A Programmable Router Architecture Supporting Control Plane Extensibility

Jun Gao Peter Steenkiste Eduardo Takahashi
Allan Fisher

March 2000
CMU-CS-00-109

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This report is an expanded version of "A Programmable Router Architecture Supporting Control Plane Extensibility," *IEEE Communications Magazine*, (38)3:152-59, March 2000.

This research was sponsored by the Defense Advanced Research Projects Agency and monitored by NCCOSC RDTE Division, San Diego, CA 92152-5000, under contract N66001-96-C-8528 and by AFRL/IFGA, Rome NY 13441-4505, under contract F30602-99-1-0518.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. government.

Keywords: Programmable Networks, Active Networks, Control Plane Extensibility, Mobile Code

Abstract

The Internet is evolving from an infrastructure that provides basic communication services into a more sophisticated infrastructure that supports a wide range of electronic services such as virtual reality games and rich multimedia retrieval services. However, this evolution is happening only slowly, in part because the communication infrastructure is too rigid. In this report, we present a programmable router architecture, in which the control plane functionality of the router can be extended dynamically through the use of *delegates*. Delegates can control the behavior of the router through a well defined router control interface, allowing service providers and third-party software vendors to implement customized traffic control policies or protocols. We describe Darwin, a system that implements such an architecture. We emphasize the runtime environment the system provides for delegate execution and the programming interface the system exports to support delegates. We demonstrate the advantages of using this system by presenting several delegate examples.

1 Introduction

The Internet has evolved from a basic bitway pipe to a more sophisticated infrastructure that supports electronic services. The services today are fairly primitive and are typically related to collecting information over the Web. Richer services such as high-quality videoconferencing, virtual reality games, and distributed simulation have been promised. Progress is slow in part because the infrastructure is inflexible. Routers are closed boxes that execute a restricted set of vendor software. Compute and storage servers are typically dedicated to support one type of service. An alternate architecture is to have an open infrastructure in which specific services can be installed and instantiated on demand, much like what we do on a PC today. One of the advantages of this approach is that it allows a larger community of people to develop services, which spurs innovation. We use some examples to motivate this approach.

The first class of examples addresses the customization of traffic control and management. Today, the range of traffic management options is fairly limited. While switches and routers increasingly have some support for classification and scheduling, these capabilities are often only used in simple ways, such as to filter out certain types of traffic, to do some simple prioritization of flows, or to implement standardized QoS mechanisms such as differentiated services. One could envision that users could employ these mechanisms to handle their traffic in specific ways. For example, one service provider could implement gold/silver/bronze service differentiation in a proprietary way, while another service provider implements communication services with stronger guarantees. Similarly, one could envision deploying a virtual private network(VPN) service, in which VPNs can use different traffic control policies or control protocols.

The second class of examples consists of value-added services, that is, services that require not only communication, but also data processing and access to storage. Examples include videoconferencing with video transcoding and mixing support, customized Web searching services, and application-specific multicast. While it is possible to deliver these services using a set of dedicated servers, it would be more efficient if services could deployed dynamically on servers leased on an as needed basis. This would allow the service to adapt to the demands and locations of the customers. Value-added services can also benefit from customized traffic management support. For example, a virtual reality game service provider may want to handle control, audio, and video traffic flows in different ways. This may require customized traffic control policies on the router.

As long as routers are closed boxes that are shipped with a set of standard protocols, it is unlikely that these examples will be realized. The above examples can best be supported by a programmable network infrastructure. Such a network will allow computing, storage, and communication resources to be allocated and programmed to deliver a specific service. Standards (e.g., ODBC, POSIX) exist to use storage servers (Web, file systems, databases) and compute servers. Routers (i.e., communicatin servers), however, are not programmable today. In this report we present a router architecture in which the control plane functionality of the router can be extended using *delegates*, code segments that implement customized traffic control policies or protocols. Delegates can affect how the router treats the packets belonging to a specific user through the router control interface (RCI). With this architecture, a broader community (e.g., third party software vendors or value-added service providers) can develop applications for routers.

The remainder of the report is organized as follows. We first define a network architecture in Section 2 in which network functions (e.g., QoS) can be selectively customized to meet the special needs of users. In Section 3, we describe Darwin, a specific instance of the above architecture. We focus on the runtime environment for delegates, code segments that can customize network control and management. In Section 4 we present some examples of how delegates can be applied to address a variety of resource management and traffic control problems, and we discuss security

issues raised by the use of delegates in Section 5. Finally, we present related work in Section 6 and conclude the report in Section 7.

2 A Programmable Network Architecture

We first characterize the network programmability requirements and introduce the concept of a delegate. We then present a programmable network architecture that can support delegates.

2.1 Network programmability

We can distinguish between two types of operations on data flows inside the network. The first class involves manipulation of the data in the packets, such as video transcoding, compression, or encryption. Since most routers do not have significant general-purpose processing power, this type of processing will typically take place on compute servers (e.g., workstation clusters inside the network infrastructure). In the future, these tasks could also be performed on routers that have been specifically built to also perform data processing, besides the usual routing functions. The second class of operations on data flows changes how the data is forwarded but typically does not require processing or even looking at the body of packets. Examples include tunneling, rerouting, selective packet dropping, and changing the bandwidth allocation of a flow. The nature of these operations is such that they are best executed on routers or switches.

We call the code segments that perform these tasks *delegates* since they represent the owner of the data flows inside the network. Data delegates perform data processing operations and execute on compute servers or specially designed routers. Control delegates execute on routers and are involved in the control of data flows. This simple classification of delegates is somewhat artificial since some delegates may fall in between these two classes. For example, a delegate that is concerned with control may need to look at the packet body to decide how to handle packets. Similarly, a "mostly control" delegate may on rare occasions have to modify a packet. Nevertheless, the distinction is useful because the two classes of delegates impose very different requirements on the system on which they run. Control delegates require an environment that provides a rich set of mechanisms to control data flows, while data delegates must run on a platform with substantial computational power.

While nobody is likely to argue against the use of data delegates on compute servers for data processing, the need for control delegates is less obvious. One could imagine routers with fixed functionality, similar to today's routers, where users can control how their traffic flows are handled by passing parameters to the routers using a signalling protocol. The examples discussed in Section 1 provide some reasons that directly executing code (i.e., control delegates) on the routers may be a more effective way to customize traffic control and management. A first reason is that control delegates can respond much more quickly to changes in the traffic conditions; it would take an entity at the edge of the network at least one and more likely several round-trip times before it could first observe and then respond to a change in the network. Second, it seems impractical to identify all possible user requirements *a priori*, so that they can be addressed by the default router software; an architecture based on delegates is more flexible and extensible. Finally, supporting customization by extending router functionality may often be a more natural and thus less error-prone solution. For example, if a service provider wants to use a routing protocol that is optimized for its traffic, doing so from the edges of the network is likely to be unnecessarily complicated.

DOCKET ALARM

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.