down in tandem with a data exchange, loosely analogous to how T/TCP optimizes the establishment and teardown of a TCP connection in a single response/reply dialogue [3]. BCFS is thus optimized for ephemeral groups that come and go rapidly and is consequently well-suited as a building block for multi-group reliable multicast schemes.

To avoid unnecessary transport-level dependence, BCFS uses an abstract "label" to identify a particular request with respect to some source. The source/label pair (S,L) thus induces an group-oriented address architecture that is precisely analogous to the source/channel (S,E) framework proposed in EXPRESS. BCFS differs from EXPRESS, however, in that messages are sent from the receivers toward the source along the multicast tree and are suppressed if a message with the same label has already been sent up the tree. The group membership protocol is exposed to and run at the application layer, and arbitrary messages can be piggy-backed onto these control messages.

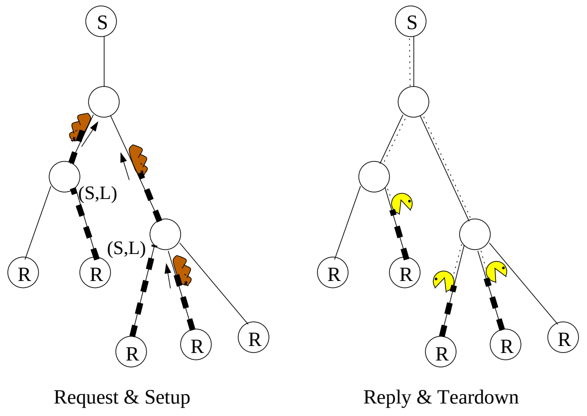

Figure 1: Basic Service Model by BCFS.

Figure 1 illustrates this breadcrumb forwarding model. Request messages for some piece of data drop breadcrumbs along the path to a source: the breadcrumbs, in turn, guide the reply message from the source back to all requesting receivers. Each breadcrumb is identified by an (S,L) pair to differentiate the forwarding paths for all labels in use.

To enhance the range of transport services that can be built on top of BCFS, the forwarding model includes a level-numbering scheme for selectively tearing down the breadcrumb state. Each breadcrumb carries with it a level number and each request and response includes a level number in the header of the packet. A request packet is propagated up the tree toward the sender only if its level number exceeds the level number in the breadcrumb stored at the router (or if no such breadcrumb exists). Similarly, breadcrumb state is torn down by a response packet only if the level number is equal to or exceeds the breadcrumb level stored in the router. The straightforward extension from PGM is tearing down the state when the count of packets forwarded becomes equivalent to the requested number. We adopt a leveling mechanism that would have more flexibility than a counting mechanism since it can have the concept of a higher level request and can tear down forwarding state by one higher level packet. It needs more study to show the usability of the mechanism and investigation of alternative mechanism for controlling forwarding state from transport protocols.

An application interacts with BCFS through the prototype interface defined in Figure 2. *bcf_bind* allows an application to register its interest in receiving breadcrumb packets sent to a particular label or set of labels using an address/mask pair. If multiple processes match a given label, a copy of the message is delivered to each process.

A request packet is sent via *bcf_request*, which includes the address of the source or tree root (i.e., the destination of the message), a label, a level number, and an optional message body. The message field in a request might include transport protocol payload like sequence numbers of requested packets.

A reply message is sent by *bcf_reply*. The reply contains a label

```
bcf_bind(lab, mask);
bcf_request(src,lab,lev,msg);
bcf_reply(lab,lev,msg );
bcf_recv(src,lab,lev,msg);
```

src:	data source address (destination of request).
lab:	BCF label.
lev:	level number.
msg:	transport message.

Figure 2: BCFS API for transport protocol

which is copied from the corresponding request packet. The level field contains the level to be torn down.

In summary, the sequence of events for effecting the BCFS forwarding service are roughly as follows:

(1) Request: A receiver sends a request packet with a label and level.

(2) Setup: A router that receives a request message maintains state for forwarding links and the level associated with the label.

(3) Suppression: The router forwards the request message toward the source if the label in the request message is new for the that router or the level number is larger than the highest level being maintained. Otherwise, the message is not forwarded.

(4) Reply: A source, in response to the request message, sends the requested data together with the label embedded in the request message and a level number to be torn down.

(5) Forwarding: A router directs a reply message to the links that are associated with the label.

(6) Teardown: The router removes the forwarding state of the link associated with the label, if the reply message includes a level number that is larger than the level maintained by the router.

