ENABLING TCP PACING USING PROGRAMMABLE DATA PLANE SWITCHES

E. Kfoury[1], J. Crichigno[1], E. Bou-Harb[2], D. Khoury[3], G. Srivastava[4]
University of South Carolina, FL, USA
Florida Atlantic University, FL, USA
American University of Science and Technology, Lebanon
Brandon University, Canada

IEEE TSP'19
Budapest, Hungary
July 1, 2019

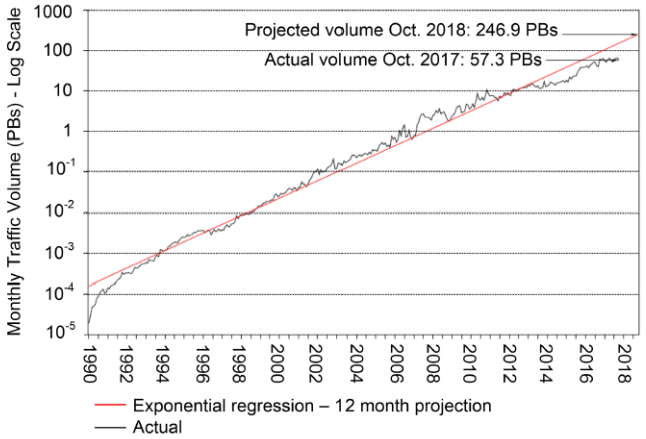
Agenda

- Motivation, Science DMZs
- Pacing
- Programmable switches
- Proposed scheme
- Preliminary results
- Ongoing work

Motivation for a High-Speed Science Architecture

- Science and engineering applications are now generating data at an unprecedented rate
- Instruments produce hundreds of terabytes in short periods of time
- Data must be typically transferred across **high-throughput high-latency** Wide Area Networks (WANs)

Applications

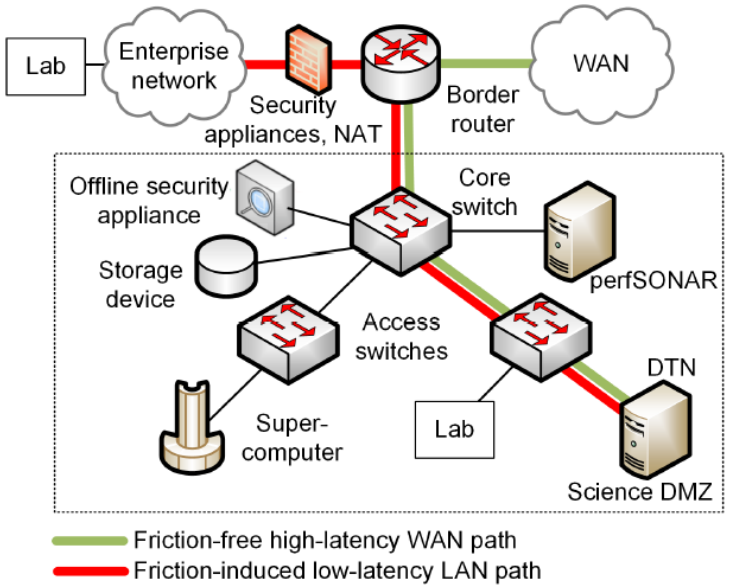


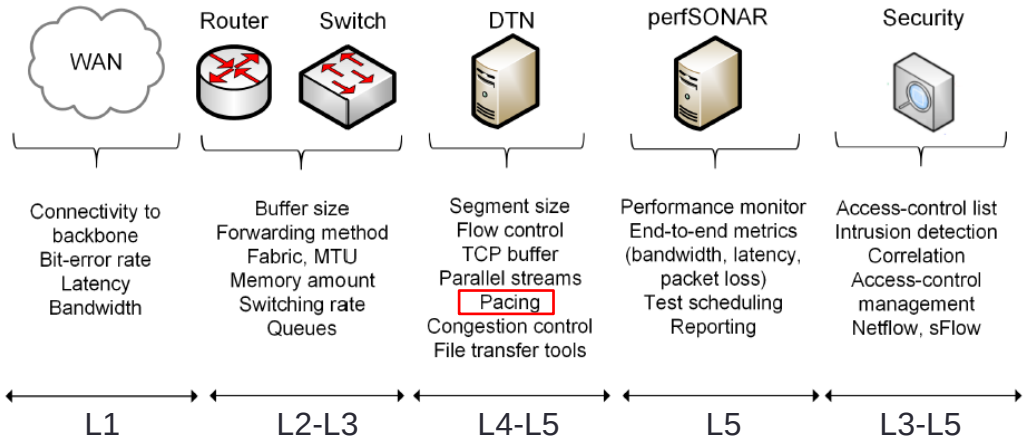
ESnet traffic

The Energy Science Network (ESnet) is the backbone connecting U.S. national laboratories and research centers

