Towards a Framework for Network Control Composition

T. S. Eugene Ng
Department of Computer Science
Rice University

Hong Yan
Department of Computer Science
Carnegie Mellon University

ABSTRACT

IP networks nowadays perform many functions in addition to best-effort datagram forwarding. These functions are typically achieved via an *ad hoc* combination of distributed protocols, database- and tool-driven router configurations, and manual configurations. In such an *ad hoc* system, it is difficult to anticipate any potential harmful interactions among the control functions or to provide any behavioral assurances.

What kind of a framework will enable the composition of network control functions for sophisticated yet robust network control? Is it possible to have a framework in which each network control function is implemented as an independent *application* that runs on top of an *operating platform*, where the operating platform serves as an interface between the applications and the underlying network routers, provides services to facilitate the composition of applications, and ensures that network-wide operational invariants are not violated by the actions of the applications?

Using an application example to illustrate, we discuss some challenges that underlie the design of such a potential operating platform. We hope this article will stimulate discussions on more principled approaches for network control composition.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Packet Switching Networks; C.2.2 [**Network Protocols**]: Routing Protocols; C.2.3 [**Network Operations**]: Network Management

Keywords

Network management, robustness, control

1. INTRODUCTION

When the Internet was still in its infancy as the ARPANET, network control was simple. The chief requirement was to provide best-effort datagram forwarding, thus only one routing protocol was needed. Over the years, however, network control has become considerably more complicated as additional requirements have emerged. For instance, the growth of the Internet has led to the use of separate intra-domain (e.g. OSPF, IS-IS) and inter-domain (i.e. BGP) routing protocols for route computation. Another example is that packet filtering and tunneling controls have been introduced to support access control policies and virtual private networking capabilities. Other sophisticated controls such as traffic engineering and network maintenance are also commonplace. Today, these sophisticated controls are achieved by using an *ad hoc* combination of distributed protocols (e.g. OSPF, IS-IS, BGP), database-driven configurations (e.g. filtering, tunneling), tool-driven configurations (e.g. traffic engineering), as well as manual configurations (e.g. maintenance).

Unfortunately, by composing these sophisticated network controls in such an *ad hoc* manner, it is also 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 example is the complex behavior resulting from the composition of intra-domain routing and inter-domain routing as discussed in [1]. The observation is that a local link failure within an autonomous system can destabilize inter-domain routing. When a link failure occurs, the routing costs of inter-domain traffic egress points can change and cause inter-domain traffic to be re-routed unnecessarily. These routing changes can trigger BGP update messages, turning a local link failure into an event that can have far reaching impact. What is far more desirable is to design and compose network controls in a way that takes away such hard-to-control interactions.

A fundamental open question then is, what kind of a framework will enable the composition of network control functions for sophisticated yet robust network control? In this article, we specifically ask, is it possible to have a framework in which each network control function is implemented as an independent *application* that runs on top of an *operating platform*, where the operating platform serves as an interface between the applications and the underlying network routers, provides abstractions and services to facilitate the composition of applications, and ensures that network-wide operational invariants are protected against the actions of the applications?

Using a specific network control application scenario as a concrete example, we sketch how such a scenario can potentially be realized by composing simple network control building blocks. We also identify a set of challenges involved in realizing an operating platform that can facilitate such network control composition. Indeed there are many interesting challenges. For example, what are the fundamental abstractions and interfaces of such an operating platform? How can the actions of the network control applications be composed systematically? How can the network control applications coordinate their actions? How can network-wide operating

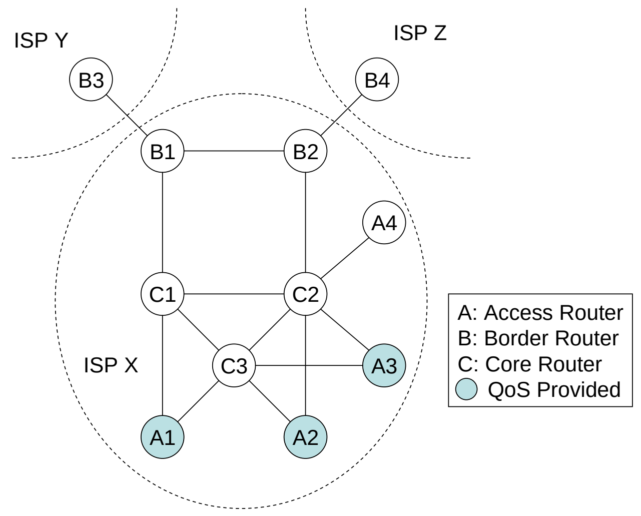

Figure 1: An example ISP network.

invariants be specified? How can such invariants be enforced? How should conflicts with the invariants be handled?

