Software Defined Transport

Chi-Yao Hong
Matthew Caesar
P. Brighten Godfrey

University of Illinois at Urbana-Champaign
Three key problems of today’s transport
Three key problems of today’s transport

1. Not flexible
   - Monolithic, run predefined algorithms
Three key problems of today’s transport

1. Not flexible
   - Monolithic, run predefined algorithms

2. Hard to reason about
   - Mismatch between (distributed) protocol properties and transport policies
Three key problems of today’s transport

1. **Not flexible**
   - Monolithic, run predefined algorithms

2. **Hard to reason about**
   - Mismatch between (distributed) protocol properties and transport policies

3. **Hard to deploy**
   - New transport protocols require custom changes at switches, end-hosts, or both
Programmable transport architecture — A killer app for datacenter network optimization
Software Defined Transport (SDT)
SDT: Architecture
SDT: Architecture

Software Defined Networking
Software Defined Networking

thin, open interface to data plane
Software Defined Networking

thin, open interface to data plane

logically centralized controller
Software Defined Networking

low-level access, but what’s the northbound API?
Software Defined Networking

low-level access, but what’s the northbound API?
SDT: Architecture

SDN controller
SDT: Architecture

SDN controller

thin interface to end hosts
SDT: Architecture

collect flow demand & allocate flow rates

thin interface to end hosts
SDT: Architecture

Host controller

SDN controller
SDT: Architecture

Transport optimizer allocating flow rates

Host controller | SDN controller
SDT: Architecture

Transport optimizer

allocating flow rates

flow demand

Host controller

SDN controller
SDT: Architecture

Transport optimizer

Host controller

SDN controller

network operator

transport policies

flow demand

topology

allocating flow rates
Scalability

How far can we push this architecture towards fine-grained flow-rate control in real-time?
Scaling transport rate control
long flows

short flows
Scaling transport rate control

long flows — handled centrally
short flows — handled locally
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

SDT controller
Scaling transport rate control

**long flows** — handled centrally

**short flows** — handled locally

SDT controller

TCP
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

SDT controller

TCP

high priority
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

SDT controller

Sent > X bytes

high priority
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

flow demand

TCP

Sent > X bytes

SDT controller

high priority
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

flow demand
flow rate allocation

SDT controller

TCP

Sent > X bytes

high priority
Scaling transport rate control

**long flows** — handled centrally

short flows — handled locally

Sent > X bytes

*SDT rate*

flow demand

flow rate allocation

*SDT controller*

*high priority*
Scaling transport rate control

long flows — handled centrally
short flows — handled locally

flow demand
flow rate allocation

Sent > X bytes

low priority

SDT rate

SDT controller
Scaling transport rate control

A fast parallel flow-rate computation algorithm

SDT controller

SDT rate

low priority
Fast parallel flow-rate computation
Controller computes flow-rate via fluid-level simulation
Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
- Multi-threaded program by mapping links to threads
Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
- Multi-threaded program by mapping links to threads
Fast parallel flow-rate computation

Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
- Multi-threaded program by mapping links to threads
Fast parallel flow-rate computation

Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
- Multi-threaded program by mapping links to threads
Controller computes flow-rate via fluid-level simulation

- Link computes output flow-rates based on input flow-rates
- Multi-threaded program by mapping links to threads

Optimization for speed

- Per-link dirty bit to avoid unnecessary re-computation
- Token-based source feedback to avoid oscillation
Evaluation platform

Testbed
• 13 OpenFlow switches (Pronto 3290s)
• 4 servers, 112 ports
• Floodlight OpenFlow Controller

Prototype implementation
• 112 Linux VMs
• tc for rate-limiting
• iptables for pkt marking
• C++11 Boost library
Controller scalability

Control interval

Number of servers

[log scale]
Controller scalability

![Graph showing controller scalability]

Control interval [log scale]

- 100 s
- 10 s
- 1 s
- 100 ms
- 10 ms
- 1 ms

Number of servers

- 100
- 1K
- 10K
- 100K

Scale to thousands of servers with sub-second control interval
Controller scalability

Scale to thousands of servers with sub-second control interval

≈ 4x speedup with 4 threads
SDT Demo in 1 minute

```
>>> >>>
```

Finishing flows faster with SDT

Root switch

ToR switches

Servers

```
cyhong@cStandard:~$ cd /ocean/hostrmgr

```

```
cyhong@cStandard:~/ocean/hostrmgr$ python

```

```
cyhong@cStandard:~/ocean/hostrmgr$ ./hostrmgr.py
```

Loading host manager.

Stat listener is up. Bringing the host agents up...

Host agents are all active. Start the host manager shell now.
SDT Demo in 1 minute

Finishing flows faster with SDT

Loading host manager.
Stat listener is up. Bringing the host agents up...
Host agents are all active. Start the host manager shell now.

>>> 
>>> 
>>>