

SANDS: Specialized Active Networking for Distributed Simulation*

S. Zabele¹, M. Dorsch², Z. Ge³, P. Ji³, M. Keaton¹, J. Kurose³, J. Shapiro³, D. Towsley³

Abstract

This paper provides an overview of SANDS (Specialized Active Networking for Distributed Simulation), a DARPA-ITO sponsored research project that is using active networking to develop a new approach to real-time, content-based information dissemination. Our approach is based on the use of active interest filtering, a publish/subscribe mechanism that uses active networks technology to install and control dynamically-established content-based filters in intermediate active routers in an IP multicast distribution tree. Active filters prune unneeded information as early as possible in the distribution tree, ensuring that only data desired (i.e., subscribed to) by a receiver actually reaches that receiver. In this paper, we describe active interest filtering, the per-node algorithms that implement active filtering, and the signaling protocol that installs interest filter state in the active routers. We describe our prototype implementation effort and present measurements from a working prototype that demonstrate the advantages of active interest filtering in a large-scale ModSAF distributed simulation scenario.

1. Introduction

This paper describes a new approach to scalable and efficient, dynamic, content-based delivery of information from a potentially large number of senders (information “publishers”) to a potentially large number of receivers (information “subscribers”). By content-based, we mean that each user may be interested in receiving only a specific subset of the data being sent. These interest subsets may change

dynamically in real time. This dynamic information dissemination problem arises in real-time distributed simulation (where update messages that are sent for specific entities are of interest to only a limited subset of the receivers), in multi-player games and data dissemination (e.g., stock quote dissemination, event notification systems) to large user communities.

This paper describes an approach that we call *active interest filtering* (AIF) to solve the dynamic information dissemination problem. AIF combines a multicast-based data delivery infrastructure with selective pruning of unwanted data within the network, based on message content. Data filtering is accomplished by dynamically installing a set of *interest filters* within network routers along the multicast distribution tree. These filters examine application-provided “content descriptors” within a packet’s application-level payload and forward over a downstream link only those packets that are desired (subscribed to) by one or more downstream receivers. Filters may also perform frequency-based filtering, forwarding only 1-out-of-every-N packets that contain a given “content descriptor.”

In this paper we describe the new architectural components required to realize active interest filtering: the per-node algorithms that actually perform filtering within an active router, and the signaling protocol that installs interest filter state in the active routers. We describe our prototype implementation effort and present measurements from a working prototype that demonstrate the advantages of active interest filtering in a large-scale ModSAF distributed simulation scenario.

The remainder of this paper is structured as follows. In section 2, we discuss several possible approaches to dynamic information dissemination and identify their advantages and disadvantages. Section 3 describes efficient algorithms that are required to make interest filtering in the forwarding path feasible. Section 4 provides an overview of a second-generation signaling protocol that we have designed for installing the interest filter state in routers. Section 5 describes a prototype implementation of our approach for the particular problem of distributed simulation, and presents measurement results. Section 6 concludes this paper.

2. Interest Management and Filtering

In many data dissemination applications, there are a potentially large number of data sources (information “publishers”) and a potentially large number of data sinks

* This work is supported in part by the Defense Advanced Research Projects Agency under contract N66001-99-C-8616

¹ Current address: ALPHATECH Inc., 50 Mall Road, Burlington, MA 01803. {zabele,mkeaton}@alphatech.com. This work was performed while the authors were with Northrop Grumman IT.

² Northrop Grumman IT Defense Enterprise Solutions, 55 Walkers Brook Dr., Reading, MA 01867-3297 mddorsch@tasc.com

³ Department of Computer Science, University of Massachusetts, Amherst MA 01003. {gezihui, jiping, kurose, jshapiro, towsley}@cs.umass.edu

(information “subscribers”), with a given receiver typically being interested in receiving data from only a small number of sources at a given time. These applications include real-time distributed simulation [Zabele 00] and multiplayer games [Levine 99], in which update messages are sent to simulated entities that are in close proximity to the sender in the simulated world; stock quote and news distribution, and large scale event notification [Carzaniga 01]. The information communication structure of these applications is often known as the publish/subscribe paradigm [Powell 96].

Several approaches have been taken towards implementing the publish/subscribe paradigm:

- **Application-Layer Publish/Subscribe.** Here, using an application-layer signaling mechanism, each data publisher (sender) advertises the types and ranges of data it can provide, and each data subscriber (receiver) sends subscription messages to the publishers to define the specific information subset in which it is interested. A sender then unicasts each data packet to only those receivers that requested the data in that packet. This approach has serious efficiency and scalability problems. Unicasting multiple copies of the same packet wastes sender capacity and network bandwidth, and the bookkeeping required at the sender may create significant state storage and signaling overhead. To overcome these difficulties, additional intermediary application-level entities can be created that route data between publishers and interested subscribers [Banavar 99, Rowstron 01].
- **Single-group multicast.** Another approach towards overcoming the inefficiency of unicasting is to multicast (or broadcast) packets. However, a typical receiver may want to receive only a subset of the data, so simple multicasting will generally result in unwanted messages being received; we refer to this situation as *over-delivery*. If the multicast session includes many receivers with highly individualized information needs, the rate of over-delivery can far exceed the rate of useful messages, wasting resources of both the network and the receiving hosts.
- **Multiple-group multicast.** In order to overcome the problems associated with application-layer and single-group-multicast approaches, recent work has focused on channel aggregation [Levine 99; Ge 01]. In this approach, a sender maps each data packet into one of a set of multicast groups or channels; each channel represents a pre-defined subset of the application data domain or content space. Receivers individually control the data subsets they will receive by joining only the appropriate multicast groups. Here multicasting is used to directly support content-based routing.

