SOFTWARE DEFINED NETWORK BASED FIREWALL TECHNIQUE

Mr. Varun S. Moruse\(^1\), Miss. A. A. Manjrekar\(^2\)

\(^1\)(M. Tech Student, Computer Science and Technology, Department of Technology, Shivaji University, Kolhapur, Maharashtra)

\(^2\)(Assistant Professor, Computer Science and Technology, Department of Technology, Shivaji University, Kolhapur, Maharashtra)

ABSTRACT

The existing networking devices (switches) are complex because they have control plane and data forwarding plane interwined in same devices. This affects the network performance in terms of delayed delivery and repeated functionality. The proposed network software system gives the technique to separate control functionality from the forwarding functionality from such devices which results in efficient network communication. OpenFlow, one of the techniques of Software Defined Network Technology, is a new approach to networking and its key attribute is: separation of data and control planes. With OpenFlow, a researcher or network administrator can introduce a new capability by writing a simple software program that manipulates the logical map of a slice of the network. The rest is taken care by the network operating system. In addition, in proposed system an openflow switch is used in network systems as firewall, which improves the network performance.

**Keywords:** Central Controller, Datapath, Forwarding Element, OpenFlow, Software Defined Network
1. INTRODUCTION

Networks have become a critical component of all infrastructures in society. However, the industry and their designs have not kept pace with ever-growing requirements. Networks are built using switches, routers, and other devices that have become exceedingly complex because they implement an ever-increasing number of distributed protocols standardized by IETF and use closed and proprietary interfaces within. In such an environment, it is too difficult for network operators, third parties including researchers, and even vendors to innovate. A failure in a network could become a failure in business processes and the consequent money loss. Many times researchers need real environments in which they can test experimental network protocols and usually encounter opposition from network administrators who forbid them to test their experiments in production networks. Operators cannot customize and optimize networks for their use cases including the application set that is relevant to their business. Even vendors cannot innovate fast enough to meet their customer requirements.

The net result is that- (a) networks continue to have serious known problems with security, robustness, manageability, mobility and evolvability that have not been successfully addressed; (b) their capital costs have not been reducing fast enough and operational costs have been growing, putting excessive pressures on network operators; and (c) network operators find it difficult to introduce new revenue generating services on their expensive infrastructures.

The software-defined networking notion introduced to solve the above mentioned problems and one of its emerging and powerful implementation is the OpenFlow. It advocates the idea of providing the control and data paths in separate planes. OpenFlow exploits this common set of functions. A network operating system running on this control plane is anticipated to provide necessary measures for scalability and reliability in order to stand against the gigantic traffic pumped by the network. [1]

Firewall is an important security tool in computer networking. By setting up policy rules as per user need into the OpenFlow switch, it can be used as OpenFlow based firewall. This firewall then Allow/Deny packets as per rules enforced in the switch.

2. RELATED WORK

The architecture of today’s Internet is relatively stagnant due to the designing principle of “Keeping the simplicity of network while leaving the complex processing tasks to hosts” [2]. The functions of the application-layer have been greatly enriched because the applications on hosts can be flexibly modified and deployed but the network devices have become like opaque black-boxes because of the lack of openness in the network-layer. Apparently today’s networks have become closed, inflexible and unmodifiable. [3].

Today, there is almost no practical way to experiment with new network protocols in sufficiently realistic settings to gain the confidence needed for their widespread deployment. The result is that most new ideas from the networking research community go untried and untested. Having recognized the problem that the networking community is hard at work developing programmable networks, such as GENI [4] a proposed
nationwide research facility for experimenting with new network architectures and distributed systems.

In the current routers, implementations of the control and forwarding functions are intertwined deeply in many ways. Communication between the control processors and the forwarding line cards is not based on any standard mechanism which makes it impossible to interchange control processors and forwarding elements. The Internet Engineering Task Force is working on standardizing a protocol between control element and the forwarding element in the ForCES working group. However unlike the SoftRouter architecture, the focus is on architecture where the control element is directly connected to the forwarding element. [5]

Internet core protocols were designed in the seventies and after four decades and a huge success, most of that initial design is still in place. New applications bring a new set of requirements that the Internet is not able to satisfy in a proper way. The Internet architecture must be reviewed and several research groups are engaged in this design. Software Defined Networking (SDN), currently materialized in OpenFlow, represents an extraordinary opportunity to rethink computer networks, enabling the design and deployment of a future Internet. [6]

There are lots of similarities between OpenFlow and previous attempts to provide an external interface for a control plane for locally controlled switches and routers. They’re all slightly different. There have also been attempts to separate the data plane from the control plane in the past, and, after all, there are many networks, like telephony networks, that already work that way. The difference here is timeliness. Now days, every network service provider company pressing need to optimize the behaviour of their networks so they can differentiate their solution from others.[7]

A few open software platforms already exist, but do not have the performance or port-density we need. The simplest example is a PC with several network interfaces and an operating system. All well-known operating systems support routing of packets between interfaces, and open-source implementations of routing protocols exist (e.g., as part of the Linux distribution, or from XORP [8]). The problem is performance: A PC can neither support the number of ports needed for a college wiring closet. [9]

Network virtualization has long been a goal of the network research community. With it, multiple isolated logical networks each with potentially different addressing and forwarding mechanisms can share the same physical infrastructure. Typically this is achieved by taking advantage of the flexibility of software or by duplicating components in (often specialized) hardware [10].