3.2 Router Behavior

We call a router that supports BCFS a *BCF Router*. A BCF Router maintains forwarding information associated with labels (*BCF Label*). End hosts exchange *BCF Messages*, which consist of *BCF Request* from receivers and *BCF Reply* from the source.

To effect BCFS, routers maintain "breadcrumb state" tied to a particular label, but they do not store a copy of the message. The breadcrumb also includes level number that corresponds to the highest request level for that label seen so far.

If a router receives a request message for a label that is already stored in the router and the level number is less than or equal to the level number stored in the breadcrumb, then the message dropped, much as DVMRP [8], EXPRESS [11], and PIM [9] join messages are coalesced in a multicast distribution tree. Otherwise, the message is propagated up the tree toward the source S using the normal unicast routing tables (i.e., along the reverse-path multicast route back to S). Thus, as in PGM, routers can fuse requests by suppressing label messages if the state has already been established.

If the message makes it all the way to the source subnet, the router incident to the source delivers it to that source and an application (which has presumably bound itself to the label in question via *bcf_bind()*) receives the message. Because suppression is carried out on a per-label basis, if different receivers send messages on the same label at the same time, only one message will be delivered to the source.

Upon receiving a request, the source may respond with an arbitrary message tied to the label in question. When a source sends a "reply" packet bound to a particular label, the routers forward the packet along all links that have breadcrumb state tied that label. As a side effect of forwarding the packet, the breadcrumb state is deleted, which allows future messages to be propagated back up the tree and frees up router resources.

Unlike PGM NACKs, labeled request messages are not sent in a hop-wise reliable fashion, which means that only breadcrumb state needs to be maintained in the router, not the entire message body. However, this also means that a lost request message that never makes it to the source suppresses further messages sent for that label. To avoid this, label state is refreshed in a soft-state fashion [24, 28]. To this end, when a router suppresses the propagation of a label because of existing breadcrumb state, it verifies that the breadcrumb state has been recently "refreshed", e.g., according to the scalable session messages algorithm in [28]. If the label needs to be refreshed, a null message for that label is sent toward the source. Thus, if the source receives a specially marked null message, it knows the original request message was dropped somewhere in the network and can invoke a higher-layer recovery process if necessary. By using data-driven state updates, the router need not manage timers to otherwise trigger soft-state updates.

To complement the soft-state update process, breadcrumb entries may be deleted by the router if no update is received after a certain time interval. Yet unlike the normal multicast group management machinery, tearing down this state is not critical because updates are data driven from the receivers and generated only if a receiver is explicitly present. Thus, the pool of breadcrumbs could either be timed out by a soft-state aging process or entries could simply be reallocated using LRU replacement. Similar time-out mechanism is used for repair state in PGM because retransmission packets are sent in the unreliable way. Thus time-out and deletion of the state allows receivers to send the same request (NACK) again without suppression in the face of retransmission lost.

As defined, BCFS is a superset of and can implement EXPRESS. To do so, each receiver periodically generates a null request message addressed to some channel (S,L) with level number 1, and the source sends data packets as responses to label L with level number 0. Thus, the breadcrumb state is maintained exactly as if it were an EXPRESS channel and packets are sent best-effort to every receiver in the source-specific group identified by the label.

3.2.1 Setup and Request Suppression

A multicast channel is set up when a request packet arrives at a router by updating state in the router.

If a label (S,L) in a request message is new for a router, the router appends a new entry for the label and maintains the link identifier for the link that the message came from as a directed link. A level number is maintained coupled with each forwarding link. The request message is then forwarded toward a source.

On the other hand, if the router's state already has a entry for (S,L), the router explores the list of forwarding links. If the link is not in the list, the link identifier is added to the list of directed links tied to the label and a level number is also maintained for the link. If the link is already designated as a directed link but the request has higher level than that in the router's state for the link, the

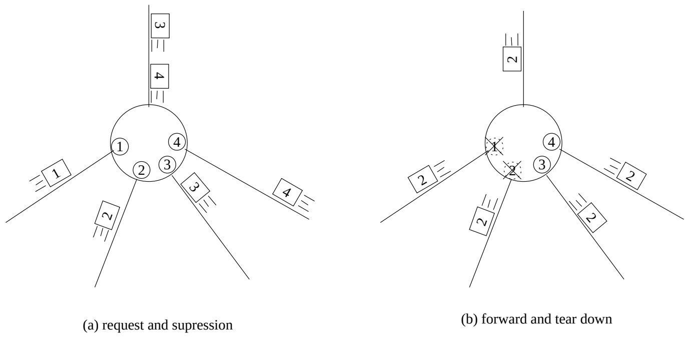

Figure 3: Setup and Teardown: the numbers show levels of packets and state in the router

level number is updated. Only in the case where the level number is larger than any level numbers of forwarding links tied to (S,L) is the message forwarded to the up-link. Otherwise, the request message is suppressed.

