Overcast: Reliable Multicasting with an Overlay Network

John Jannotti David K. Gifford Kirk L. Johnson
M. Frans Kaashoek James W. O'Toole, Jr.
Cisco Systems
{jj,gifford,tuna,kaashoek,otoole}@cisco.com

Abstract

Overcast is an application-level multicasting system that can be incrementally deployed using today's Internet infrastructure. These properties stem from Overcast's implementation as an *overlay network*. An overlay network consists of a collection of nodes placed at strategic locations in an existing network fabric. These nodes implement a network abstraction on top of the network provided by the underlying *substrate* network.

Overcast provides scalable and reliable single-source multicast using a simple protocol for building efficient data distribution trees that adapt to changing network conditions. To support fast joins, Overcast implements a new protocol for efficiently tracking the global status of a changing distribution tree.

Results based on simulations confirm that Overcast provides its added functionality while performing competitively with IP Multicast. Simulations indicate that Overcast quickly builds bandwidth-efficient distribution trees that, compared to IP Multicast, provide 70%-100% of the total bandwidth possible, at a cost of somewhat less than twice the network load. In addition, Overcast adapts quickly to changes caused by the addition of new nodes or the failure of existing nodes without causing undue load on the multicast source.

1 Introduction

Overcast is motivated by real-world problems faced by content providers using the Internet today. How can bandwidth-intensive content be offered on demand? How can long-running content be offered to vast numbers of clients? Neither of these challenges are met by today's infrastructure, though for different reasons. Bandwidth-intensive content (such as 2Mbit/s video) is impractical because the bottleneck bandwidth between content providers and consumers is considerably less than the natural consumption rate of such media. With currently available bandwidth, a 10-minute news clip might require an hour of download time. On the other hand, large-scale (thousands of simultaneous viewers) use of even moderate-bandwidth live video streams (perhaps 128Kbit/s) is precluded because network costs scale linearly with the number of consumers.

Overcast attempts to address these difficulties by combining techniques from a number of other systems. Like IP Multicast, Overcast allows data to be sent once to many destinations. Data are replicated at appropriate points in the network to minimize bandwidth requirements while reaching multiple destinations. Overcast also draws from work in caching and server replication. Overcast's multicast capabilities are used to fill caches and create server replicas throughout a network. Finally Overcast is designed as an *overlay network*, which allows Overcast to be incrementally deployed. As nodes are added to an Overcast system the system's benefits are increased, but Overcast need not be deployed universally to be effective.

An Overcast system is an overlay network consisting of a central source (which may be replicated for fault tolerance), any number of internal Overcast nodes (standard PCs with permanent storage) sprinkled throughout a network fabric, and standard HTTP clients located in the network. Using a simple tree-building protocol, Overcast organizes the internal nodes into a distribution tree rooted at the source. The tree-building protocol adapts to changes in the conditions of the underlying network fabric. Using this distribution tree, Overcast provides large-scale, reliable multicast groups, especially suited for on-demand and live data delivery. Overcast allows unmodified HTTP clients to join these multicast groups.

Overcast permits the archival of content sent to multicast groups. Clients may specify a starting point when joining an archived group, such as the beginning of the content. This feature allows a client to "catch up" on live content by tuning back ten minutes into a stream, for instance. In practice, the nature of a multicast group will most often determine the way it is accessed. A group containing stock quotes will likely be accessed live. A group containing a software package will likely be accessed from start to finish; "live" would have no meaning for such a group. Similarly, high-bandwidth content can not be distributed live when the bottleneck bandwidth from client to server is too small. Such content will always be accessed relative to its start.

We have implemented Overcast and used it to create a data distribution system for businesses. Most current users distribute high quality video that clients access on demand. These businesses operate geographically distributed offices and need to distribute video to their employees. Before using Overcast, they met this need with low resolution Web-accessible video or by physically reproducing and mailing VHS tapes. Overcast allows these users to distribute high-resolution video over the Internet. Because high quality videos are large (Approximately 1 Gbyte for a 30 minute MPEG-2 video), it is important that the videos are efficiently distributed and available from a node with high bandwidth to the client. To a lesser extent, Overcast is also being used to broadcast live streams. Existing Overcast networks typically contain tens of nodes and are scheduled to grow to hundreds of nodes.

The main challenge in Overcast is the design and implementation of protocols that can build efficient, adaptive distribution trees without knowing the details of the substrate network topology. The substrate network's abstraction provides the appearance of direct connectivity between all Overcast nodes. Our goal is to build distribution trees that maximize each node's bandwidth from the source and utilize the substrate network topology efficiently. For example, the Overcast protocols should attempt to avoid sending data multiple times over the same physical link. Furthermore, Overcast should respond to transient failures or congestion in the substrate network.

Consider the simple network depicted in Figure 1. The network substrate consists of a root node (R), two Overcast nodes (O), a router, and a number of links. The links are labeled with bandwidth in Mbit/s. There are three ways of organizing the root and the Overcast nodes into a distribution tree. The organization shown optimizes bandwidth by using

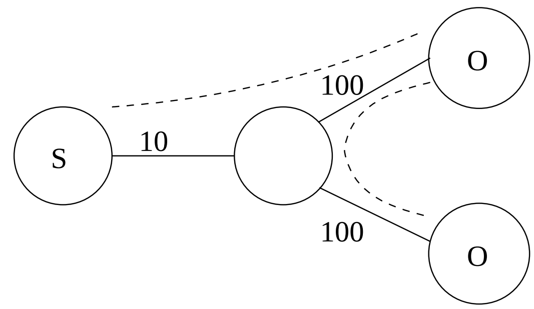

Figure 1: An example network and Overcast topology. The straight lines are the links in the substrate network. These links are labeled with bandwidth in Mbit/s. The curved lines represent connections in the Overlay network. **S** represents the source, **O** represents two Overcast nodes.

the constrained link only once.

The contributions of this paper are:

- A novel use of overlay networks. We describe how reliable, highly-scalable, application-level multicast can be provided by adding nodes that have permanent storage to the existing network fabric.
- A simple protocol for forming efficient and scalable distribution trees that adapt to changes in the conditions of the substrate network without requiring router support.
- A novel protocol for maintaining global status at the root of a changing distribution tree. This state allows clients to join an Overcast group quickly while maintaining scalability.
- Results from simulations that show Overcast is efficient. Overcast can scale to a large number of nodes; its efficiency approaches router-based systems; it quickly adjusts to configuration changes; and a root can track the status of an Overcast network in a scalable manner.

Section 2 details Overcast's relation to prior work. Overcast's general structure is examined in Section 3, first by describing overlay networks in general, then providing the details of Overcast. Section 4 describes the operation of the Overcast network performing reliable application-level multicast. Finally, Section 5 examines Overcast's ability to build a bandwidth-efficient overlay network for multicasting and to adapt efficiently to changing network conditions.

2 Related Work

Overcast seeks to marry the bandwidth savings of an IP Multicast distribution tree with the reliability and simplicity of store-and-forward operation using reliable communication between nodes. Overcast builds on research in IP multicast, content distribution (caching, replication, and content routing), and overlay networks. We discuss each in turn.

