Revisiting IP Multicast

Sylvia Ratnasamy
Intel Research

Andrey Ermolinskiy
U.C.Berkeley

Scott Shenker
U.C.Berkeley and ICSI

ABSTRACT

This paper revisits a much explored topic in networking – the search for a simple yet fully-general multicast design. The many years of research into multicast routing have led to a generally pessimistic view that the complexity of multicast routing – and inter-domain multicast routing in particular – can only be overcome by restricting the service model (as in single-source) multicast. This paper proposes a new approach to implementing IP multicast that we hope leads to a reevaluation of this commonly held view.

Categories and Subject Descriptors: C.2.2 [Network Protocols]: Routing Protocols

General Terms: Design.

Keywords: Routing, Multicast.

1. INTRODUCTION

In 1990, Deering proposed IP multicast – an extension to the IP unicast service model for efficient multipoint communication [1]. The multicast service model offered two key benefits: (1) the efficient use of bandwidth for multipoint communication and, (2) the indirection of a group address which allows for network-level rendezvous and service discovery. Deering's proposal triggered an era of research on the implementation and applications of IP multicast. In terms of actual deployment, this research has had somewhat mixed success. On the one hand, support for multicast is built into virtually every endhost and IP router and the service is often deployed within enterprise networks. However there is little cross-provider global deployment of multicast, and today, fifteen years after Deering's seminal work, the vision of a ubiquitous multicast "dialtone" remains an elusive, if not altogether abandoned, goal.

Theories abound for why this vision was never realized (*e.g.*, [2–4]). Very broadly, most of these can be viewed as questioning the viability of IP multicast on two fronts. The first is its practical *feasibility* given the apparent complexity of deploying and managing multicast at the network layer. The second is the *desirability* of supporting multicast with many questioning whether the demand for multicast applications justified the complexity of its deployment, whether ISPs could effectively charge for the service, the adequacy of alternate solutions, and so forth.

SIGCOMM'06, September 11–15, 2006, Pisa, Italy.

This paper directly addresses the issue of feasibility, proposing a simpler approach to implementing IP multicast that we call Free Riding Multicast (FRM). We focus on inter-domain multicast for which complexity proved particularly acute but (as we describe later) FRM can be extended to the intra-domain scenario as well. FRM offers two key advantages over existing solutions:

- by leveraging existing unicast routes, FRM virtually eliminates the need for a distributed multicast route computation thus side-stepping much of the network layer complexity associated with traditional solutions (hence the name "Free Riding").
- a domain's participation and use of inter-domain multicast is effected via the same channel as in the unicast case, namely BGP, thus offering network operators a familiar framework within which to tackle the management (access control, accounting, *etc.*) of a multicast service.

These advantages however come at a cost and the core tradeoff FRM makes is to tilt the complexity of route computation to the *internals* of a router (as opposed to distributed protocol mechanism). Consequently, FRM requires more storage and algorithmic sophistication at routers and can be less efficient in bandwidth consumption than traditional multicast solutions. However this tradeoff – the avoidance of distributed computation and configuration at the cost of optimal efficiency – is one we believe is worth exploring given technology trends [5] that can endow routers with significant memory and processing on the one hand and our continued difficulties taming wide-area routing algorithms on the other [6, 7].

The primary focus of this paper is the design and evaluation of FRM. We lay the context for our work in Section 2, then discuss prior work and our overall approach in Sections 3 and 4 respectively. We present the design, evaluation and implementation of FRM in Sections 5, 6 and 7 respectively. Finally, we observe that FRM represents a more general approach to supporting network-layer services such as anycast or data-centric routing. We touch on this and other directions in Section 8.

Our contribution is a new approach to implementing multicast that we hope would lower the technical barriers to its deployment. At the same time, our exploration is triggered in part by the suspicion that the desirability of multicast too might merit rescrutiny. We briefly touch on this in the following section.

2. IN DEFENSE OF IP MULTICAST

While we make no claims to understand the "market" for multicast, we observe that many of the applications that originally motivated the research on multicast have (finally) arrived and would still be well served by native multicast support.

One example is massive multiplayer games (MMORPGs) with reports of 30-100% [8, 9] annual subscription growth and upto 5 million active subscriptions in a year [10]. In these games, a player's moves must be propagated to those in its "virtual" vicinity. Currently, game operators achieve this by deploying multiple servers, each assigned a region of the virtual world, that relay communication between players. Thus, for n nodes in a virtual region, the corresponding server's bandwidth requirements vary from O(n) to O(n^2) depending on the extent to which timeliness constraints allow multiple updates to be aggregated [11, 12]. Such scaling can be problematic and indeed numerous reports cite overloaded servers affecting the user experience [8, 12].[1] In a simple scenario, game operators might use multicast to cut server bandwidth to between O(1) to O(n). In a more sophisticated scenario, players might directly multicast updates thus offloading data forwarding from servers. In short, IP Multicast can aid game operators in building more lightweight, and hence ultimately cheaper infrastructure.

Another example is the adoption of Internet TV technology [15] with several providers already in customer trials. These efforts use IP multicast within their networks but currently rely on pre-provisioned channels from the content source in to their networks. Such provisioning allows ISPs to deliver content from major content providers to their immediate access customers. Supporting multicast across domains would further allow ISPs to transitively extend this delivery to more viewers and content providers without requiring each content provider to partner with individual ISPs.

File-sharing, software updates, RSS dissemination, video conferencing, grids are additional examples of deployed services that could potentially leverage multicast delivery.

It has been argued however that it is difficult for ISPs to charge for the use of multicast, leaving them with little incentive for deployment. As Diot *et al.* observe [3], this has much to do with the open, available-to-all usage model of earlier research. We assume that ISPs will instead enforce a more closed access model enabling them to charge for and better control usage (Section 8). Given a closed usage model, the emergence of ISP hosting services and IPTV lend hope that viable charging models exist; *i.e.*, ISPs charge server operators and/or endusers for multicast connectivity.

A rejoinder is that alternate techniques such as source-specific (SSM) or application-layer multicast can meet the needs of the above applications. Regarding SSM, we note that multi-source applications do exist (*e.g.*, game servers exchanging summary updates, P2P, video conferencing) and FRM could support these with complexity comparable (although different in nature) to SSM. SSM also loses the rendezvous features of the more general service model; while difficult to assess precisely, the many uses of a DHT's indirection capabilities [16–18] and the interest in auto-discovery mechanisms that enable opportunistic [19, 20] or configuration-free networking [17] suggest that low-level rendezvous might be a broadly useful feature.

Which leaves us with application-layer solutions. While both network and application layer solutions offer the benefits of multicast – efficient multipoint communication and indirection – they do so with very different tradeoffs. While application layer solutions are less constrained by the operational concerns of ISPs, scaling these to a global user population with even modest per-user bandwidth requirements represents a serious investment in bandwidth, server resources, and management. For example, the global adoption of IPTV with no support for multicast would require the ubiquitous deployment of video servers down at the DSLAM level. Moreover, deployments of such scale are likely beyond the resources of any single application provider but independent deployments bring with them non-trivial issues of application-level peering and interoperability. Network-layer solutions by contrast, allow the deployment of services that scale by augmenting an existing global ecosystem of infrastructure, services, customers, and peering arrangements. As a deployment vehicle, this can be particularly attractive for general-purpose services such as multicast or rendezvous, that can serve a variety of applications. The clear downside is that any evolution to this complex ecosystem is inevitably constrained.

While these tradeoffs are well-recognized, the reputed complexity of IP multicast has had the unfortunate consequence of transforming the debate on the desirability of IP multicast into one of whether it is strictly *necessary* to support multicast in routers. By lowering the complexity of network-layer multicast, we hope instead to revert back to debating its utility. In this context, the above discussion offers examples of existing applications that stand to gain from ISP deployment of IP multicast. We conjecture that ultimately both network and application-layer solutions to multicast – each used as appropriate – have much to offer in the implementation and scaling of networked applications such as network games, IPTV, *etc.* and it would be valuable to leave the door open to both approaches.

3. FRM: BACKGROUND, APPROACH

IP Multicast offers endhosts a simple abstraction: a host can *join* or *leave* a multicast group G and any host can *send* to a group G. As with unicast, the internals of the network provide the foundational packet delivery service atop which richer functionality may be implemented at endsystems. The multicast routing problem is thus key to supporting the service model and has been the subject of much research over the years [4, 21–27]. We start with a brief review of this literature. In the following, we distinguish between multicast routing and forwarding – the former to refer to the construction of distribution trees, the latter to the process by which routers propagate packets.