Figure 3 (a) shows an example of how requests with different levels are forwarded (or suppressed) by a router and which state remains in it. In Figure 3, we suppose all requests have the same label (S,L) but different levels that the numbers show. At first, when the request message with level 3 arrives at the router, the level 3 state tied with the (S,L) label is maintained and the request is forwarded toward the source. After the state is set up, the requests with level 1 and 2 arrive but these are suppressed because the levels are lower than the maximum level maintained in the router. However, forwarding state with level 1 and 2 is retained for each respective link. Finally in this example, the request with level 4 arrives at the router and is forwarded toward the source since it has a higher level than the maximum level that the router maintains.

3.2.2 Teardown and Forwarding

A forwarding path is torn down by deleting links from the list of directed links in a router's state when a reply packet arrives at the router or a timer expires.

When a BCF reply message arrives at a router, the BCF reply message is forwarded to the links listed as directed links tied to (S,L). Furthermore the router compare the level in the message with the level of each directed link. If the level number in the reply is equal to or larger than the level number for a link, the forwarding state of the link is deleted. Otherwise, the link is retained as a directed link.

Figure 3 (b) shows an example of forwarding packets and tearing down the state. In this example the router maintains different levels from 1 to 4 for each link and the message from the source with level 2 arrives at the router. Because all the links have forwarding state tied with the label of the arrival packet, the packet is forwarded to all the down links. Only the states that is equal to or less than 2 are deleted and as a result the forwarding states for the two links which are leveled more than 2 are still maintained.

4. APPLICATION EXAMPLES

To illustrate the power and flexibility of BCFS as an independent and reusable network service, we explain general deployment of the concept of label and level from applications or transport protocols. Then we describe some application examples, which range from PGM to a different type of multicast transport protocol.

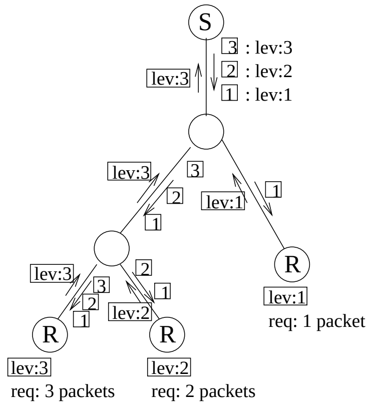

Figure 4: Request by the same label with different levels.: each receiver receives different number of packets

4.1 Label and Level Use

How applications and protocols generate labels depends on how the protocol designer wishes to differentiate and aggregate higher-layer messages. For example, in PGM, the transport sequence number could be hashed with a port number or some other application specific identifier to produce a label for a retransmission request. Of course, hashing can result in address conflicts so applications must be prepared to deal with superfluous data coming from other, unrelated applications or sessions. But, if the label space is large enough and the label generation functions are chosen well, then the probability of collision will remain low and not adversely impact protocol performance. Moreover, a separate, independent label space exists for each source, so the impact of collisions is quite limited. This contrasts with the existing IP multicast service model where any host in the network can send arbitrary data to an arbitrary group and collisions are prone to happen especially in the absence of a globally consistent multicast address allocation scheme (which remains a difficult research problem).

The level numbering scheme controls request forwarding and tear-down timing. For example, Figure 4 shows how a level number can be used to request a specific number of packets on some label in a receiver-specific fashion. Here, a request message for N packets uses a level number of N, where N can vary among the different receivers, say $N_1 \dots N_R$. Let $M = \max_{k=1\dots R} N_k$. The request with the largest level M then reaches the source without suppression, and as a result the source learns M. The source then generates M packets with level numbers $1 \dots M$, and each receiver receives exactly the number of packets requested.

