

Towards an Operating Platform for Network Control Management

T. S. Eugene Ng Alan L. Cox
Department of Computer Science
Rice University

1. INTRODUCTION

One of the key challenges in designing a new architecture for the future Internet is manageability. This position statement is concerned with the manageability of network controls within a single autonomous system.

Network controls are not systematically managed in the current Internet architecture, they are realized by an *ad hoc* combination of protocols, automated configurations, and manual configurations. For instance, intra-domain and inter-domain routing protocols are used for basic route computation; database-driven configurations of packet filters and tunnels are used to support access control policies and virtual private networking capabilities; manual configurations are used to support network maintenance operations. Unfortunately, creating these sophisticated network controls in such an *ad hoc* manner makes it difficult to control the interactions among them or to provide concrete behavioral assurances. Ultimately, the entire system might exhibit harmful emergent behaviors that are difficult to anticipate, test for, debug, or correct. One well-known example is the complex behavior resulting from the composition of intra-domain and inter-domain routing as described by Teixeira and Rexford in [2]. They observe that a local link failure within an autonomous system can cause inter-domain traffic to be re-routed unnecessarily and turn a local link failure into an event that can have far reaching impact. Even planned maintenance of network nodes by the operator is prone to causing service outages elsewhere in the network due to unexpected network behaviors. Incrementally creating custom interfaces between each pair of existing mechanisms to control their interactions does not scale well and may further increase complexity.

We believe the architecture for the future Internet should eliminate the current *ad hoc* practices and start from a clean slate. We propose to develop a new architecture for managing network controls to achieve two goals: (1) make network control creation systematic and simple, and (2) protect the network from potential harmful actions of network controls. In this architecture, each network control function is implemented as an *application* that runs on top of an *operating platform*. The operating platform provides a set crucial services:

- Provides a standard interface between the applications and the underlying network nodes. Informs applications of the state of the underlying network nodes. Generates network node configuration commands based on the actions of the applications.
- Provides services to facilitate the interactions among applications. Supports composition of applications. Supports concurrent executions of applications. Supports application scheduling policies. Supports isolation among unrelated applications.
- Monitors all application actions and ensures that network-wide operational invariants are protected against the actions of the applications.

The operating platform architecture moves the environment for network controls from the *ad hoc* practice of today to become much more structured like a modern operating system. Moreover, allowing new network controls to be easily created as applications can potentially create a new market for a variety of 3rd-party network control applications. Making the operating platform possible require many new areas of research, such as new abstractions and interfaces for the new way of programming network control applications, solutions to address the scheduling, synchronization, and inter-application communication and resource multiplexing problems for the new breed of network control applications, efficient approaches to verify network operational invariants on the operating platform, and new network control applications for specific network environments.

2. THE OPERATING PLATFORM

The operating platform is centralized. This is appropriate since many network control and management operations are centralized. The operating platform may be replicated to provide redundancy. The salient features of such an operating platform is that each network control function is implemented as an independent *application*. These applications can exchange information among each other, but they interact with the underlying network routers only via the abstractions and interfaces provided by the operating platform. The operating platform provides information on the underlying

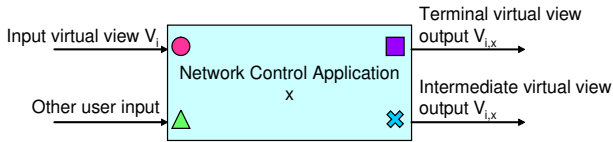


Figure 1: Network control application abstraction

network to the applications, coordinates and composes the actions of the applications, enforces network-wide invariants, and communicates with the underlying network routers to retrieve and install control state.

2.1 Objectives

Abstraction and Interface: Good abstractions and interfaces are critical to enabling application composition. We need a powerful abstraction and interface for the network view, which must encapsulate properties like network connectivity (both intra-domain and inter-domain), link characteristics (e.g. delay, bandwidth, loss rate), router capabilities (e.g. switching capacity, processing delay, buffer space, queuing capability, packet classification capability, packet filtering capability, etc.), traffic demand matrix, etc. We need an abstraction and interface for interacting with the inter-domain routing protocol that communicates with neighboring networks. We also need an abstraction and interface for the network control state, which encapsulates the desired actions of a network control application and allows different sets of network control state to be composed.

Inter-Application Coordination: The operating platform needs to provide interfaces and services to allow applications to coordinate and synchronize their actions. Applications may concurrently execute and generate conflicting outputs. Applications may have many inter-dependencies. They may also have vastly different computation resource and timeliness requirements.

Providing Protection: Providing the kind of network-wide protection that we envision is a complex problem. We can imagine many such protections such as simple link overload protection, or more sophisticated ones such as pricing-based inter-domain traffic routing protection. Enforcing these protections at runtime needs to be efficient, and when a violation occurs, the system needs to have a systematic way of either automatically resolving it, or notifying the human operator.