3.1 Multicast Routing

Deering's early work defined DVMRP, a broadcast-and-prune approach in which a packet multicast by a source S is forwarded towards *all* endhosts and those that receive unwanted packets send "prune" messages up the distribution tree toward the source [1, 21]. DVMRP constructs efficient shortest-path trees from *any* source but scales poorly for which reason it is typically limited to intra-domain routing.

Another intra-domain protocol is MOSPF, a multicast extension to unicast OSPF in which a router augments its link state advertisement with the list of groups for which it has local receivers which allows all routers to compute the shortest path tree from *any* source to all receivers. MOSPF is a fairly incremental extension and builds efficient trees but is limited to networks that run link-state protocols.

Shared tree protocols such as Core-Based Trees (CBT) [22] and PIM-SM [23] emerged to address the poor scaling of flood-and-prune tree construction. These protocols associate a special rendezvous point (RP) router that serves as the root of a single tree shared across all senders for a group. A new receiver sends a JOIN message along the unicast path towards the group's RP, instantiating forwarding state at routers along the way. While shared-tree protocols offer a dramatic improvement in scalability, they give rise to non-trivial issues regarding the placement and discovery of RPs.

[1] A revealing anecdote is the virtual demonstration in which, to protest issues with Warcraft's [13] operators, players overloaded and crashed operator servers by assembling in one virtual location [14].

Perhaps more importantly, the RP is the nerve center that determines the very availability of a PIM-SM tree and hence ISPs proved reluctant to depend on RPs run by other ISPs. This led to the development of the Multicast Source Discovery Protocol [24] that allows domains to discover and interconnect multiple RPs in a loose mesh. To accommodate the incremental deployment of PIM-SM/MSDP, multi-protocol extensions were introduced in BGP-4 (MBGP) [27]. MSDP has its own scaling problems and was thus originally intended as a temporary measure pending the deployment of a more scalable inter-domain solution.

BGMP [25] is one such proposal and incorporates many of the above ideas [25]. BGMP supports source-rooted, shared and bidirectional shared trees. Key to BGMP is the association of a group to a "home" AS responsible for allocating the group address. A group's home AS acts as the domain-level RP for the group's routing tree. To map a group to its home AS, BGMP proposes address advertisement (AAP) [25] that may be used in conjunction with MASC [28], a dynamic address allocation protocol.

The ever increasing complexity of multicast routing led Holbrook *et al.* [4] to challenge the wisdom of Deering's service model. They argued that many large-scale applications only require delivery from a single, often well-known, source. By exposing the identity of this source to the endpoints, routing can be greatly simplified by having receivers just send JOIN messages directly towards the source, moving RP discovery out of routers. Their Express protocol (now PIM-SSM) thus proposes a *single-source* service model in which a multicast "channel" is identified by both a group (G) and source (S) IP address. Endhost `joins/leaves` specify an (S,G) channel address and only the source S may transmit to a channel.

Holbrook *et al*'s insight represents a practical compromise that has done much to further ISP adoption of IP Multicast. The price is a loss in generality – with SSM, a group address is tied to a specific endhost IP address and hence the value of multicast as a network-layer rendezvous mechanism is largely lost. FRM makes a different compromise – retaining generality and seeking simplicity by accepting higher bandwidth and (off-the-fast-path) storage costs at routers.

3.2 Multicast Forwarding

The above centered on efforts to scale multicast routing. Of at least equal concern is the scalability of multicast forwarding state within routers. Because group membership need not be topologically contained, multicast forwarding entries are not easily aggregatable and, left unchecked, forwarding state grows linearly in the number of groups that pass through a router. Thaler and Handley [29] propose an interface-centric implementation model applicable to shared-bus router architectures which allows some aggregation. Their implementation model however does not apply to switched router architectures nor implementations which store forwarding state as a list of per-group incoming-outgoing interfaces. Moreover, in the absence of careful address allocation, forwarding state remains fundamentally linear in the number of active groups and can hence be non-trivial. Radoslavov [30] proposes "leaky" aggregation that tradesoff bandwidth for scalability in state while Briscoe *et al.* [31] propose a scheme wherein applications cooperate to select addresses that aid aggregation. To the best of our knowledge, none of these schemes have been adopted in common router implementations.

Discussion. The quest for a satisfactory multicast routing solution thus led down an increasingly tortuous path. Perhaps reflective of this is the somewhat daunting list of multicast protocols found in most commercial routers; *e.g.*, Cisco routers advertise implementations of PIM-SM, PIM-DM, Bidir-PIM, PIM-SSM, AutoRP, MBGP, MSDP and IGMP v1,v2,v3 (while still lacking support for address allocation (MASC/AAP) and scalable inter-domain (BGMP) protocols!) It is hard, even in retrospect, to cleanly lay the blame for this abundance of mechanism at the feet of any one problem or issue as each solution addresses a very real concern. Unfortunately, the complexity of the ensemble greatly raises the barrier to deployment of a multicast service. Our primary goal with FRM was thus to provide a "leaner" solution while retaining acceptable performance. Next, we describe our approach to achieving this.

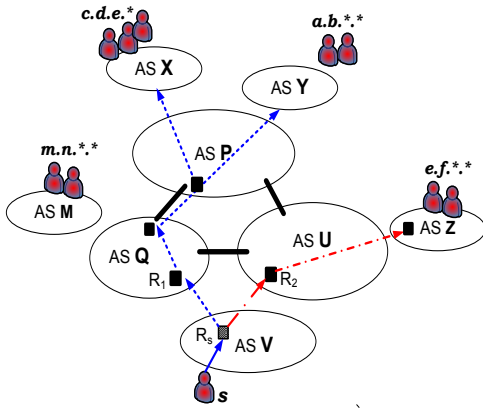

Figure 1: ***FRM: group membership and forwarding.***

4. FRM: APPROACH AND TRADEOFFS

4.1 Approach

In the abstract, multicast delivery requires knowledge of: (1) which end hosts are group members and, (2) how to reach these member hosts or domains. While most solutions combine these two components into a single from-the-ground-up protocol, FRM decouples membership discovery from route discovery. This separation offers the advantage that, once group members are known, any source can construct the multicast tree from its unicast routes to each member host. This is easily done for path-vector or link-state unicast protocols that reveal the required paths; as we discuss in Section 8, this approach can also be adapted to distance-vector protocols.

As stated earlier, we focus on inter-domain routing in which scenario the basic FRM scheme operates as follows: a domain's BGP advertisements are augmented with a description of the multicast groups currently in use within the domain. These adverts are then propagated as per the normal operation of BGP thereby giving every (border) router a description of the groups present in each destination prefix. To discover the dissemination tree for a group G, the the border router at the source (denoted R_s) scans its BGP table to identify those prefixes with members of G. Having thus identified all destination domains, R_s simply computes the dissemination tree from the union of the BGP unicast paths to all destination domains. R_s then forwards a single copy of the packet to each next hop on this dissemination tree along with an encoding of the subtree each next hop must in turn forward the packet along.

Figure 1 illustrates this process: A packet multicast to G by host s arrives at R_s. From their BGP advertisements, R_s learns that prefixes $a.b.*.*$, $c.d.e.*$, and $e.f.*.*$ have members in G and computes the multicast tree from the BGP paths from V to each of the

above prefixes and forwards one copy of the packet to Q along with an encoding of the subtree in dashed-line and another copy to U with an encoding of the subtree in dash-dot-dash style.

4.2 Discussion

To some extent, FRM can be viewed as extending MOSPF to the inter-domain arena. This extension however is non-trivial because we do not have a complete network map at the inter-domain level. The path-vector nature of BGP allows a router to compute the shortest path(s) from *itself* to a set of receivers but not from *any* source to a set of receivers. This complicates forwarding as a router that receives a packet has no way of knowing which subset of receivers it should forward towards since it does not know whether it lies on the shortest path from the source to those receivers. For example, in Figure 1: R1 and R2 both have BGP entries for prefixes $c.d.e.*$, $a.b.*.*$, and $e.f.*.*$, and can hence infer the presence of group members in these prefixes. However, when R1 receives a packet from R_s, it has no easy way of knowing not to forward toward $e.f.*.*$ and likewise R2 towards $c.d.e.*$ and $a.b.*.*$. While one might employ a limited form of flood-and-prune this raises issues similar to DVMRP in terms of scalability and vulnerability to dynamics. An alternate option might be to change BGP to a policy compliant link-state protocol however this represents a major overhaul of BGP which we avoid. FRM's forwarding is instead designed to exploit and live within the constraints of the information BGP offers. Finally, we note that while PIM and Express too leverage existing unicast routes they do so only in forwarding JOIN messages towards the rendezvous point; packet delivery still relies on group-specific forwarding state laid down by JOINs.