The content space is an application-specific multi-coordinate parameter space; each data packet effectively corresponds to a point in this space. There are several approaches to establishing the mapping from the content space into multicast groups. One approach, grid-based

filtering [Van Hook 94], subdivides the content space into unit cells by assigning quantized values to each of its coordinates. Each point in the content space is then defined by a vector of coordinates of a specific unit cell.

While the channel aggregation approach decreases over-delivery compared with the use of a single multicast group, there can still be significant over-delivery, since a receiver may not be interested in all of the information being sent by all of the receivers that are using a given multicast group. Channel aggregation can also suffer from excessive use of the multicast address space and excessive multicast routing overhead (when the number of groups is large). Applying the channel aggregation approach to the distributed simulation problem, for example, may easily require millions of multicast groups to achieve an acceptable over-delivery rate. Every change of interest by a receiver must be handled by the multicast tree maintenance algorithms, which may create a high rate of multicast routing churn. We also note that since calculating optimal aggregation solutions has been shown to be NP-hard [Ge 01], aggregation must be accomplished using suboptimal approximate methods, which will generally increase over-delivery in comparison to an optimal solution. Channel aggregation provides only a roughly linear improvement in delivery efficiency, while the aggregation problem potentially scales exponentially with the number of users (as implied by its NP-complete complexity). While there may be particular circumstances in which channel aggregation is feasible, this approach lacks generality and is ultimately unscalable.

3. Active Interest Filtering

Given the limitations noted above for application-layer publish subscribe, single-group multicast, and multiple group multicast, we conclude that there has been no truly scalable approach to large-scale information dissemination applications such as distributed simulation. The Active Interest Filtering (AIF) approach [Zabele 00] that will be described in the remainder of this paper has the potential to fill this gap; a related approach known as router-level filtering [Oliveira 00] has been subsequently proposed independently by others.

3.1 The AIF Model

The basis of AIF is to augment the single-group multicast with the use of active networking technology to prune unneeded application data packets that are flowing along the multicast tree, preventing almost all over-delivery. AIF has the following components:

- **Embedded Content Descriptors.** The application layer payload of each data packet includes a set of *content descriptors* that specify the coordinates of the point in content space occupied by that packet. Content descriptors are embedded at predefined locations within

S: sender
 AR: active router
 R: receiver

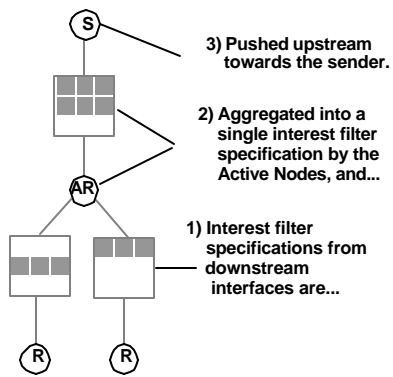


Figure 1a: Interest Filter Signaling

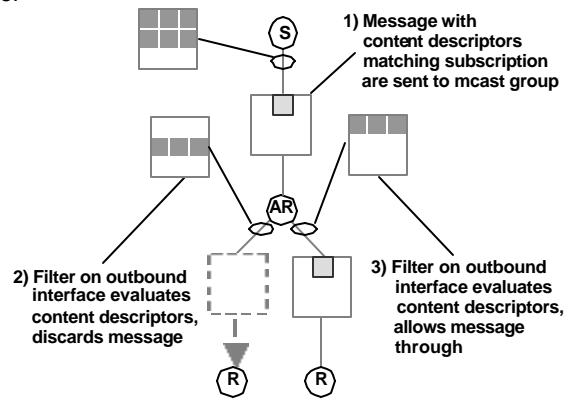


Figure 1b: Data Filtering Operations

the application payload. They may be fields that are intrinsically in the data or application-layer header fields added explicitly for this purpose.

- Interest Filter Operation.** Active interest filters within active routers examine the content descriptors of each packet in a particular multicast group in order to determine whether or not to forward a given packet. A packet is forwarded out a given interface if there is a multicast route through that interface for the multicast group and if the content descriptors in the message match the interest filters associated with that interface. The forwarding decision made by the interest filter could use simple Boolean combinations of interval tests applied to each content descriptor independently, or could use more complicated operations. For the applications we have considered in our implementations (see section 5), simple operations suffice.
- Interest Filter Signaling.** AIF interest filters are installed in routers by the Active Filter Signaling Protocol (AFSP). AFSP is used by receivers to declare or modify subscriptions to content descriptors of interest, and by routers to propagate the interest filter requests along the data delivery tree. Propagation along the multicast tree upstream towards a sender allows interest filters to be installed at the earliest possible points within the multicast tree, such that delivery efficiency is optimized. AFSP is used to dynamically modify these filters as the receiver's interest changes. AFSP merges interest filters that it receives from downstream entities (subscribers or other active routers) and sends the collective set of filter specifications upstream. A merged filter set may be either an exact representation of the collective set of subscriber interest filters, or it may be a simpler set that covers the merged filters, if needed to accommodate signaling or filtering constraints. The latter case may cause over-delivery. Merging of information flowing towards and

upstream destination in an active network is a service provided in the more general framework of the Concast architecture [Calvert 2001].