Motivation for a High-Speed Science Architecture

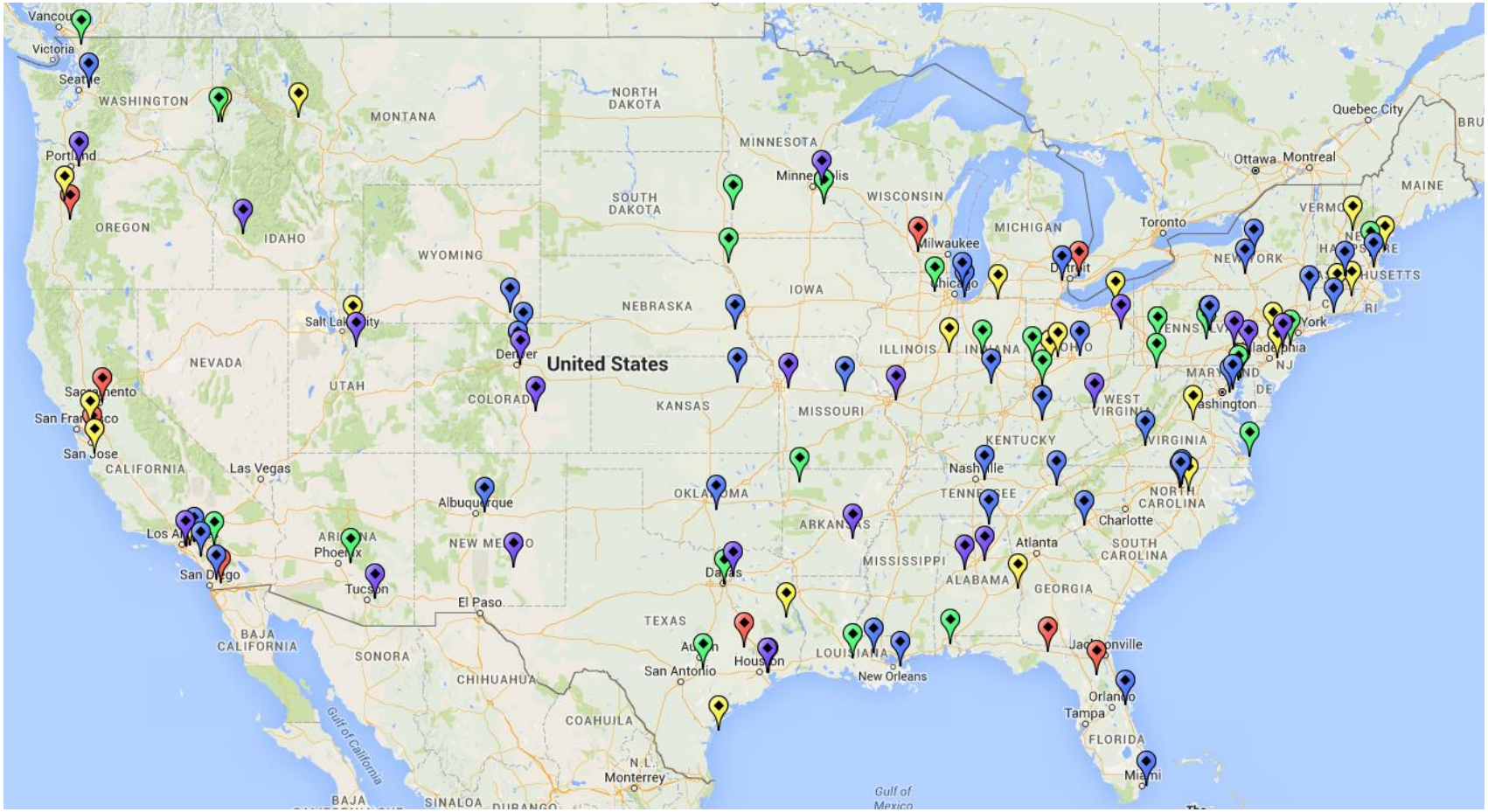
- The Science DMZ is a network designed for big science data
- Main elements
  - High throughput, friction free WAN paths
  - Data Transfer Nodes (DTNs)
  - End-to-end monitoring = perfSONAR
  - Security tailored for high speeds





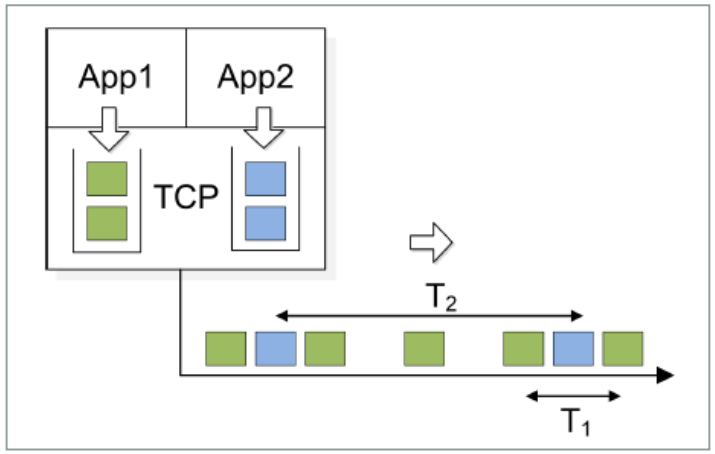
Motivation for a High-Speed Science Architecture

- Science DMZ deployments, U.S.



