Notes on Software Defined Networking (SDN)

Narek Galstyan and Scott Shenker

Caveats: This overview focuses on the conceptual foundations of SDN and presents a very idealized view of the basic SDN mechanisms. Talk to a real networking person for the practical details, which are far more complicated. Also, please see the selection of SDN-related papers referenced in the Appendix of these notes.

Context: Internet functionality can be divided into the dataplane and the control plane. The dataplane defines how a router/switch handles an arriving packet, based on local forwarding state; this is an inherently local process, with no external dependencies. The control plane defines how a router/switch establishes that forwarding state, which is an inherently nonlocal process because forwarding state must be based on knowledge of the overall network (i.e., how can you determine where to forward a packet without knowledge of the rest of the network?).

What is Software Defined Networking (SDN)? SDN is nothing more than how to arrange functionality in the Internet's control plane. This may not seem like much of an innovation, but remember that the Internet architecture is nothing more than how to arrange functionality in the Internet's dataplane. The dataplane is organized in layers, and we will soon see that SDN organizes the control plane into layers.

Why was SDN needed? The original control plane had only one goal: deliver packets from one host to another. In this case, the control plane merely needs routing protocols, such as those you've learned about in this class (e.g., link-state, distance-vector). Later in its evolution, the Internet was broken into autonomous systems (ASes) or domains, and the control plane had to deal with two routing problems: intradomain and interdomain. At this point, these two classes of routing protocols were all that was needed for the early Internet. However, as the Internet started to be used by enterprises, several new requirements arose. These include:

- Isolation of virtual LANs (see the appendix for some background on vLANs): this is to limit the scope of L2 broadcasts, which is done by defining virtual LANs (vLANs), with each broadcast packet indicating which vLAN it belongs to. This requires establishing forwarding state telling switches which ports to use to forward broadcast packets for each vLAN. This was typically done by manual configuration (i.e., a sysadmin entering them into a switch's forwarding state).

- Access control: Enterprises don't want all hosts to reach all other hosts (e.g., your bank doesn't want your laptop in the lobby to be able to reach their backend database that keeps account records). This is done by inserting Access Control Lists (ACLs) in routers that specify which packets can be forwarded to a particular destination. For instance, a forwarding entry might indicate the logical equivalent of "drop all packets from host X to host Y" if the network operator does not want packets from X to reach host Y. These ACLs were typically installed by manual configuration.

- Traffic engineering: Once the Internet became a serious business, the ISPs had to take great care that their links did not get overloaded. This requires making routing decisions that spread the load over all links to maximize the total capacity. To accomplish this, algorithms were designed that would take as input the map of the network and the traffic matrix of the network (how much traffic was going between every two endpoints) and then compute the forwarding state of each router. Because of the complexity of these computations, they were done in a centralized manner.

Thus, soon after going commercial the Internet's control plane required a collection of largely disjoint mechanisms: distributed algorithms (for intradomain and interdomain routing), manual configuration (for ACLs and vLANs), and centralized computation (for traffic engineering). There was no modularity (i.e., no reuse of components between these mechanisms), so each mechanism was largely designed from scratch. As networks became larger, the ad hoc nature of the control plane caused significant problems with network management. SDN arose from a desire to develop a modularity for the control plane. See the lecture for a more detailed description of the use case (multi-tenant datacenters) that eventually led to SDN's adoption.

What is the scope of SDN? This is an issue that Scott forgot to mention in the lecture. Almost all control plane issues (aside from interdomain routing) involve what happens within a given particular domain's network (e.g., the internal network used by a company, university, or ISP) or even just a portion of that network (e.g., a specific datacenter or building). Thus, SDN is designed to manage the control plane in only a narrowly defined scope under the control of a single administrative entity. For now, think of SDN as controlling UC Berkeley's network or that of a large datacenter (by large we mean hundreds of thousands of servers and tens of thousands of routers/switches).