IP Multicast IP Multicast [11] is designed to provide efficient group communication as a low level network primitive. Overcast has a number of advantages over IP Multicast. First, as it requires no router support, it can be deployed incrementally on existing networks. Second, Overcast provides bandwidth savings both when multiple clients view content simultaneously and when multiple clients view content at different times. Third, while reliable multicast is the subject of much research [19, 20], problems remain when various links in the distribution tree have widely different bandwidths. A common strategy in such situations is to decrease the fidelity of content over lower bandwidth links. Although such a strategy has merit when content *must* be delivered live, Overcast also supports content types that require bit-for-bit integrity, such as software.

Express [15] is a single-source multicasting system that addresses some of IP Multicast's deficits. Express alleviates difficulties relating to IP Multicast's small address space, susceptibility to denial of service attacks, and billing difficulties which may lie at the root of IP Multicast's lack of deployment on commercial networks. In these three respects Overcast bears a great deal of similarity to Express. Overcast differs mainly by stressing deployability and flexibility. Overcast does not require router modifications, simplifying adoption and increasing flexibility. Although Overcast provides a useful range of functionality, we recognize that there needs for which Overcast may not be suited. Express standardizes a single model in the router which works to lock out applications with different needs.

Content Distribution Systems Others have advocated distributing content servers in the network fabric, from initial proposals [10] to larger projects, such as Adaptive Caching [26], Push Caching [14], Harvest [8], Dynamic Hierarchical Caching [7], Speculative Data Dissemination [6], and Application-Level Replication [4]. Overcast extends this previous work by building an overlay network using a self-organizing algorithm. This algorithm, operating continuously, not only eliminates the need for manually determined topology information when the overlay network is created, but also reacts transparently to the addition or removal of nodes in the running system. Initialization, expansion, and fault tolerance are unified.

A number of service providers (e.g., Adero, Akamai, and Digital Island) operate content distribution networks, but in-depth information describing their internals is not public information. FastForward's product is described below as an example of an overlay network.

Overlay Networks A number of research groups and service providers are investigating services based on overlay networks. In particular, many of these services, like Overcast, exist to provide some form of multicast or content distribution. These include End System Multicast [16], Yoid [13] (formerly Yallcast), X-bone [24], RMX [9], FastForward [1], and PRISM [5]. All share the goal of providing the benefits of IP multicast without requiring direct router support or the presence of a physical broadcast medium. However, except Yoid, these approaches do not exploit the presence of permanent storage in the network fabric.

End System Multicast is an overlay network that provides small-scale multicast groups for teleconferencing applications; as a result the End System Multicast protocol (Narada) is designed for multisource multicast. The Overcast protocols different from Narada in order to support large-scale multicast groups.

Yoid is a generic architecture for overlay networks with a number of new protocols, which are in development. The most striking difference between Yoid and Overcast is in approach. Yoid strives to be a general purpose overlay network and content distribution toolkit, addressing applications as diverse as netnews, streaming broadcasts, and bulk email distribution. While these goals are laudable, we believe that because Overcast is more focused on providing single-source multicast our protocols are simpler to understand and implement. Nonetheless, there remains a great deal of similarity between Overcast and Yoid, including url-like group naming, the use of disk space to "time-shift" multicast distribution, and automatic tree configuration.

X-bone is also a general-purpose overlay network that can support many different network services. The overlay networks formed by X-bone are meshes, which are statically configured.

RMX focuses on real-time reliable multicast. As such, its focus is on reconciling the heterogenous capabilities and network connections of various clients with the need for reliability. Therefore their work focuses on *semantic* rather than *data* reliability. For instance, RMX can be used to change high resolution images into progressive JPEGs before transmittal to underprovisioned clients. Our work is less concerned with interactive response times. Overcast is designed for content that clients are interested in only at full fidelity, even if it means that the content does not become available to all clients at the same time.

FastForward Networks produces a system sharing many properties with RMX. Like RMX, FastForward focuses on real-time operation and includes provisions for intelligently decreasing the bandwidth requirements of rich media for low-bandwidth clients. Beyond this, FastForward's product differs from Overcast in that its distribution topology is statically configured by design. Within this statically configured topology, the product can pick dynamic routes. In this way FastForward allows experts to configure the topology for better performance and predictability while allowing for a limited degree of dynamism. Overcast's design seeks to minimize human intervention to allow its overlay networks to scale to thousands of nodes. Similarly, FastForward achieves fault tolerance by statically configuring distribution topologies to avoid single points of failure, while Overcast seeks to dynamically reconfigure its overlay in response to failures.

PRISM is an architecture for distributing streaming media over IP. Its architecture bears some similarity to Overcast, but their work appears focused on the naming of content and the design of interior nodes of the system. PRISM's high level design includes an overlay based content distribution mechanism, but it is assumed that such a system can be "plugged in" to the rest of PRISM. Overcast could provide that mechanism.

Active Services Active Services [2] is a framework for implementing services at the application-level throughout the fabric of the network. In that sense, there is a strong similarity in mindset between our works. However, Active Services must contend with the difficulty of sharing the resources of a single computer among multiple services, a difficulty we avoid by using dedicated nodes. Perhaps because of this challenge, Active Service applications have focused on real-time multimedia streaming, an application with transient resource needs. Our application uses large amounts of disk space for long periods of time which is problematic in a shared environment.

Our observation is that one-time hardware costs do not drive the total costs of systems on the scale that we propose. Total cost is dominated by bandwidth, maintenance, and continual hardware obsolescence. Therefore Overcast seeks to minimize the use of bandwidth, cut maintenance costs by simplifying node deployment, and avoid obsolescence by structuring the system to allow older nodes to continue to contribute to the total efficiency of the overlay network.

Active Networks One may view overlay networks as an alternative implementation of active networks [23]. In active networks, new protocols and application-code can dynamically be downloaded into routers, allowing for rapid innovation of network services. Overcast avoids some of the hard problems of active networks by focusing on a single application; it does not have to address the problems created by dynamic downloading of code and sharing resources among multiple competing applications. Furthermore, since Overcast requires no changes to existing routers, it is easier to deploy. The main challenge for Overcast is to be competitive with solutions that are directly implemented on the network level.

3 The Overcast Network

This section describes the overlay network created by the Overcast system. First, we argue the benefits and drawbacks of using an overlay network. After concluding that an overlay network is appropriate for the task at hand, we explore the particular design of an overlay network to meet Overcast's demands. To do so, we examine the key design requirement of the Overcast network—single source distribution of bandwidth-intensive media on today's Internet infrastructure. Finally we illustrate the use of Overcast with an example.

3.1 Why overlay?

Overcast was designed to meet the needs of content providers on the Internet. This goal led us to an overlay network design. To understand why we chose an overlay network, we consider the benefits and drawbacks of overlays.

An overlay network provides advantages over both centrally located solutions and systems that advocate running code in every router. An overlay network is:

Incrementally Deployable An overlay network requires *no* changes to the existing Internet infrastructure, only additional servers. As nodes are added to an overlay network, it becomes possible to control the paths of data in the substrate network with ever greater precision.