Pacing

- Pacing is a technique by which a transmitter evenly spaces or paces packets at a pre-configured rate
- If the network bottleneck is known, end devices can be set to transfer at a pacing rate rather than ‘discovering’ the rate
- Pacing also helps to mitigate packet bursts

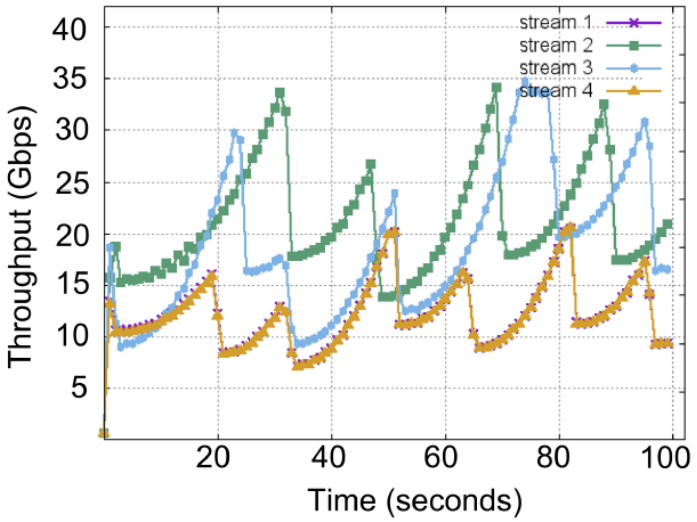


Pacing

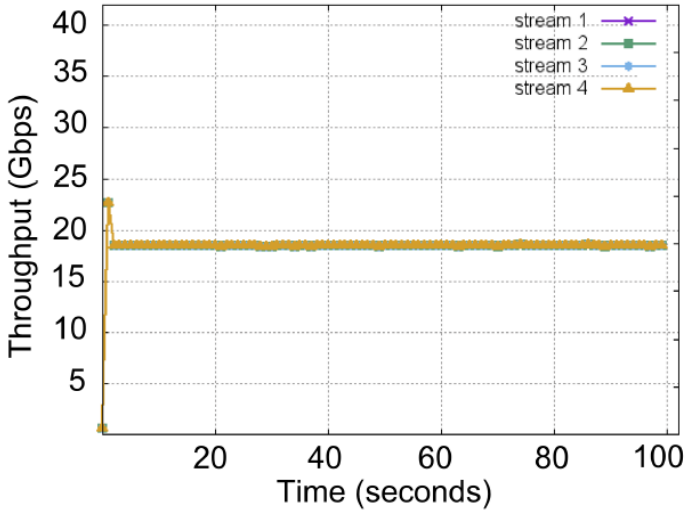
Consider tests over ESnet backbone[1]

Four flows on a 100 Gbps network, 92 msec RTT

- "Consistent loss on the network with four streams, no pacing…"
- "Pacing to match bottleneck link works better yet…"
- ESnet approach requires the network operator to statically set the pacing rate, based on the number of big flows



(a)



(b)

1. https://meetings.internet2.edu/media/medialibrary/2016/10/24/20160927-tierney-improving-performance-40G-100G-data-transfer-nodes.pdf

Programmable Switches

- P4 is a programming language for switches
- P4 permits operators/developers to program the data plane

  Add proprietary features: invent, *develop custom protocols*
- USC partnered with Barefoot Networks to use Tofino's chip to develop custom protocols