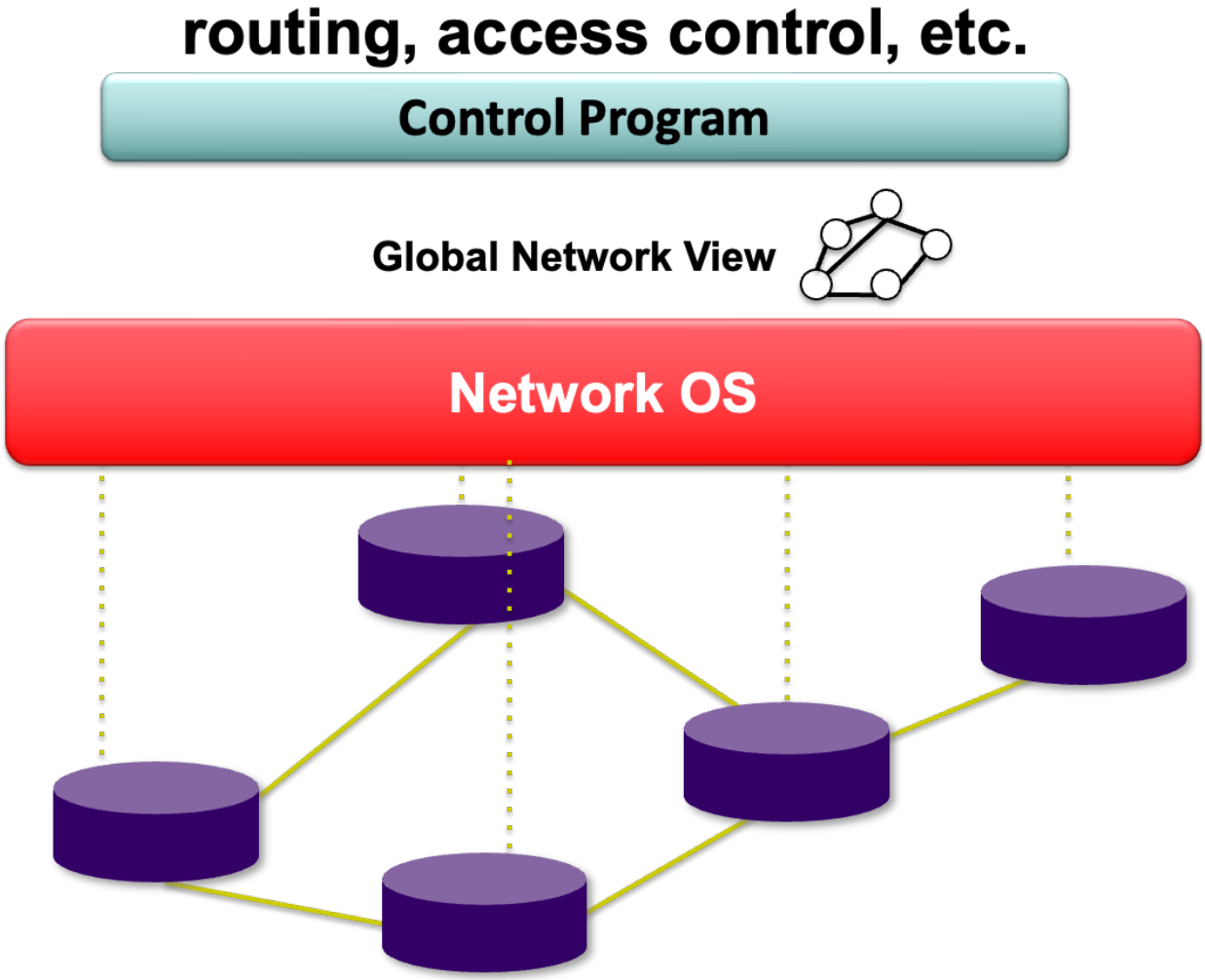
What is the modularity of the control plane? *This is the central question of SDN*. The fundamental task of the control plane is to compute the forwarding state. We can break this down into several subtasks (the ordering here is different from the lecture, but the content is the same).

First, we recognize that the control plane needs to know the overall topology of the network in order to accomplish many of its tasks. Thus, one task is to collect this topology information and present it to whatever control program will compute the forwarding state (such as a routing algorithm or an access control algorithm). This is accomplished by something called a Network Operating System (NOS), which runs on some servers within the network. Think of the NOS as logically centralized (i.e., it can be represented by a single logical server) but is physically distributed (using various replication algorithms). Each router/switch reports to the NOS which neighbors it is connected to, along with information about those links (latency, bandwidth, etc.). The NOS can then construct (much like in link-state routing algorithms) the overall network topology.

Second, once this topology information is gathered, and the control program has computed the forwarding state, this state must be communicated to the switches/routers. Vendors of networking equipment often have their own proprietary hardware and software, which use their own formats for specifying forwarding entries. Thus, the second subtask is to (i) allow the NOS to communicate the set of forwarding entries to each router/switch, and then (ii) convert these generic forwarding entries to the vendor-specific format of the receiving router/switch. OpenFlow is the proposal for this protocol.