Recall that the basic AIF approach uses a single multicast group for all data. If we assume that the population of receivers in the group changes slowly compared with the rate of change of interest filters, then the multicast routing is essentially fixed on the AIF signaling time scale. Note also that AIF signaling is *path-oriented*, that is, it creates and modifies state strictly along the path of the data flow. We will see in section 4, however, that since not all nodes on the multicast data distribution tree may be active, the active routers in the tree will maintain a separate signaling tree amongst themselves that is overlaid on the multicast data distribution tree. Because AIF is path-oriented, signaling should have an intrinsically lower cost than the multicast routing calculations required by multiple-group multicast, which requires a distributed calculation among all routers of the domain. Another advantage of the AIF approach is that it allows AFSP to evolve independently of multicast routing protocols.

Figures 1a and 1b illustrate the operation of AIF for one sender and two receivers that are members of the same IP multicast group. For clarity, these examples assume two-dimensional Cartesian coordinates in the content space. Within the content space, subscribed subregions are shown as shaded cells within these squares.

In Figure 1a, interest filters from each receiver are forwarded upstream towards the sender. An active router along the path accepts interest filters from subscribers (and downstream active routers in the general case) and merges the filters into a single, more general interest filter that it forwards upstream. The resulting data delivery function is illustrated in Figure 1b. The sender transmits a message with embedded content descriptors to the multicast group. Upon receipt of the message, the active router compares the message content

descriptors with the interest filters associated with each outbound interface. The message is forwarded only out those interfaces with matching interest filters. In this example, the active router has forwarded an aggregate filter specification that is exact in the sense that no simplifications were made, e.g., to meet performance goals. Hence, even though both subscribers are in fact members of the same single multicast group, they receive only data in which they have specified an interest, i.e., there is no over-delivery.

3.2 Requirements for AIF

The design of AIF can be refined by imposing additional system requirements. The following requirements are characteristic of the distributed simulation application in particular. However, in varying degree they should be appropriate to any dynamic information dissemination problem.

- **Signaling for Many-to-many Delivery.** Within a distributed simulation environment, each end-host will typically act as both a sender and a receiver on the same multicast group. This requires that AFSP accommodate many-to-many multicast distribution.
- **Datagram Service Model.** For real-time distributed simulations, minimizing communication latency is critical. Hence these applications use a datagram network service model (e.g., UDP) rather than reliable delivery for the majority of the data traffic. This allows pruning of unneeded data to any given receiver without disrupting the underlying transport mechanism.
- **Efficient Filtering.** A key goal of large-scale distributed simulation is supporting large numbers of simultaneous entities. Hence the signaling and filtering operations required by AIF must not significantly impact throughput or delivery latency.
- **Correctness Property.** The data dissemination problems with which we are concerned have an implicit correctness property that must be preserved: all data needed by a receiver host must be deliverable. (We cannot guarantee that it actually is delivered, because of the datagram delivery model.) Thus, while some over-delivery is allowed, under-delivery must be minimal.
- **Graceful Service Degradation.** AIF should continue to provide benefit when active routers have insufficient resources to fully support the requested filtering and signaling services. This can be achieved by approximating complex filters or filtering requests with simpler expressions, as described below. This will maintain the correctness property if the approximations are a superset (i.e., a covering) of the exact filters. For example, an active router can react to overload conditions by simply bypassing the filtering operations on multicast packets being forwarded. All downstream entities are still provided all needed data, with the downside that they may experience an increased level of over-delivery. Moreover, it should be possible to configure filters so that if filtering operations can only be partially completed (e.g., because

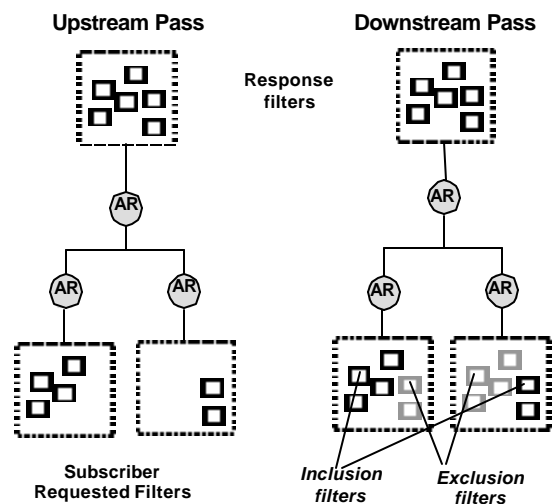


Figure 2: NBMA link below uppermost active router causes overdelivery to each of the two downstream routers

of deadline scheduling), the filtering still provides some benefit.

- **Operation with Partial AIF Availability.** AIF must provide enhanced performance even if some routers along the path do not support active interest filtering. At worst, the result of such partial coverage should be some additional over-delivery. In the limit of no AIF-capable routers, of course, the problem reduces to the single-group multicast scenario described earlier.
- **Robustness.** Signaling must be robust. The system must continue to function correctly in the presence of lost signaling messages (e.g., lost interest filter specifications) and be able to recover from temporary or permanent host, router or link outages.

3.3. Distributed Filtering

In order to accomplish the requirements described above, AIF takes a *distributed* approach towards filtering. To illustrate the distributed approach, suppose that all receivers in a multicast distribution tree have identical interest filters. Then each router node downstream from the node closest to the source will needlessly repeat an identical set of filtering operations. Instead, we can require that each node provide a description of the filtering that has been performed upstream to nodes immediately downstream. Each downstream node then needs to perform only *incremental* filtering. This approach, which we call *distributed filtering*, implies a bi-directional exchange of filtering information among neighboring active nodes in the multicast distribution tree, in order for them to set up the appropriate filtering state.