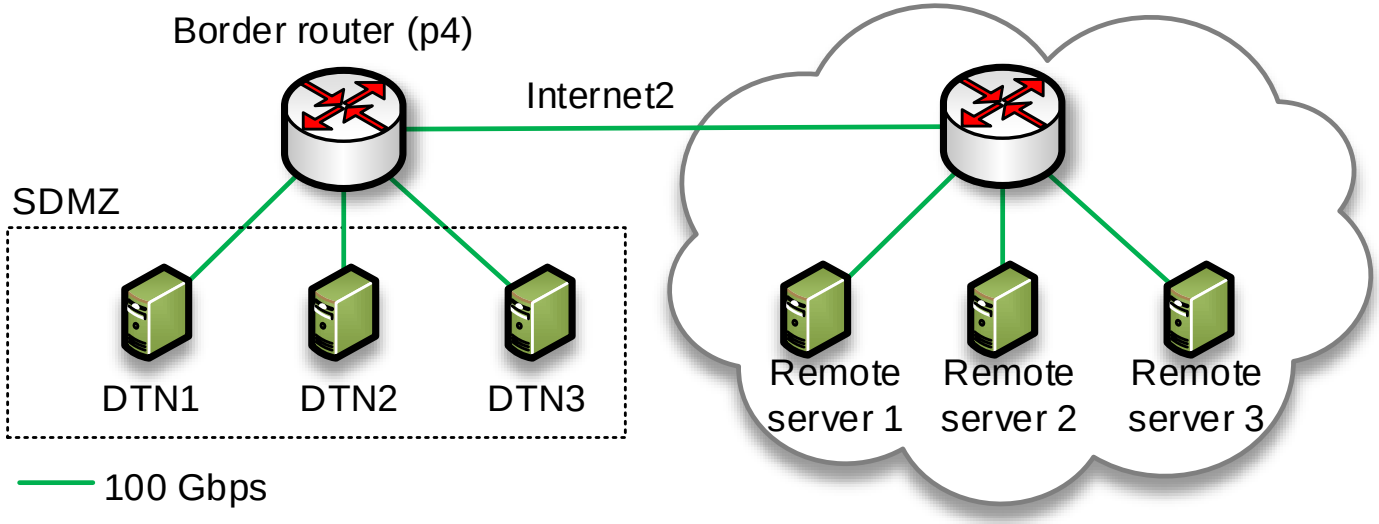
```
136  /*************************************************************************
137  ***********************  P A R S E R  ***********************************
138  *************************************************************************/
139
140      state parse_ethernet {
141          packet.extract(hdr.ethernet);
142          transition select(hdr.ethernet.etherType) {
143              TYPE_IPV4: parse_ipv4;
144              default: accept;
145          }
146      }
147
148      state parse_ipv4 {
149          packet.extract(hdr.ipv4);
150          verify(hdr.ipv4.ihl >= 5, error.IPHeaderTooShort);
151          transition select(hdr.ipv4.ihl) {
152              5             : accept;
153              default       : parse_ipv4_option;
154          }
155      }
```

P4 code

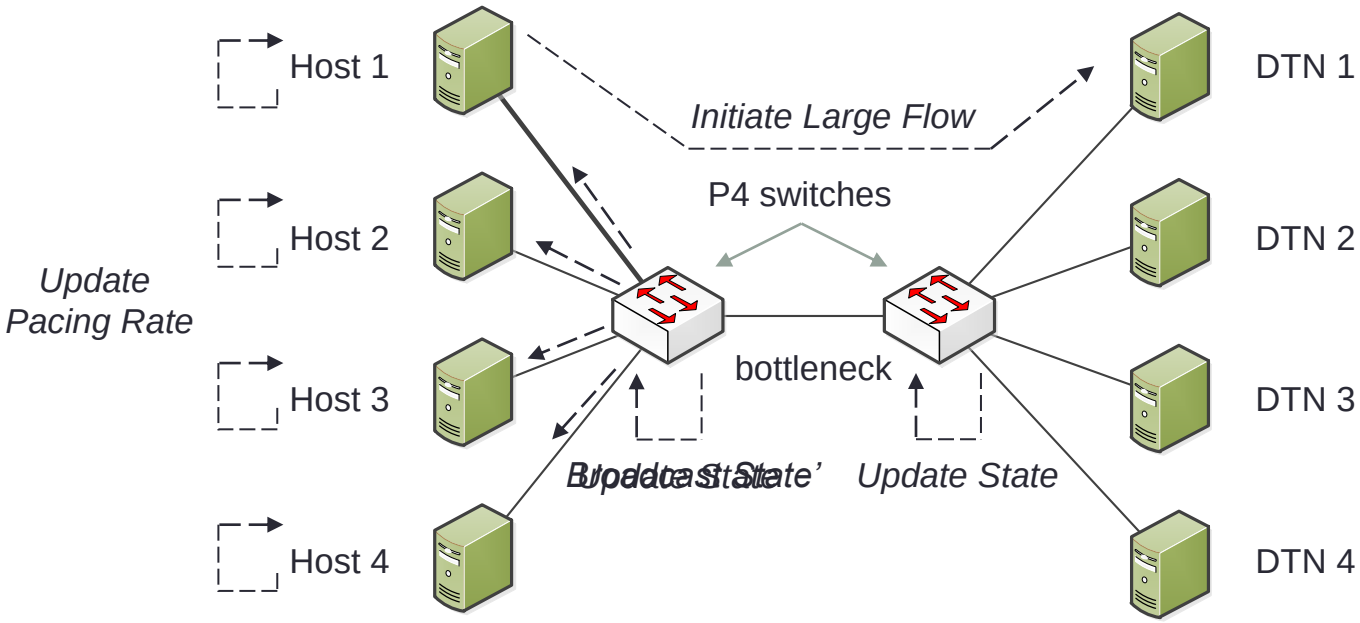
Barefoot's Tofino (2016)

Pacing using Programmable Switches

- What if the rate at a sender node is adjusted based on feedback provided by a P4 switch?
- Feedback includes number of large flows and more