Combining these two abstractions, you end up with the picture on the following page. The control plane consists of a control program (routing, access control, etc.) that uses the network graph constructed by the NOS. After the control program computes the desired forwarding entries (in the OpenFlow format), it then passes them to the NOS which uses OpenFlow to communicate these flow entries to the individual routers/switches. The OpenFlow implementation on these routers/switches convert these flow entries to their formats used by those particular routers/switches.

The control programs and the NOS can be thought of as the initial two “layers” of the control plane. When we have a layered architecture like this, we can think of the northbound interface to mean the interface to the layer above (with the northbound interface of the top layer being exposed to the network operator), and the southbound interface referring to the interface to the layer below (with the southbound interface of the bottom layer being exposed to the physical routers/switches).



However, this approach requires a sophisticated control program that can compute all of the forwarding entries. In the hope of simplifying the task of writing control programs, we can insert a new abstraction called the virtualization layer. The virtualization layer is tailored to the specific task (e.g., vLANs or access control) and presents a simplified version of the network that contains only enough information to allow the control program (or network operator) to specify the desired functionality. For instance, for access control, the virtualization layer can just present a single switch with all hosts showing up as endpoints, and the control program merely states which pairs of hosts

can communicate. The virtualization layer then translates these specifications into a set of forwarding entries on the full network graph that achieves the desired goal. Think of the virtualization layer as a compiler that translates between high-level control plane goals and low-level implementations of those goals.

With this design we then can describe the full SDN architecture as consisting of three layers, where we use the term “configuration” to refer to the set of forwarding entries of the network.

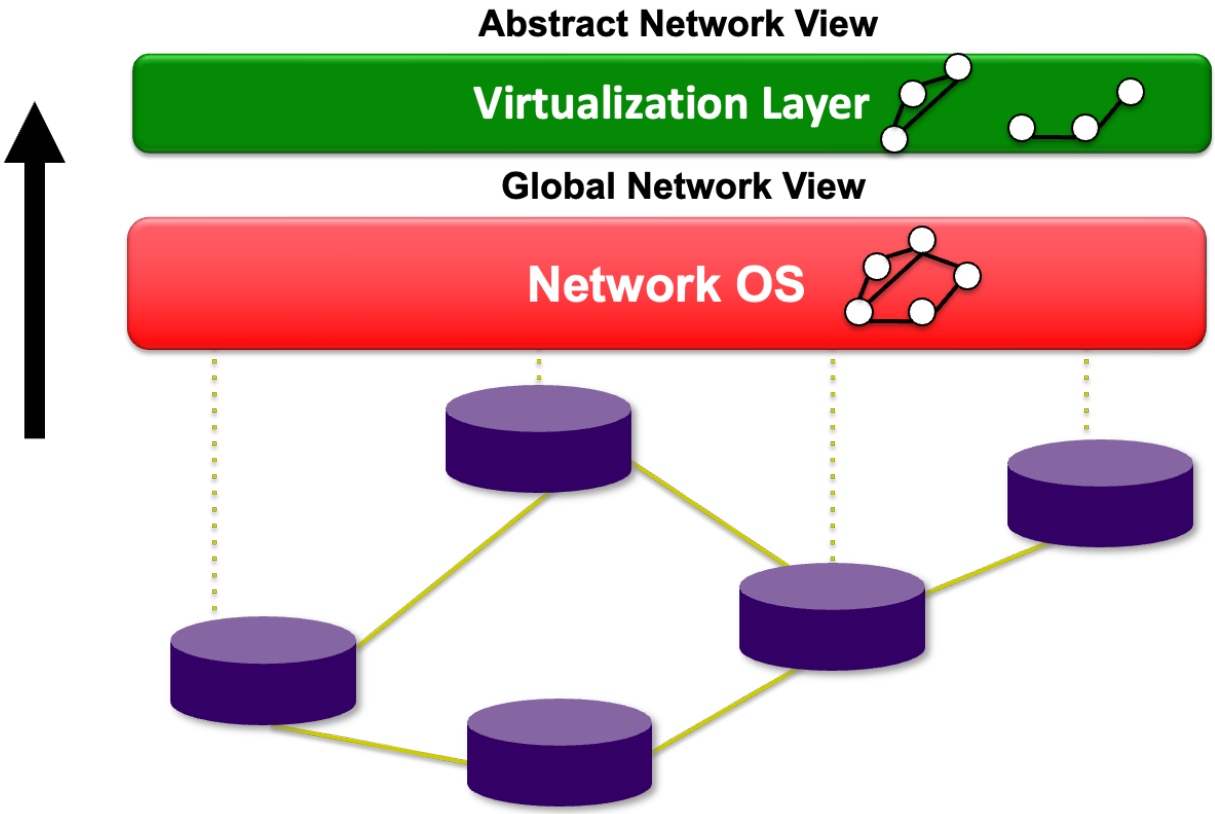
1. A set of control programs that dictate access control, traffic engineering, etc.
    a. Northbound interface: Allows network operator to specify their goals
    b. Southbound interface: The control program configures the abstract network view as provided by the virtualization layer.