In distributed filtering, filter information is passed upstream as shown in the basic AIF signaling illustrated in Figure 1a. Interest filter specifications, or *request filters*, are forwarded upstream towards the senders, with each active router merging

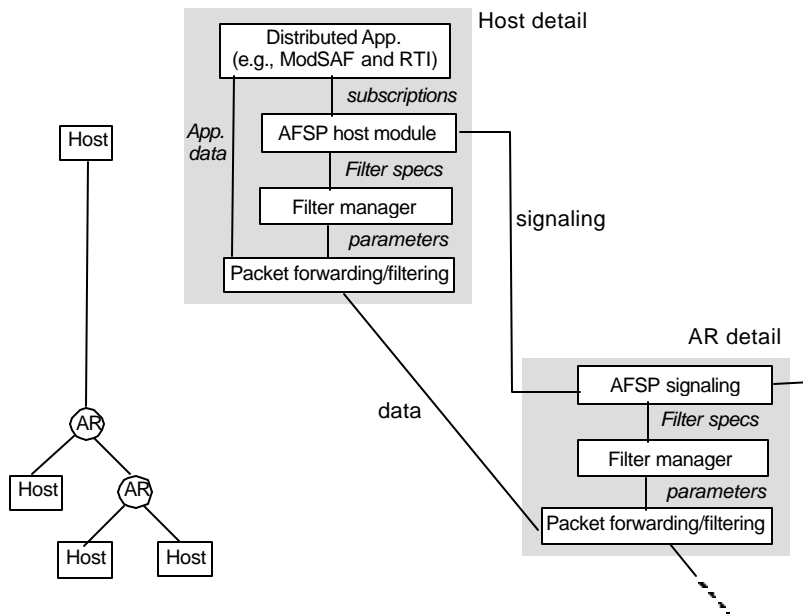


Figure 3: Host and active router detail

request filters it has received from downstream receivers (with respect to a given source).

Once an active node has received request filters from all downstream receivers, it can determine the actual content that it will be sending over a given downstream interface, and thus passes this information to active downstream nodes. The resulting filter specification that is then sent downstream towards the receivers to characterize the content being forwarded to the next-hop downstream active node is known as a *response* filter. Each active router along the forwarding path computes the differences between the response filter(s) it receives from upstream and the set of request filters it receives on each of its downstream interfaces. This yields the incremental filtering needed for that interface.

At first glance, it might seem unnecessary for a downstream active router to receive any filtering information from its immediate upstream neighbor. After all, the downstream active router has already informed (via its upstream request filter) its upstream active router of precisely the content it (the downstream router) wishes to receive. If the upstream active router can perform content filtering, why would the downstream router expect to receive anything from the upstream router other than that which it specified in its request filter? The reasons for such over-delivery to a router are threefold: (i) an upstream node may install an approximate filter for performance reasons (Section 4), (ii) there may be non-active routers in the path upstream, or (iii) there may be a broadcast or non-broadcast multi-access (NBMA) link upstream. In cases (ii) and (iii), a multicast branch point that does not have corresponding interest filter merging may force an upstream active router to over-deliver. This is shown in Figure 2. Here, the uppermost active router will be transmitting content over its single downstream interface that matches the *union* of the filters received from the two downstream routers.

Thus, each of the downstream routers will be receiving content that it did not request.

Note that the response filter will generally cover the request filters, i.e., it will define the same or a larger content space region than the request filters. Therefore, portions of the incremental filtering process may best be performed using *exclusion* filtering, where packets not falling within a given set of filters are forwarded rather than dropped. This is an inversion of the *inclusion* filtering implied by single-pass filter setup, where packets not falling within the given set of filters are dropped.

4. Active Filtering Algorithms

Our approach distinguishes filter management functions that occur in the control plane from filtering operations that occur in the data plane, as shown in Figure 3. Section 5 describes the AFSP signaling protocol that operates in the control plane to deliver interest filter specifications to each node in the path. A software component in each node called the Filter Manager (FM) then computes and installs the appropriate content filters for operations within the data plane. We now discuss algorithms used by FM to improve active interest filtering efficiency within the data plane.

The basic performance requirement for filtering is sustained operation at router line speeds with useful filter complexity. Our research has focused on filtering mechanisms that use only simple bit-field comparisons on fixed and predefined locations in the data packets. The complexity of filtering operations depends upon the application area, but it generally scales with the number of receivers. The results in Section 6 show that highly complex filtering can be performed on existing router platforms with negligible throughput loss. Furthermore, the graceful service degradation principle means that

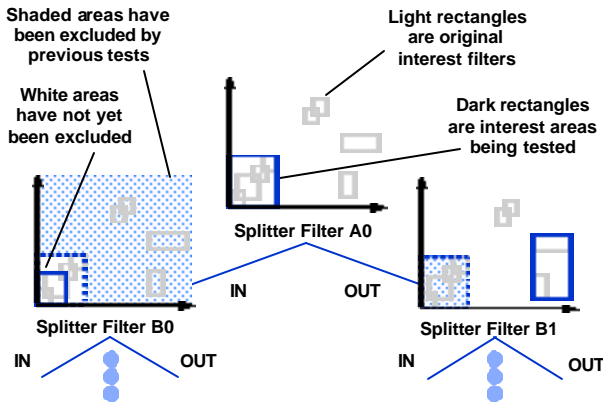


Figure 4: filter trees

throughput loss can be limited at the expense of some over-delivery. The result is a highly scalable mechanism.