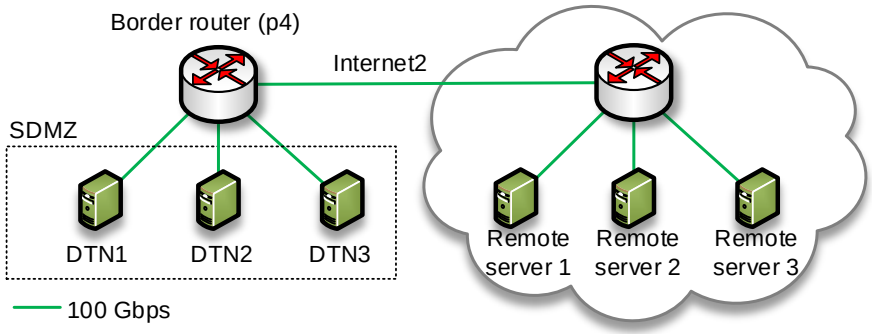


Pacing using Programmable Switches

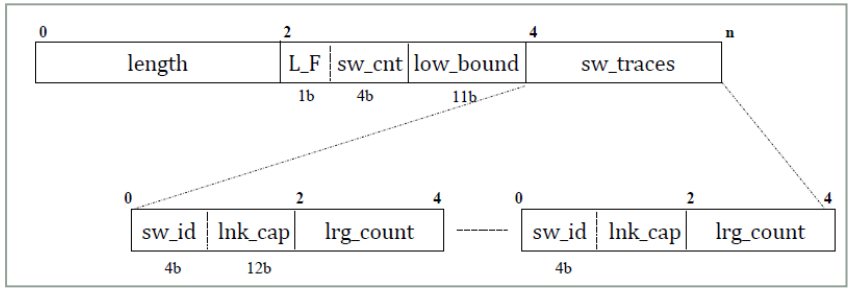


Pacing using Programmable Switches

- Switches store network's state (number of large flows)
- To initiate a large flow, a DTN inserts a custom header during the TCP 3-way handshake, using the IP options field
- Switches parse custom header, update number of large flows
- Number of large flows is returned in the SYN-ACK message, and sent to all DTNs. DTNs update their *pacing* rate



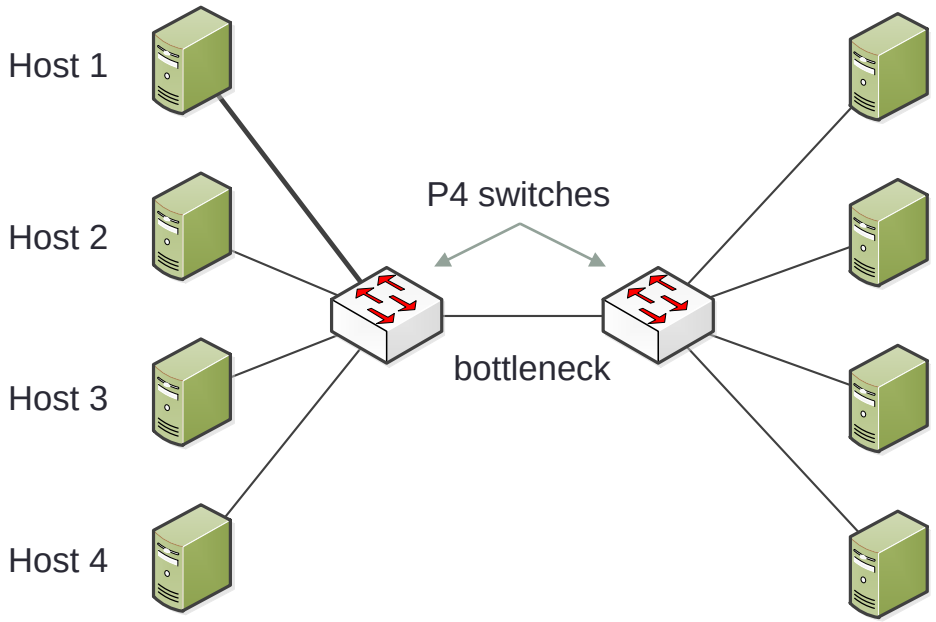
Sample topology



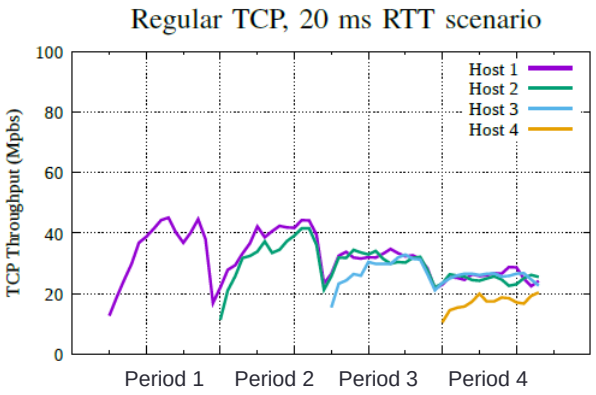
Custom protocol built using IP options field