4.3 Tradeoffs

The core tradeoff FRM makes is to cut down on distributed protocol mechanism at the cost of demanding more from the *internal* capabilities of routers. This offers both advantages and challenges. On the positive side, we offer the following observations:

Parsimony in protocol mechanism. In terms of protocol complexity the basic FRM framework requires: (1) extending BGP to carry group membership information and (2) that an AS on occasion filter some group for a downstream customer AS (for reasons described in Section 5).

ISP control. Because group membership is explicitly advertised through BGP, an ISP has ultimate (and easy) control over which groups its customers subscribe to; *e.g.*, to block an undesired group, an ISP can simply drop it from its BGP advertisement. FRM also allows ISP control over sources in its domain as border routers have knowledge of (and control over!) the destination domains included in the dissemination tree. As articulated by Holbrook *et al.*, this assists in source-based charging as an ISP can now infer the traffic "amplification" due to a multicast transmission.

Ease of configuration. FRM avoids the contentious selection of RPs and new inter-domain protocols, instead piggybacking membership state over BGP.

Centralized route construction. In FRM, the multicast tree is computed in its entirety by the source's border router using existing unicast routes. This not only eliminates the need for a separate multicast routing algorithm but also spares us new routing anomalies [6, 7].

General service model. FRM supports a multi-source service model with efficient source-rooted trees.

The key challenges FRM faces include:

State requirements. FRM incurs the overhead of advertising and maintaining group membership. While true for all multicast protocols, FRM disseminates membership information more widely than traditional protocols and hence incurs greater overhead. Specifically, group state in FRM is aggregated per destination prefix rather than on the basis of topology.

Unorthodox packet forwarding. Traditional packet forwarding involves a (longest prefix match) lookup on the destination address to obtain the next hop along which to send the packet. By contrast, in FRM, finding the next hop(s) requires that the access border router scan its entire BGP table and that intermediate nodes decipher the encoded tree. FRM faces the challenge of achieving this in a manner that is both scalable and amenable to high-speed forwarding.

Bandwidth overhead. FRM's use of what is effectively a form of multicast source routing incurs additional bandwidth costs.

The remainder of this paper presents the design and evaluation of a protocol that addresses the above concerns.

5. DESIGN

The design of FRM comprises two (mostly separable) components – group membership discovery and multicast packet forwarding. This section presents our solutions for each along with a qualitative evaluation of their resource requirements. Our design assumes the license to quite significantly modify a router's internal operation though we do not modify unicast processing and require only a modest (and hardware-friendly) upgrade to forwarding plane. This appears reasonable given vendors' past willingness to incorporate new multicast routing into their routers. Our discussion of the overhead due to packet processing in routers follows standard assumptions – that high speed forwarding is assisted if packets are processed entirely on line cards and that the memory and processing available at the route processor may be comparable to high end machines but is more limited at line cards.

5.1 Advertising Group Membership

To leverage unicast routes, group membership information must be maintained at the same granularity as unicast routing destinations. For this, FRM augments BGP to include per-prefix group membership information. A border router augments its BGP advertisements with a description of the group addresses active – *i.e.*, with at least one member host – within its domain. Because simple enumeration leaves little opportunity for scaling to large numbers of groups, we encode active group addresses using bloom filters which allow advertisements to be compressed in a manner that introduces false postives but no false negatives and hence never results in service being denied to valid group members. The possibility of false positives however implies that a domain may on occasion receive traffic for a group it has no interest in. To handle this, the receiving domain R can either simply drop the unwanted traffic or, similar to DVMRP, can inform the upstream domain U to cease forwarding traffic for that particular group. This latter can be implemented by installing an explicit filter rule at U or by having R recode its advertisement to U into multiple bloom filters such that the offending false positive is eliminated. In this paper, we assume the use of filter rules.

Through the intra-domain multicast protocol (Section 5.3), a border router discovers which groups are active in its local domain and

encodes these addresses into a group bloom filter, denoted GRP_BF. The length of a GRP_BF is selected by reasoning in terms of the number of filter entries an AS is allowed by its upstream ASes. Each false positive results in a filter being installed at the upstream provider's network and hence, if an AS is allowed f upstream filters, then we set its target false positive rate to $\text{MIN}(1.0, f/(A-G))$ where G is the number of groups to be encoded and A is the total size of the multicast address space. This choice follows from the observation that a false positive can only be triggered by one of $A-G$ addresses which improves scalability by allowing for appropriately smaller GRP_BFs at large G; *e.g.*, a domain with $G \sim A$ ought only use a single bit that tells upstream domains to forward all multicast traffic its way. The filter size L is then computed using the above false positive rate. For efficient manipulation (compression, aggregation, expansion), we require that L be a power of two and assume a well known maximum length L_{max}.

A border router then piggybacks GRP_BFs on its regular BGP advertisements. If customer prefixes are aggregated, a corresponding aggregate GRP_BF is computed as the bitwise-OR of the individual customer GRP_BFs . Finally, based on its available memory, a router can independently choose to compress a GRP_BF of length L by repeated halving wherein the filter is split in two halves that are then merged by a bitwise-OR. Inversely, a previously compressed bloom filter can be expanded by repeated concatenation to obtain the desired length. Of course, both aggregation and compression result in a corresponding increase in the false positive rate.

Memory requirements. The total memory due to GRP_BF state at a participant border router is on the order of the number of destination prefixes times the average GRP_BF length. This can be non-trivial – for example, our evaluation in Section 6 estimates GRP_BF memory for 170,000 prefixes and 1 million active groups at approximately 2 GB. Fortunately, FRM's forwarding scheme does not require that GRP_BF state be stored in the forwarding tables on individual line cards and instead places GRP_BF state in the BGP RIB on the route processor. As such, the main impact due to the memory requirements for GRP_BF state is the monetary cost of memory. At even current memory prices, this should to be a minor increment to overall router costs [5].

Bandwidth and processing requirements. In keeping with the incremental nature of BGP, changes in GRP_BFs are communicated as deltas and hence the rate of updates depends primarily on the rate at which groups are added to, or removed from, a GRP_BF. Advertisements are for the domain as a whole and hence require updating only when the number of group members drops below one or rises above zero and hence unlikely to fluctuate rapidly, particularly if withdrawals are damped (as is likely [32]). Moreover, updates are small – on the order of the number of bloom filter hash functions for each added/deleted group. In terms of processing, GRP_BF updates, unlike BGP route updates, do not trigger route recomputations and only rarely require updating the actual forwarding tables on line cards (we describe when this is needed in the following section). Instead, processing GRP_BF updates is largely a matter of updating the BGP RIB in the route processor's memory. Thus, both the frequency and processing overhead due to GRP_BF updates should be tractable.

5.2 Multicast Forwarding

FRM processes packets differently at the border router in the access domain for the source (R_s), and border routers in the transit core (R_t). We discuss each in turn.

Forwarding on GRP_BF *state at* R_s. A packet multicast by source s to a group G is delivered via the intra-domain multicast routing protocol to R_s, the border router in the source's domain. R_s scans its BGP RIB, testing each GRP_BF entry to identify the destination prefixes with members in G and constructs the AS-level multicast tree $T(G)$ from the union of the individual AS-level paths to each member prefix. $T(G)$ can be computed in $O(p \times d)$ where p is the number of prefixes and d the average AS path length. We assume these operations are performed by the route processor where GRP_BF state is stored. While rather expensive, two factors render this computational complexity manageable. First is simply that, as an access router, R_s is under less forwarding load (in terms of both number of groups and total packets) than core routers and is hence better positioned to absorb this overhead. Second, and more valuable, is that R_s can cache, or even precompute, the results of the lookup so that this computation is only invoked on the first packet sent to each group. Thus, the complexity of lookups on GRP_BF state is incurred only by access border routers and, even there, only once for each group with active sources in the local domain.