Our approach to filtering incorporates several performance acceleration techniques: tree-based filtering constructs within the active routers, filter simplification to reduce the overall number of filters to accommodate physical storage or signaling constraints, and exclusion/inclusion minimization. We now present the basic ideas behind each of these techniques.

Filter Trees

The scalability of filtering performance with filter complexity is significantly improved if interest filters are organized as decision trees. These decision trees may be created by recursively dividing sets of interest filters into smaller subsets. Each division defines a simple branching decision to assess subset membership, e.g., a decision half-space or a single covering filter containing all members of one subset. Such a branching decision forms a node in the decision tree, and the two disjoint subsets resulting from that decision are represented by links attached to the node. Tree filters are applied by sequentially evaluating the branching decisions using the message content descriptor values, following the resulting links until a terminal node is reached. The first few stages of a filtering tree for a nominal set of eight interest filters is shown in Figure 4.

Assuming the decision tree that is constructed is a balanced binary tree, tree filtering scales logarithmically (\log_2) with the number of filter regions. This is a significant improvement over applying each interest filter in sequence, which scales linearly. Experimentally, this approach often achieves the theoretical lower bound on the average number of required decisions [Kennett 02].

Filter Simplification

The final strategy for limiting the increase of AIF load with increasing filter complexity is to simplify filter specifications during merging. For example, a single larger interest region may be used in lieu of a large number of small interest regions as long as the larger region covers the smaller regions.

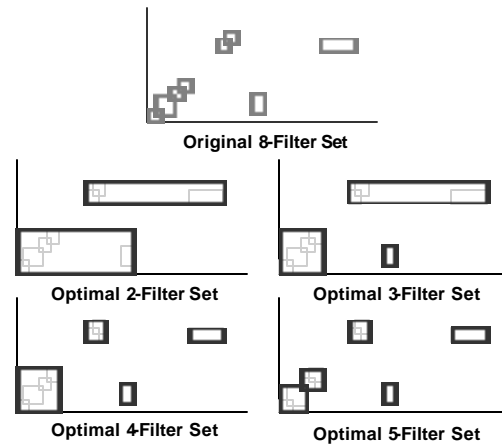


Figure 5: Optimal Minimum Volume Specification

The results of optimally merging a set of 8 interest filters into a smaller target number of filters are shown in Figure 5. Here, optimal means that the added volume (i.e., the volume within the simplified filter that was not in one of the original filters) is minimized. Assuming a uniform traffic density model, minimizing the volume is equivalent to minimizing over-delivery. Although we have implemented an optimal N -dimensional filter simplification capability as part of our experimental work, optimal filter simplification in general has NP-hard computational complexity [Burkard 99]. To circumvent this problem, we have adapted the Fast Pairwise Nearest Neighbor (FastPNN) algorithm [Equitz 89] to perform fast, approximate filter simplification [Zabele 02]. The FastPNN algorithm is attractive as its results have proven quite good (at least over the range where comparisons with optimal performance are feasible), and has an attractive $O(N \log(N))$ computational complexity: for N filters.

Inclusion and Exclusion Filtering

It was noted earlier that distributed filtering, which installs incremental filters, can result in exclusion filtering. Since the number of exclusion regions can be much smaller than the number of inclusion regions, this may allow the tree constructor to produce a smaller tree using exclusion tests, which may significantly reduce overall filter processing load. More generally, with some caveats, the filter tree constructor is free to combine inclusion tests (i.e., a packet is within a user requested region in the content space) with exclusion tests (i.e., a packet is within a region *not* requested by any user) at each branch point in the tree, allowing minimization of the number of tests needed to implement the tree, similar to combinatorial logic minimization. See Figure 6. Although constructing optimal decision trees that allow the use of inclusion or exclusion filters at any given decision point can require significant computational resources (the load likely increases exponentially with the number of filters N), we have developed tree construction mechanisms with computational requirements that scale linearly with the number of filters that have demonstrated performance near the theoretical bounds.

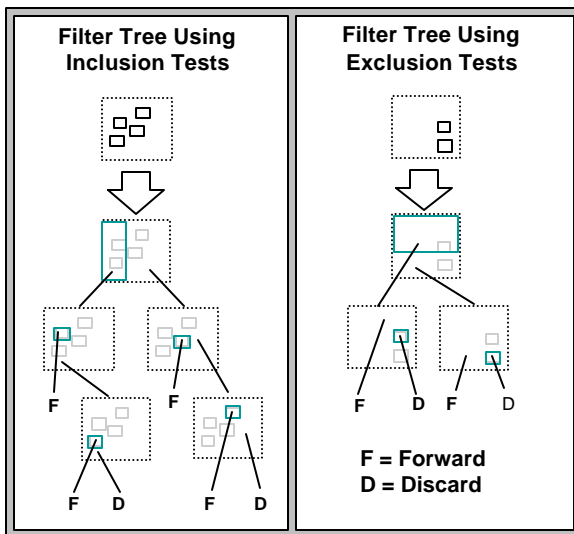


Figure 6: Filter Trees constructed using only Inclusion tests (left) and with reduced overhead using exclusion tests (right)

The theoretical bounds and tree construction algorithms are presented in [Kennett 02].