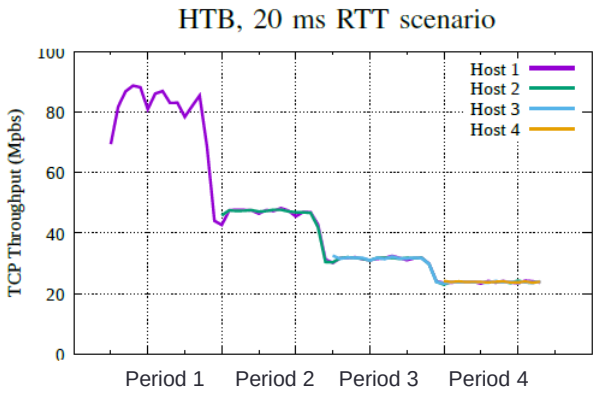
Emulation Results

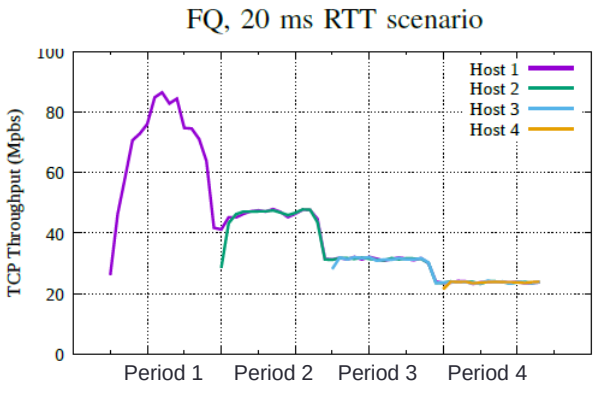
- The custom protocol was implemented in Mininet
- The P4 switch is the BMv2 from P4.org
- Four hosts (DTNs) generating flows; 100 Mbps, 20ms RTT
- Hosts adjusted their pacing rate using two pacing disciplines
  - Fair Queue (FQ)
  - Hierarchical Token Bucket (HTB)



Emulation Results







Throughput

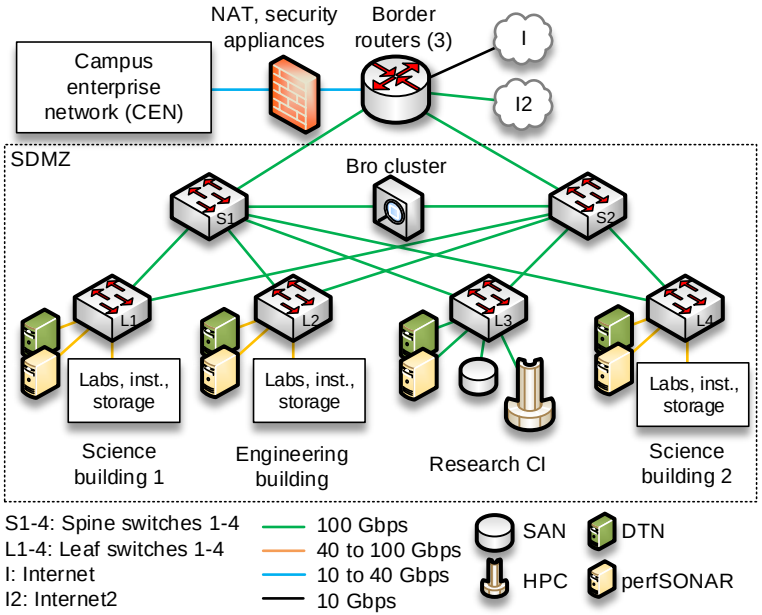
	Regular TCP					HTB					FQ				
Period	$\sum T_i$	T_1	T_2	T_3	T_4	$\sum T_i$	T_1	T_2	T_3	T_4	$\sum T_i$	T_1	T_2	T_3	T_4
P_1 (01-15 sec)	**33.62**	33.62	N/A	N/A	N/A	**81.25**	81.25	N/A	N/A	N/A	**66.59**	66.59	N/A	N/A	N/A
P_2 (16-30 sec)	**67.27**	36.06	31.21	N/A	N/A	**93.1**	46.40	46.70	N/A	N/A	**89.91**	45.85	44.06	N/A	N/A
P_3 (31-45 sec)	**88.83**	31.27	30.61	26.95	N/A	**94.42**	31.40	31.37	31.65	N/A	**93.72**	31.40	31.36	30.96	N/A
P_4 (46-60 sec)	**91.86**	25.32	24.63	25.32	16.59	**95.12**	23.78	23.75	23.73	23.86	**94.52**	23.71	23.71	23.67	23.43