Forwarding on cached state at R_s. As described above, R_s caches the results of the initial lookup on a group address G. Cached forwarding state is indexed by group address and hence accessed by exact-match lookups. Many well-known techniques exist for efficient exact-match lookups and we assume that FRM would employ any of these as appropriate – *e.g.*, CAMs and direct-memory data structures offer O(1) exact-match lookups while more compact data structures achieve exact-match lookups in logarithmic time [33, 34]. The total memory requirements for cached forwarding state depends on the number of groups with active sources within the domain and the per-group forwarding state. The latter of these depends on the size of the tree T(G) (we enumerate the exact forwarding state R_s must cache in the discussion on forwarding at R_t that follows). Our evaluation in Section 6 suggests that this state could be mostly accommodated in RAM on line cards – for example, our evaluation estimates a 400MB cache for a domain that has simultaneously active sources for 1 million groups [35–37]. If the memory on line cards cannot accommodate the entire cache, one might only cache state for high data rate groups on line cards leaving the route processor to handle forwarding for low data rate groups. Our implementation achieves this with LRU cache replacement.

In summary, caching replaces the linear scan of the BGP RIB's GRP_BF state by an exact-match lookup on cached forwarding state and, if needed, should be mostly achievable in line cards. We note that R_s maintains per-group forwarding state. However, as mentioned earlier, we believe this scaling is reasonable here because the number of groups (with active sources) in R_s's domain is likely lower than in core transit domains. In fact, the intra-domain multicast protocol is likely to impose similar scaling.

Forwarding at R_t. Multicast delivery is now a matter of forwarding the packet along $T(G)$, the AS-level tree computed by R_s, with appropriate packet replication at fanout domains. However, as described in Section 4, R_s cannot simply forward the packet to each of its next hop ASes on the tree as an interior AS does not know which subset of destination prefixes it should in turn forward to. Moreover, such an approach would impose forwarding state and complexity akin to that at R_s on all routers – a scenario we've argued against. We instead adopt an approach in which R_s communicates T(G) to intermediate routers. FRM implements this using a "shim" header above the IP header into which R_s encodes the *edges* from $T(G)$. A tree edge from autonomous system A to B is

State	scaling	lookup	used at	stored in	when used
GRP_BFs	O($\|p\|.g$)	linear scan	R_s	route proc.	per group
cached GRP_BFs	O($g_s.T(g_s)$)	exact match	R_s	line card	per pkt
encoded links	AS degree	filter match	R_t	line card	per pkt

Table 1: ***FRM: packet processing requirements.*** $|p|$ ***is the total number of prefixes at a BGP router and*** g ***the average groups per prefix.*** g_s ***is the number of groups with active sources in domain*** s ***and*** $T(g_s)$ ***the average size of the dissemination trees for groups with source in*** s***.***

assigned the unique label 'A:B' and R_s encodes these edge labels into the shim header it constructs for each of its next hops. Hence, in Figure 1, R_s would encode 'Q:P', 'P:X' and 'P:Y' in its packets to $R1$ and 'U:Z' in those to $R2$. Note that our choice of encoding edge labels is actually crucial in allowing R_s to disambiguate forwarding responsibility amongst interior ASes and allows the shim header inserted at R_s to be carried unchanged all the way to the destination(s) with no updating at intermediate routers (*e.g.*, this would not be possible were R_s to encode only the AS numbers of *nodes* in the tree).

For scalability reasons similar to those discussed in Section 5.1, we encode the dissemination tree into the shim header using a bloom filter (denoted TREE_BF) and deal with false positives as described in Section 5.1. However, unlike the GRP_BF advertisements, we require that the TREE_BF be of fixed length – or one of a small set of well-known lengths – so as to be amenable to fast processing in hardware. This raises the issue of picking an appropriate TREE_BF length. A too small header can lead to high false positive rates for large groups while a TREE_BF length selected to accommodate even the largest groups would be needlessly wasteful in the per-packet overhead the shim header imposes. Our solution instead is the following: we pick a fixed TREE_BF size of h bits, a target false positive rate f and compute e, the number of edges that can be encoded in h bits while maintaining a false positive rate $\leq f$. We then use a standard bin-packing algorithm to decompose the tree into groups of subtrees such that the number of edges in each group is less than e. This bin-packing can be computed in a single (typically partial) pass over $T(G)$. Each group of subtrees is then encoded into a single shim header and transmitted as a separate packet. Note that this approach can cause certain links to see multiple copies of a single multicast transmission.

The "tax" due to our *source-encoded* forwarding is thus twofold (we quantify these in Section 6):

- in its bandwidth consumption, source-encoded forwarding can be more inefficient than traditional multicast due to the per-packet shim header and redundant transmissions (on certain links) for groups too large to be encoded into a single shim header.
- the per-group forwarding state cached at R_s must now include the shim header(s). *I.e.*, for each cached group G, R_s caches the list of next hop ASes and the shim header(s) associated with each next hop.

The payoff is highly scalable and efficient packet processing at intermediate routers – to forward a packet, R_t need only check which of its AS neighbor edges are encoded in the shim header's TREE_BF. *I.e.*, if A is R_t's AS number, then, for each neighbor AS B, R_t checks whether 'A:B' is encoded in the packet's shim header in which case it forwards a copy of the packet to B. This offers two advantages. The first is that R_t's "forwarding" state is essentially a list of its neighbor edges. This state is independent of any particular group G and hence the number of such forwarding entries at R_t depends only on its domain's AS degree. Measurements report per-domain AS degree distributions ranging from 1 to under 10,000 with a power-law distribution and hence we can expect the number of forwarding entries at R_t to be low– potentially several orders of magnitude lower than the number of multicast groups – and easily accommodated on line cards.

For efficient packet processing, we store neighbor edges in their encoded representation; *i.e.*, each edge is inserted into, and stored as, a separate TREE_BF bloom filter. The lookup operation at R_t is then similar to standard filter matching – for each neighbor edge, R_t checks whether the corresponding bits are set in the packet's TREE_BF. There are a variety of options by which to implement this but perhaps the simplest is to use TCAM with the bloom filter for each neighbor edge stored in one TCAM row and all zero bits set to the "don't care" value [40]. With this, all edges can be matched in parallel with a single TCAM access. Alternately, this state can be stored in RAM and an edge matched in logarithmic time. Finally, as mentioned before, the shim header remains unmodified along the entire path and requires no updating at R_t.

The second advantage to source-encoded forwarding is that, because the forwarding state at R_t depends only on its (mostly static) set of AS neighbors, no wide-area protocol mechanism is required to construct and maintain this state. Source-encoded forwarding thus achieves sparseness in protocol mechanism and scalable forwarding state though at the cost of some additional bandwidth and memory (at R_s) usage.

Table 1 summarizes the state requirements due to FRM. We note that while FRM tilts the burden of forwarding state and complexity onto source access domains, this is a not displeasing arrangement as the benefit of multicasting is greatest at the source (a receiver's bandwidth consumption is unchanged with multicast). Finally, we note that source-encoded forwarding (somewhat unlike IP source routing) is easily implemented in hardware and selects paths compliant with the policy choices of intermediate ISPs.

5.3 FRM and Intra-domain Protocols

Many of the operational and scaling issues that complicate inter-domain multicast routing are less acute in the intra-domain scenario and hence it appears reasonable to retain existing solutions (*e.g.*, PIM-SM, DVMRP) at the intra-domain level. These can interface to FRM in a straightforward manner; *e.g.*, a group's internal RP could notify border routers of domain-wide group membership and packets could be relayed to/from FRM border routers via tunneling to the RP or by having border routers join groups with active sources. If desired however, FRM could be extended to the intra-domain scenario; we briefly discuss this in Section 8.

6. EVALUATION

In this section, we use simulation and trace-driven calculation to estimate the storage and bandwidth overhead due to FRM's group membership and forwarding components. Due to space constraints, we present only key results for likely usage scenarios. a more detailed parameter exploration is presented in [41].

Setup. To relate performance directly to end-user behavior, we allow U=2^{32-p} users in a domain of prefix length p and assume that each user joins k groups selected using some group popularity distribution from a total of A *simultaneously* active groups. Unless stated otherwise, we model group popularity using a zipfian distri-

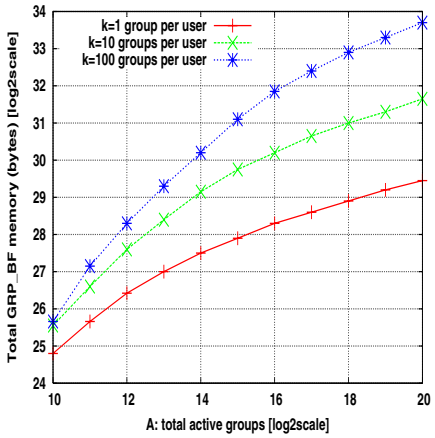

Figure 2: ***Total GRP-BF storage per border router.***

bution akin to the Web [42] and pessimistically assume no locality in group membership; any locality would only improve scalability. We use Subramanian *et al.*'s Oct'04 snapshots of BGP routing tables and their AS-level topologies annotated with inter-AS peering relationships [43].