Although we do not propose any specific designs in this article, we believe that exposing the challenges will stimulate discussions on the feasibility and potentials of such an operating platform-based approach, as well as encourage the exploration of other principled approaches for network control composition.

2. OPERATING PLATFORM FOR NETWORK CONTROL COMPOSITION

To make the ideas for an operating platform for network control composition more concrete, we use a specific network control application example to motivate the discussions.

2.1 An ISP Network Scenario

Consider the network of a hypothetical ISP X depicted in Figure 1. The network consists of 2 border routers (B1 and B2, connected to the neighboring ISPs Y and Z respectively), 3 core routers (C1, C2, C3), and 4 access routers (A1, A2, A3, A4). The network control requirements of ISP X are:

1. By the service agreements with the neighboring ISPs, any external address can normally be reached via either ISP Y or ISP Z. However, ISP X defines an inter-domain routing policy such that ISP Y is the preferred egress unless the destination is not reachable via ISP Y.

2. For ingress inter-domain traffic, traffic can be received from ISP Y or ISP Z without care.

3. For traffic destined for addresses within ISP X, routing is based on the shortest path routing policy by default.

4. ISP X provides a QoS routing service among the customers connected to access routers A1, A2, and A3.

5. ISP X occasionally needs to take down a router for maintenance.

Although these requirements are somewhat artificial, they serve to illustrate several points. Let us consider how each requirement can be realized today. To implement requirement 1, a well known method [1] is to artificially inflate the OSPF link weight of link B2-B4 to make ISP Y the more attractive egress for routing calculations. This *ad hoc* method however creates a delicate dependency between the inter-domain routing policy and the internals of the intra-domain routing protocol. For example, if the B2-B4 link weight is not set high enough, ISP Z can still become the more attractive egress when the network topology or the other link weights change.

Requirement 2 and 3 are straight-forward and can be achieved by sending BGP route announcements to both ISP Y and ISP Z, and by using OSPF for intra-domain routing calculations, respectively. Requirement 4 can be achieved by tool-driven MPLS path configurations in the routers. Of course, when the topology or traffic load changes, the MPLS paths may need to be recalculated.

Requirement 5 is challenging. For example, in order to take down router C1, care must be taken such that ISP Y will remain the preferred egress even when C1 is down. This would require careful inspection and/or setting of the OSPF link weights. Also, the QoS routed traffic must be re-routed before C1 is taken down. Finally, the performance impact of taking down C1 due to traffic re-routing is difficult to know in advance since several independent routing mechanisms are at work.

In summary, achieving the requirements by combining all the aforementioned mechanisms results in a fairly complex system whose behavior is quite hard to predict and control.

2.2 Starting from a Clean Slate

What if we start from a clean slate? What kind of a framework would be suitable to support the composition of network controls in a principled manner? We believe such a framework should be based on at least two core principles: clear abstraction and explicit protection.

- **Clear Abstraction** - The *ad hoc* composition of network controls in practice today is partially due to the lack of good abstractions and interfaces. For instance, if there are clear abstraction and interface for the interactions between inter-domain and intra-domain routing, then one would not have to overload the link weight parameters in OSPF as an indirect and fragile means for choosing preferred egress points. There is no fundamental reason why the preferred egress points cannot be explicitly communicated to the intra-domain routing control. Making abstractions and interfaces explicit helps to prevent unintended side effects and remove implicit dependencies between network controls. The resulting system is more predictable and easier to manage.