3. PROPOSED SYSTEM

The proposed network software system consists of two modules which are as follows:

3.1 Forwarding Element (FE)

3.2 Central Controller (CC)

The architecture of the system is shown in Fig 1.
3.1 Forwarding Element (FE)

The forwarding element is the OpenFlow Switch itself, its main function is to forward the packets to particular destination. The data path portion resides on the FE. It will only forward the packet to the controller. The FE contains a flow-table with the defined flow entries and the actions to be performed.

Network administrator can control packet flow by selecting the routes for the packets and then process it.

The main functions of FE are as follows:
- Address lookup and mapping
- Policy enforcement
- Tunneling

The Forwarding element has following important entities:
- Flow Table
- A Secure Channel
- The OpenFlow Protocol

3.1.1 Flow Table

Flow means combination of rules, actions and statistics. Flow table describes the components of flow table entries and the process by which incoming packets are matched against flow table entries.

Each flow table entry contains:
- Header fields to match against packets
- Counters to update for matching packet
- Actions to apply to matching packets

In Flow Table an action is associated with each flow entry. Action decides how to process the flow. The FE can perform three different actions associated to the flow entries.
- Forward the flow's packets to a given port (or ports).
- Encapsulate and forward the flow's packets to a controller
- Drop the flow's packets.
Usually each flow table entry has following entities as shown in Table 1:

**TABLE 1: A Flow entry consists of Header Fields, Counters and Actions.**

<table>
<thead>
<tr>
<th>Category</th>
<th>Sr. No</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>1</td>
<td>Session Id</td>
<td>Identifier for current session</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>Flags</td>
<td>Indicate behaviour of physical port</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Type</td>
<td>Type of message</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>Cookies</td>
<td>History of current hosts</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>Duration</td>
<td>Amount of time flow has been installed</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>Table Id</td>
<td>Identifier of flow table</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>Priority</td>
<td>Priority of the entry</td>
</tr>
<tr>
<td>Counter</td>
<td>8</td>
<td>No. of packets</td>
<td>Count of packets</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>No. of bytes</td>
<td>Count of bytes</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>Idle timeout</td>
<td>Indicates when entry should be removed due to a lack of activity,</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Hard timeout</td>
<td>Indicates when the entry should be removed, regardless of activity.</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>Protocol</td>
<td>Type of protocol</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>In port</td>
<td>Input port number</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>Vlan id</td>
<td>Virtual lan identifier(if any)</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>Data link src</td>
<td>MAC address of source</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>Data link dst</td>
<td>MAC address of destination</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>Nw src</td>
<td>N/W address of source</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>Nw dst</td>
<td>N/W address of destination</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>Tos</td>
<td>Terms of service</td>
</tr>
<tr>
<td>Action</td>
<td>20</td>
<td>Action</td>
<td>Action taken for the flow</td>
</tr>
</tbody>
</table>

3.1.2 A Secure Channel

The secure channel is OpenFlow channel, which is the interface that connects each FE to CC. Through this interface, the CC configures and manages the FE, receives events from the FE, and sends packets out to the FE. All channel messages between FE and CC must be formatted according to the OpenFlow protocol. The channel is usually encrypted using TLS, but may be run directly over TCP. It also allows commands and packets to be sent between a controller and the FE.

3.1.3 The OpenFlow protocol

It is the protocol which governs rules to communicate between the FE and the CC. It has the header and the data fields as per the type of message.

It supports three message types, CC-to-FE, asynchronous, and symmetric, each with multiple sub-types. CC-to-FE messages are initiated by the controller and used to directly manage or inspect the state of the FE. Asynchronous messages are initiated by the FE and used to update the controller of network events and changes to the FE state. Symmetric messages are initiated by either the FE or the controller and sent without solicitation.
3.2 Central Controller (CC)

The Controller is the network element of above system. It is responsible for managing the Forwarding Elements. It takes high level routing decisions regarding data received from FE, it is also called as server. It makes control function independent of the hardware it controls. It speeds up the forwarding and routing process. The OpenFlow reference distribution includes a controller that acts as an Ethernet learning FE in combination with FEs.

The FEs are connected to the CC before the actual communication takes place between the different hosts. CC decides whether the communication between hosts should be allowed or not. The controller adds and removes flow-entries from the Flow Table.

CC also maintains policy rules. CC stores control information such as addresses, location and policy. Such information is distributed to different FEs as needed. Although CC is conceptually one entity in the given architecture, it can be implemented in a distributed way. CC provides control and configuration function. It also provides security at lower layers, which is more promising than providing security at higher levels.