6.1 Group Membership

Memory overhead. Per-prefix GRP_BFs are required to store group membership information. We compute the GRP_BF size for a single prefix as follows: the number of groups advertised per prefix – denoted G – is the expected number of distinct groups given that U users each pick k-from-A as per the selected group popularity distribution and hence the corresponding GRP_BF size is the bloom filter size needed to encode G items for a target false positive rate of $f/(A-G)$ (recall that f is the target number of filters per prefix). Then, to estimate the *total* storage due to GRP_BF state at a BGP router, we use real BGP tables [43] and compute the total storage per router as the sum of the GRP_BF size corresponding to each prefix entry. Figure 2 plots this total storage for increasing A for f =10 and k =1, 10, and 100 groups per user.

Overall, we see that the memory required to maintain group membership state, while not trivial, is very manageable given current storage technology and costs. For example, 1 million simultaneously active groups and 10 groups per user requires approximately 3 GB – an amount of memory found today on even user machines. Moreover, the trend in memory costs should allow FRM to handle the relatively slower growth in BGP table size.

Bandwidth costs. We use back-of-the-envelope calculations to show that the bandwidth due to updating group membership is tractable. Recall that a domain updates its membership for group G only when the number of members of G within the domain falls to, or rises above, zero. Moreover, some domain-level damping of group departures is likely. We thus generously assume a prefix sees a new group appear or an existing group depart every second. Updates are conveyed as the set of GRP_BF bit positions to be set/reset. Hence if we assume GRP_BFs use 5 hash functions and bit positions are represented as 24 bit values (in multiples of 256-bytes), then updating membership for a single prefix requires approximately 15 bytes per second (Bps). If we assume a router with full BGP routes has 200,000 prefix entries (current reports indicate ∼170,000 FIB entries [44]) then the total bandwidth consumed due to updates would be approximately 3MBps – a small fraction of the bandwidth capacity at core BGP routers.

The first node to join a group within its prefix/domain incurs the latency due to inter-domain GRP_BF update propagation. (The latency of subsequent joins is that of an *intra*-domain join.) Unlike regular BGP updates, GRP_BF updates do not trigger distributed route recomputations and hence their rate of propagation will likely be limited primarily by protocol constraints (if any) used to bound update traffic (as opposed to concerns about routing loops, inconsistencies, and the like). Our current prototype limits inter-AS GRP_BF updates to once per second which would lead to a "first-time" join latency of ∼ 1-6 seconds given current AS path lengths [44]. Further deployment experience would be required to better gauge appropriate update intervals.

Group size	Ideal multicast	FRM	per-AS unicast
100	28	28	38
1000	158	159	246
10,000	1000	1012	1962
100,000	4151	4233	9570
1M	8957	9155	21754
10M	15353	15729	39229

Table 2: `total-tx`***: the total number of packet transmissions for increasing group sizes.***

6.2 Forwarding Overhead

Bandwidth costs. The bandwidth overhead due to FRM forwarding stems from: (1) the per-packet shim header and, (2) the redundant transmissions required when subtrees are too large to be encoded in a single shim header. We assume fixed 100 byte shim headers and measure the overhead in packets transmitted; our results extrapolate to different shim header sizes in a straightforward manner.[2]

We use two metrics to quantify FRM's overhead due to redundant transmissions:

- `total-tx`: the total number of packet transmissions required to multicast a single packet from the source to all receivers
- `per-link-tx`: the number of transmissions *p*er link used to multicast a single packet from source to all receivers.

To calibrate FRM's performance, we measure the above for: (1) "ideal" multicast in which exactly one packet is transmitted along each edge of the source-rooted tree and, (2) per-AS unicast in which the source unicasts each member AS individually. This latter can be achieved using only FRM's group membership component and thus represents a simple network layer solution that requires no multicast-specific forwarding at routers (as does FRM).

Table 2 lists `total-tx` for increasing group sizes. We see that for all group sizes, the overall bandwidth consumed by FRM is very close to that of ideal multicast (between 0-2.4% higher) while per-AS unicasts can require more than twice the bandwidth of ideal multicast. As expected, the difference between FRM and ideal multicast grows with increasing group size due to the multiple shim headers needed to encode the larger trees.

[2]100 bytes represents ∼10% overhead on typical data (*i.e.*, non-ack) packets which appears reasonable. In practice, for greater efficiency, a source might choose from a few well-known shim header sizes; *e.g.*, we find even 20B headers would suffice for groups of upto a few thousand.

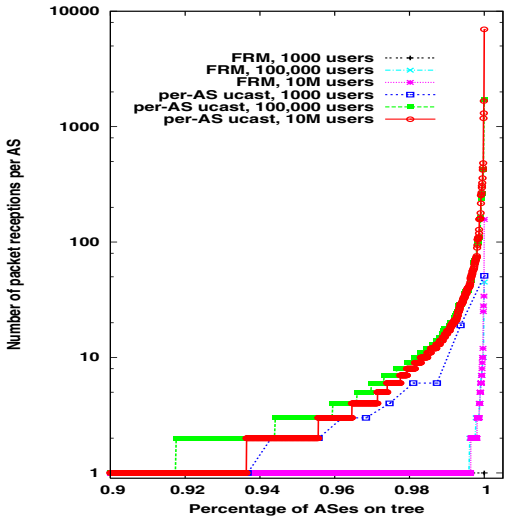

Figure 3: ***CDF of*** `per-link-tx`***, the transmissions per AS link for FRM and per-AS unicasts.***

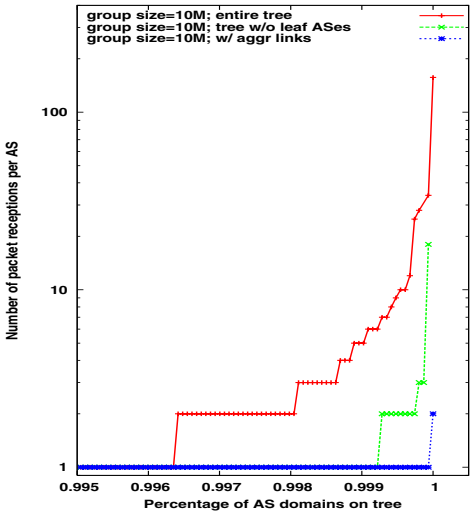

Figure 4: ***CDF of transmissions per (AS) link with optimizations to reduce the size of the encoded tree.***

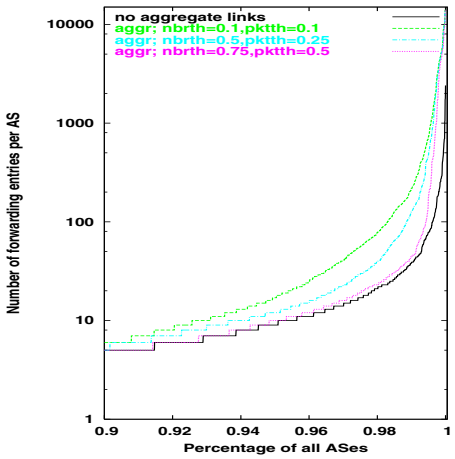

Figure 5: ***CDF of forwarding entries per AS. Tests with aggregate links use a group size of 10 million.***

Figure 3 plots the CDF of `per-link-tx` for FRM and per-AS unicasts for different group sizes (`per-link-tx` is always one for ideal multicast). In all cases, over 90% of links see exactly one transmission per link. However, we see that with per-AS unicasts, the worst-case `per-link-tx` can be over 40 for group sizes of just 1,000 and almost four orders of magnitude greater than ideal multicast for very large group sizes. FRM's tree-encoded forwarding significantly reduces this overhead as over 99.5% of links see exactly one transmission and the worst-case `per-link-tx` (at 10M users) drops to 157 relative to 6950 transmissions for per-AS unicasts.

We note that this is a stressful scenario – for our trace, 10 million users selected with no topologically locality results in every AS having a group member and is thus equivalent to broadcasting to the *entire* Internet. In such cases, FRM's overhead of $\sim$ 150 transmissions on a single link might well represent a reasonable penalty. Nonetheless, we look for techniques to further reduce this overhead. Examination reveals that the highest `per-link-tx` occurs at large ISPs that have both high degree and a large number of downstream ASes (*e.g.*, ATT, Usenet, Level-3). This leads us to propose two performance optimizations – one fairly trivial and another that, while light on mechanism, requires more forwarding state at core routers.