Coefficient of variation and Jain's fairness

	Regular TCP					HTB					FQ				
Period	**F**	**CV_1**	**CV_2**	**CV_3**	**CV_4**	**F**	**CV_1**	**CV_2**	**CV_3**	**CV_4**	**F**	**CV_1**	**CV_2**	**CV_3**	**CV_4**
P_1 (01-15 sec)	1.00	32.32	N/A	N/A	N/A	1.0000	8.188	N/A	N/A	N/A	1.0000	28.427	N/A	N/A	N/A
P_2 (16-30 sec)	.994	22.63	30.08	N/A	N/A	.99998	3.773	2.998	N/A	N/A	.99960	4.351	14.142	N/A	N/A
P_3 (31-45 sec)	.994	9.349	10.90	19.69	N/A	.99998	2.065	2.081	1.985	N/A	.99960	1.618	1.317	3.879	N/A
P_4 (46-60 sec)	.974	7.806	5.260	6.447	17.27	.99999	1.168	1.138	.755	.684	.99997	1.022	1.020	.996	3.336

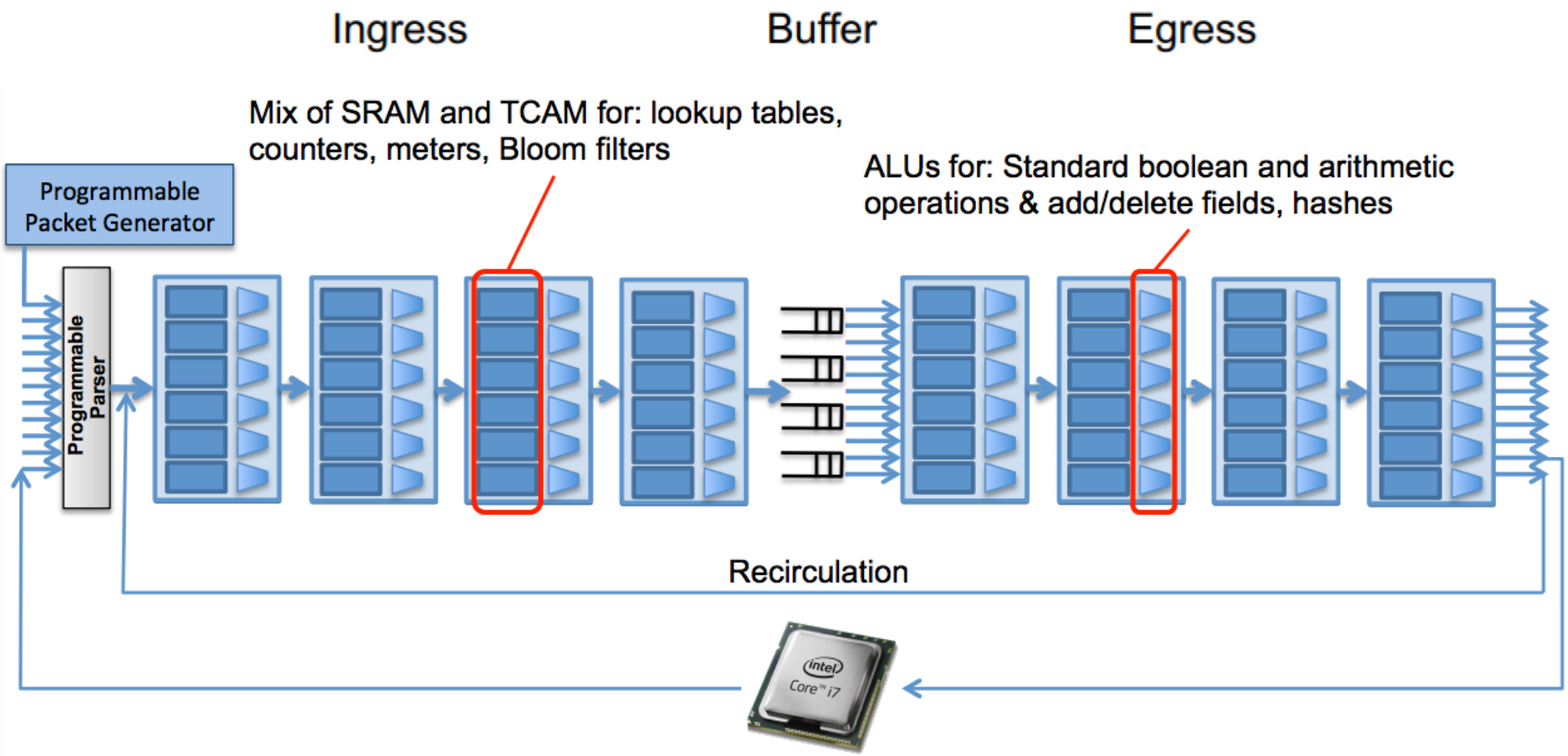
Ongoing Work

- Implement proposed protocol using real P4 switches
- Extend the protocol to support general cases
- Extend the sharing bandwidth scheme for scenarios where an uneven allocation is desirable (priorities)
- Use proposed protocol in USC's production network