4.2 PGM-like Multicast over BCFS

A PGM-like reliable multicast transport can be naturally adapted to BCFS, because BCFS has an aspect of generalized PGM. We briefly describe how a PGM-like reliable multicast can be realized over BCFS. In this approach, original data packets are sent to the entire multicast group. When a receiver detects lost packets, a NACK is sent via a BCF request message with a label generated by hashing sequence number of the lost packet into the low-bits of a label, with some well-known upper prefix (perhaps selected as a hash of a session-specific identifier). In response to the NACK, the source retransmits the lost data via a BCF reply message with the same label. In turn, the BCF routers forward the reply packet to precisely those receivers that earlier sent a request.

The label generated by transport sequence numbers can prevent conflicts between different NACKs. Even if the label space is smaller than the sequence number space, the label with sequential order can avoid conflict because a source is expected to send data packets sequentially. In PGM, the size of transmit window is large enough for label space to avoid conflict because only data packets within the transmit window are supposed to be provided for loss recovery.

4.3 FEC-based Loss Recovery

The level-numbering scheme shown in Figure 4 interacts nicely with previously proposed schemes for FEC-based loss recovery [21]. Each receiver generates a NACK for a block of packets using BCF request with a level indicating how many packets were omitted from the block (i.e. corresponding to the N_k's in section 4.1). Then the source would generate as many parity packets as the maximum of lost packet number among receivers (i.e. corresponding to M in section 4.1). The levels of recovery packets for the same block increase by one for each packet (i.e. $1 \ldots M$).

These packets can recover the lost packets for each receiver, no matter which packets were lost, and each receiver can receive the exactly same number of recovery packets as necessary. Details of this mechanism as applied to the traditional multicast service model are described at length in [21].

4.4 Rainbow on Digital Fountain

Not only can a PGM-like transport be built on top of BCFS, but because BCFS is a generic network service, other transport protocols can exploit it as well. In this section, we describe a reliable multicast transport that differs quite substantially from PGM even though it is built upon precisely the same network service. In particular, our protocol exhibits a viable solution to one of the hardest problems in reliable multicast, namely congestion control.

Multicast congestion control is greatly confounded by heterogeneity amongst receivers in a group: if using only a single multicast group, a single, uniform sending rate cannot satisfy the conflicting requirement of a diverse set of receivers attached to the network at different bit rates. That is, a congestion control strategy must force the sender to transmit data according to the most constrained receiver [2, 25]. This solution is inherently unsatisfying for large-scale deployment in heterogeneous environments. Alternatively, the source can send to multiple multicast groups allowing receivers to individually adjust their reception rate by joining and leaving multicast groups [20, 31, 26]. Unfortunately, the granularity of the layers limits the degree of adaptation and the design of a control law that can manage receiver membership in a scalable and robust fashion is a hard problem that has not been satisfactorily solved.

To address these problems, we propose a reliable multicast transport based on BCFS, called *Rainbow*, (ReliAble multicast by INdividual Bandwidth adaptation using windOW), which includes a congestion control scheme. Rainbow is designed to accomplish the following:

- A receiver receives data at its available rate as if there were a unicast TCP connection between a source and a receiver (i.e., the protocol dynamics are "TCP friendly").
- The bottleneck link or links in a multicast distribution tree are efficiently shared by data aggregation among many receivers.
- The source need not manage state on a per-receiver basis, which would otherwise limit the protocol's scalability.

4.4.1 Digital Fountain

The Rainbow congestion control scheme utilizes a Digital Fountain [4] on top of BCFS. To establish the context for Rainbow, we first outline the Digital Fountain abstraction.

A Digital Fountain provides a robust mechanism for "implicit" multicast loss recovery as it requires no feedback from the source. Here, a sender simply multicasts a stream of data packets that are generated by the fountain as a function of a fixed input (e.g., a file). A receiver tunes in at any point and gathers up some fixed number of packets. Once this critical number of packets is attained, the receiver leaves the group and decodes the file from the packets.

A key property of the digital fountain is that (almost) any subset of packets may be used to decode thereby alleviating any need for feedback from the receivers to the source. From the perspective of a sender's load, the scheme is extremely efficient because a sender simply transmits packets without involving any sort of loss recovery scheme. Moreover, heterogeneity can potentially be tamed since the fountain can stripe packets across multiple rates and receivers can adjust their reception rate using multiple groups as described above. Though a scheme based on receiver-driven adaptation across multiple multicast groups may eventually be shown to be viable, this approach has not been fully and comprehensively developed and we felt it worthwhile to look for alternatives.