Optimization#1: no leaves. Here, customer ASes at the leaves of the dissemination tree are not encoded into the shim header. This could be acceptable because a provider AS that receives traffic for a group G can easily determine which of its immediate customer ASes have advertised membership in G and forward traffic appropriately. Now however, a multi-homed customer AS may on occasion receive traffic from more than one upstream provider. In this case the customer AS can, as in the event of a false positive, push filter requests to the provider sending it unwanted traffic. From figure 4, we see that this improves the worst-case transmissions per link by approximately an order of magnitude.

Optimization#2: aggregate links. If the number of tree edges from an AS A is a large fraction of either A's total edges (*nbrthresh*) or the total edges per packet (*pktthresh*), then the encoding router R_s replaces the edges from A by an *aggregate* edge 'A:$*$' that tells A to forward the received packet on all outgoing edges. Figure 4 plots the transmissions per link for *nbrthresh* $=$ *pktthresh* $= 0.5$ while Table 3 reports the worst-case transmissions per links for different *nbrthresh* and *pktthresh*.[3] We see that the use of aggregate links can allow FRM to match optimal multicast.

Aggregate links implicitly include non-tree edges. To avoid A sending packets out along non-tree edges, when A receives a packet matching 'A:$*$', it forwards the packet to a neighbor B only if the packet also matches 'B:X' for some X, neighbor of B. This requires that A know of B's edges that lie on the path from A to various destinations. Fortunately, this information is locally available from A's

[3]We note that the parameters (*nbrthresh* and *pktthresh*) do not require to be globally consistent and are instead selected independently by R_s. Moreover, the effectiveness of a particular parameter choice is immediately evident when decomposing the tree and hence R_s can experiment with a few parameter choices to achieve a target overhead.

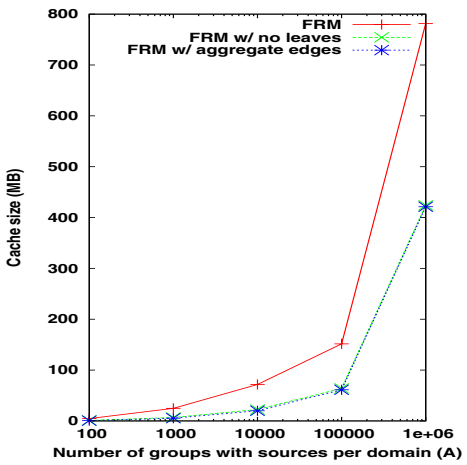

Figure 6: ***Cache size for increasing A, the number of groups with active sources in a domain.***

BGP table and can hence be obtained with no additional protocol mechanism but requires that A store additional AS edges in its forwarding table. To control this increase, A can maintain 2-hop edges for only a few neighbors and indicate these through (for example) a flag associated with a BGP path it advertises. In our tests, we assume an AS maintains 2-hop edges for only its customers and measure the corresponding increase in forwarding state.

In summary, for very large groups, aggregate edges can improve the efficiency FRM to match optimal multicast at the cost of additional forwarding state but little new mechanism (specifically, an additional BGP flag attribute, a new conditional clause in the tree decomposition at R_s and an additional matching rule in the forwarding at transit R_t routers).

Storage costs. The forwarding state at a core router R_t is made up of its AS neighbor edges and hence the number of forwarding entries at R_t is the AS degree of its domain. The use of aggregate links adds additional 2-hop edges to the forwarding table. Figure 5 plots the cummulative distribution of the number of forwarding entries per AS for both basic FRM, and FRM using aggregate edges. We see that the power-law AS degree distributions means that the vast majority of ASes have remarkably small forwarding tables – in all cases, over 90% have less than 10 entries. We also see that for most ASes the number of forwarding entries is unchanged by the use of aggregate edges. The worst-case number of entries however increases from approximately 2,400 without aggregate links to 14,071 with aggregate links. While a significant relative increase, this is still a small number of forwarding entries in the absolute. The corresponding memory requirements can be computed as the number of entries times the size of the bloom filter (recall we store each edge as a bloom filter). With 100 byte bloom filters, this gives a worst-case forwarding table of 2,400 entries, $\sim$ 240KB for FRM and 14,071 entries, 1.4MB for FRM with aggregate edges both of which can be comfortably accommodated with current TCAM usage [37, 45].

The forwarding state at the source's border router R_s consists of the cached shim header(s) for those groups with active sources within the domain. To compute the amount of cached state, we assign a domain a total of A groups with active sources and assume, as before, that users join each group based on a zipfian group popularity distribution and enforce a minimum group size (of all 8 domains) to avoid empty groups. For each resultant group size, we

nbrthresh $\Rightarrow$ pktthresh $\Downarrow$	0.1	0.25	0.5	0.75
0.1	1	1	2	2
0.25	1	2	2	3
0.5	1	2	2	6

Table 3: ***Worst-case transmissions per (AS) link with aggregate links and different*** *nbrthresh* ***and*** *pktthresh.*

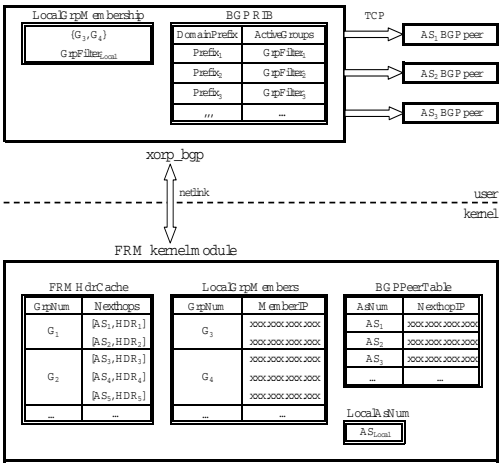

Figure 7: ***Software architecture of the FRM prototype***

compute the corresponding number of shim headers as above. Figure 6 plots the cache size for increasing A. If we assume on the order of several hundred megabytes of RAM on line cards, then we see that R_s could support line-card-only forwarding for upto several hundred thousand groups and over a million groups using the above optimizations. The initial sub-linear scaling trend is because cache requirements for highly popular groups dominate the initial cache size while the later linear scaling reflects our limit on the minimum group size. We note that our tests are stressful in that groups 1-25 all have over 10 million users; *i.e.*, every domain has 25 groups with sources simultaneously multicasting the entire Internet.

In summary, caching should allow source border routers to handle forwarding in the line cards for at least several hundred thousand groups.

7. IMPLEMENTATION

We have built a prototype FRM router that runs under the Linux operating system using the eXtensible Open Router Platform (XORP) [46]. Figure 7 illustrates the overall structure of the FRM prototype. A Linux kernel module implements the FRM forwarding plane and a user-level component manages group membership state and propagates membership updates to neighboring ASes. The user-level module runs in the execution context of the XORP BGP daemon (`xorp_bgp`) and communicates with the kernel-side FRM module via the Linux netlink mechanism. At kernel level, the `FRMHdrCache` table caches forwarding state for groups that have sources in the router's local domain while the `BGPPeerTable` holds the encoded AS edges used to forward transit packets. The GRP_BFs are stored in the BGP RIB in XORP. Our prototype currently lacks support for interfacing FRM to intra-domain multicast routing protocols; instead, as an interim mechanism, we maintain a local table (`LocalGrpMembers`) that stores the IP addresses of local group members. A more scalable implementation might, for example, store the IP address of the group's local RP. We mod-

ify the designated router (DR) side of the IGMP implementation to insert/remove shim headers. Endhosts are thus unchanged and FRM routers only *update* shim headers. Our impementation adds 3500 lines of code to the Linux kernel and 1900 lines to the BGP daemon.

7.1 Packet Processing

The kernel delivers incoming multicast packets to the FRM module. If the source address indicates that the packet originated in the router's local domain, then we first check for forwarding state in the `FRMHdrCache` cache.

Source domain: cache miss. In the event of a cache miss, the kernel upcalls to `xorp_bgp` to request the multicast tree for the packet's destination group. `xorp_bgp` responds with a set of structures of the form AS_x : $SubTree_x$, where AS_x is the AS number of a direct child node and $SubTree_x$ is a list of edges in the subtree at AS_x. The kernel parses the daemon's response and constructs the FRM shim headers for every AS_x.

Our shim header consists of 32 control bits, followed by the TREE_BF. The first 4 control bits hold the number of bloom filter hash functions, followed by 4 bits for the length of the TREE_BF in multiples of 16 bytes. The next 16 bits carry a checksum computed over the shim header; the last 8 bits are currently left for future protocol extensions.