Following are the messages (shown in Table 2) which are communicated between FE and CC which can be observed and analysed by using packet analyser i.e. Wireshark.

In proposed system, an OpenFlow switch is used as firewall. A firewall keeps track of the packets it has seen in the past. Each packet triggered by host is sent through FE and it is matched against the set of existing firewall rules. The firewall operates in a reactive manner. Firewall rules are sorted by priority at the time they are created (via API).

**TABLE 2: Messages communicated between Central Controller (CC) and Forwarding Element (FE)**

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Message</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hello</td>
<td>CC-&gt;FE</td>
<td>Following the TCP handshake, the CC sends its version number to the FE.</td>
</tr>
<tr>
<td>2</td>
<td>Hello</td>
<td>FE-&gt;CC</td>
<td>The FE replies with its supported version number.</td>
</tr>
<tr>
<td>3</td>
<td>Features Request</td>
<td>CC-&gt;FE</td>
<td>The CC asks FE to see which ports are available.</td>
</tr>
<tr>
<td>4</td>
<td>Set Config</td>
<td>CC-&gt;FE</td>
<td>In this case, the CC asks the FE to send flow expirations.</td>
</tr>
<tr>
<td>5</td>
<td>Features Reply</td>
<td>FE-&gt;CC</td>
<td>The FE replies with a list of ports, port speeds, and supported tables and actions.</td>
</tr>
<tr>
<td>6</td>
<td>Port Status</td>
<td>FE-&gt;CC</td>
<td>Enables the FE to inform that CC of changes to port speeds or connectivity. Ignore this one, it appears to be a bug.</td>
</tr>
<tr>
<td>7</td>
<td>Packet-In</td>
<td>FE-&gt;CC</td>
<td>A packet was received and it didn't match any entry in the FE's flow table, causing the packet to be sent to the CC.</td>
</tr>
<tr>
<td>8</td>
<td>Packet-Out</td>
<td>CC-&gt;FE</td>
<td>CC sends a packet out one or more FE ports.</td>
</tr>
<tr>
<td>9</td>
<td>Flow-Mod</td>
<td>CC-&gt;FE</td>
<td>Instructs a FE to add a particular flow to its flow table.</td>
</tr>
<tr>
<td>10</td>
<td>Flow-Expired</td>
<td>FE-&gt;CC</td>
<td>A flow timed out after a period of inactivity.</td>
</tr>
</tbody>
</table>
Each incoming packet will be compared against the list from the highest priority until either a match is found or the list is exhausted. If a match is found, the rule's action (ALLOW or DENY) is stored in a file. This information is considered for rest of the packet processing. The above decision eventually reaches packet forwarding. Forwarding pushes a regular forwarding flow entry if the decision is ALLOW, or a drop flow entry if the decision is DENY. In either case, the flow entry must be sent to the FE and matched with existing rules. In advance, the first packet of each connection will be handled by the controller, but all other connection packets will be handled by the FE without contacting CC every time.

As shown in Fig 2, two hosts are connected via FE using datapath. FE is controlled by CC. Suppose the communication policy is that all incoming packets from host A to host B should be allowed, but outgoing packets from host B to host A are not allowed. Such rule is installed at the FE, through which all the packets are forwarded. This scenario can be extended for organizational network where OpenFlow switch based firewall will play crucial role. As OpenFlow is containing API’s, the policy rules can be changed as per need.

**Fig 2:** Block diagram of system- FE behaves as firewall

Following is the list of messages which are communicated in Fig 2;

1) Request from Host A to FE for Host B.
2) FE communicates with CC and CC checks policy.
3) CC responses with result.
4) Rule installation at FE: data from Host A can be sent to Host B.
5) FE responses to Host A.
6) Data is forwarded from host A to FE.
7) Data is forwarded by FE to host B.
8) Data is forwarded from host B to FE but no data is forwarded by FE to host A, because no such rule present.
4. RESULTS

As shown in above Fig 3, the messages (mentioned in Table 2) are transmitted between host 1 and 2, when host 1 ping for the host 2. The ARP request is broadcasted through FE. FE then asks to CC, CC replies with MAC address of host 2 and the policy rule (whether the host is allowed or not). This rule is installed in the FE; the reply is forwarded to host 1. Now the next PING request from host 1 is directly sent to the host 2, this time without contacting CC. Thus the policy rule of communication between host 1 and 2 is set by CC.

5. CONCLUSION

The system decouples control plane from data forwarding plane. The CC controls the packet forwarding through the FEs by setting policy rules. Users can specify and manage their individual security and QoS policy settings the same way as they manage the conventional on-site networks. In above system it has shown that FE can be used as firewall with the help of CC by setting up the firewall rules. The architecture takes advantage of network virtualization and centralized control, using enhanced FE switches. It uses existing infrastructure which avoids new investment on network devices. The system gives an idea for real life experiments in network without disturbing existing infrastructure. The system makes innovation easier and makes deployed networks not just configurable but also programmable.
REFERENCES