4.4.2 Congestion Control by using BCFS

In a heterogeneous environment, it is difficult to satisfy all receiver bandwidth requirements with a single multicast channel. To provide different data rates for each receiver without deteriorating network condition, congestion control using BCFS is designed as follows:

- **Individual TCP-like window control:** Each receiver independently executes TCP-like window control [13]. Data transmissions are triggered by the arrival of breadcrumbs at the sender. In turn, packet arrivals at the receiver cause that host to increase its congestion window (either by one for each packet received in slow start or one packet per round-trip in congestion avoidance mode). The invariant we maintain is that the number of breadcrumbs outstanding is less than or equal to the congestion window. Thus, the number of packets in transit from the source to the receiver is bounded by the congestion window. In addition, the congestion window is controlled in response to lost packets according to measured congestion conditions on the path from the source to that receiver. Since the window control behaves as if there were a TCP session between a source and a receiver, each receiver utilizes bandwidth in a TCP-friendly way.
- **Transmission request by BCF messages:** A receiver sends a TRQ (transmission request) as a BCF request using as many labels as its window size. This means that receivers that have the same window size use the same labels for TRQs, and a receiver that has a smaller window uses a subset of the labels that are used by a receiver with a larger window size. If receivers send TRQs with a label after another receiver sends a TRQ with the same label and the TRQs sent later arrives at a BCF router before the reply message of the former TRQ arrives, the TRQs are aggregated and the copies of the identical data packet are sent to all receivers which send the TRQs with the same label. TRQ corresponds to ACK in TCP in the sense that it is sent at packet reception. However it does not need to include sequence numbers of received packets.
- **Simple reply by a Digital Fountain source:** By using a Digital Fountain, the source can merely respond to each TRQ

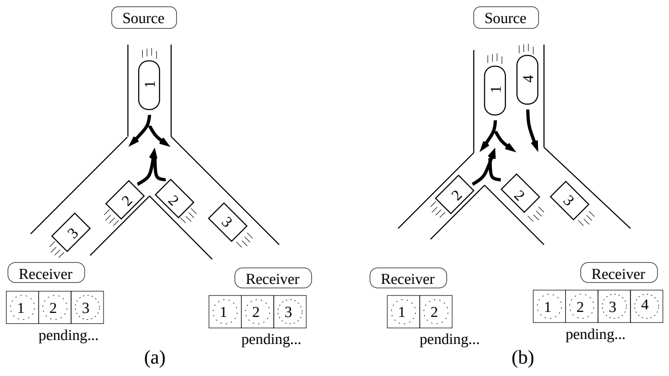

Figure 5: Multicast Congestion Control using BCFS: receivers maintain their individual window and TRQ's are spontaneously aggregated at a BCF router.

by sending one packet after another as a BCF reply, which includes the same label as the TRQ.

Figure 5 (a) illustrates Rainbow/BCFS data aggregation, where two receivers have a shared bottleneck link, they are probable to have the same window size and it is expected that most TRQs are aggregated at the link. As a result, most of the data packets from a source are directed to both receivers. In case that two receivers have bottlenecks at down-links and one link has half the bandwidth of the other link as shown in Figure 5 (b), the slower receiver receives half of the data directed to the faster receiver, copied at the diverging point. Through using Digital Fountain source, a receiver receives different packets with high probability and reliability is guaranteed by continuing to send TRQs until enough packets arrival to reconstruct the original data,

Of course, there is no guarantee that all packets are delivered efficiently as in the example above, because window control at each receiver is not synchronized in any explicit coordinated way, and a receiver accesses data asynchronously. But when receivers have a shared bottleneck link, it could happen that the same packet loss pattern causes same window control timing, and behave in a nearly synchronous way. As a result, more data aggregation occurs at the shared link, networks can enjoy efficient transmission of multicast, and receivers can receive data at the end-to-end available bandwidth.

4.4.3 Simulation Results

We ran simulations on ns [30] to investigate Rainbow's performance. We show some of the results which explore the behavior and dynamics of Rainbow based on two scenarios with relatively small scale sessions, as shown in Figure 6. More comprehensive scenarios for a large scale session and heterogeneous receivers are currently being investigated [32].