Adaptable Although an overlay network abstraction constrains packets to flow over a constrained set of links, that set of links is constantly being optimized over metrics *that matter to the application.* For instance, the overlay nodes may optimize latency at the expense of bandwidth. The Detour Project [21] has discovered that there are often routes between two nodes with less latency than the routes offered by today's IP infrastructure. Overlay networks can find and take advantage of such routes.

Robust By virtue of the increased control and the adaptable nature of overlay networks, an overlay network can be *more* robust than the substrate fabric. For instance, with a sufficient number of nodes deployed, an overlay network may be able to guarantee that it is able to route between any two nodes in two independent ways. While a robust substrate network can be expected to repair faults eventually, such an overlay network might be able to route around faults immediately.

Customizable Overlay nodes may be multipurpose computers, easily outfitted with whatever equipment makes sense. For example, Overcast makes extensive use of disk space. This allows Overcast to provide bandwidth savings even when content is not consumed simultaneously in different parts of the network.

Standard An overlay network can be built on the least common denominator network services of the substrate network. This ensures that overlay traffic will be treated as well as any other. For example, Overcast uses TCP (in particular, HTTP over port 80) for reliable transport. TCP is simple, well understood, network friendly, and standard. Alternatives, such as a "home grown" UDP protocol with retransmissions, are less attractive by all these measures. For better or for worse, creativity in reliable transport is a losing battle on the Internet today.

On the other hand, building an overlay network faces a number of interesting challenges. An overlay network must address:

Management complexity The manager of an overlay network is physically far removed from the machines being managed. Routine maintenance must either be unnecessary or possible from afar, using tools that do not scale in complexity with the size of the network. Physical maintenance must be minimized and be possible by untrained personnel.

The real world In the real world, IP *does not* provide universal connectivity. A large portion of the Internet lies behind firewalls. A significant and growing share of hosts are behind Network Address Translators (NATs), and proxies. Dealing with these practical issues is tedious, but crucial to adoption.

Inefficiency An overlay *can not* be as efficient as code running in every router. However, our observation is that when an overlay network is small, the inefficiency, measured in absolute terms, will be small as well — and as the overlay network grows, its efficiency can approach the efficiency of router based servcies.

Information loss Because the overlay network is built on top of a network infrastructure (IP) that offers nearly complete connectivity (limited only by firewalls, NATs, and proxies), we expend considerable effort deducing the topology of the substrate network.

The first two of these problems can be addressed and nearly eliminated by careful design. To address management complexity, management of the entire overlay network can be concentrated at a single site. The key to a centralized-administration design is guaranteeing that newly installed nodes can boot and obtain network connectivity without intervention. Once that is accomplished, further instructions may be read from the central management server.

Firewalls, NATs and HTTP proxies complicate Overcast's operation in a number of ways. Firewalls force Overcast to open all connections "upstream" and to communicate using HTTP on port 80. This allows an Overcast network to extend exactly to those portions of the Internet that allow web browsing. NATs are devices used to multiplex a small set of IP addresses (often exactly one) over a number of clients. The clients are configured to use the NAT as their default router. At the NAT, TCP connections are rewritten to use one of the small number of IP addresses managed by the NAT. TCP port numbers allow the NAT to demultiplex return packets back to the correct client. The complication for Overcast is that client IP addresses are obscured. All Overcast nodes behind the NAT appear to have the same IP address. HTTP proxies have the same effect.

Although private IP addresses are never directly used by external Overcast nodes, there are times when an external node must correctly report the private IP address of another node. For example, an external node may have internal children. During tree building a node must report its childrens' addresses so that they may be measured for suitability as parents themselves. Only the private address is suitable for such purposes. To alleviate this complication all Overcast messages contain the sender's IP address *in the payload* of the message.

The final two disadvantages are not so easily dismissed. They represent the true tradeoff between overlay networks and ubiquitous router based software. For Overcast, the goal of instant deployment is important enough to sacrifice some measure of efficiency. However, the amount of inefficency introduced is a key metric by which Overcast should be judged.

3.2 Single-Source Multicast

Overcast is a single-source multicast system. This contrasts with IP Multicast which allows any member of a multicast group to send packets to all other members of the group. Beyond the fact that this closely models our intended application domain, there are a number of reasons to pursue this particular refinement to the IP Multicast model.

Simplicity Both conceptually and in implementation, a single-source system is simpler than an any-source model. For example, a single-source provides an obvious rendezvous point for group joins.

Optimization It is difficult to optimize the structure of the overlay network without intimate knowledge of the substrate network topology. This only becomes harder if the structure must be optimized for all paths [16].

Address space Single-source multicast groups provide a convenient alternative to the limited IP Multicast address space. The namespace can be partitioned by first naming the source, then allowing further subdivision of the source's choosing. In contrast, IP Multicast's address space is flat, limited, and without obvious administration to avoid collisions amongst new groups.

On the other hand, a single-source model clearly offers reduced functionality compared to a model that allows any group member to multicast. As such, Overcast is not appropriate for applications that require extensive use of such a model. However, many applications which appear to need multi-source multicast, such as a distributed lecture allowing questions from the class, do not. In such an application, only one "non-root" sender is active at any particular time. It would be a simple matter for the sender to unicast to the root, which would then perform the true multicast on the behalf of the sender. A number of projects [15, 17, 22] have used or advocated such an approach.

3.3 Bandwidth Optimization

Overcast is designed for distribution from a single source. As such, small latencies are expected to be of less importance to its users than increased bandwidth. Extremely low latencies are only important for applications that are inherently two-way, such as video conferencing. Overcast is designed with the assumption that broadcasting "live" video on the Internet may actually mean broadcasting with a ten to fifteen second delay.

Overcast distribution trees are built with the sole goal of creating high bandwidth channels from the source to all nodes. Although Overcast makes no guarantees that the topologies created are optimal, our simulations show that they perform quite well. The exact method by which high-bandwidth distribution trees are created and maintained is described in Section 4.2.

3.4 Deployment

An important goal for Overcast is to be deployable on today's Internet infrastructure. This motivates not only the use of an overlay network, but many of its details. In particular, deployment must require little or no human intervention, costs per node should be minimized, and unmodified HTTP clients must be able to join multicast groups in the Overcast network.

To help ease the human costs of deployment, nodes in the Overcast network configure themselves in an adaptive distributed tree with a single root. No human intervention is required to build efficient distribution trees, and nodes can be a part of multiple distribution trees.

Overcast's implementation on commodity PCs running Linux further eases deployment. Development is speeded by the familiar programming environment, and hardware costs are minimized by continually tracking the best price/performance ratio available in off-the-shelf hardware. The exact hardware configuration we have deployed has changed many times in the year or so that we have deployed Overcast nodes.

The final consumers of content from an Overcast network are HTTP clients. The Overcast protocols are carefully designed so that unmodified Web browsers can become members of a multicast group. In Overcast, a multicast group is represented as an HTTP URL: the hostname portion names the root of an Overcast network and the path represents a particular group on the network. All groups with the same root share a single distribution tree.

Using URLs as a namespace for Overcast groups has three advantages. First, URLs offer a hierarchal namespace, addressing the scarcity of multicast group names in traditional IP Multicast. Second, URLs and the means to access them are an existing standard. By delivering data over a simple HTTP connection, Overcast is able to bring multicasting to unmodified applications. Third, a URL's richer structure allows for simple expression of the increased power of Overcast over tradition multicast. For example, a group suffix of `start=10s` may be defined to mean "begin the content stream 10 seconds from the beginning."