4. AFSPv2: The Active Filter Signaling Protocol

The Active Filter Signaling Protocol (AFSP) is used to reliably install, maintain and remove interest filter states on active routers or end hosts along data dissemination paths. In this section, we describe AFSPv2, which builds upon the lessons learned with AFSPv1 [Zabele 01].

Given our discussions above, AFSP has been designed with the following considerations in mind:

- **Path-Oriented Signaling.** AFSP performs path-oriented signaling to control filter distribution for the general case of many-to-many delivery of data packets. AFSP signaling interacts with the filter manger, so that together they can set up interest filters for data packets in a particular session. Here, a session is defined by the multicast IP destination address and the UDP destination port to which the application data is addressed. Signaling is independent for different sessions.
- **Receiver Orientation.** AFSP carries interest filter requests upstream and may carry interest filter responses downstream, relative to the direction of data flow. Since the upstream signaling (“publication”) is fundamental, we say that AFSP is receiver-oriented.
- **Filter Merging.** As shown in Figure 1, AFSP merges filter requests for the given multicast group as they flow upstream along the multicast tree. Similarly, AFSP merges filter responses from an upstream router, since every sender may publish a different region of the content space. Merging may simply form the union of the individual filters sets, or it may simplify the union as needed to accommodate resource constraints.

- **Robustness.** AFSP must provide reliable installation and modification of filter state, performing hop-by-hop reliable signaling between AFSP components. Currently, TCP is used to provide reliable communication between AFSP components. To achieve robust removal of state, AIF signaling uses soft-state techniques in which routers and hosts periodically refresh their filter specifications.
- **Dynamic Variation.** The normal case in active information dissemination will involve changing request and response filters. Thus, AFSP must be able to provide signaling in support of incremental addition, deletion, and modification of filters.

AFSP signaling propagates publish/subscribe requests from end users, triggers filter installation and update on active nodes, and adapts to the changes of underlining data dissemination topology. It communicates with two adjacent modules, the Active Topology Discovery Protocol (ATDP) module and the Filter Manager (FM). FM has been described above, and ATDP will be described below. Based on the active topology reported by ATDP, AFSP notifies FM of the filters that should be installed, updated, or removed.

By querying ATDP, AFSP acquires the virtual interfaces (VIFs) and the peer AFSPs that connect to the corresponding VIFs on the data path of a given multicast session. An AFSP peer p is a *parent* of an active node q if p is the nearest active node located at the upstream of q along the data transmission path from one or more publisher(s). Similarly, an AFSP peer p is a *child* of q if p is the nearest active node located downstream of current node q along the data transmission path to one or more subscriber(s). An AFSP peer can be both a parent and a child depending on its position in the multicast trees associated with different senders in the multicast session. Since the peer relationship is logical, a pair of peers is not necessarily connected directly. Thus, on each VIF, there may be multiple parent and child associations.

AFSP has two primary signaling messages, the subscription request, `IF_REQUEST`, and the response message, `IF_RESPONSE`. These messages are transmitted hop-by-hop reliably between a pair of peer AFSPs or between an AFSP on an end host and its applications. Each subscription or response message carries a list of interest filters, a requested operation (e.g., install, modify, remove), information about the initiator of this filter's request, as well as the identifier (IP address and VIF identifier) of the message sender. When AFSP receives an `IF_REQUEST` or an `IF_RESPONSE` message, it contacts ATDP, querying the VIFs that are related to this message, so that AFSP can correctly propagate this control message along the multicast tree. In the meantime, on receiving an `IF_REQUEST` or an `IF_RESPONSE` message, an AFSP calls FM, passing it a list of filters and the requested operation, so that FM can update (add, remove, modify) its filter states. Based on the updated filter state information returned by FM and the VIFs returned by ATDP, an AFSP decides whether or not, and where, to forward a control message, as well as adjusting the filter states that AFSP maintains.

For FM to build optimal filtering trees within a router (see section 4), it is desirable to know not only the data requested by downstream receivers, but also the data that will be transmitted by the upstream peers. To achieve this, AFSP uses a *two pass signaling* algorithm. In the first pass, an AFSP propagates the filter setup requests received from downstream towards all the parent peers (except those connecting to the same VIF where the peer sending these requests is attached). Meanwhile, AFSP notifies FM that the interest filter states have been updated for that VIF. In the second pass, an AFSP sends `IF_RESPONSE` messages to each of the child peers connected to a VIF, notifying them of changed filters states on that VIF (installations, modifications, and removals).

The Active Topology Discovery Protocol

The Active Topology Discovery Protocol (ATDP) isolates those aspects of AFSP signaling that are tied to the underlying multicast routing. The reasons for this modularization are twofold. First, the mechanisms for maintaining pointers to neighboring active routers and recovering from topology changes are logically distinct from the filter signaling that must make use of these pointers. Second, many active multicast protocols share the need to maintain a signaling overlay. Thus, a generic protocol for this purpose is of independent interest.

ATDP provides a general purpose mechanism for establishing reliable communication channels between neighboring active routers and making these channels available to higher-level protocols - AFSP in this case. The effect of ATDP executing on all active routers within a multicast tree is to create a bi-directional signaling overlay that traces the underlying active tree topology and adapts to changes in its topology. Each hop in the signaling overlay connects two active routers that are typically separated by non-active routers.