[Topologies]

To satisfy each receiver, the receiving rate should become close to end-to-end available bandwidth for each sender-receiver pair. However, 100 % utilization of available bandwidth is not expected because of TCP-like oscillating window control. In the perfect scenario, receivers sharing the same link would observe complete aggregation. However, this cannot be always realized because of lack of explicit synchronization mechanism. It is still expected that a "bottleneck" link should be shared efficiently by BCFS data aggregation mechanism. Another point to explore is whether clusters of receivers under the same link can share the link fairly.

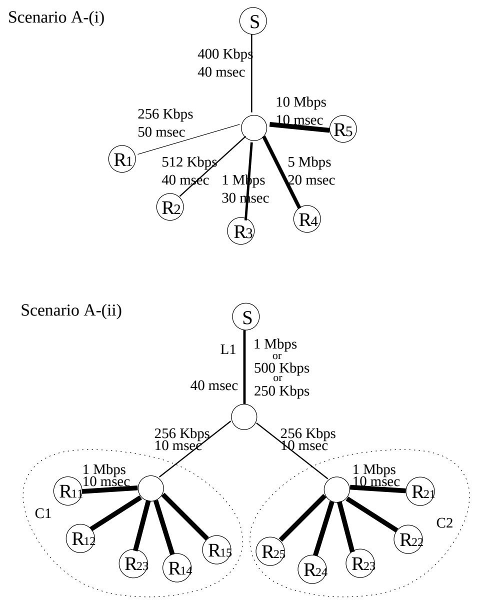

Figure 6: Simulation Settings: Exploring behavior.

The topology of Scenario A-(i) consists of one shared up-link and five different down-links. One down-link is narrower than the shared up-link, but others have broader link capacity than the shared bottleneck. Through this topology, adaptation to heterogeneous receivers and sharing bandwidth at a bottleneck link is investigated. While four faster receivers should receive the same service at up-link capacity, the slowest receiver should receive a portion of the data directed to faster ones at its down-link capacity.

Scenario A-(ii) has two clusters of five receivers with the shared backbone link (L1) by all receivers and the same capacity down-link for each cluster. For the shared backbone link, we use three different bandwidth. Receivers in both clusters should receive data at the same rate in all situations in terms of intra-session fairness, and the degree of data aggregation should change depending on the backbone capacity. We expect that as the backbone capacity becomes narrower, more packets are aggregated at the link and all the receivers come to receive the same data packets if the backbone becomes an end-to-end bottleneck.

In all simulations, the data packet size is 512 bytes, and the simulation run comprises 2000 packets, which means a receiver stops sending TRQs after receiving 2000 packets. All routers are RED gateways with a queue size of 10 packets. To "randomize" each run, each receiver initiates its session at a uniformly random start time in [0...5] seconds. In each scenario, 100 simulations are executed with randomized different start time of receivers and average results are shown in the following sections.

[Scenario A-(i)]

The results for scenario A-(i) are illustrated in Figure 7. The slowest receiver receives data at over 200 Kbps against its 256 Kbps bottleneck capacity and the average receiving rate throughout the duration of the simulation is 224.1 Kbps. This average is calculated from all 100 simulation runs, and we use the average rate for later explanation of simulation results. The receiving rate of all four faster receivers reaches around 350 Kbps against 400 Kbps up-link bottleneck bandwidth after slow start phase and the average rate of four receivers is 346.2 Kbps.

In this scenario, four faster receivers should be dealt with as if on a single multicast channel because they share the identical bottleneck

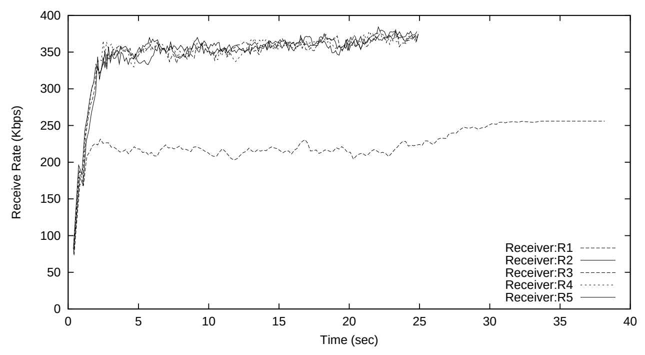

Figure 7: Simulation Result (scenario A-(i)): Receiving rate of five different receivers.

link. The overhead of sent packets through the shared link is 8.3 %, which is calculated by four times of the packet number sent from the source over the total packet number received by the four faster receivers during the stable condition, between 10 and 20 second in the simulation. Furthermore, 1237.4 packets out of 1265.6 packets for the slowest receiver (R1) are shared with at least one of other faster receivers while at least one other receiver is receiving data.