2.2 Meta-Management System

We assume a single autonomous network is under the control of an instance of the operating platform. The operating platform runs on a node that is connected to the autonomous network. This autonomous network may be connected to other neighboring networks. We assume an orthogonal protocol such as E-BGP is used to interface with these neighboring networks to exchange reachability information.

We assume each node in the autonomous network provides certain per hop behaviors (PHBs). We assume PHBs

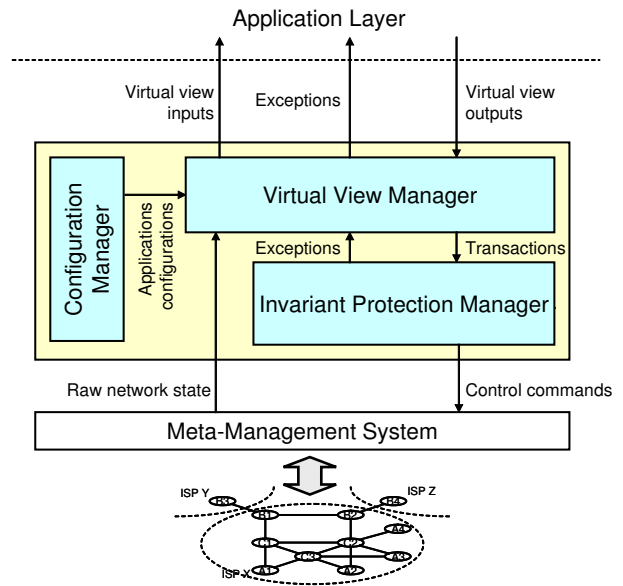


Figure 2: Operating platform components

are abstractly implemented via a multi-packet field lookup table. Each entry matches certain incoming packet(s), and each entry is associated with a particular behavior such as packet forwarding, packet dropping, packet tunneling, packet queuing and scheduling, etc. We assume the abstract lookup table that implements the PHBs can be dynamically configured to realize different network control policies.

To transport the control communications between the operating platform and the network nodes, the operating platform relies on an orthogonal subsystem called the Meta-Management System (MMS) [1]. The MMS is like a “BIOS” for the network. The MMS consists of a thin layer of self-contained software that pervades the autonomous network. When the network is powered up, the MMS automatically establishes and maintains a communication channel between network nodes and the management nodes as long as there is physical network connectivity. Using the service provided by the MMS, the network nodes can report their state to the operating platform, and control commands from the operating platform, including PHB configurations, node reboot, link shutdown, etc. can reach the network nodes.

Software that runs on a network node can collect its state (e.g. lookup table entries, link status, link bandwidth and delay, neighbor network reachability, traffic demands, etc.) and report to the operating platform via the MMS. This software also accepts control commands from the operating platform and configures the node’s hardware accordingly.

2.3 Operating Platform Architecture

2.3.1 Abstract Virtual View Object

The basic goal of a network control application is to monitor and/or effect changes to the underlying network accord-

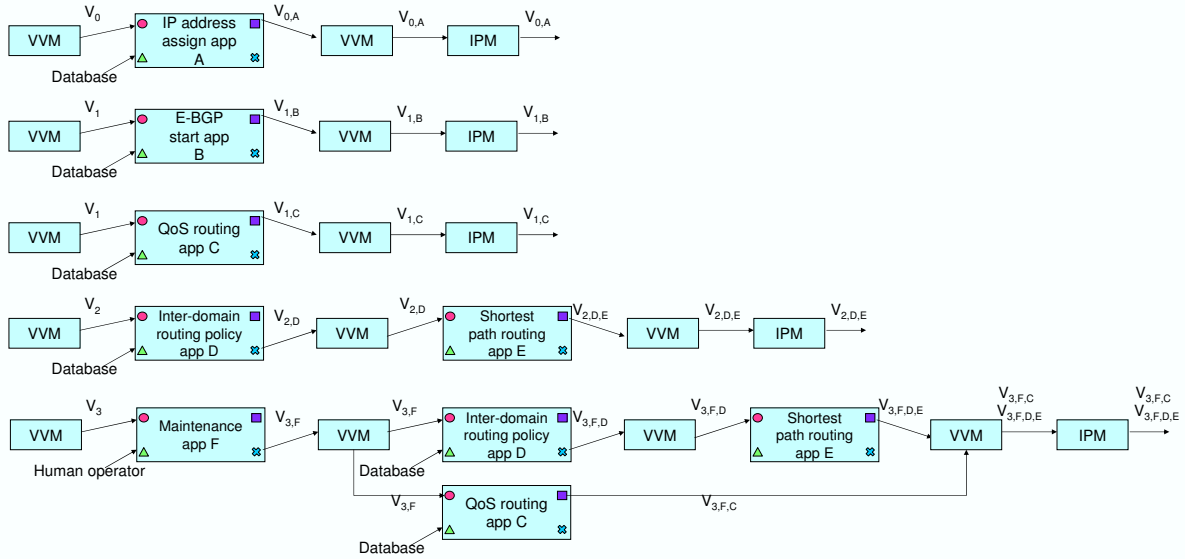


Figure 3: Operating platform usage examples

ing to its algorithms. To support these activities, the operating platform communicates with an application by passing virtual view objects. A virtual view object encodes the state of a virtual network that maps onto a portion of the underlying physical network. The ability to provide a virtual view instead of a complete view is useful and more efficient because an application may only care about a specific type of state (e.g. link connectivity status) of a portion of the network.