The essential topology discovery mechanism of ATDP uses *source path messages* (SPMs) originally introduced by PGM [Speakman 00] to allow an active router to discover its upstream neighbor. SPMs originate at each multicast source and are sent periodically to the group address by ATDP. Each SPM packet contains a `PARENT` field that is overwritten by the active node with its own contact IP address prior to forwarding. In this way, each node learns the identity of its parent and, as SPMs are multicast repeatedly, learns if the parent has changed. The identity of the parent is indexed by the pairing (S, G) of source S and multicast group G . In addition, the SPM contains an outgoing interface (`OIF`) field that is overwritten with the identifier of the outgoing interface identifier for each forwarded copy. The `OIF` field allows a child to identify the correct parent downstream interface when signaling its parent.

Having learned of its parent via an SPM, the active node will attempt to establish a reliable bidirectional communication channel - which we refer to as an *ATDP channel* - with that parent. ATDP makes this channel available to other local protocol modules for hop-by-hop reliable data exchange. Furthermore, ATDP multiplexes signaling traffic between two

hosts over a single channel, even if those hosts have parent-child relationships in multiple source-based trees. Thus, if the ATDP channel is connection-oriented, the connection is shut down only if it is no longer used for any (S, G) pair.

A simple programming interface provides higher-level protocol modules with access to the ATDP channels for sending and receiving signaling data to neighbors. While messages can be addressed directly to specific neighbors, it is often more convenient to address messages to a group of neighbors based on topological properties without knowing their explicit identities. For example, a message may be addressed to all children or to the parent for a particular (S, G) . Most important for AFSP, a wildcard can be used in place of an explicit source, effectively addressing neighbors in the union of all source-based trees for a particular multicast group. This feature is important for AFSP signaling as it allows, for example, subscription messages to be forwarded toward all potential sources. Although a single message may thus be destined for multiple neighbors, the protocol ensures that each message is transmitted at most once over any ATDP channel.

A more complete description of ATDP is available in [Shapiro 01].

5. A Prototype Implementation

The active filtering approach described in this paper was implemented to demonstrate its use in solving DARPA's challenge problem of leveraging active networks to address distributed simulation performance and scalability limitations. The resulting demonstration architecture not only included the prescribed filtering, filter management, and active signaling components, but was also integrated with a DoD-standard High Level Architecture (HLA)-compatible [DoD 98] Run Time Infrastructure (HLA/RTI) [Fujimoto 98]. The architecture was used to demonstrate achievable performance gains using the U.S. Military standard Modular Semi-automated Forces (ModSAF) simulation package.

In order to measure the performance gains of our active filtering implementation, we constructed a ModSAF simulation containing 18 ModSaf federates, with each federate running on a separate host. To ensure realism, accuracy, and the defensibility of our results, the simulation was derived from historical data of the U.S. Seventh Corps attack on the Iraqi Republican Guard during Operation Desert Storm, a battle that remains of considerable interest and the subject of much analysis within the U.S. Military. Each federate hosted approximately 70 simulated entities (e.g., tanks, jeeps, etc) for an overall simulation size in excess of 1200 active entities.

The network topology connected three hosts to an edge active router, with three edge active routers connected to a core active router; the two core routers were then interconnected. The 18 connected hosts were running in the Utah Emulab testbed [Emulab 2002]. AFSPv1 signaling was performed for these measurements.

Figure 7 plots the host number on the X-axis versus the number of packets per second of unreliable, multicast, UDP ModSAF packets received at each host on the Y-axis. With

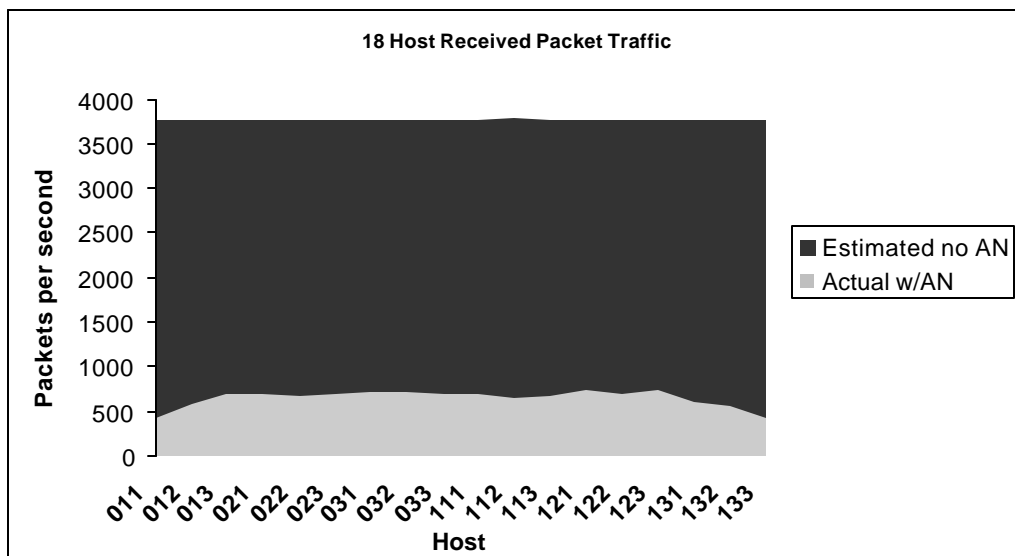


Figure 7: Traffic intensities at simulation hosts with and without active networking

active filtering, each host receives on the order of 700 packets per second. Without active filtering, each host would receive all MoDSAF packets transmitted by the other hosts, yielding a receive rate of approximately 3700 packets per second. For this configuration, *active filtering yields more than a five-fold decrease in received traffic*. As the size of the simulation scales, and/or the number of filtering points increases, the gains achievable with active filter will increase further. The data presented here represents a demonstration and first reporting of the actual gains achievable in a specific configuration. As discussed below, we are currently working on instrumenting and measuring an even larger-scale distributed simulation, where we expect even larger gains to be realized, as noted above.