2. The virtualization layer that accepts the configuration of the abstract network from the control program and converts it into a full set of network forwarding entries for the network graph.
    a. Northbound interface: An abstract network view
    b. Southbound interface: A set of forwarding entries for the full network view

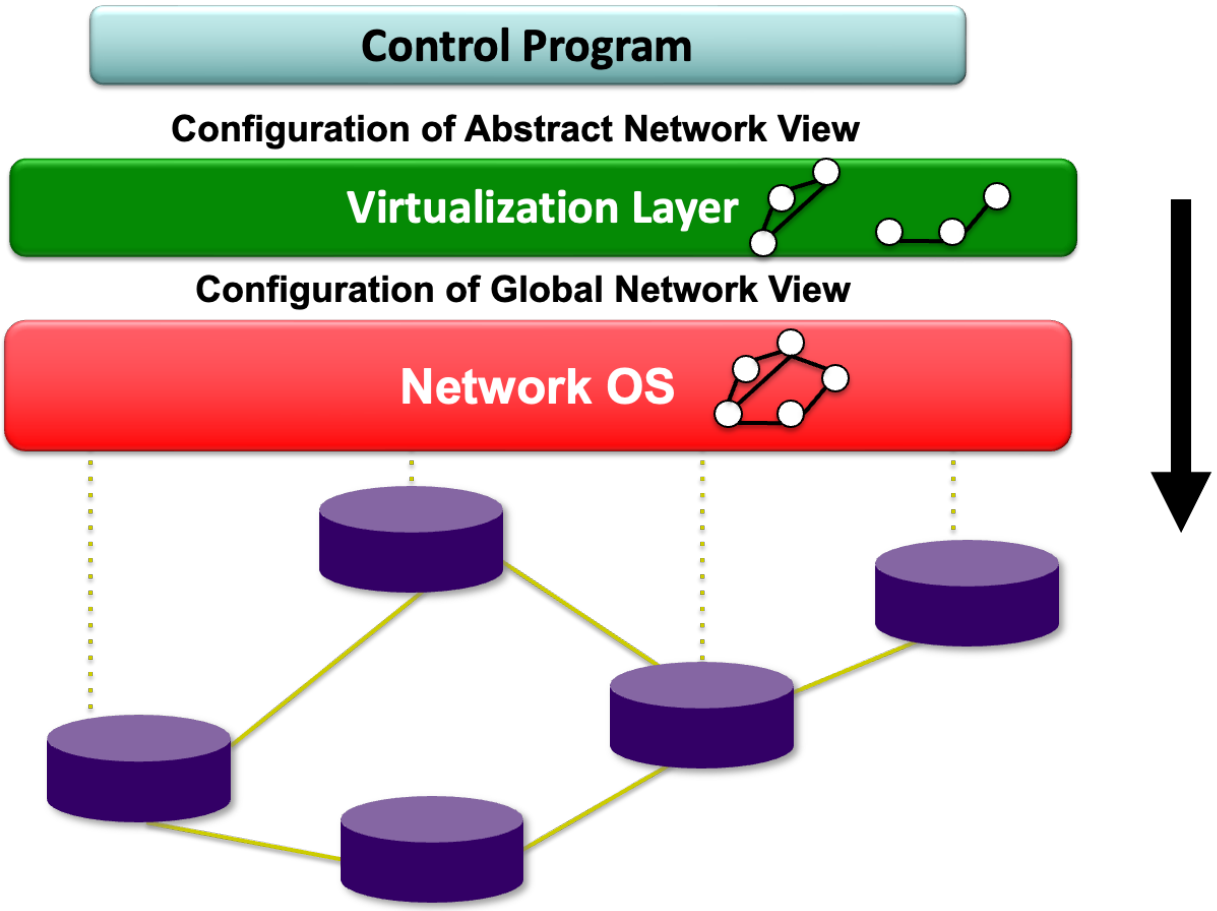
3. The NOS that accepts the configuration of the full network view from the virtualization layer and communicates the specific forwarding entries to each individual router/switches.
    a. Northbound interface: An abstract network view
    b. Southbound interface: A set of forwarding entries for the full network view

