Software Defined Networking

Ananth Balashankar - 3/8/2018

This week, we will talk about Software Defined Networking. In previous classes, we have talked about the control and data plane and how they are handled in a router. The control plane which does the bookkeeping of how packets need to be routed using tables and the data plane which looks up these tables based on header prefix matching and does the actual packet forwarding. The key idea in Software defined networking is to separate the two planes so that they are done on separate processors. Doing so affords both these planes to be more flexible and tunable by network administrators.

Ethane

Ethane was the first paper which introduced the idea of SDN, although doesn't explicitly mention it in the paper. Its motivation was based on security in enterprise networks, where defence against unauthorized break-ins is the prime issue. It later turned out that the same architecture could be used for traffic engineering and network function virtualization (NFVs) in middleboxes.

Data Plane

In the data plane, Ethane implemented the notion of "Match Action processing" which allowed network admins to map matching of most network headers to do certain types of actions. This was a major extension from tradition routers which were capable of doing only destination address matching for forwarding purposes. In Ethane, it is possible to write matching rules using wildcard expressions against TCP header, IP header and Ethernet MAC. The actions specified could be one of "Drop" or "Forward" or "Prioritize". While drop and forward are self-explanatory, prioritize allowed routers to maintain separate buffer queues depending on the tasks and the quality of service required for them (for e.g. VoIP vs FTP). It was also possible to do a combination of these tasks using a policy language, which looks like a very simple program. No new actions other these could be added as the underlying routers (NetFPGA) at the time couldn't support them. For example, use cases like weighted fair queueing and XCP couldn't be implemented as they either needed more information in the headers or more actions to be supported. We will see more on this in the next lecture. It turned out that this principle of using existing hardware turned out to be the main USP of Ethane for faster adoption in the industry.

Control Plane

However, the true software part of SDN lies in the control plane. A centralized controller which captures the entire network on a single processor was very useful to implement complex security actions which would otherwise be difficult in a distributed setting. It also turned out that the limitation of a single controller could be overcome in follow up works using multi-threaded and fault tolerant implementations. As a centralized controller, it would need to be aware of the authenticity of the nodes in the network. This was accomplished through registration. At a user level, authentication was done using username/password of the Stanford network. For switches, credentials were registered and individual hosts were registered using mac addresses. Users might map to a set of hosts. Policies were then defined on users, which would be less brittle and easy to understand as hosts might change ownership within a network. In order to construct the network at the controller, a minimum spanning tree algorithm was run rooted at the controller consisting of all routers. This ensured that network pathways were created to/from the controller and each of the routers. In order to check the authenticity of the nodes, the controller checked if the user/host existed in the registry.

Flow maintenance

For every packet from the host, the router can't find the forwarding address, it forwards the packet to the controller. Flow entry table is updated by the controller to all the switches in the path that is determined based on the shortest route. Controller can also route it through middleboxes (Network function chains) to check for spurious traffic instead of the shortest path. The downside of this method is that latency is high when a new flow comes up. Although controller is the bottleneck and needs to handle a large rate of updates, it might not be a problem in real networks - 10K new flows/second in a 20K host network as shown in the evaluation of the paper. However, when the controller needs to be updated, there still exists a spike in latency. Broadcasts are handled by forwarding to all hosts from the controller.
The upside to this approach is that with increase in the number of active flows through the switch, the size of flow table is still moderate as the controller which has the complete picture of the network can make updates to the flow table based on wildcard matches. This significantly reduces the memory requirement on switch chip. This directly translates to savings in cost & reductions in defects associated with lower memory.

Shortcomings

Certain broadcast based protocols based on ARP floods the network. This decision was made to trade off choosing a complicated controller vs more traffic on the network. Security is not guaranteed if the hosts resort to application layer routing. For example if host A can talk to B, but A is not supposed to talk to C. If B is allowed to talk to C, A and B can agree upon application level packet forwarding such that A can communicate with C through B. This is not easy to overcome especially when the packets are encrypted. Inferring user intent is hard when it comes to blocking certain users from doing certain actions. For example, to check if all SSH connections are whitelisted, we can check port 22. But if another port is agreed upon for communication by the end hosts (which are also supported by the protocol), there is no way to disallow SSH between unauthorized hosts. The policy becomes unclear as there are some legitimate reasons to go around security policy (to access blocked content in a network), in which case the network administrator is the bad actor.