Once the headers are computed, a copy of the packet is made for each AS_x, its shim header updated appropriately, and then sent out. We use an auxiliary data structure (`BGPPeerTable`) in the kernel to map from the AS number of a BGP peer to its corresponding next-hop IP address. Finally, we add the destination group address and the set of shim headers for each AS_x into `FRMHdrCache`. The `FRMHdrCache` cache is indexed by group address and uses a basic LRU replacement scheme.

Source domain: cache hit. In the event of a cache hit, processing is simple – a copy of the packet is made for each AS_x entry associated with the destination group, the packet's shim header is updated with the appropriate shim header, and the packet sent to AS_x.

Transit domain processing. If the packet did not originate in the router's local domain, processing is straightforward: we decrement the IP TTL, update the IP checksum and finally traverse the `BGPPeerTable` checking for the presence of the edge denoted 'AS_{local} : AS_x' in the packet's FRM header. If present, we forward a copy of the packet to the next-hop address for AS_x. As the last step,a copy of the packet is sent to every local member listed in the `LocalGrpMembers` table.

We measure the forwarding latency for the above code paths. Our measurements were performed on a 1.8GHz IBM Thinkpad with 1GB RAM running FRM under Linux RedHat 9, kernel level 2.4.20-8. Table 4 lists the forwarding time for packets that hit in the `FRMHdrCache` cache under increasing fanout (*i.e.*, outgoing copies) for different payload sizes. Relative to unmodified Linux, FRM exhibits similar scaling behavior but is always slower in the absolute. Examination reveals this is primarily because our FRM implementation incurs one additional buffer copy for every packet sent – in standard multicast, an identical copy of the packet is sent to all outgoing next hops while generates a distinct copy of the packet (with appropriate shim header) for every neighbor and hence replicates the original buffer.

To measure the forwarding time for packets that suffer a cache miss, we populate the RIB with an Oct'04 snapshot of a BGP table with 117519 prefix entries and initialize a fraction of prefixes to indicate membership in the packet's destination group. Table 5 lists the forwading time for an increasing number of prefixes included in the tree. We see that, in the worst case where every prefix has a group member, it takes approximately 303.2 ms to forward the packet. Further investigation revealed this time is dominated by the cost of the BGP RIB scan. Although clearly expensive, we do not view the processing latencies of cache misses as cause for concern due to two reasons: First, these measured latencies are entirely dependent on the processor speed and other hardware characteristics of the router which is, in our case, a uniprocessor IBM Thinkpad. In reality, header construction can be parallelized and optimized on SMPs. Second, this latency is only incurred on the first packet sent to a group, and can be rendered even more infrequent by avoiding cache misses through pre-computation and an appropriate choice of the cache size.

Finally, Table 6 lists the forwarding latency for transit packets for different tree fanout values and different sizes of the table `BGPPeerTable`. We observe that transit forwarding is efficient and only marginally more expensive than a cache hit at the source router for the same tree fanout. As with source forwarding, the processing time scales linearly with the number of outgoing packet copies. As expected (given our software implementation) the results are linearly dependent on the domain's AS degree though TCAM would avoid this.

In summary, the design of FRM admits a straightforward implementation of the cache hit and transit forwarding code paths that achieve efficiency comparable to that of the native kernel forwarding. For cache misses, we believe a combination of hardware and software optimizations, along with a sufficient cache memory allotment can make the performance impact of misses negligible but an exploration and evaluation of performance optimizations merits further study, particularly in the context of realistic router forwarding engines.

Fanout	Linux mcast 1-byte pkts	FRM 1-byte	FRM 128-bytes	FRM 1024 bytes
1	0.4	0.7	0.8	1.2
128	25.4	64.8	76.2	89.5
256	50.7	132.5	154.2	177.5
512	101.2	262.7	308.6	351.4

Table 4: ***Forwarding time (in μsecs) at R_s when the group is in*** `FRMHdrCache`**.**

#entries in tree	117519	29296	7300	1831	471	0
proc. time	303.2	124.8	89.1	74.5	68.3	65.8

Table 5: ***Forwarding time (in milliseconds) at R_s when the group is not in*** `FRMHdrCache`***. Packets are 512 bytes.***

Fanout $\Rightarrow$ AS deg. $\Downarrow$	1	32	128	256	512	1024
1	7.6					
32	10.9	38.8				
128	17.0	43.8	127.1			
256	27.7	54.5	137.1	220.7		
512	45.6	73.5	159.4	248.8	402.2	
1024	81.4	113.4	204.5	308.0	465.2	748.7

Table 6: ***Forwarding time (in μsecs) at R_t for 512-byte packets.***

7.2 Advertising group membership changes

An endhost's IGMP reports are delivered to its designated router (DR). In our current implementation, we modify DRs to relay these reports directly to the source FRM router R_s which updates its `LocalGrpMembers` table. We define a new optional transitive path attribute `FRM_UPDATE` for communicating incremental group membership changes and `FRM_GRP_BF` for the initial transfer of GRP_BFs at the start of a peering session.

To avoid a full scan of `FRMHdrCache`, we use an auxiliary data structure that efficiently resolves a bit position into a set of pointers to cached groups associated with that bit.

In our evaluations, the processing cost of an update message for a single group activation event that modifies 6 bits in the membership Bloom filter and invalidates a single `FRMHdrCache` entry (with 1024 entires present in the cache) requires total processing time of 18.6 μsec. It takes 0.34 μsec to update the Bloom filter and 18.33 μsec to perform the invalidation.

Finally, to test FRM end-to-end, we set up a local testbed of 4 interconnected FRM routers, with 2 Windows desktops running unmodified VAT [47] that connect to our FRM network via our modified DRs. We observed packet delivery from the VAT source to receivers demonstrating that FRM can forward packets end-to-end using legacy endhost stacks and applications.

8. DISCUSSION:

Usage model. It is likely that a multicast service deployed today, would not adopt an open usage model. We speculate on possible usage models but stress that issues of address allocation, access control and charging merit much greater scrutiny than we can provide here.

ISPs might control use of multicast at two levels – per-user and per-group. The first determines whether a user is allowed to send and/or receive multicast traffic (independent of which groups). As with unicast connectivity, users sign up with their local ISP for multicast service and the local ISP handles access control and charging of users. ISPs might distinguish between service offerings that allow users to both send and receive traffic from those that only allow a user to receive multicast traffic. For senders, ISPs might choose to charge based in proportion to the group size or include limits on the (AS-level) group size in the service agreement. FRM assists ISPs in this regard as it allows the access provider to accurately compute and control the extent of the dissemination tree.

Access control at the group level controls which group addresses are routable. ISPs might each be allocated a portion of the multicast address space and, to create a group, a user must explicitly obtain an address from some ISP. The role of the allocating ISP is merely to legitimize the group and does not constrain membership of the group in any way. ISPs only route group addresses that can be proven to have been legitimately allocated by a recognizable ISP. For this, an allocating ISP signs the group address with its private key; group members may retrieve this signature via the same channel (*e.g.*, DNS) used to discover the group address and can present the signature to its local ISP when it joins and/or sends to a group. To verify signatures, ISPs use the signing ISP's public key which can be disseminated along with an ISP's BGP adverts. Allocation of a group address can be associated with a fee and a lease period allowing prices to be driven by demand.

The above serves to limit service to legitimate users and legitimate groups but does not attempt to regulate which users are allowed access to which groups. We conjecture that this may be a tractable level of control for ISPs to implement while leaving more fine-grained access control to be handled by applications as per their (different) needs. At the same time, the above access control schemes could accommodate some extensions for more fine-grained control; *e.g.*, a user's service contract could limit the groups it may join or the allocating ISP's signature could include a list of authorized sender IP addresses.

Finally, while the above assumes ISPs control address allocation, this is not strictly required as FRM imposes no structural restrictions on the allocation and use of group addresses.

Attacks on the FRM protocol. With the above, malicious attempts to trigger frequent GRP_BF would be limited to legit groups which should make it harder to cause domain-wide fluctuations in membership. Moreover, this is tantamount to a user attacking its local ISP which increases attacker exposure. The same is true for malicious users that send to many different (valid) groups so as to burden routers with the more expensive tree construction operations.

Intra-domain FRM. FRM may be applied unmodified within domains that run link-state protocols. For domains with distance-vector-based protocols, FRM requires modification to work in the absence of complete path information. For this, we could encode destination nodes, as opposed to tree edges, in the shim header. As mentioned in section 5 this would require that intermediate routers repartition the set of encoded leaves to avoid duplicate forwarding though the results of this could be cached.