  Having clear abstractions and interfaces also creates a better environment for innovation and evolution. Different controls that implement the same interface can be substituted for each other without affecting the operations of other controls in the network and they can be platform independent. This can stimulate a wider variety of control software to be developed. With well-defined interfaces, there is also less of a danger for getting *locked in* by a particular combination of network controls. In contrast, without clear interfaces, custom middleware may be needed to glue pieces together, creating a system that is much harder to evolve.

- **Explicit Protection** - Since there may always be faults in control software and composing the actions of multiple network controls may produce an unexpected outcome, a good framework for network control composition should provide explicit protection for the network and enforce certain network-wide invariants. An example of such protection is that the

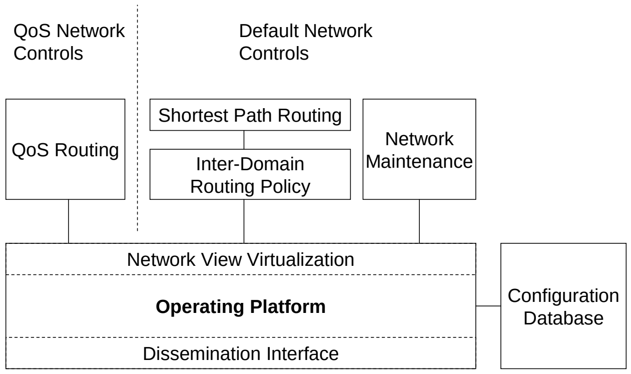

Figure 2: A potential operating platform organization.

combined actions of the network controls should not be allowed to route more traffic on a link than its capacity. Another example is that the network controls should not produce more routing table entries for a network router than it can handle. Providing such explicit protections for invariants that the network operator specifies would go a long way to improve the robustness of the network.

These core principles about abstractions and protections are certainly not new. In fact, they are inspired by the abstractions and protections found in today's computer operating systems. We believe the environment for network controls should move away from the *ad hoc* practice of today to become much more structured like a modern operating system.

2.3 The Operating Platform Approach

In this section, we discuss the particular ideas for an operating platform for network control composition. 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 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.

This approach is inspired by the 4D architecture [2], and the operating platform takes the role of the decision element in 4D. The communication interface with the underlying network routers can leverage the dissemination plane in 4D. The dissemination plane in the 4D architecture provides a robust mechanism for remote communications with network routers that is independent of the underlying network protocol. Using the dissemination plane, the operating platform can retrieve network state and send commands to the network routers to directly set their control state (e.g. forwarding tables).

Let us consider the example presented in Section 2.1 again in the context of the operating platform approach for network control composition. Figure 2 illustrates how the requirements might be achieved in this environment. In this figure, each box above the operating platform is an independent network control application. When the operating platform starts, it consults a configuration database to retrieve the network control requirements and then instantiates the required network control applications.

Via the dissemination interface, it collects the latest information about the underlying network view such as the intra- and inter-domain connectivity topology and the latest traffic demand matrix. Once the network view data is collected, it feeds this information to the applications through a Network View Virtualization layer.

The job of the Network View Virtualization layer is to appropriately augment the network view for each application. For instance, the network view for the QoS Routing application would exclude routers B3, B4, and A4 as they are irrelevant. The applications that deal with default network control, however, would be provided the complete network view.

The Inter-Domain Routing Policy application augments the complete network view to explicitly use preferred egress points for inter-domain traffic based on the required policy (requirement 1). Assuming BGP is the inter-domain routing protocol, the application also outputs commands to routers B1 and B2 to announce the appropriate inter-domain routes to the neighboring ISPs (requirement 2).

The augmented network view is then passed to the Shortest Path Routing application. This application would not even know that B2 can be an inter-domain traffic egress since such information has already been removed by the Inter-Domain Routing Policy application. Thus, it simply computes shortest paths based on the provided network view and generates the corresponding forwarding tables for each router. These forwarding tables are passed to the operating platform (requirement 3).

The QoS Routing application computes the special routes among A1, A2, and A3, generates the corresponding router configurations, and passes them to the operating platform (requirement 4).

The operating platform combines the outputs from the two stacks of routing controls, realizing that the QoS routing decisions take priority over the default routing decisions, to produce the combined control state for the network routers.

