CSC 2209, Course Project - Network Visualization

Due: Dec 14, 2012

Yujia Li, 998835435
yujiali@cs.toronto.edu

1 Introduction

In this project, I created a tool named "NetVis", that can detect the connections between components in a small network built using OpenFlow switches. Using the OpenFlow protocol, NetVis can control the switches to send out probe messages, and collect information from incoming packets sent by other switches as well as the hosts. By inspecting the incoming packets, NetVis is able to infer the connections between switches and between hosts and switches. The structure of the whole network is then visualized using a graphical user interface (GUI), and the users can further interact with the network components in the GUI to get more detailed information, e.g. IP and MAC addresses, on each of them.

NetVis is implemented in Python using the POX API and tested with Mininet 1.0, a light weight network simulator that supports OpenFlow switches. Mininet 2.0 is now also available. But our software has yet to be tested on it and more importantly on real networks.

The software is available for download at `http://www.cs.toronto.edu/~yujiali/proj/netvis.html`.

2 How to Run it

The content of this section is almost the same as the "Run NetVis" part on the project website. It is kept here for completeness.

To run NetVis, one should first make sure OpenFlow and POX are correctly installed. Mininet is also required to be installed to run the simulation.

The recommended setting to run the software is to use a virtual machine that runs the system image provided in OpenFlow tutorial[1], which is also the environment used to develop NetVis.

After uncompressing the package, the user should copy the `ext` directory and `pox_netvis.py` to the root directory of POX (on the recommended system, it is `/home/openflow/pox/`). NetVis can be started in this directory by command

```
./pox_netvis.py --no_cli net_discovery
```

The program will print a lot of information out on the command line. A GUI with a blank canvas is started and waiting to display network structure.

One can use Mininet to test NetVis. For example, the following command

```
sudo mn --topo linear,3
```

[1] Available for download from `http://www.openflow.org/wk/index.php/OpenFlow_Tutorial#Install_Required_Software`

will start a network with 3 switches connected in a chain, and each switch has a host attached to it.

After the network is set up, one can see NetVis already detected all the switches and all connections between them. Then go back to the CLI interface of Mininet and do a pingall test. Now NetVis should have detected all hosts and connections.

In the package, there is a nice tool `myminiedit` that allows users to customize the network used to test NetVis. `myminiedit` is based on the `miniedit` example provided in Mininet package, which is slightly changed to run a CLI after the user click the "Run" button. With `myminiedit`, a lot of complicated networks can be used to test NetVis very easily.

In the NetVis GUI, switches are labeled as 's' followed by their unique data path IDs, while hosts are labeled as 'h' followed by a host number. Here the host number is set to be the lower 3 bytes of the IP address of the host, which matches the host number used by Mininet. The user can move the cursor on a switch, a host or a link, to view further information, including MAC address, IP address and port number, about them.

3 How it Works - In More Details

The "How it Works" part on the project website provides some ideas about how NetVis works. In this section, I will describe it with more technical details.

The starting point of this project is the observation that every packet in the network carries some information about its sender in its header, e.g. IP and MAC addresses. By using these information and information about the receiving ports (network interfaces), NetVis is able to discover connections in the network. After that, NetVis will visualize the network nicely in a GUI.

The necessary information is collected using OpenFlow protocol. In a network built with OpenFlow switches, all switches are connected to a central controller through a secure channel. The switches will send messages about incoming packets and other events to the controller and the controller decides how the switches should behave and sends back control messages. NetVis is integrated with the central controller, which has access to almost all information flowing in the network.

3.1 Switch Discovery

NetVis only handles two types of messages: handshake and packet-in. Handshake messages are exchanged after the connection between the controller and a switch is established. During the handshake, the controller will send feature request (`OFPT_FEATURES_REQUEST`) message to the switch, requesting more detailed information about it. The switch will respond in a feature reply (`OFPT_FEATURES_REPLY`) message, which contains the unique data path ID of the switch, the detailed description about each port on the switch, and the buffer size, etc. NetVis analyzes this feature reply message and create a node for the switch using its own internal representation. All the ports on the switch along with their MAC addresses are also registered. This is the first step of network discovery. After all switches have established connection with the central controller, NetVis should have collected information about all switches and the ports on them.

Note that OpenFlow switches do not have IP addresses for every port. This makes the switches only able to work in the link layer. NetVis will only use the MAC addresses and port numbers to identify the ports.

3.2 Connection and Host Discovery

OpenFlow switches use a flow table to forward (or modify or drop, etc.) packets. Whenever a packet is received from a port, the switch will try to match the packet with one of the entries in

the flow table and handle the packet as described by the flow entry. If no match has been found for that packet, a packet-in (`OFPT_PACKET_IN`) message will be sent to the controller. This packet-in message contains the complete header of the incoming packet, along with information about the receiving port. By inspecting the packet header, NetVis is able to discover who the sender is. This is then combined with information about the receiving port to create a connection between the pair of network components, which can be a port-port connection or port-host connection. When the network is started, all flow tables are empty, so switches will send packet-in messages to the controller whenever a packet arrives. To discover port-port and port-host connections, the switches should receive packets from both other switches and the hosts.

