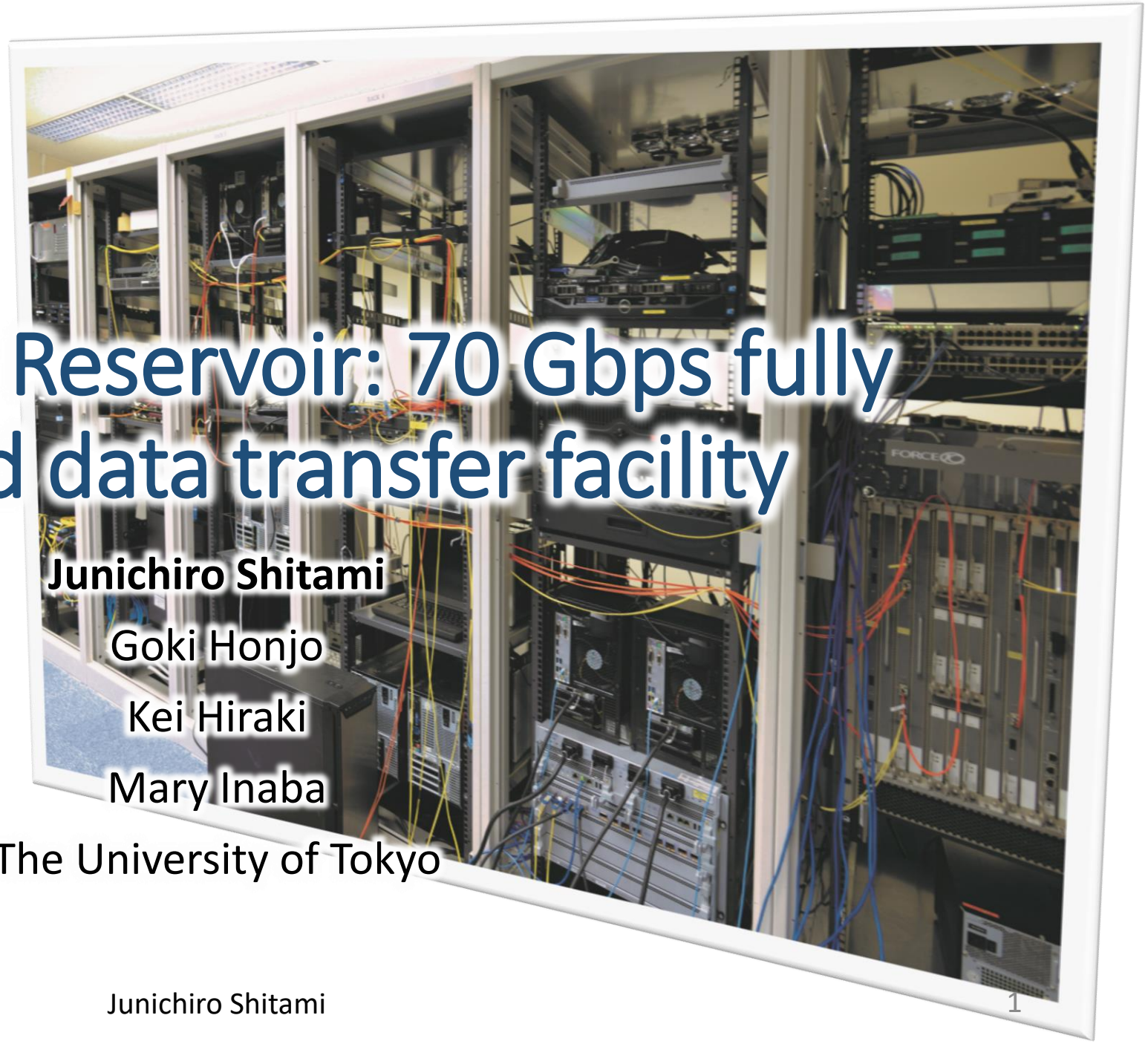
Secure Data Reservoir: 70 Gbps fully encrypted data transfer facility