Relative to running intra-domain link-state MOSPF, FRM's source-encoded forwarding reduces the state and computational load at intermediate routers but requires a shim header. Admittedly, these are modest advantages and hence replacing intra-domain MOSPF by FRM would more likely be motivated by a desire for uniformity in intra and inter-domain solutions. Relative to intra-domain PIM-SM, FRM avoids the need to configure RPs.

Other routing services. As FRM makes no use of hierarchical address allocation or aggregation, its implementation represents a fairly general abstraction – subscription to, and location of – flat identifiers and could thus be applied to more general routing services such as IP layer anycast, data-centric or name-based routing. The main difference is that multicast requires matching *all* subscriptions while the above require matching *any*. The only implication to our design is that false positives would be undesirable; a simple solution would be to instead, enumerate subscriptions or use compression that admits only false negatives.

9. CONCLUSION

FRM represents a different approach to implementing multicast. It is simpler in the wide area (no distributed tree construction), easier to configure (no need to place RPs), and allows providers to work within the familiar BGP framework to handle inter-provider issues. These features come at a cost of reduced efficiency and greater demands on border routers; a tradeoff that we believe is worth exploring given technology trends.

FRM tackles a purely technical barrier to deployment and other barriers do exist. However, given the growing adoption of Internet broadcasting, massively multiplayer games, and other networked applications we conjecture the time may be right to revisit IP multicast and re-evaluate its chances.

10. ACKNOWLEDGMENTS

We thank Katerina Argyraki, Kevin Fall, Zhi Li, Timothy Roscoe and the anonymous reviewers for their valuable input that helped improve this work. We would also like to thank Hitesh Ballani for helpful discussions on exploiting router resources.

11. REFERENCES

[1] Stephen Deering and David Cheriton. Multicast routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[2] Yang hua Chu, Sanjay Rao, and Hui Zhang. A Case for End System Multicast. In *Proceedings of SIGMETRICS 2000*, CA, June 2000.

[3] Christophe Diot, Brian Levine, Bryan Lyles, H. Kassem, and D. Balensiefen. Deployment issues for IP multicast service and architecture. *IEEE Network Magazine. Special Issue on Multicasting*, 2000.

[4] Hugh Holbrook and David Cheriton. Ip multicast channels: Express support for single-source multicast applications. In *Proceedings of SIGCOMM '99*, Cambridge, MA, September 1999.

[5] ISC Domain Survey, January 2005.

[6] Craig Labovitz, Abha Ahuja, Abhijit Abose, and Farnam Jahanian. An experimental study of delayed Internet routing convergence. 2000.

[7] Matthew Caesar, Donald Caldwell, Nick Feamster, Jennifer Rexford, Aman Shikh, and Jacobus van der Merwe. Design and Implementation of a Routing Control Platform. In *Proc. of NSDI*, 2005.

[8] E. Castronova. Network Technology, Markets and the Growth of Synthetic Worlds. In *Second Workshop on Network and Systems Support for Games (NetGames)*. ACM, May 2003.

[9] MMOGCHART. http://www.mmogchart.com , http://terranova.blogs.com/terra_nova/2003/10/growth_rates_of.html.

[10] Blizzard Entertainment. WoW Surpasses 5 Million Customers Worldwide. 2005. http://www.blizzard.com/press/051219.shtml.

[11] N. Sheldon, E. Girard, S. Borg, M. Claypool, and E. Agu. The Effect of Latency on User Performance in Warcraft III. In *Second Workshop on Network and Systems Support for Games (NetGames)*. ACM, May 2003.

[12] J. Pellegrino and C. Dovrolis. Bandwidth Requirement and State Consistency in Three Multiplayer Game Architectures. In *Second Workshop on Network and Systems Support for Games (NetGames)*. ACM, May 2003.

[13] Blizzard Entertainment. World of Warcraft. http://www.blizzard.com.

[14] Synthetic Statehood and the Right to Assemble. http://terranova.blogs.com/2005/02/the_right_to_as.html.

[15] Microsoft IPTV Edition.

[16] Ion Stoica, Dan Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet Indirection Infrastructure. In *Proceedings of SIGCOMM*, August 2002.

[17] Bryan Ford. Unmanaged Internet Protocol: Taming the edge network management crisis. In *HotNets*, November 2003.

[18] A. Rowstron, A-M. Kermarrec, M. Castro, and P. Druschel. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. In *Proceedings of NGC*, London, UK, November 2001.

[19] Hui, Chaintreau, Scott, Gass, Crowcroft, and Diot. Pocket switched networks and the consequences of human mobility in conference environments. In *Workshop on Delay Tolerant Networking*, 2005.

[20] Kevin Fall. A Delay Tolerant Networking Architecture for Challenged Internets. In *Proceedings of SIGCOMM*, August 2003.

[21] D. Waitzman, C. Partridge, and S. Deering. *Distance Vector Multicast Routing Protocol*. ARPANET Working Group Requests for Comment, DDN Network Information Center, November 1988. RFC-1075.

[22] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core based trees (CBT) an architecture for scalable inter-domain multicast routing. Technical report, San Francisco, CA, September 1993.

[23] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas. Protocol Independent Multicast – sparse mode (PIM-SM): Protocol specification, October 2003. Internet Draft.

[24] B. Fenner and D. Meyer. *Multicast Source Discovery Protocol (MSDP)*. ARPANET Working Group Requests for Comment, DDN Network Information Center, 2003. RFC-3618.

[25] K. Kumar, P. Radolavov, D. Thaler, D. Alaettinoglu, D. Estrin, and M. Handley. The MASC/BGMP architecture for inter-domain multicast routing. In *Proceedings of SIGCOMM '98*, Vancouver, BC CANADA, September 1998.

[26] Dina Katabi. The Use of IP Anycast for Building Efficient Multicast Trees. In *Proceedings of Global Internet*, 1999.

[27] T. Bates et al. *Multiprotocol Extensions for BGP-4*. ARPANET Working Group Requests for Comment, 2000. RFC-2858.

[28] Radoslavov et al. *The Multicast Address-Set Claim Protocol*. RFC-2909.

[29] David Thaler and Mark Handley. On the aggregatability of multicast forwarding state. In *Proceedings IEEE Infocom*, Israel, March 2000.

[30] Pavlin Radoslavov, Deborah Estrin, and Ramesh Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical Report TR99-697, University of Southern California, 1999.

[31] Briscoe and Tatham. *End-to-end aggregation of multicast protocols*, 1997. Internet Draft.

[32] W. Fenner. *Internet Group Management Protocol, Version 2*. Internet Engineering Task Force, Inter-Domain Multicast Routing Working Group, February 1996. Internet Draft.

[33] Pankaj Gupta. *Algorithms for routing lookups and packet classification*. PhD thesis, Stanford University, December 2000.

[34] M. Waldvogel ad G. Varghese, J. Turner, and B. Plattner. Scalable high speed IP routing lookups. In *Proceedings of SIGCOMM '97*, Cannes, France, September 1997. ACM.

[35] Cisco Systems. *Cisco 1200 Series 3GigE Line Card*. (linecard with 512MB buffer and 256MB route memory).

[36] Cisco Systems. *Cisco 1200 Series One-Port OC-192 Line Card*. (reports 512MB route memory).

[37] Katerina Argyraki and David R. Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *Proc. of USENIX Annual Technical Conference*, 2005.

[38] S. Keshav and Rosen Sharma. Issues and Trends in Router Design. *IEEE Communications Magazine*, May 1998.

[39] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S.Shenker. Off by Default! In *Fourth Workshop on Hot Topics in Networks*, November 2005.

[40] Content Addressable Memory Cypress Semiconductor. http://www.cypress.com.

[41] S. Ratnasamy, A. Ermolinskiy, and S.Shenker. Revisiting IP Multicast. Intel Research Technical Report.

[42] V. Padmanabhan and L. Qiu. The content and access dynamics of a busy web site: Findings and implications. In *Proceedings of SIGCOMM*, Stockholm, Sweden, August 2000.

[43] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz. Characterizing the Internet Hierarchy from Multiple Vantage Points. In *Proc. of IEEE Infocom*, 2002.

[44] Route Views Project Page.

[45] Cisco Systems. *Access list configuration in Cisco's Gigabit Ethernet Interface*. (reports GigE module supports up to 256K TCAM entries).

[46] Handley, Kohler, Ghosh, Hodson, and Radoslavov. Designing Extensible IP Router Software. In *Proceedings of NSDI*, 2005.

[47] Van Jacobson and Steven McCanne. *Visual Audio Tool*. Lawrence Berkeley Laboratory.