3.5 Example usage

We have used Overcast to build a content-distribution application for high-quality video and live streams. The application is built out of a publishing station (called a *studio*) and nodes (called *appliances*). Appliances are installed at strategic locations in their network. The appliances boot, contact their studio, and self-organize into a distribution tree, as described below. No local administration is required.

The studio stores content and schedules it for delivery to the appliances. Typically, once the content is delivered, the publisher at the studio generates a web page announcing the availability of the content. When a user clicks on the URL for published content, Overcast redirects the request to a nearby appliance and the appliance serves the content. If the content is video, no special streaming software is needed. The user can watch the video over standard protocols and a standard MPEG player, which is supplied with most browsers.

An administrator at the studio can control the overlay network from a central point. She can view the status of the network (*e.g.*, which appliances are up), collect statistics, control bandwidth consumption, etc.

Using this system, bulk data can be distributed efficiently, even if the network between the appliances and the studio consists of low-bandwidth or intermittent links. Given the relative prices of disk space and network bandwidth, this solution is far less expensive than upgrading all network links between the studio and every client.

4 Protocols

The previous section described the structure and properties of the Overcast overlay network. This section describes how it functions: the initialization of individual nodes, the construction of the distribution hierarchy, and the automatic maintenance of the network. In particular, we describe the "tree" protocol to build distribution trees and the "up/down" protocol to maintain the global state of the Overcast network efficiently. We close by describing how clients (web browsers) join a group and how reliable multicasting to clients is performed.

4.1 Initialization

When a node is first plugged in or moved to a new location it automatically initializes itself and contacts the appropriate Overcast root(s). The first step in the initialization process is to determine an IP address and gateway address that the node can use for general IP connectivity. If there is a local DHCP server then the node can obtain IP configuration directly data using the DHCP protocol [12]. If DHCP is unavailable, a utility program can be used from a nearby workstation for manual configuration.

Once the node has an IP configuration it contacts a global, well-known registry, sending along its unique serial number. Based on a node's serial number, the registry provides a list of the Overcast networks the node should join, an optional permanent IP configuration, the network areas it should serve, and the access controls it should implement. If a node is intended to become part of a particular content distribution network, the configuration data returned will be highly specific. Otherwise, default values will be returned and the networks to which a node will join can be controlled using a web-based GUI.

4.2 The Tree Building Protocol

Self-organization of appliances into an efficient, robust distribution tree is the key to efficient operation in Overcast. Once a node initializes, it begins a process of self-organization with other nodes of the same Overcast network. The nodes cooperatively build an overlay network in the form of a distribution tree with the root node at its source. This section describes the tree-building protocol.

As described earlier, the virtual links of the overlay network are the only paths on which data is exchanged. Therefore the choice of distribution tree can have a significant impact on the aggregate communication behavior of the overlay network. By carefully building a distribution tree, the network utilization of content distribution can be significantly reduced. Overcast stresses bandwidth over other conceivable metrics, such as latency, because of its expected applications. Overcast is not intended for interactive applications, therefore optimizing a path to shave small latencies at the expense of total throughput would be a mistake. On the other hand, Overcast's architecture as an overlay network allows this decision to be revisited. For instance, it may be decided that trees should have a fixed maximum depth to limit buffering delays.

The goal of Overcast's tree algorithm is to maximize bandwidth to the root for all nodes. At a high level the algorithm proceeds by placing a new node as far away from the root as possible without sacrificing bandwidth to the root. This approach leads to "deep" distribution trees in which the nodes nonetheless observe no worse bandwidth than obtaining the content directly from the root. By choosing a parent that is nearby in the network, the distribution tree will form along the lines of the substrate network topology.

The tree protocol begins when a newly initialized node contacts the root of an Overcast group. The root thereby becomes the `current` node. Next, the new node begins a series of rounds in which it will attempt to locate itself further away from the root without sacrificing bandwidth back to the root. In each round the new node considers its bandwidth to `current` as well as the bandwidth to `current` *through each of* `current`*'s children*. If the bandwidth through any of the children is about as high as the direct bandwidth to `current`, then one of these children becomes `current` and a new round commences. In the case of multiple suitable children, the child closest (in terms of network hops) to the searching node is chosen. If no child is suitable, the search for a parent ends with `current`.

To approximate the bandwidth that will be observed when moving data, the tree protocol measures the download time of 10 Kbytes. This measurement includes all the costs of serving actual content. We have observed that this approach to measuring bandwidth gives us better results than approaches based on low-level bandwidth measurements such as using ping. On the other hand, we recognize that a 10 Kbyte message is too short to accurately reflect the bandwidth of "long fat pipes". We plan to move to a technique that uses progressively larger measurements until a steady state is observed.

When the measured bandwidths to two nodes are within 10% of each other, we consider the nodes equally good and select the node that is closest, as reported by *traceroute*. This avoids frequent topology changes between two nearly equal paths, as well as decreasing the total number of network links used by the system.

A node periodically reevaluates its position in the tree by measuring the bandwidth to its current siblings (an up-to-date list is obtained from the parent), parent, and grandparent. Just as in the initial building phase, a node will relocate below its siblings if that does not decrease its bandwidth back to the root. The node checks bandwidth directly to the grandparent as a way of testing its previous decision to locate under its current parent. If necessary the node moves back up in the hierarchy to become a sibling of its parent. As a result, nodes constantly reevaluate their position in the tree and an Overcast network is inherently tolerant of non-root node failures. If a node goes off-line for some reason, any nodes that were below it in the tree will reconnect themselves to the rest of the routing hierarchy. When a node detects that its parent is unreachable, it will simply relocate beneath its grandparent. If its grandparent is also unreachable the node will continue to move up its ancestry until it finds a live node. The ancestor list also allows cycles to be avoided as nodes asynchronously choose new parents. A node simply refuses to become the parent of a node it believes to be it's own ancestor. A node that chooses such a node will forced to rechoose.

While there is extensive literature on faster fail-over algorithms, we have not yet found a need to optimize beyond the strategy outlined above. It is important to remember that the nodes participating in this protocol are dedicated machines that are less prone to failure than desktop computers. If this becomes an issue, we have considered extending the tree building algorithm to maintain backup parents (excluding a node's own ancestry from consideration) or an entire backup tree.

By periodically remeasuring network performance, the overlay network can adapt to network conditions that manifest themselves at time scales larger than the frequency at which the distribution tree reorganizes. For example, a tree that is optimized for bandwidth efficient content delivery during the day may be significantly suboptimal during the overnight hours (when network congestion is typically lower). The ability of the tree protocol to automatically adapt to these kinds of changing network conditions provides an important advantage over simpler, statically configured content distribution schemes.

4.3 The Up/Down Protocol

To allow web clients to join a group quickly, the Overcast network must track the status of the Overcast nodes. It may also be important to report statistical information back to the root, so that content providers might learn, for instance, how often certain content is being viewed. This section describes a protocol for efficient exchange of information in a tree of network nodes to provide the root of the tree with information from nodes throughout the network. For our needs, this protocol must scale sublinearly in terms of network usage at the root, but may scale linearly in terms of space (all with respect to the number of Overcast nodes). This is a simple result of the relative requirements of a client for these two resources and the cost of those resources. Overcast might store (conservatively) a few hundred bytes about each Overcast node, but even in a group of millions of nodes, total RAM cost for the root would be under $1,000.

