software defined networking

CS 6410

Hakim Weatherspoon

Slides by Eric Campbell and Rolph Recto
software defined networking
software defined networking
(OpenFlow, originally)
TR10: Software-Defined Networking

Nick McKeown believes that remotely controlling network hardware with software can bring the Internet up to speed.

4 comments

KATE GREENE
Tuesday, February 24, 2009
“Stanford computer scientist Nick McKeown and colleagues developed a standard called OpenFlow that essentially opens up the Internet to researchers, allowing them to define data flows using software—a sort of ‘software-defined networking.’ Installing a small piece of OpenFlow firmware (software embedded in hardware) gives engineers access to flow tables, rules that tell switches and routers how to direct network traffic.”
software defined radio
software defined networking
software defined storage
software defined data center
software defined everything
software defined anything
software defined storage for dummies
software defined architecture
software defined definition
software defined storage solutions
control plane

routing

isolation

traffic engineering

routing: determine route taken by packets from source to dest

data plane

packet forwarding

packet scheduling

forwarding: move packets from router’s input to appropriate router output
traditional networking
**SDN**

*Routing*: determine route taken by packets from source to destination.

*Forwarding*: move packets from router’s input to appropriate router output.
software defined networking
(programmable)
Active Networks

Separating the Data and Control Planes

OpenFlow

SDN Today
use pulls

technology pushes

network ossification

desire for unified middlebox interface
use pulls

technology pushes

lower compute costs

advances in programming languages

DARPA Active Networks
programmable switches
with `sniff.java` installed, switch maintains table of packet counts by source IP.
<table>
<thead>
<tr>
<th>srcIP</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.1</td>
<td>1</td>
</tr>
</tbody>
</table>
packet contains instructions to push switch info at every hop
whither active networks?
whither active networks?

performance and security concerns

no “killer app”

no practical deployment plan

“The misconception that packets would necessarily carry Java code written by end users made it possible to dismiss active network research as too far removed from real networks and inherently unsafe.”

“The Road to SDN,” Feamster et al 2014
performance and security concerns

no “killer app”

no practical deployment plan

"The misconception that packets would necessarily carry Java code written by end users made it possible to dismiss active network research as too far removed from real networks and inherently unsafe."

“The Road to SDN,” Feamster et al 2014
- Tennenhouse & Wetherall
- Smart Packets
- NetScript
- ANTS
- Tiny Packet Programs
- SIGCOMM 2014
- In-Band Network Telemetry
- Whippersnapper P4FPGA
Active Networks

Separating the Data and Control Planes

OpenFlow

SDN Today
Use pulls

Technology pushes

- burgeoning network speeds
- insufficient network reliability
- specialized services (VPNs)
open interface between control and data planes
logically centralized control
Open Shortest Path First (OSPF)
OSPF

Diagram showing network topology with labels for latency times: 1μs, 2μs, 3μs, 5μs, and 10μs.
SoftRouter

Network Entity (NE)  Control Element (CE)

Forwarding Entity (FE)
Shortest Path Routing
whither SoftRouter (and others)?

vendors didn’t adopt ForCES (and others)

not general enough

no practical deployment plan
Active Networks

Separating the Data and Control Planes

OpenFlow

SDN Today
OpenFlow: enabling innovation in campus networks

Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, Jonathan Turner

SIGCOMM 2008
use pulls

technology pushes

networking research

market factors

datacenter networks
use pulls

technology pushes

backwards compatible
general packet processing
(more fields to match on)
learning switch
controller maintains hash table of MAC to port number mappings
src: spongebob
dst: patrick
msg: I'M READY
packet_in

MAC port
MAC
spongebob
port

packet_out

1 2 3
MAC port

spongebob 1

...
install rule: forward packets for spongebob to port 1
**src:** patrick  
**dst:** spongebob  
**msg:** i'm patrick
no flooding required!
firewall
install rule: drop packets destined for squidward
src: spongebob
dst: squidward
msg: hello
Active Networks

Separating the Data and Control Planes

OpenFlow/SDN

SDN Today
Frenetic
Abstractions for Network Update
NetKAT
Propane
NetKAT

Forwarding Policy (F)

if sw==A then
   (if pt==1 then pt=2 elif pt==2 then pt=1)
elif sw==B then
   (if pt==1 then p=2 elif pt==2 then pt=1)
else drop

if sw==A and pt==2 then (sw=B; pt=1)
elif sw==B and pt==1 then (sw=A; pt=2)
else drop

Network Behavior
run( F; T)

Topology (T)
NetKAT

Forwarding Policy (F)

if sw==A then
    (if pt==1 then pt=2 elif pt==2 then pt=1)
elif sw==B then
    (if pt==1 then p=2 elif pt=2 then pt=1)
else drop

if sw==A and pt==2 then (sw=B; pt=1)
elif sw==B and pt==1 then (sw=A; pt=2)
else drop

Topology (T)

Intuition: These are functions from packets to sets of packets

Network Behavior

run( F; T)
Want to show:
SSH packets sent from H1 get to H2

Invariant true when P1 is equivalent to P2.

P1

if typ==SSH and @h1
then run(F; T);
eventually(@h2)

let @1 be sw==A and pt==1
let @2 be sw==B and pt==2

P2

if typ==SSH and @l
then run(F; T)
Want to show:
SSH packets sent from H1 get to H2

In invariant true when P1 is equivalent to P2.

Invariant true when P1 is equivalent to P2.

if typ==SSH and @h1
then run(F; T);
finally(@h2)

if typ==SSH and @h1
then run(F; T)
Want to show:
SSH packets sent from \textbf{H1} get to \textbf{H2}

\begin{itemize}
\item \textbf{P1}
  \begin{align*}
  \text{if typ==SSH and @h1} \\
  \text{then run( F; T) ;} \\
  \text{eventually( @h2) }
  \end{align*}
\end{itemize}

\begin{itemize}
\item \textbf{P2}
  \begin{align*}
  \text{if typ==SSH and @h1} \\
  \text{then run( F; T)}
  \end{align*}
\end{itemize}

Invariant true when \textbf{P1} is equivalent to \textbf{P2}.

“run” the network

H1 \quad 1 \quad \text{A} \quad 2 \quad \text{B} \quad 1 \quad 2 \quad \text{H2}
Want to show:
SSH packets sent from H1 get to H2

P1
if typ==SSH and @h1
then run(F; T);
eventually(@h2)

Invariant true when P1 is equivalent to P2.

P2
if typ==SSH and @l
then run(F; T)

packets received by H2
if sw == A then:
    if port == 1 then port := 2
    elif port == 2 then port := 3
    else drop
elif sw == B then:
    if port==1 drop
    else port := 1
elif sw == C
    if port == 1 then port := 3
    elif port == 3 then port := 2
    else drop;
if sw == A then:
    if port == 2 then alert_ctrl
    else port := 3

if sw == B then:
    if port == 1 then drop
    elif port == 2 then port := 3
    elif port == 3 then port := 2

if sw == C
    if port == 1 then alert_ctrl
    elif port == 2 then port := 1
    elif port == 3 then port := 2
whither SDN?

Why VMware is spending $1B-plus to buy Nicira
VMware makes strong software-defined networking play by purchasing Nicira
whither SDN?

Silicon Valley Makes a Rare Bet on Silicon

By Ian King
September 13, 2017, 5:00 AM EDT Corrected September 15, 2017, 2:22 PM EDT

→ Barefoot Networks emerges as leading startup chipmaker
→ Company has big backers in Goldman Sachs, Google, Alibaba
questions?
thanks!