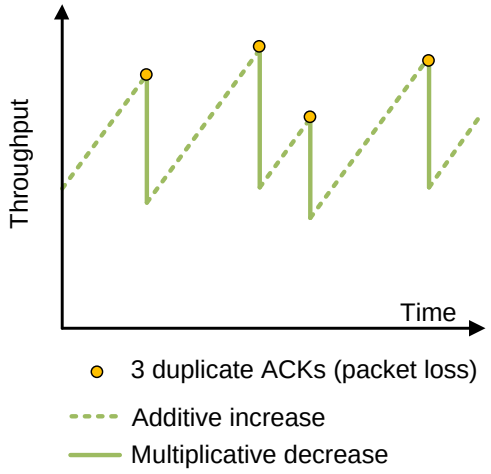
Additional Slides

PISA Architecture



Pacing

- Packet loss is expensive in high-throughput high-latency networks



Sawtooth behavior

Internet2

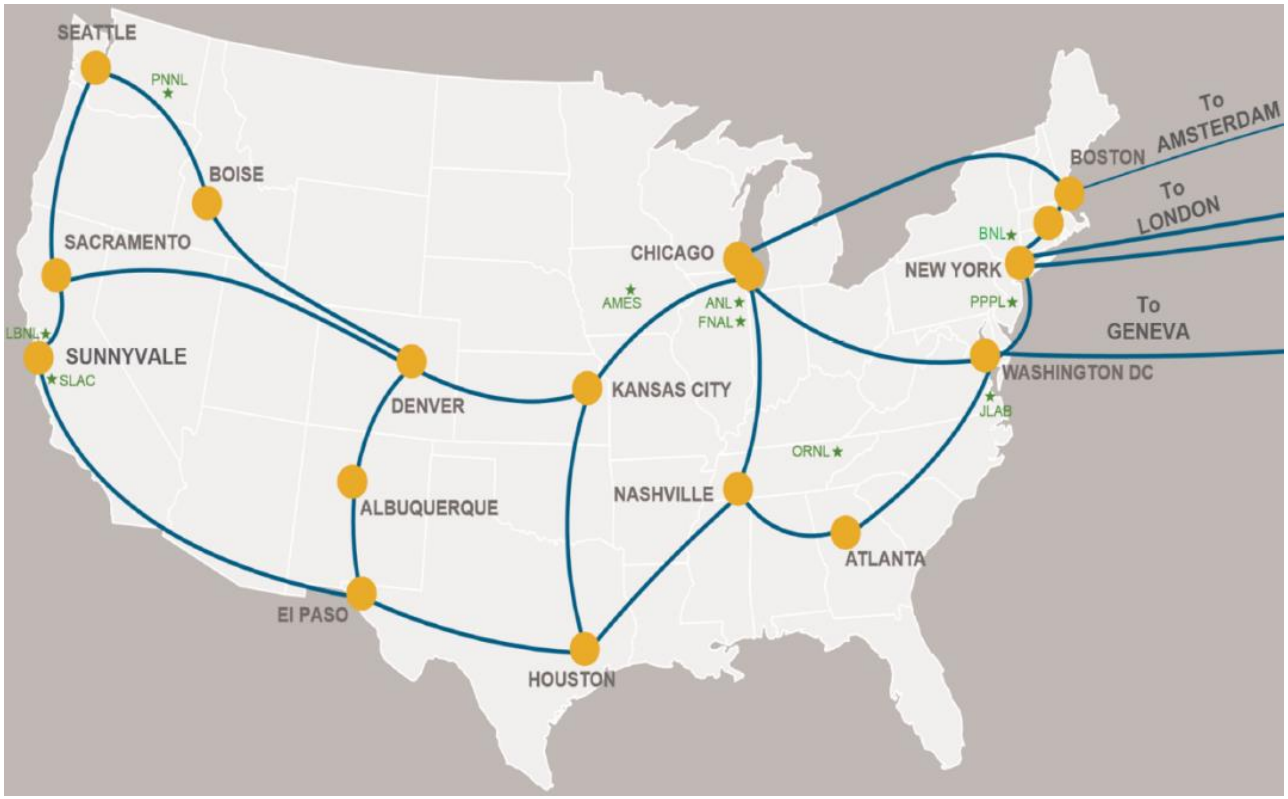
$$\text{TCP throughput} = \frac{c \cdot MSS}{RTT \cdot \sqrt{p}}$$

MSS: maximum segment size
RTT: round-trip time
p: loss rate
c: constant

(c) Average throughput

M. Mathis, J. Semke, J. Mahdavi, T. Ott, "The macroscopic behavior of the tcp congestion avoidance algorithm," *ACM Computer Communication Review*, vol. 27, no 3, pp. 67-82, Jul. 1997.

ESnet


SEATTLE
PNNL
BOISE
SACRAMENTO
LBNL
SUNNYVALE
SLAC
DENVER
KANSAS CITY
CHICAGO
AMES
ANL
FNAL
BOSTON
To AMSTERDAM
To LONDON
BNL
NEW YORK
PPPL
To GENEVA
WASHINGTON DC
JLAB
ORNL
NASHVILLE
ATLANTA
ALBUQUERQUE
El PASO
HOUSTON