6. Summary

This paper has provided an overview of the SANDS project, which uses active networking to develop a new approach to real-time, content-based information dissemination. Our approach is based on installing and controlling dynamically-established content-based active filters in intermediate active routers in an IP multicast distribution tree. Active filters prune unneeded information as early as possible in the distribution tree, ensuring that only data desired by a receiver actually reaches that receiver. We described the per-node algorithms that implement active filtering, the AFSP signaling protocol that installs interest filter state, the ATDP protocols that maintains the AFSP signaling tree topology, a prototype implementation effort. Our measurements demonstrated the advantages of active interest filtering in an actual ModSAF distributed simulation scenario.

Active networks can provide scalable and efficient mechanisms for dynamic information dissemination within applications such as distributed simulation. This approach offers several important benefits. *(i)* Because data forwarding is decoupled from multicast route setup, an entity can rapidly and dynamically select which data a router will forward. This selection can be done on a much shorter timescale than existing multicast group join/leave mechanisms. *(ii)* Because each entity can install its own filters, information filtering is accomplished in a receiver-driven manner, allowing each entity to customize its filters according to its own need. This decentralized approach allows active filtering to scale well as the number of entities grows large. *(iii)* Because active filtering is performed at a routing point, filtering can also be dependent on the state (e.g., congestion-level) at that router. In particular, this allows both entities and network routers to determine which data should be shed in times of congestion.

Our current work is aimed at implementing AFSPv2, measuring active filtering performance in even larger scale environments, and in extending and generalizing the notion of congestion-level filtering.

References

- [Banavar 99] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajaram, R. Strom, D. Sturman, "An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems," *1999 Int. Conference on Distributed Computing Systems*.
- [Burkard 99] R. E. Burkard and M. M. Miatselski, "Volume Maximization and Orthoconvex Approximation of Orthogons," *Computing*, Vol. 63, 1999.
- [Calvert 2001] K. Calvert, J. Griffioen, B. Mullins, A. Sehgal, S. Wen, "Concast: Design and Implementation of an Active

- Network Service," *IEEE Journal on Selected Area in Communication (JSAC)*, Volume 19, No. 3. March, 2001.
- [Carzaniga 2001]** A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Transactions on Computer Systems*, 19(3):332-383, Aug 2001.
- [DoD 98]** Department of Defense High Level Architecture Interface Specification, Version 1.3, DMSO, April 1998
- [Emulab 2002]** Emulab.net: The Utah Emulab Testbed, <http://www.emulab.net/pubs.php3>
- [Equitz 89]** W. H. Equitz, A New Vector Quantization Clustering Algorithm, *IEEE Trans. ASSP*, Vol. 37, No. 10, October 1989
- [Fujimoto 98]** R. M. Fujimoto and P. Hoare, HLA RTI Performance in High Speed LAN Environments, *Proceedings of the Fall Simulation Interoperability Workshop*, September 1998
- [Gauthier 98]** L. Gauthier, C. Diot, "Design and Evaluation of MiMaze, a Multi-Player Game on the Internet," *Proc. 1998 International Conference on Multimedia Computing and Systems*.
- [Ge 2001]** Z. Ge, M. Adler, J. Kurose, D. Towsley, and S. Zabele, "Channelization Techniques for Large Scale Data Dissemination," *Proceedings of ICNP 2001*.
- [Kennett 02]** R. Kennett and S. Sabele, "Use of Decision Trees for Packet Filtering in Active Networks," manuscript, 2002.
- [Levine 99]** B. Levine, J. Crowcroft, C. Diot, J.J. Garcia-Luna-Aceves, J. Kurose, "Consideration of Receiver Interest for IP Multicast Delivery," *Proc. 1999 IEEE Infocom*.
- [Oliveira 00]** M. Oliveira, J. Crowcroft, C. Diot. "Router level filtering for receiver Interest Delivery," Proceeding of *Workshop on Networked Group Communications (NGC) 2000*.
- [Powell 96]** David Powell (Guest editor). "Group Communication", pages 50-97, *Communications of the ACM*, Vol. 39, No. 4, April 1996.
- [Rowstron 01]** A. Rowstron, A-M. Kermarrec, P. Druschel and M. Castro, "SCRIBE: The design of a large-scale event notification infrastructure," submitted for publication June 2001.
- [Shapiro 01]** J. Shapiro, J. Kurose, D. Towsley, S. Zabele, "Topology Discovery Service for Router-Assisted Multicast Transport," Technical Report, Department of Computer Science, University of Massachusetts, December 2001. To appear in *Proceedings IEEE OpenArch 2002*.
- [Speakman 00]** T. Speakman et al., "PGM Reliable Transport Protocol," IETF Internet draft draft-speakman-pgm-spec-04.txt, 2000.
- [Van Hook 94]** D. Van Hook, S. Rak, and J. Calvin, Approaches to Relevance Filtering, 94-11-144, Eleventh Workshop on Standards for the Interoperability of Distributed Simulations, September 26-30, 1994.
- [Zabele 00]** S. Zabele, T. Stanzione, J. Kurose, and D. Towsley, "Improving Distributed Simulation Performance Using Active Networks," *Proceedings of World Multi Conference 2000*, January 23-27, 2000, San Diego, CA.