We call this protocol the "up/down" protocol because our current system uses it mainly to keep track of what nodes are up and what nodes are down. However, arbitrary information in either of two large classes may be propagated to the root. In particular, if the information either changes slowly (*e.g.*, up/down status of nodes), or the information can be combined efficiently from multiple children into a single description (*e.g.*, group membership counts), it can be propagated to the root. Rapidly changing information that can not be aggregated during propagation would overwhelm the root's bandwidth capacity.

Each node in the network, including the root node, maintains a table of information about all nodes lower than itself in the hierarchy and a log of all changes to the table. Therefore the root node's table contains up-to-date information for all nodes in the hierarchy. The table is stored on disk and cached in the memory of a node.

The basis of the protocol is that each node periodically checks in with the node directly above it in the tree. If a child fails to contact its parent within a preset interval, the parent will assume the child and all its descendants have "died". That is, either the node has failed, an intervening link has failed, or the child has simply changed parents. In any case, the parent node marks the child and its descendants "dead" in its table. Parents never initiate contact with descendants. This is a byproduct of a design that is intended to cross firewalls easily. All node failures must be detected by a failure to check in, rather than active probing.

During these periodic check-ins, a node reports new information that it has observed or been informed of since it last checked in. This includes:

- "Death certificates" - Children that have missed their expected report time.
- "Birth certificates" - Nodes that have become children of the reporting node.
- Changes to the reporting node's "extra information."
- Certficates or changes that have been propagated to the node from its own children since its last checkin.

This simple protocol exhibits a race condition when a node chooses a new parent. The moving node's former parent propagates a death certificate up the hierarchy, while at nearly the same time the new parent begins propagating a birth certificate up the tree. If the birth certificate arrives at the root first, when the death certificate arrives the root will believe that the node has failed. This inaccuracy will remain indefinitely since a new birth certificate will only be sent in response to a change in the hierarchy that may not occur for an arbitrary period of time.

To alleviate this problem, a node maintains a sequence number indicating of how many times it has changed parents. All changes involving a node are tagged with that number. A node ignores changes that are reported to it about a node if it has already seen a change with a higher sequence number. For instance, a node may have changed parents 17 times. When it changes again, its former parent will propagate a death certificate annotated with 17. However, its new parent will propagate a birth certificate annotated with 18. If the birth certificate arrives first, the death certificate will be ignored since it is older.

An important optimization to the up/down protocol avoids large sets of birth certificates from arriving at the root in response to a node with many descendants choosing a new parent. Normally, when a node moves to a new parent, a birth certificate must be sent out for each of its descendants to its new parent. This maintains the invariant that a node knows the parent of all its descendants. Keep in mind that a birth certificate is not only a record that a node exists, but that it has a certain parent.

Although this large set of updates is required, it is usually unnecessary for these updates to continue far up the hierarchy. For example, when a node relocates beneath a sibling, the sibling must learn about all of the node's descendants, but when the sibling, in turn, passes these certificates to the original parent, the original parent notices that they do not represent a change and quashes the certificate from further propagation.

Using the up/down protocol, the root of the hierarchy will receive timely updates about changes to the network. The freshness of the information can be tuned by varying the length of time between check-ins. Shorter periods between updates guarantee that information will make its way to the root more quickly. Regardless of the update frequency, bandwidth requirements at the root will be proportional to the number of changes in the hierarchy rather than the size of the hierarchy itself.

4.4 Replicating the root

In Overcast, there appears to be the potential for significant scalability and reliability problems at the root. The up/down protocol works to alleviate the scalability difficulties in maintaining global state about the distribution tree, but the root is still responsible for handling all join requests from all HTTP clients. The root handles such requests by redirection, which is far less resource intensive than actually delivering the requested content. Nonetheless, the possibility of overload remains for particularly popular groups. The root is also a single point of failure.

To address this, overcast uses a standard technique used by many popular websites. The DNS name of the root resolves to any number of replicated roots in round-robin fashion. The database used to perform redirections is replicated to all such roots. In addition, IP address takeover may be used for immediate failover, since DNS caching may cause clients to continue to contact a failed replica. This simple, standard technique works well for this purpose because handling joins from HTTP clients is a read-only operation that lends well to distribution over numerous replicas.

There remains, however, a single point of failure for the up/down protocol. The functionality of the root in the up/down protocol cannot be distributed so easily because its purpose is to maintain changing state. However the up/down protocol has the useful property that all nodes maintain state for nodes below them in the distribution tree. Therefore, a convenient technique to address fault tolerance is to specially construct the top of the hierarchy.

Starting with the root, some number of nodes are configured linearly, that is, each has only one child. In this way all other overcast nodes lie below these top nodes. Figure 2 shows a distribution tree in which the top three nodes are arranged linearly. Each of these nodes has enough information to act as the root of the up/down protocol in case of a failure. This technique has the drawback of increasing the latency of content distribution unless special-case code skips the extra roots during distribution. If latency were important to Overcast this would be an important, but simple, optimization.

"Linear roots" work well with the need for replication to address scalability, as mentioned above. The set of linear nodes has all the information needed to

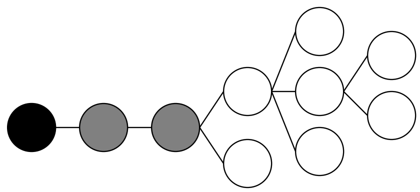

Figure 2: A specially configured distribution topology that allows either of the grey nodes to quickly stand in as the root (black) node. All filled nodes have complete status information about the unfilled nodes.

perform Overcast joins, therefore these nodes are perfect candidates to be used in the DNS round-robin approach to scalability. By choosing these nodes, no further replication is necessary.

4.5 Joining a multicast group

To join a multicast group, a Web client issues an HTTP GET request with the URL for a group. The hostname of the URL names the root node(s). The root uses the pathname of the URL, the location of the client, and its database of the current status of the Overcast nodes to decide where to connect the client to the multicast tree. Because status information is constantly propagated to the root, a decision may be made quickly without further network traffic, enabling fast joins.

Joining a group consists of selecting the best server and redirecting the client to that server. The details of the server selection algorithm are beyond the scope of this paper as considerable previous work [3, 18] exists in this area. Furthermore, Overcast's particular choices are constrained considerably by a desire to avoid changes at the client. Without such a constraint simpler choices could have been made, such as allowing clients to participate directly in the Overcast tree building protocol.

Although we do not discuss server selection here, a number of Overcast's details exist to support this important functionality, however it may actually be implemented. A centralized root performing redirections is convenient for an approach involving large tables containing collected Internet topology data. The up/down algorithm allows for redirections to nodes that are known to be functioning.

4.6 Multicasting with Overcast

We refer to reliable multicasting on an overcast network as "overcasting". Overcasting proceeds along the distribution tree built by the tree protocol. Data is moved between parent and child using TCP streams. If a node has four children, four separate connections are used. The content may be pipelined through several generations in the tree. A large file or a long-running live stream may be in transit over tens of different TCP streams at a single moment, in several layers of the distribution hierarchy.