OpenFlow protocol allows the switches to send out packets on their own. So NetVis asks the switches to flood probe messages out of every port after the handshake. By doing this, other switches that are connected to the sending switch will be able to receive the probe messages, which are then send to the controller via a packet-in message. By inspecting the packet-in message, NetVis will discover the connection between switches.

Since all packets received by the switches are handled by the controller, the probe messages do not need to have a valid format of any higer layer protocols as long as they got the Ethernet header right (they can even be empty Ethernet packets!). On the other hand, it is hard to send valid higher layer packets because the ports do not have IP addresses. To make the solution more "standard" looking, NetVis chooses to use ARP probe messages.

An ARP probe message is an ARP request with all-zero sender IP address. Here the destination IP address is also set to zero, and we only use the ARP probe message to announce the MAC address of the sending port, which is the only unique identifier for that port. When such a message is received by a host, it will be ignored. When a switch receives such a message, it will be handled by the controller correctly to establish the connection between the pair of ports.

To get packets from the hosts using the switches alone, however, is not easy. Because the switch ports do not have IP addresses, it is hard to get the hosts reply to any packets they received from the switches. Therefore to complete the network discovery, each host is required to do a ping test to all the other hosts. To do a ping test, a host will first send an ARP request message to the switch connected with it to ask for the MAC address of the target host. This packet will also be delivered to the central controller. The IP and MAC address of the sending host can be read from the header, so the sending host along with the connection between the host and the receiving port will be discovered by NetVis.

NetVis will then create visual representation for the hosts and connections in the GUI. The data path ID of the switches, the MAC addresses of the ports, and IP and MAC addresses of the hosts will be displayed on the GUI, which can help the users identify switches and hosts in a real network.

4 Optimizing the System

4.1 Spanning Tree Algorithm

The system is tested on Mininet, which has a nice tool called `pingall` that allows the user to run a ping test for each pair of hosts. However, if NetVis only collects information from the packets received from the hosts and never respond, the pingall test will take a very long time to finish because it will always wait for a timeout for every pair of hosts.

This can be improved by enabling the hosts to forward packets. The simplest approach is to flood any packets received from a port out of all other ports. However, this flooding strategy only works for networks that do not have loops in it. If there is a loop in a network, and packets are

flooded, any unanswered packets will be forwarded in the loop forever.

This problem is solved using the Ethernet spanning tree algorithm. In NetVis, the implementation of the spanning tree algorithm is a simpler version of the standard, where nodes and links can only be added to the network, but not removed from the network. In the begining, all ports on all switches are labeled as "floodable". Whenever a new link is added to the network and when the two end switches are already connected in the discovered network, its two end ports will be changed to "unfloodable" to maintain a tree structure.

This spanning tree algorithm can significantly speed up the pingall tests.

Note that whenever a packet is forwarded, the sender address in the Etherent header should be updated to be the MAC address of the sending port. Otherwise the receiver of this forwarded packet will be mislead to think that the sending host is directly connected to it.

4.2 IP Forwarding

With the spanning tree algorithm, the problem of forwarding loops is solved nicely. However, packets are still flooded on the whole network. This can be made more efficient by enabling the switches to forward packets based on target IP.

Directed forwarding is not available in the Ethernet layer, because whenever a switch forwards a packet, the sender MAC address will be changed (otherwise there will be confusion about how hosts are connected to switches as mentioned above). Therefore IP layer is our choice for packet forwarding.

To implement IP forwarding, each switch will maintain a forwarding table that maps target IPs to ports. Whenever an IP or ARP packet is received, the mapping from the sender's IP to receiving port is learned and added to the table. If the target IP address is in the table, the packet will be forwarded accordingly. If the target IP address is not available in the table, an ARP request will be flooded out of "floodable" ports, with source MAC address being the sending port's MAC address and the source IP being the sender's IP address. The IP packet will be stored in a queue, and when the ARP reply comes, it will be notified and sent out.

IP forwarding solved the flooding packet problem, and makes the packet forwarding much more efficient.

Note that although the switches can forward packets based on target IP, they are not complete layer 3 switches. The problem is still that ports do not have IP addresses, and the switches need to forward ARP requests. This makes the switches somewhere between layer 2 and layer 3.

5 Limitations of the Current System

First limitation of the current system is that it cannot distinguish a host with multiple network interfaces and multiple different hosts. This problem comes from the fact that every network interface on a host machine has a unique IP address and MAC address that is different from other interfaces. This makes it almost impossible for the switches to distinguish a host with multiple interfaces from multiple different hosts. So the assumption made in NetVis is that every host has only one network interface.

The second limitation comes from the spanning tree algorithm. It will only work for point-to-point networks. When a switch port is connected to a Ethernet, where more than one other switches are also connected to it, it is possible for the switch ports to disagree on whether it is "floodable" or not. Then new loops will be created. However for Mininet, all connections are point-to-point connections, so the algorithm works fine.