Before the operating platform injects the control state into the network routers via the dissemination interface, it verifies that the control state does not violate any network-wide invariants. For example, based on the traffic demand matrix and the proposed forwarding tables, it can determine whether any link will be overloaded. If such invariants are violated, a failure notification will be delivered. Potentially the network operator may decide how to handle the problem.

The Network Maintenance application can instruct the operating platform to augment the network view for other control applications. For example, if router C1 needs to be taken down for maintenance, the Network Maintenance application instructs the operating platform to mark the router C1 to be in maintenance state. This change in network view is passed to the other network control applications so that they can re-compute their network control state. Once the updated network control state has been injected into the network, the Network Maintenance application is notified that its request has been successful and the router C1 can now be taken down (requirement 5).

The five network control requirements in the example are thus satisfied in a systematic manner with all the interactions explicitly coordinated. The control state is not injected into the network unless it satisfies the network-wide invariants. Note that applications that perform the same function can easily substitute for each other. For instance, different QoS routing algorithms can be plugged in easily. Likewise, the inter-domain routing policy can be changed easily without having to change the other applications. Maintenance is handled in a step-by-step orderly fashion automatically. If the re-routing actions due to maintenance will overload certain network links, the operator will be notified to handle the problem.

2.4 Challenges

Although the approach seems promising, it is not surprising that, in reality, there are many challenges in building such an operating platform. The first set of challenges are related to facilitating the design and implementation of network control applications:

- **Abstraction and Interface Design** - Good abstractions and interfaces are critical to enabling application composition. We need 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 abstraction and interface for interacting with the inter-domain routing protocol that communicates with neighboring networks. We also need 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.

- **Composition Language Design** - As network control applications are treated as components running on the operating platform, we need a language to define how the components are connected via their data and control interfaces and how their network control state outputs should be composed. The composition language needs to be flexible enough to allow reasonably sophisticated network control compositions. The language should be capable of specifying composition ordering, prioritization of actions, and conflict resolution rules.

- **Inter-Application Coordination** - The operating platform needs to provide interfaces and services to allow applications to coordinate and synchronize their actions. An event broadcast and notification mechanism might be an appropriate component for this purpose. For example, in the network maintenance scenario, the Network Maintenance application triggers a router maintenance event, which needs to be processed by all the other network routing controls. Subsequently, if the network routing controls cause an invariant violation, then the Network Maintenance application needs to be notified that its action has caused an exception.

The second set of challenges concerns the operating platform itself:

- **Handling Application Diversity** - Different applications might have very different time complexity thus require different scales of running times. For example, finding the shortest paths in a network with hundreds of routers takes less than one second, while optimizing paths to balance load can take hours. Applications might also differ in priorities: when a link fails it is critical to change routes to bypass the failed link; when traffic patten changes, it is not as urgent to update routes as long as the traffic pattern change does not cause network congestion. The operating platform needs to provide interfaces for applications to express their desired run-time support and schedule the executions of the applications accordingly.

- **Providing Protection** - Providing the kind of network-wide protection that we envision is a complex problem. First, we need an abstraction and a language to describe the protections desired by the network operator. 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 run-time 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.

- **Performance, Robustness, and Security** - Last but not least, achieving high performance, robustness and security is a huge challenge. The first question we must face is, to what extent should the operating platform be replicated or distributed. Replication and distribution help to improve system robustness, but whether there is a trade-off with performance is not yet clear. A naive implementation of a distributed operating platform may have such a high communication overhead that diminishes the robustness benefits.

  The system must be designed to resist denial-of-service attacks. A promising approach is to achieve this in the underlying dissemination plane that connects the operating platform to the routers. Since the underlying dissemination plane can be protocol independent, the operating platform need not be IP capable. That is, even though the system is physically connected to the underlying IP network where malicious attackers reside, it is logically disconnected from the underlying IP network and no IP data traffic can reach it. The only way to reach it is via the dissemination plane, which can have a much more restricted and secure access model than IP. This should go a long way to enhance system security.