If a failure occurs during an overcast, the distribution tree will rebuild itself as described above. After rebuilding the tree, the overcast resumes for on-demand distributions where it left off. In order to do so, each node keeps a log of the data it has received so far. After recovery, a node inspects the log and restarts all overcasts in progress.

Live content on the Internet today is typically buffered before playback. This compensates for momentary glitches in network throughput. Overcast can take advantage of this buffering to mask the failure of a node being used to Overcast data. As long as the failure occurs in a node that is not at the edge of the Overcast network, an HTTP client need not ever become aware that the path of data from the root has been changed in the face of failure.

5 Evaluation

In this section, the protocols presented above are evaluated by simulation. Although we have deployed Overcast in the real world, we have not yet deployed on a sufficiently large network to run the experiments we have simulated.

To evaluate the protocols, an overlay network is simulated with increasing numbers of overcast nodes while keeping the total number of network nodes constant. Overcast should build better trees as more nodes are deployed, but protocol overhead may grow.

We use the Georgia Tech Internetwork Topology Models [25] (GT-ITM) to generate the network topologies used in our simulations. We use the "transit-stub" model to obtain graphs that more closely resemble the Internet than a pure random construction. GT-ITM generates a transit-stub graph in stages, first a number of random backbones (transit domains), then the random structure of each back-bone, then random "stub" graphs are attached to each node in the backbones.

We use this model to construct five different 600 node graphs. Each graph is made up of three transit domains. These domains are guaranteed to be connected. Each transit domain consists of an average of eight stub networks. The stub networks contain edges amongst themselves with a probability of 0.5. Each stub network consists of an average of 25 nodes, in which nodes are once again connected with a probability of 0.5. These parameters are from the sample graphs in the GT-ITM distribution; we are unaware of any published work that describes parameters that might better model common Internet topologies.

We extended the graphs generated by GT-ITM with bandwidth information. Links internal to the transit domains were assigned a bandwidth of 45Mbits/s, edges connecting stub networks to the transit domains were assigned 1.5Mbits/s, finally, in the local stub domain, edges were assigned 100Mbit/s. These reflect commonly used network technology: T3s, T1s, and Fast Ethernet. All measurements are averages over the five generated topologies.

Empirical measurements from actual Overcast nodes show that a single Overcast node can easily support twenty clients watching MPEG-1 videos, though the exact number is greatly dependent on the bandwidth requirements of the content. Thus with a network of 600 overcast nodes, we are simulating multicast groups of perhaps 12,000 members.

5.1 Tree protocol

The efficiency of Overcast depends on the positioning of Overcast nodes. In our first experiments, we compare two different approaches to choosing positions. The first approach, labelled “Backbone”, preferentially chooses transit nodes to contain Overcast nodes. Once all transit nodes are Overcast nodes, additional nodes are chosen at random. This approach corresponds to a scenario in which the owner of the Overcast nodes places them strategically in the network. In the second, labelled “Random”, we select all Overcast nodes at random. This approach corresponds to a scenario in which the owner of Overcast nodes does not pay attention to where the nodes are placed.

The goal of Overcast’s tree-building protocol is to optimize the bottleneck bandwidth available back to the root for all nodes. The goal is to provide each node with the same bandwidth to the root that the node would have in an idle network. Figure 3 compares the sum of all nodes’ bandwidths back to the root in Overcast networks of various sizes to

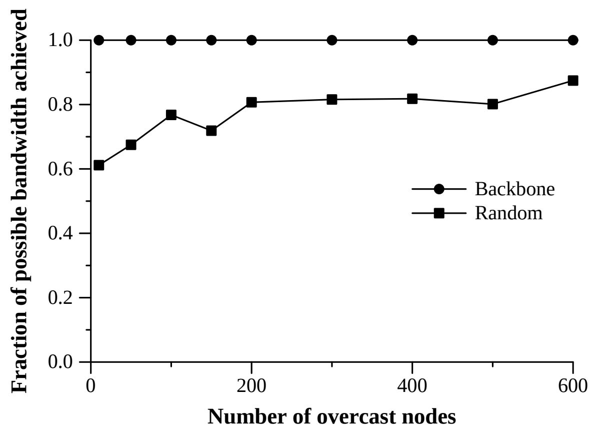

Figure 3: Fraction of potential bandwidth provided by Overcast.

the sum of all nodes’ bandwidths back to the root in an optimal distribution tree using router-based software. This indicates how well Overcast performs compared to IP Multicast.

The main observation is that, as expected, the backbone strategy for placing Overcast nodes is more effective than the random strategy, but the results of random placement are encouraging nonetheless. Even a small number of deployed Overcast nodes, positioned at random, provide approximately 70%-80% of the total possible bandwidth.

It is extremely encouraging that, when using the backbone approach, no node receives less bandwidth under Overcast than it would receive from IP Multicast. However some enthusiasm must be withheld, because a simulation artifact has been left in these numbers to illustrate a point.

Notice that the backbone approach and the random approach differ in effectiveness even when all 600 nodes of the network are Overcast nodes. In this case the same nodes are participating in the protocol, but better trees are built using the backbone approach. This illustrates that the trees created by the tree-building protocol are not unique. The backbone approach fares better by this metric because in our simulations backbone nodes were turned on first. This allowed backbone nodes to preferrentially form the “top” of the tree. This indicates that in future work it may be beneficial to extend the tree-building protocol to accept hints that mark certain nodes as “backbone” nodes. These nodes would preferentially form the core of the distribution tree.

Overcast appears to perform quite well for its intended goal of optimizing available bandwidth, but

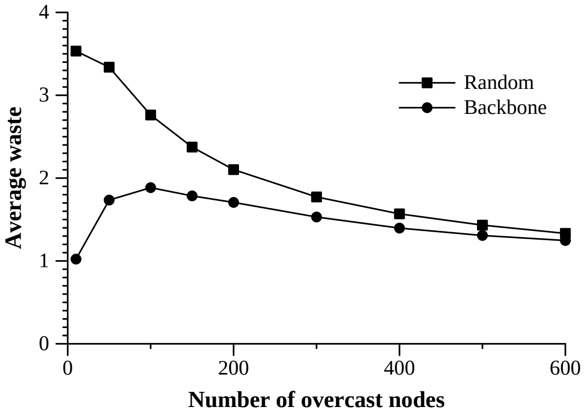

Figure 4: Ratio of the number of times a packet must "hit the wire" to be propagated through an Overcast network to a lower bound estimate of the same measure for IP Multicast.

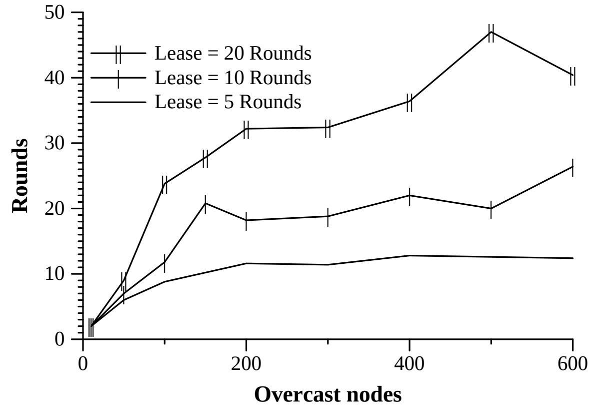

Figure 5: Number of rounds to reach a stable distribution tree as a function of the number of overcast nodes and the length of the lease period.

it is reasonable to wonder what costs are associated with this performance.