OpenFlow

The authors of Ethane approached existing network equipment manufacturers to deploy some of the functionalities in their networks. However, a centralized controller was antithetical to their philosophy at the time, even for enterprise networks. This led to the founding of the company NICIRA which also established the standard of OpenFlow to guide network vendors to allow for flexibility for experimentation in college campuses for research purposes. However, it later came to be used for more than research purposes too.

Protocol
OpenFlow works on the basic principle that all vendor switches need to implement flow tables for forwarding. By allowing the network administrator to separate production and research flows, the protocol allows flexibility for researchers to update the research entries as per their protocol while the production flows are unaffected. It specifies a standard through which controllers can communicate with switches of all vendors. This ensures that researches do not need to program the switch every time there is a protocol change. Actions supported in OpenFlow are forward, encapsulate and forward to controller for new packets or drop. An entry in the flow table also has the header which is a 10 tuple 2 fields for TCP header, 3 fields of IP and Ethernet, 1 field each for in-port and VLAN-id. The rules to match can be mentioned as wildcards or exact matches of any of these fields. Each entry also keeps statistics of bytes transmitted, etc to drop inactive flows.

Outcomes
Some of the outcomes of the OpenFlow standard directly impacted cloud computing. A client of cloud computing would want to share the same underlying infrastructure with different virtual machine networks. (Network virtualization). E.g different departments within an enterprise network. Some of the functionalities that might be required like dropping packets between networks can be easily implemented using a controller. When it turned out that not all switches need to be configured as per the OpenFlow standard due to the emergence of virtual machines running within each host, virtual switches were used to route within host traffic. Virtual switches are written entirely in software and can be modified by enterprise clients themselves similar to a VL2 agent. So, when it became hard to convince network vendors to even standardize a subset of their API, it became much easier and scalable to modify the edge which was an open virtual switch than modifying the core which was supposed to be an OpenFlow switch.

Virtual switches are also the perfect fit for virtual machine hypervisor to modify. We will see in the next class if programmable edge virtual switches are enough or do we need actual switches also to be programmable. However, one of the drawbacks of the virtual switch is that it is running inside a virtual machine and can be much slower than an actual switch. But, it has very less load than an actual switch (only the one within a VM). Microsoft has recently implemented accelerated FPGA to boost the network part of virtual switches. Access control can be easily enforced in a virtual switch. However, we still need the controller to be aware of the entire network, both actual and virtual switches in order to operate and maintain the flow tables. Some of the operations like encapsulation, NAT and pattern matching not supported in OpenFlow.

Thoughts and discussion:

1. Where is OpenFlow implemented?
   Network equipment industry is closed, which made it hard for enterprise networks to customize based on their requirements. OpenFlow allows to use existing chips/APIs to implement SDN, no modifications required.

2. Ethane has shown preliminary work on many threads, single server controller. Did it scale?
   Since the publication, SDN is used in industry. Each of the ideas mentioned in the Ethane paper - Controller theory, consistency and policy language have led to follow up work. Although inspiration is owed to 4D, Ethane managed to go beyond solving the problems of the ISP and generalize it.

3. What if the govt takes over the network?
   Network controller is trusted.

4. Ethane had a deployment
   Testbed is the norm, but they got operational experience at Stanford, which influenced some of the decisions they made.

5. What functionality was achieved by a central command?
   Although there were no new things to do, Ethane provided a better (manageable) way to do the same functions. For example, instead of running distance vector in a distributed manner, you can run Djikstra's on the controller.

6. Related work: Some of the earlier work was to mine switch configuration based on traffic statistics to get the policy language, which proved to be cumbersome and unnecessary in hindsight.

7. What's different between Enterprise vs Datacenter networks?
   DC networks are much larger, inhouse and automated and run third party applications. However, enterprise networks are more cluttered originally, more for cloud computing use cases.