The following two pictures show the flow of information up (as information is collected from routers/switches, used to construct a network view, which is then passed to the virtualization layer) and the flow of information down (as the configuration of the abstract network view is converted into the configuration of the full network, and then passed to switches/routers).

Information Flowing Up



Information Flowing Down



Why is SDN useful? SDN modularizes the control plane, but why does this help? SDN delivers two benefits, both of which are standard results of modularization in other large-scale systems. The first benefit is a separation of concerns so that one can evolve parts of the SDN design without disturbing other parts. For instance, one can introduce new control programs without changing the NOS or OpenFlow (and in some cases without changing the virtualization layer, if the abstract view does not need to change). This is just like in the Internet's dataplane, where changing an application or a transport protocol requires no change in IP or in L2 designs.

The second benefit is that SDN pushes all the complexity into reusable code. The hard parts of the control plane are in the virtualization layer (to translate configurations of abstract network views into configurations of full network views) and the NOS (to collect the necessary topology). These need not change very often. Just as compilers are complicated beasts that make writing programs easier, the SDN layers are complicated pieces of code that make writing control programs easier (e.g., a routing protocol because just a graph algorithm).

The End: This concludes our short summary of the technical content of the lecture. The lecture itself contains more information on how we managed to get this deployed without any hardware changes.

Appendix

Some notes on vLANs: This course has previously covered local area networks (LANs), particularly how the Spanning Tree Protocol is used to enable loop-free flooding. As discussed, LANs provide seamless connectivity, host discovery, and communication, all independent of physical topology and wiring. However, these previous discussions assumed that LANs were small, so the issues of security or scaling did not arise.

These issues do arise when there are hundreds or thousands of hosts on a single LAN. For example, maybe you do not want all computers in the entire building to be able to discover the printer in your office. We can solve this problem by breaking down the single physical LAN into multiple logical LANs, called virtual LANs or vLANs. In the example given, it might be preferable to have a separate vLAN per floor of the building.

To achieve this, routers and switches must be configured to behave as if these vLANs were indeed separate. In particular, when packets are broadcast each router/switch must ensure that packets are only sent on the appropriate vLAN. This was typically

done by having network operators manually configure the routers/switches. As LANs became bigger, this manual configuration of vLANs in routers became painful.

Want to learn more about SDN? Below is a select list of publications on SDN. None of the content from these papers will be on the exam. These are here as further reading, in case you find the topics discussed in lecture interesting or would like to know what approaches have been proposed in the community.

NOX: Towards an Operating System for Networks - This was the first paper on a network operating system. NOX itself was not commercially deployed, but was a forerunner for the commercial systems that followed.

Onix: A Distributed Control Platform for Large-scale Production Networks - The paper describes the first commercial instantiation of an SDN controller. It provided a general purpose high level API that provided a global view of the network against which network control logic could be implemented. At a high level, Onix addresses the exact pain points described in the lecture but the paper goes into more detail about certain tradeoffs, comparisons to alternatives and important challenges.

RCP: The Case for Separating Routing from Routers - The paper notices that IP routers currently have two jobs: running a distributed route selection algorithm and forwarding packets along found routes. Motivated by similar pain points described in lecture, the authors propose a mechanism that removes routing from routers and puts it in *Routing Control Platform (RCP)* which selects routes on behalf of routers and installs the selected routes on IP routers. RCP focuses on software defined *interdomain* control planes while the lecture focused on software defined *intradomain* control planes.

A clean slate 4D approach to network control and management - The paper is one of the first attempts to address the control plane management and scalability issues discussed in lecture. The authors identify four pillars - decision, dissemination, discovery, and data - around which network management should be built.

Fabric: A Restrospective on Evolving SDN - This paper takes a critical look at the original SDN formulation and concludes that while “a significant step forward in some respects, it was a step backwards in others” and proposes how to change SDN to avoid those problems. This is a short paper that is easy to read.