To explore this question we measure the network load imposed by Overcast. We define network load to be the number of times that a particular piece of data must traverse a network link to reach all Overcast nodes. In order to compare to IP Multicast Figure 4 plots the ratio of the network load imposed by Overcast to a lower bound estimate of IP Multicast's network load. For a given set of nodes, we assume that IP Multicast would require exactly one less link than the number of nodes. This assumes that all nodes are one hop away from another node, which is unlikely to be true in sparse topologies, but provides a lower bound for comparison.

Figure 4 shows that for Overcast networks with greater than 200 nodes Overcast imposes somewhat less than twice as much network load as IP Multicast. In return for this extra load Overcast offers reliable delivery, immediate deployment, and future flexibility. For networks with few Overcast nodes, Overcast appears to impose a considerably higher network load than IP Multicast. This is a result of our optimistic lower bound on IP Multicast's network load, which assumes that 50 randomly placed nodes in a 600 node network can be spanned by 49 links.

Another metric to measure the effectiveness of an application-level multicast technique is *stress*, proposed in [16]. Stress indicates the number of times that the same data traverses a particular physical link. By this metric, Overcast performs quite well with average stresses of between 1 and 1.2. We do not present detailed analysis of Overcast's performance by this metric, however, because we believe that network load is more telling for Overcast. That is, Overcast has quite low scores for average stress, but that metric does not describe how often a longer route was taken when a shorter route was available.

Another question is how fast the tree protocol converges to a stable distribution tree, assuming a stable underlying network. This is dependent on three parameters. The *round period* controls how long a node that has not yet determined a stable position in the hierarchy will wait before evaluating a new set of potential parents. The *reevaluation period* determines how long a node will wait before reevaluating its position in the hierarchy once it has obtained a stable position. Finally the *lease period* determines how long a parent will wait to hear from a child before reporting the child's death.

For convenience, we measure all convergence times in terms of the fundamental unit, the round time. We also set the reevaluation period and lease period to the same value. Figure 5 shows how long Overcast requires to converge if an entire Overcast network is simultaneously activated. To demonstrate the effect of a changing reevaluation and lease period, we plot for the "standard" lease time—10 rounds, as well as longer and shorter periods. Lease periods shorter than five rounds are impractical because children actually renew their leases a small random number of rounds (between one and three) before their lease expires to avoid being thought dead. We expect that a round period on the order of 1-2 seconds will be practical for most applications.

We next measure convergence times for an existing Overcast network in which overcast nodes are added or fail. We simulate overcast networks of various

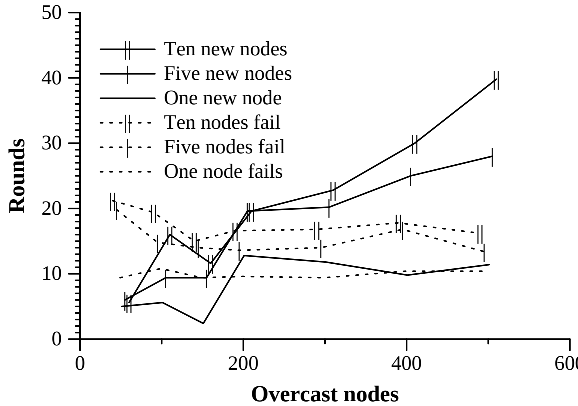

Figure 6: Number of rounds to recover a stable distribution tree as a function of the number of nodes that change state and the number of nodes in the network.

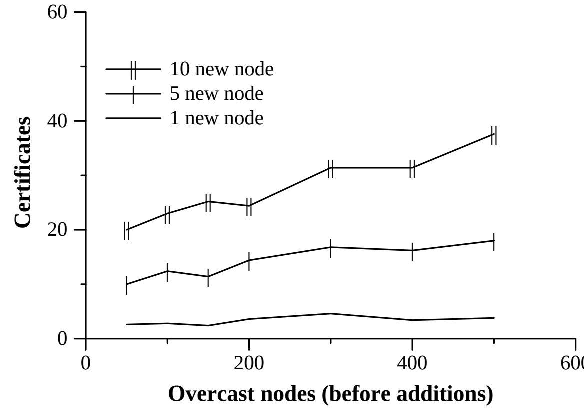

Figure 7: Certificates received at the root in response to node additions.

sizes until they quiesce, add and remove Overcast nodes, and then simulate the network until it quiesces once again. We measure the time, in rounds, for the network to quiesce after the changes. We measure for various numbers of additions and removals allowing us to assess the dependence of convergence on how many nodes have changed state. We measure only the backbone approach.

Figure 6 plots convergence times (using a 10 round lease time) against the number of overcast nodes in the network. The convergence time for node failures is quite modest. In all simulations the Overcast network reconverged after less than three lease times. Furthermore, the reconvergence time scaled well against both the number of nodes failing and the total number of nodes in the overcast network. In neither case was the convergence time even linearly affected.

For node additions, convergence times do appear more closely linked to the size of the Overcast network. This makes intuitive sense because new nodes are navigating the network to determine their best location. Even so, in all simulations fewer than five lease times are required. It is important to note that an Overcast network continues to function even while stabilizing. Performance may be somewhat impacted by increased measurement traffic and by TCP setup and tear down overhead as parents change, but such disruptions are localized.

5.2 Up/Down protocol

The goal of the up/down algorithm is to minimize the bandwidth required at the root node while maintaining timely status information for the entire network. Factors that affect the amount of bandwidth used include the size of the overcast network and the rate of topology changes. Topology changes occur when the properties of the underlying network change, nodes fail, or nodes are added. Therefore the up/down algorithm is evaluated by simulating overcast networks of various sizes in which various numbers of failures and additions occur.

To assess the up/down protocol's ability to provide timely status updates to the root without undue overhead we keep track of the number of certificates (for both "birth" and "death") that reach the root during the previous convergence tests. This is indicative of the bandwidth required at the root node to support an overcast network of the given size and is dependent on the amount of topology change induced by the additions and deletions.

Figure 7 graphs the number of certificates received by the root node in response to new nodes being brought up in the overcast network. Remember, the root may receive multiple certificates per node addition because the addition is likely to cause some topology reconfiguration. Each time a node picks a new parent that parent propagates a birth certificate. These results indicate that the number of certificates is quite modest: certainly no more than four certificates per node addition, usually approximately three. What is more important is that the number of certificates scales more closely to the number of new nodes than the size of the overcast network. This gives evidence that overcast can scale to large networks.