From this simulation, we have seen all receivers can receive at their appropriate rates according to each available bandwidth through data aggregation at bottleneck link. Some overhead exists even for receivers sharing a bottleneck link, but this overhead is not expected to grow as the number of receivers increases, because the more receivers exist under the same bottleneck link, the higher the probability TRQs are aggregated. As a result, the same data packet is sent to more receivers at the same time.

nineb [Scenario A-(ii)]

In Table 1, the average receiving rate for each cluster of five receivers and the average total packet number sent from the source are shown for different bandwidth backbone (L1) cases of scenario A-(ii). According to the average receiving rate in Table 1, receivers on both subtrees (C1,C2) attain approximately equal throughputs, around 225 Kbps, for all bandwidth cases.

When the shared backbone link has enough bandwidth (1 Mbps) to accommodate the two down-links, all receivers would receive data at the down-link capacity rate. In this case, packet aggregation at link L1 is not expected because there is no mechanism for synchronization. The packets for receivers under the bottleneck link (i.e. within C1) is totally aggregated as if they were on the same multicast channel as receiving rate around 225 Kbps in Table 1 shows. As a result entire behavior becomes similar to the situation where two multicast groups are formed for each subtree. The average total number of sent packets is 3326.9, which is less than the double of the necessary packet number for a single receiver because some packets are eventually aggregated across the clusters.

When the bandwidth of the shared backbone link is 500 Kbps, its capacity is a little less than aggregation of down-links bandwidth. Even in this case, the same receiving rate is realized as in 1 Mbps case, as shown in Table 1. The reason is that more packets are aggregated at the the backbone link (L1) and directed to more receivers at the same time, which is also evident by less total sent packets than 1 Mbps case in Table 1.

When the shared backbone link is 250 Kbps, the link becomes the bottleneck and all receivers should be dealt with as if on one multicast channel. In Table 1, a decrease in overall packets shows that more data are aggregated at the node of the backbone link. Further, average receiving rates for both clusters are consistent independent of backbone capacity.

As the results show, depending on the placements of bottleneck link, Rainbow aggregates data packets in different ways, and as a result, traffic behaves as if a subtrees of bottleneck link is formed as the same multicast channel without explicit coordinated mechanism for synchronization.

Table 1: Simulation Results (scenario A-(ii)): Rate for two clusters and total packet number.

L1 Bandwidth	Average rate (Kbps)		Packet number
1 Mbps	C1	226.2	3326.9
	C2	224.8	
500 Kbps	C1	224.2	3241.7
	C2	223.5	
250 Kbps	C1	226.3	2411.2
	C2	226.0	

5. CONCLUSION

In this paper, we have presented an alternative multicast service model, BCFS, and application examples built on top of the BCFS.

As future work, we plan to tackle practical implementation issues for BCFS in terms of a router's required memory and processing load, and describe more detailed protocol specification. Other efficient application examples on BCFS will enforce the significance of the service model and examine the usability of the BCFS functionality.

We believe BCFS provides a new direction for multicast forwarding service. By factoring PGM into a reusable network component that is modeled after EXPRESS, we have created a network service that is not only a good building block for PGM, but also provides a foundation for new transport protocols like Rainbow.

6. REFERENCES

[1] T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT): An Architecture for Scalable Inter-Domain Multicast Routing. In *Proceedings of Sigcomm '93*, pages 85–95, San Francisco, CA, Sept. 1993. ACM.

[2] S. Bhattacharyya, D. Towsley, and J. Kurose. The Loss Path Multiplicity Problem for Multicast Congestion Control. In *Proceedings of IEEE Infocom '99*, New York, NY, March 1999.

[3] R. Braden. *T/TCP – TCP Extensions for Transactions Functional Specification*, Jul 1994. RFC-1644.

[4] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A Digital Fountain Approach to Reliable Distribution of Bulk Data. In *Proceedings of Sigcomm '98*, Vancouver, Canada, September 1998.

[5] B. Cain, T. Speakman, and D. Towsley. Generic Router Assist (GRA) Building Block, Oct. 1999. Internet Draft (Work in Progress).

[6] Y. Chawathe, S. Fink, S. McCanne, and E. Brewer. A Proxy Architecture for Reliable Multicast in Heterogeneous Environments. In *Proceedings of ACM Multimedia*, Bristol, England, September 1998.