3. RELATED WORK

Programmable networks developed by the Opensig and Active Network communities (see [3] for a good survey) have exploited concepts like network virtualization and service composition. However, the focus of programmable network research is on enabling the dynamic deployment and composition of high level services (e.g. QoS-aware video conferencing services) and making the network extensible via dynamic deployment of new protocols (e.g. mobile IP). In contrast, this article is focused on refactoring the existing nuts and bolts functionalities in IP networks (e.g. packet forwarding and router maintenance) into simple applications to make low level network control more principled and less error-prone. Supporting the composition of these low level network control applications will require synchronization, conflict resolution, and protection mechanisms to be developed.

Teixeira and Rexford [1] studied the problems of routing disruptions in ISP networks and described the challenges faced by network operators. They suggest the research community to "investigate alternative approaches instead of proposing incremental enhancements to the protocol that fix one aspect of the problem at a time." We believe that the root cause of the difficulties is the lack of a unified framework, and we take one step further to describe a potential operating platform that can systematically solve network control problems by composing control applications.

Feldmann et. al. [4] built a network traffic engineering tool called NetScope, which collects network configuration files, generates network models, and runs simulation on the models to find configuration errors and answer "what if" questions (e.g. to estimate the impact of configuration changes). NetScope is a powerful offline traffic engineering tool, but it is not designed to serve as a control platform that runs all types of applications to directly control the network.

Our work is inspired by a clean slate 4D architecture [2] which re-factors today's routers into four planes: data, dissemination, discovery, and decision. Network information is collected by the discovery plane and disseminated to the decision plane where network configuration is computed and pushed to the data planes of the routers. The focus of this work, however, is to zoom into one critical aspect of any 4D-like network architecture, namely composing network control applications. We believe that in order to build a versatile network control system, the composition problem must be addressed.

The idea of composing network modules has been explored in many previous works. For example, the *x*-Kernel protocol framework [5] defines interfaces for network protocols to operate on one another. Network protocols that implement the *x*-Kernel interfaces can be composed in the framework to serve an end host or a router. As another example, Lakshminarayanan et. al. [6] have studied the problem of composing network services such as filtering, intrusion detection, anonymization, transcoding, and caching. The idea is to deploy intermediate processing points (middleboxes) so that packets traversing the boxes receive a sequence of services. Although protocol composition and service composition are different problems from network control composition, some underlying issues such as composition language design are similar. The insights gained in these previous works should shed light on the network control composition problem.

4. SUMMARY

The current practice of *ad hoc* network control composition creates much unwanted complexity that the practice is unsustainable. A systematic framework is needed to achieve robust network control and to create an environment that favors innovation and evolution. Such a framework must define clear abstractions and provide explicit network-wide protections.

We explore the idea of creating an operating platform for network control composition. The operating platform serves as an interface between the network control applications and the underlying network routers, facilitates the composition of applications, and enforces network-wide invariants to protect the network. Exploring these ideas highlights the promising potentials of such an operating platform, and leads to a better understanding of the challenges involved in realizing it. We believe these challenges are not unique to the operating platform approach. Other approaches will likely share many of the same issues.

We hope that by formulating the problem, discussing a particular approach, its potentials, and highlighting the challenges, we can stimulate discussions and research on future architectures and mechanisms for network control composition.

5. REFERENCES

[1] Renata Teixeira and Jennifer Rexford. Managing routing disruptions in internet service provider networks. *IEEE Communication Magazine*, Mar 2006.

[2] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4D approach to network control and management. *ACM Computer Communication Review*, October 2005.

[3] Andrew T. Campbell, Herman G. De Meer, Michael E. Kounavis, Kazuho Miki, John B. Vicente, , and Daniel Villela. A survey of programmable networks. *ACM SIGCOMM Computer Communications Review*, 29(2):7–23, April 1999.

[4] Anja Feldmann, Albert Greenberg, Carsten Lund, Nick Reingold, and Jennifer Rexford. NetScope: Traffic engineering for IP networks. *IEEE Network Magazine*, pages 11–19, March 2000.

[5] Norman C. Hutchinson and Larry L. Peterson. The *x*-Kernel: An architecture for implementing network protocols. 17(1):64–76, January 1991.

[6] Karthik Lakshminarayanan, Ion Stoica, and Klaus Wehrle. Support for service composition in i3. In *MULTIMEDIA '04: Proceedings of the 12th annual ACM international conference on Multimedia*, pages 108–111, New York, NY, USA, 2004. ACM Press.