Figure 1 illustrates the overview of an application. An application x requests from the operating platform the network state it is interested in and receives an input virtual view V_i from the operating platform. The application may modify an input virtual view to produce an output virtual view $V_{i,x}$. The new virtual view ID (i, x) allows the operating platform to track changes. The output virtual view of an application is configured at instantiation time to either be *terminal* or *intermediate*. A terminal output virtual view is meant to effect changes to the underlying network immediately. On the other hand, an intermediate output virtual view is meant to be consumed by another application. By passing virtual views, complex functions can be realized by composing elementary applications. The composition is systematically handled by the operating platform.

2.3.2 Architecture Overview

The high level design of the operating platform is shown in Figure 2. The Virtual View Manager (VVM) uses the raw network state received from the MMS to generate the application requested virtual views and pass them to the applications. The applications produce output virtual views that are either terminal or intermediate. The Configuration Manager (CM) manages the compositions of applications and

informs the VVM. When the VVM receives an intermediate virtual view, it is passed to the intended application(s) accordingly. When related terminal output virtual views are collected, they are bundled into a single transaction and passed to the Invariant Protection Manager (IPM). If a transaction is accepted by the IPM, it sends control commands to the MMS to implement the configuration changes requested by the transaction. On the other hand, if the transaction is rejected, an exception is passed up to the applications involved which can inform the operator if necessary.

2.3.3 Virtual View Manager

The services provided by the VVM is illustrated by a series of examples shown in Figure 3. Each row in Figure 3 represents the execution of a set of related applications. When the network is powered up, the devices report their state to the operating platform via the MMS. The VVM collects the raw state and generates virtual view V_0 . V_0 contains only basic information about network topology and device capabilities. Thus, most applications would ignore it except the “IP address assign” application, which consults a database and generates the IP address assignments for the network interfaces (row 1). As soon as IP addresses are assigned, the VVM generates an updated virtual view V_1 . V_1 has IP addresses assigned and indicates to the “E-BGP start” (row 2), “QoS routing” (row 3), and “Shortest path routing” (not shown) applications that they can now start. By CM configuration, the QoS routing decisions override the shortest path routing decisions. Thus, the VVM generates the transactions accordingly and the IPM allows these routing decisions as long as they do not violate any invariants.

When E-BGP discovers reachability for remote destinations, it informs the operating platform via the MMS. This

leads to another updated view V_2 . The “Inter-domain routing policy” application reacts to V_2 and produces an intermediate output view $V_{2,D}$ that reflects the inter-domain routing policies. From the CM, the VVM knows that $V_{2,D}$ should be passed to the “Shortest path routing” application, which produces the inter-domain forwarding tables for the network. At this point, the network provides a best effort packet forwarding service as well as a QoS forwarding service.

Finally, suppose the operator uses the “Maintenance” application to attempt to remove a node from active service. The “Maintenance” application produces an intermediate view $V_{3,F}$ that is configured to be passed to the “Inter-domain routing policy” and the “QoS routing” applications. The VVM collects the terminal views $V_{3,F,C}$ and $V_{3,F,D,E}$ and creates a single transaction. If the transaction is accepted by the IPM, the node will be removed from active service and the forwarding tables will be updated accordingly. At this point the operator can perform any maintenance actions on the node via the MMS. If not, the entire transaction will be rejected and the “Maintenance” application will be notified of the failure.

3. SUMMARY

Network controls are not systematically managed in the current Internet architecture, they are realized by an *ad hoc* combination of protocols, automated configurations, and manual configurations. The architecture for the future Internet should eliminate the current *ad hoc* practices and start from a clean slate.

This position statement outlines a new operating platform architecture for managing network controls. The operating platform architecture moves the environment for network controls from the *ad hoc* practice of today to become much more structured like a modern operating system. We believe this architecture allows powerful network controls to be developed and deployed in a simple and systematic way, and the network can be protected from potential harmful actions of network controls.

4. REFERENCES

- [1] David A. Maltz, T. S. Eugene Ng, Hemant Gogineni, Hong Yan, and Hui Zhang. Meta-management system for geni. *GENI Design Document 06-37, Backbone Working Group*, April 2007.
- [2] Renata Teixeira and Jennifer Rexford. Managing routing disruptions in internet service provider networks. *IEEE Communication Magazine*, Mar 2006.