[7] A. Costello and S. McCanne. Search Party: Using Randomcast for Reliable Multicast with Local Recovery. In *Proceedings of IEEE Infocom '99*, New York, NY, March 1999.

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

[9] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*, Jun 1998. RFC-2362.

[10] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, 1995.

[11] H. Holbrook and D. Cheriton. IP Multicast Channels: EXPRESS Support for Large-scale Single-source Applications. In *Proceedings of Sigcomm '99*, Cambridge, MA, September 1999.

[12] H. Holbrook, S. Singhal, and D. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceedings of Sigcomm '95*, Boston, MA, Sept. 1995. ACM.

[13] V. Jacobson. Congestion avoidance and control. In *Proceedings of Sigcomm '88*, Stanford, CA, Aug. 1988.

[14] S. Kasera, J. Kurose, and D. Towsley. Scalable Reliable Multicast Using Multiple Multicast Groups. In *Proceedings of ACM Sigmetrics Conference*. ACM, June 1997.

[15] K. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley. The MASC/BGMP Architecture for Inter-Domain Multicast Routing. In *Proceedings of Sigcomm '98*, Vancouver, Canada, September 1998.

[16] B. Levine, , and J. Garcia-Luna-Aceves. Improving Internet Multicast with Routing Labels. In *Proceedings of IEEE International Conference on Network Protocols*, Atlanta, GA, October 1997.

[17] B. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The Case for Concurrent Reliable Multicasting Using Shared Ack Trees. In *Proceedings of ACM Multimedia*, Boston, MA, Nov. 1996. ACM.

[18] J. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings IEEE Infocom '96*, pages 1414–1424, San Francisco, CA, Mar. 1996.

[19] A. Mankin, A. Romanow, S. Bradner, and V. Paxson. *IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols*, Jun 1998. RFC-2357.

[20] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proceedings of Sigcomm '96*, Stanford, CA, Aug. 1996. ACM.

[21] J. Nonnenmacher, E. Biersack, and D. Towsley. Parity-Based Loss Recovery for Reliable Multicast Transmission. In *Proceedings of Sigcomm '97*, Cannes, France, Sep 1997. ACM.

[22] C. Papadopoulos, G. Parulkar, and G. Varghese. An Error Control Scheme for Large-Scale Multicast Applications. In *Proceedings IEEE Infocom '98*, San Francisco, CA, March 1998.

[23] R. Perlman, C.-Y. Lee, A. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green. Simple Multicast: A Design for Simple, Low-Overhead Multicast, Feb. 1999. Internet Draft (Work in Progress).

[24] S. Raman and S. McCanne. A Model, Analysis, and Protocol Framework for Soft State-based Communication. In *Proceedings of Sigcomm '99*, Cambridge, MA, September 1999.

[25] I. Rhee, N. Ballaguru, and G. Rouskas. MTCP: Scalable TCP-like Congestion Control for Reliable Multicast. In *Proceedings of IEEE Infocom '99*, New York, NY, March 1999.

[26] D. Rubenstein, J. Kurose, and D. Towsley. The Impact of Multicast Layering on Network Fairness. In *Proceedings of Sigcomm '99*, Cambridge, MA, September 1999.

[27] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4), Nov. 1984.

[28] P. Sharma, D. Estrin, S. Floyd, and V. Jacobson. Scalable timers for soft state protocols. In *Proceedings IEEE Infocom '97*, Kobe, Japan, Apr. 1997.

[29] T. Speakman, D. Farinacci, S. Lin, and A. Tweedly. PGM Reliable Transport Protocol Specification, Aug. 1998. Internet Draft (Work in Progress).

[30] UCB/LBNL/VINT. Network Simulator - ns (version 2). http://www-mash.cs.berkeley.edu/ns/.

[31] L. Vicisano, L. Rizzo, and J. Crowcroft. TCP-like congestion control for layered multicast data transfer. In *Proceedings of Infocom '98*, San Francisco, CA, March 1998.

[32] K. Yano and S. McCanne. The Breadcrumb Forwarding Service and the Digital Fountain Rainbow: Toward a TCP-Friendly Reliable Multicast. Technical report, University of California, Berkeley, Oct. 1999.

[33] R. Yavatkar, J. Griffioen, and M. Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. In *Proceedings of ACM Multimedia '95*, San Francisco, CA, Nov. 1995. ACM.