Similarly, Overcast requires few certificates to react to node failures. Figure 8 shows that in the common case, no more than four certificates are required per node failure. Again, because the number of certificates is proportional to the number of failures rather

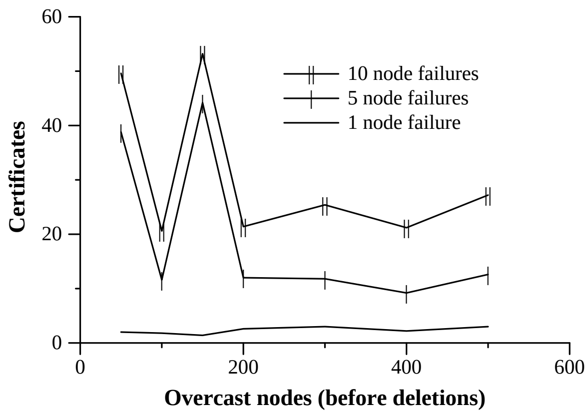

Figure 8: Certificates received at the root in response to node deletions.

than the size of the network, Overcast appears to offer the ability to scale to large networks.

On the other hand, Figure 8 shows that there are some cases that fall far outside the norm. The large spikes at 50 and 150 node networks with 5 and 10 failures occurred because of failures that happened to occur near the root. When a node with a substantial number of children chooses a new parent it must convey it's entire set of descendants to its new parent. That parent then propagates the entire set. However, when the information reaches a node that already knows the relationships in question, the update is quashed. In these cases, because the reconfigurations occurred high in the tree there was no chance to quash the updates before they reached the root. In larger networks such failures are less likely.

6 Conclusions

We have described a simple tree-building protocol that yields bandwidth-efficient distribution trees for single-source multicast and our up/down protocol for providing timely status updates to the root of the distribution tree in scalable manner. Overcast implements these protocols in an overlay network over the existing Internet. The protocols allow Overcast networks to dynamically adapt to changes (such as congestion and failures) in the underlying network infrastructure and support large, reliable single-source multicast groups. Geographically-dispersed businesses have deployed Overcast nodes in small-scale Overcast networks for distribution of high-quality, on-demand video to unmodified desktops.

Simulation studies with topologies created with the Georgia Tech Internetwork Topology Models show that Overcast networks work well on large-scale networks, supporting multicast groups of up to 12,000 members. Given these results and the low cost for Overcast nodes, we believe that putting computation and storage in the network fabric is a promising approach for adding new services to the Internet incrementally.

Acknowledgements

We thank Hari Balakrishnan for helpful input concerning the tree building algorithm; Suchitra Raman, Robert Morris, and our shepherd, Fred Douglis, for detailed comments that improved our presentation in many areas; and the many anonymous reviewers whose reviews helped us to see our work with fresh eyes.

References

[1] FastForward Networks' broadcast overlay architecture. Technical report, FastForward, 2000. `www.ffnet.com/pdfs/BOA-whitepaperv6.PDF`.

[2] Elan Amir, Steven McCanne, and Randy H. Katz. An active service framework and its application to real time multimedia transcoding. In *Proc. ACM SIGCOMM Conference (SIGCOMM '98)*, pages 178–190, September 1998.

[3] Yair Amir, Alec Peterson, and David Shaw. Seamlessly selecting the best copy from Internet-wide replicated web servers. In *The 12th International Symposium on Distributed Computing (DISC'98)*, pages 22–23, September 1998.

[4] Michael Baentsch, Georg Molter, and Peter Sturm. Introducing application-level replication and naming into today's web. In *Proc. 5th International World Wide Web Conference*, May 1996.

[5] A. Basso, C. Cranor, R. Gopalakrishnan, M. Green, C.R. Kalmanek, D. Shur, S. Sibal, C.J. Sreenan, and J.E. van der Merwe. PRISM, an IP-based architecture for broadband access to TV and other streaming media. In *Proc. IEEE International Workshop on Network and Operating System Support for Digital Audio and Video*, June 2000.

[6] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic, and response time in distributed information systems. In *Proc. of the 1996 International Conference on Data Engineering (ICDE '96)*, March 1996.

[7] M. Blaze. *Caching in Large-Scale Distributed File Systems*. PhD thesis, Princeton University, January 1993.

[8] Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Michael F. Schwartz, and Kurt J. Worrell. A hierarchical Internet object cache. In *Proc. USENIX 1996 Annual Technical Conference*, pages 153–164, January 1996.

[9] Yatin Chawathe, Steven McCanne, and Eric Brewer. RMX: Reliable multicast for heterogeneous networks. In *Proc. IEEE Infocom*, March 2000.

[10] P. Danzig, R. Hall, and M. Schwartz. A case for caching file objects inside internetworks. In *Proc. ACM SIGCOMM Conference (SIGCOMM '93)*, pages 239–248, September 1993.

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

[12] R. Droms. Dynamic host configuration protocol. RFC 2131, Internet Engineering Task Force, March 1997. `ftp://ftp.ietf.org/rfc/rfc2131.txt`.

[13] Paul Francis. Yoid: Your Own Internet Distribution. Technical report, ACIRI, April 2000. `www.aciri.org/yoid`.

[14] J. Gwertzman and M. Seltzer. The case for geographical push-caching. In *Proc. 5th Workshop on Hot Topics in Operating Systems (HotOS-V)*, pages 51–57. IEEE Computer Society Technical Committee on Operating Systems, May 1995.

[15] Hugh W. Holbrook and David R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *Proc. ACM SIGCOMM Conference (SIGCOMM '99)*, pages 65–78, September 1999.

[16] Yang hu Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proc. ACM SIGMETRICS Conference (SIGMETRICS '00)*, June 2000.

[17] M. Frans Kaashoek, Robbert van Renesse, Hans van Staveren, and Andrew S. Tanenbaum. FLIP: an internetwork protocol for supporting distributed systems. *ACM Trans. Computer Systems*, 11(1):77–106, February 1993.

[18] D. R. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web. In *Proc. 29th ACM Symposium on Theory of Computing*, pages 654–663, May 1997.

[19] Steven McCanne and Van Jacobson. Receiver-driven layed multicast. In *Proc. ACM SIGCOMM Conference (SIGCOMM '96)*, pages 117–130, August 1996.

[20] Jörg Nonnenmacher, Ernst W. Biersack, and Don Towsley. Parity-based loss recovery for reliable multicast transmission. In *Proc. ACM SIGCOMM Conference (SIGCOMM '97)*, pages 289–300, September 1997.

[21] Stefan Savage, Tom Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: A case for informed Internet routing and transport. *IEEE Micro*, 19(1):50–59, January 1999.

[22] Michael D. Schroeder, Andrew D. Birrell, Michael Burrows, Hal Murray, Roger M. Needham, Thomas L. Rodeheffer, Edwin H. Satterthwaite, and Charles P. Thacker. Autonet: A high-speed, self-configuring local area network using point-to-point links. *IEEE/ACM Trans. Networking*, 9(8):1318–1335, October 1991.

[23] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[24] J. Touch and S. Hotz. The X-bone (white paper). Technical report, SIS, May 1997. `www.isi.edu/x-bone`.

[25] Ellen W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *Proc. IEEE Infocom*, pages 40–52, March 1996.

[26] Lixia Zhang, Scott Michel, Khoi Nguyen, Adam Rosenstein, Sally Floyd, and Van Jacobson. Adaptive web caching: Towards a new global caching architecture. In *Proc. 3rd International World Wide Web Caching Workshop*, June 1998.