Hardwarepraktikum

Andre Löffler, Fabian Helmschrott, Nils Wisiol

20. Mai 2012

Aufgabe B

Unser Testsetup für diese Aufgabe war wie folgt: Zwei Maschinen wurden verwendet. Maschine A ist ein Linux-Host, Maschine B ist ein Windows-Host.

Was ist der Unterschied zwischen Packet- und Flow-Switching?

Beim Packet-Switching werden die zu übertragenden Daten unabhängig von Struktur in Blocks verpackt. Beim Flow-Switching werden die Datenströme analysiert und je nach Bedarf die Ressourcen frei gemacht oder umverteilt.

Unterstützt OpenFlow das Packet-Switching?

OpenFlow ist in der Lage verschiedene Formen des Switching zu unterstützen, darunter auch Packet- und Circuit-Switching.

Bekommt h2 eine Antwort? Warum?

Es kommt keine Antwort, da noch keine Flows bekannt sind. Dieses Ergebnis konnten wir mit beiden Maschinen realisieren.

Warum bekommt h2 nun eine Antwort?

h2 bekommt nun eine Antwort, da die Flowregeln erstellt wurden und der Switch nun weiß, wohin gesendet werden muss. Die Flowtabelle entspricht den Erwartungen, es wurden zwei entsprechende Flows hinzugefügt – in jede Richtung einen. Jedoch wurden für 3 Pings 7 Pakete verschickt.

Welche Pakete sind zu sehen? Welche Bedeutung haben sie?

[**Packet In**] Ein Paket von 2 nach 3 vom Typ OFP+ICMP geht ein.

[**Packet Out**] Der Controller weißt den Switch an, das gerade empfangene Paket an allen Ports weiterzuversenden.

[**Paket In**] Ein ICMP-Paket von 3 nach 2 geht ein.

[**Flow Mod**] Der Controller weißt den Switch an, einen neue Flow anzulegen.

Controller und Switch begrüßen sich zunächst gegenseitig mit "Hello". Danach wird ausgehandelt, welche Features zur Verfügung stehen und welche Konfiguration verwendet wird. Anschließend wird die Verbindung mit "echo" offen gehalten.

Ist ein Unterschied sichtbar?

	kernel space	user space
Maschine A	3660 Mb/s	389Mb/s
Maschine B	244 Mb/s	19,9Mb/s

Hub-Benchmark

Maschine A wurde auf 8 Mb/s und Maschine B auf 1 Mb/s verlangsamt. Ist das nox-Terminal nicht minimiert sondern sichtbar, so wird durch das Neuzeuchnen der Anzeige im Terminal nochmals etwa 40% der Performance eingebüßt.

Learn and Forward

```
def learn_and_forward(self, dpid, inport, packet, buf, bufid):
        """Learn MAC src port mapping, then flood or send unicast."""

        # Learn the port for the source MAC
        self.mac_to_port[mac_to_str(packet.src)] = inport
        if (mac_to_str(packet.dst) in self.mac_to_port.keys()):
            # Send unicast packet to known output port
            self.send_openflow(dpid, bufid, buf,
                    self.mac_to_port[mac_to_str(packet.dst)], inport)
            # Later, only after learning controller works:
            # push down flow entry and remove the send_openflow command above.
            attrs = {}
            attrs[core.IN_PORT] = inport
            attrs[core.DL_DST] = packet.dst
            action = [[openflow.OFPAT_OUTPUT, [0,
                    self.mac_to_port[mac_to_str(packet.dst)] ]]]
            self.install_datapath_flow(dpid, attrs, 15, 90, action)
        else:
            # flood packet out everything but the input port
            self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)
```

Maschine A erreichte mit dieser Implementierung im kernel space Modus eine Performanz von 3580 Mb/s.

Für mehrere Switches

```
    def learn_and_forward(self, dpid, inport, packet, buf, bufid):
        """Learn MAC src port mapping, then flood or send unicast."""

        # Learn the port for the source MAC
```

```
if (not (dpid in self.dpid_mac_to_port.keys())):
    self.dpid_mac_to_port[dpid] = {}
self.dpid_mac_to_port[dpid][mac_to_str(packet.src)] = inport
if (mac_to_str(packet.dst) in self.dpid_mac_to_port[dpid].keys()):
    # Send unicast packet to known output port
    self.send_openflow(dpid, bufid, buf,
         self.dpid_mac_to_port[dpid][mac_to_str(packet.dst)], inport)
    # Later, only after learning controller works:
    # push down flow entry and remove the send_openflow command above.
    attrs = {}
    attrs[core.IN_PORT] = inport
    attrs[core.DL_DST] = packet.dst
    action = [[openflow.OFPAT_OUTPUT, [0,
              self.dpid_mac_to_port[dpid][mac_to_str(packet.dst)] ]]]
    self.install_datapath_flow(dpid, attrs, 15, 90, action)
else:
    # flood packet out everything but the input port
    self.send_openflow(dpid, bufid, buf, openflow.OFPP_FLOOD, inport)
```

Diese Implementierung läuft auf Maschine A mit 3000 Mb/s

CBench

Unser eigener python-Controller lieferte bei cbench im average 4800 responses/s. Die Referenzimplementierung lieferte im average 80000 responses/s. Die nox-C++ Implementierung lieferte im parallelen Betrieb im average 30000 responses/s, die nicht-parallele nur 2000 responses/s. Ein Router verteilt eingehende Pakete anhand der Ziel-IP-Adresse, während für einen Switch die MAC-Adresse entscheidend ist.