Yoid: Extending the Internet Multicast Architecture

Paul Francis

ACIRI
francis@aciri.org, www.aciri.org

April 2, 2000

Contents

0.1 Abstract

If we take a broad view of the term “multicast” to mean any distribution of content to more than one machine, we find that multicast is proceeding along two distinct architectural tracks. On one track is IP multicast, which mainly targets realtime non-reliable applications, but for which hopes run high for reliable applications as well. On the other are a plethora of open or proprietary host- or server-based approaches, each typically targeting a specific application or product line.

IP multicast suffers from a number of technical problems, lacks applications, and in general is having trouble reaching critical mass, especially regarding anything resembling a global infrastructure. Server-based approaches are valuable and wide-spread, but there is no synergy in terms of multiple distinct groups working within the same architecture. As a result, progress is not as fast as it could be, and consumers are strapped with multiple application-specific infrastructures to deal with.

This paper presents an architecture, called yoid, that aims to unify both tracks under a single umbrella architecture. Yoid attempts to take the best from both tracks—reliable and asynchronous distribution from the server-based track, and dynamic auto-configuration via a simple API from the IP multicast track.

A key component of yoid is that it allows a group of endhosts (the hosts where the content-consuming application resides) to auto-configure themselves into a tunneled topology for the purpose of content distribution. Yoid can run over IP multicast, but does not require it. This allows application developers to bundle yoid into their applications, giving their applications robust and scalable configuration-free out-of-the-box multicast. This is key to the initial acceptance of yoid and for allowing it to reach critical mass.

Yoid is not limited, however, to endhost-based distribution. It can also work in infrastructure servers (boxes that receive, replicate, and forward content but that are not consumers of the content). This allows improved performance for applications that require it. It can also provide other benefits such as better security. The endhost- and server-based modes of operation taken together, along with yoid’s ability to utilize local islands of IP multicast, allow yoid to support the broadest possible range of applications.

1 Changes

April 2, 2000 Updated for name change from Yallcast to Yoid.

2 Introduction

2.1 Document Roadmap

Lest the reader be immediately put off by the size of this document, I offer the following brief roadmap:

If, having read the abstract, you want to go straight to a very brief technical overview of yoid, look at Subsection 2.4 ("Yoid in a Nutshell") and perhaps the subsection preceding it (2.3). For a deeper technical overview, see Subsections 3.1 and 3.2.

Otherwise, this paper pretty much proceeds in increasing detail, and you can just start at the beginning and continue until you've read enough. The first Section (2, "Introduction"), tells you why we need better distribution, what yoid is, and why yoid provides better distribution.

The second Section (3) takes you through yoid in increasing technical detail. Of particular interest might be Subsection 3.4 ("Yoid Tree Management Protocol (YTMP)"), and to some extent the following subsection, which describe how yoid tunneled topologies are dynamically configured—the only really novel thing about yoid.

The last section outlines an assortment of enhancements that may be made to the basic architecture to increase yoid's capabilities or solve various problems endemic to yoid. This section can easily be skipped by all but the most interested readers.

This document is in an early draft stage. It is still missing references, and has not gone through any peer review. Any references or comments on the document or yoid itself are greatly appreciated.

2.2 Motivation

Let's take a broad view of the term "multicast" and take it to mean every instance where content is moved from one machine to more than one other machine. For lack of a better term, and to avoid long and clumsy phrases, let's refer to this broad view multicast as simply *distribution*.

Viewed this way, the majority of what gets transmitted over the internet is distribution: mail, news, web pages (HTML) and files of all types (jpg, mp3, etc.), chat, channels, DNS records, audio-video broadcasts, and so on. While strictly 2-party exchanges are obviously important, it is not an exaggeration to say that the internet is what it is because of its distribution functionality.

In spite of this, virtually every distribution application in the internet today runs over the unicast infrastructure and derives its distribution functionality from mechanisms internal to and specific to the application itself. For instance, RFC822 mail headers have mechanisms for detecting loops among mail forwarders, NNTP has the NEWNEWS command to manage flooding of news articles, HTTP has redirection and mechanisms for handling caches, and so on. Practically speaking, though, there is no general "infrastructure" support for distribution in existence today. The exception that proves this rule is IP multicast, globally available tunneled in the form of the mbone, and privately available as either tunneled or native IP multicast.

The reason this exception proves the rule is that IP multicast has so far not lived up to early expectations, to say the least. And this in spite of the fact that it is now available on most host operating systems and in most routers (though it is usually not "turned on"). Different people have different ideas on why IP multicast has not taken off, ranging from a lack of tools for managing IP multicast installations to insufficient protocol development to a lack of IP multicast-ready applications (all examples of reasons voiced by participants in the IETF maddogs ad hoc group). Many people, myself included, have labored under the tacit assumption that if we could only fix this or that problem, and add this or that functionality, then IP multicast would reach some sort of critical mass and take off, in the same sense that HTTP/HTML at some point reached a critical mass and took off. It would then serve as the infrastructure foundation upon which various kinds of distribution applications would be built.

I no longer believe this is a realistic possibility. IP multicast, in its role as "the architectural foundation for internet distribution", suffers from one fundamental problem, one major problem, and a lot of nagging problems.

The fundamental problem is that IP multicast works only across space, not across time, whereas most distribution on the internet (almost everything mentioned above), works across both space and time. What I mean by this is that the recipients of most types of distribution content (mail, web pages, etc.) want to receive it at different times. Even content that under ideal conditions would reach all recipients immediately (i.e., "push" content like mail) can't because not all recipients are ready to receive all the time (mainly because they are not connected to the internet all the time). IP multicast, on the other hand, requires that all recipients receive the content at the same time. (I understand that there is a way around this, namely multicasting the content multiple times until all recipients have got it. But this is not a very attractive thing to have to do, and rather proves my point.)

The major problem referred to above is that IP multicast addresses are too small. Basing the global architectural foundation for distribution on a 27-bit address space is, frankly, silly. It may very well be that some of the current initiatives for IP multicast address assignment will satisfactorily solve the problem (I did say this was only a major problem, not a fundamental problem), but it really is bending over backwards unnecessarily. And of course there is IPv6, but I wouldn't want to assume that that will become ubiquitous any time soon.

The nagging problems include:

- Large IP multicast routing tables.
- Congestion control over IP multicast.
- Reliable data transport over IP multicast.
- Good access control and security mechanisms.

I want to repeat that all of the above discussion of IP multicast is in its role as "the architectural foundation for internet distribution." To be fair, I understand that its not like some person or group ever sat down and, working from a clean whiteboard, analyzed all the options and decided that IP multicast, with its 27 bit address space and space-only distribution mechanism, was the best choice for all internet distribution. Rather, we've incrementally backed ourselves into this corner via some set of historical decisions, or non-decisions, that have long since been overtaken by events.

IP multicast is ideal for applications that cannot tolerate (much) delay, can tolerate some loss, and have throughput that is relatively high but upper-bounded. This mainly includes interactive things like audio-video conferencing and internet games. A host- or server-based approach works poorly or not at all for these applications. On the other end of the spectrum, a host- or server-based approach is great for applications that can tolerate significant delay (minutes or hours) and cannot tolerate loss. This includes things like mail and file distribution. IP multicast works poorly or not at all for these applications. For stuff in between (chat, distributed whiteboard, audio-video one-way broadcast), both can suffice, and reasonable people could agree to disagree on which is better.

For this stuff in between, however, it just turns out that the path of least resistance for obtaining distribution funtionality is the server-based approach. This may be in part because of the technical

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**