Junichiro Shitami

Goki Honjo

Kei Hiraki

Mary Inaba

The University of Tokyo

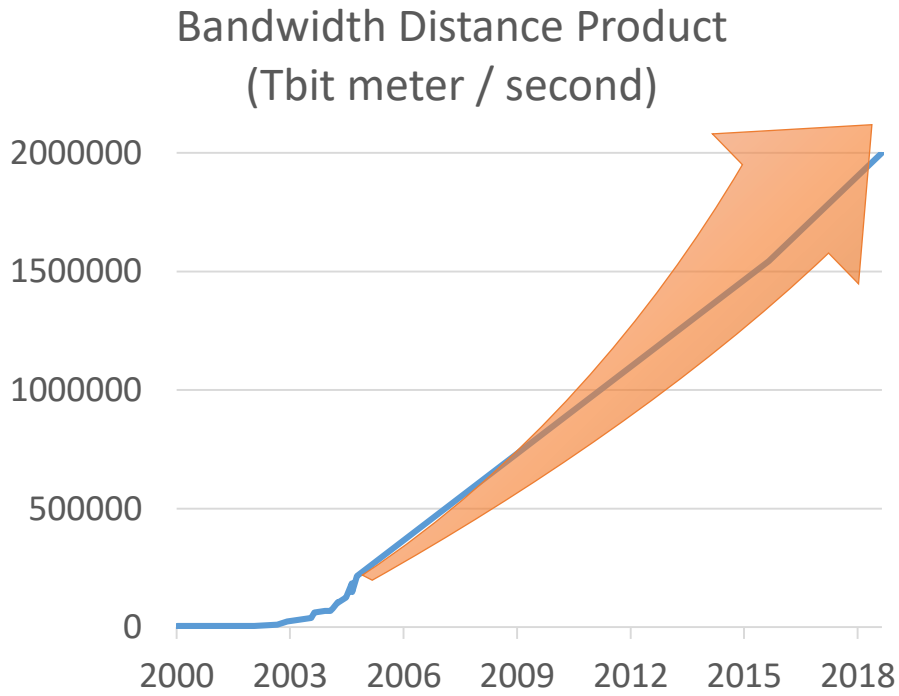
東京大学
THE UNIVERSITY OF TOKYO

Agenda

- Background
- Our Goal
- Preliminary Evaluations
- Design of Secure Data Reservoir
- Evaluation on Real Network
- Conclusion
- Future Work

Background

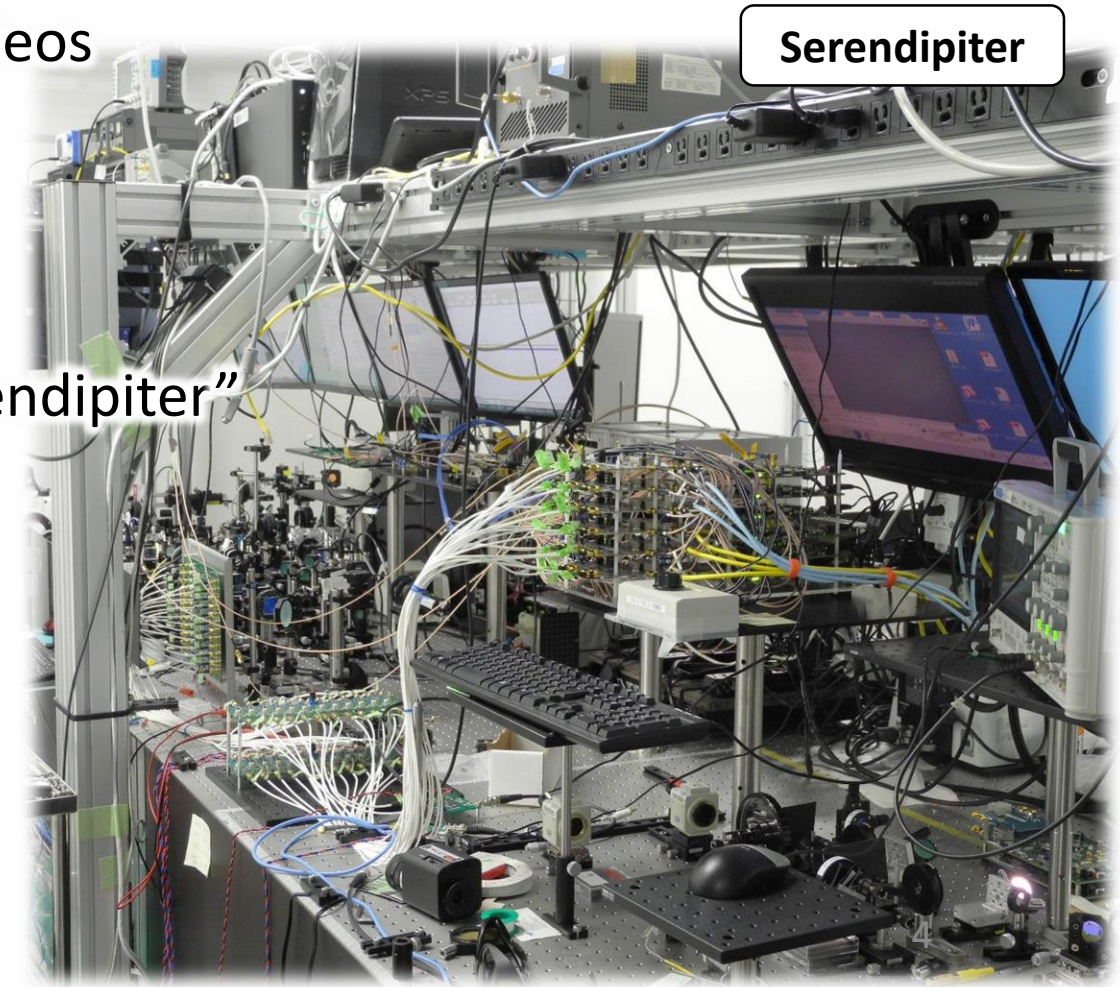
- High performance data transfer is essential
  - We have been working to efficiently utilze high-speed network
- Data Reservoir Project
  - Started on 1 Gbps Tokyo – US network
  - Achieved Land Speed Record of that date
- We can get enough performance
  - for memory transfer even 100G network
  - also for unencrypted storage transfer



Change of target application

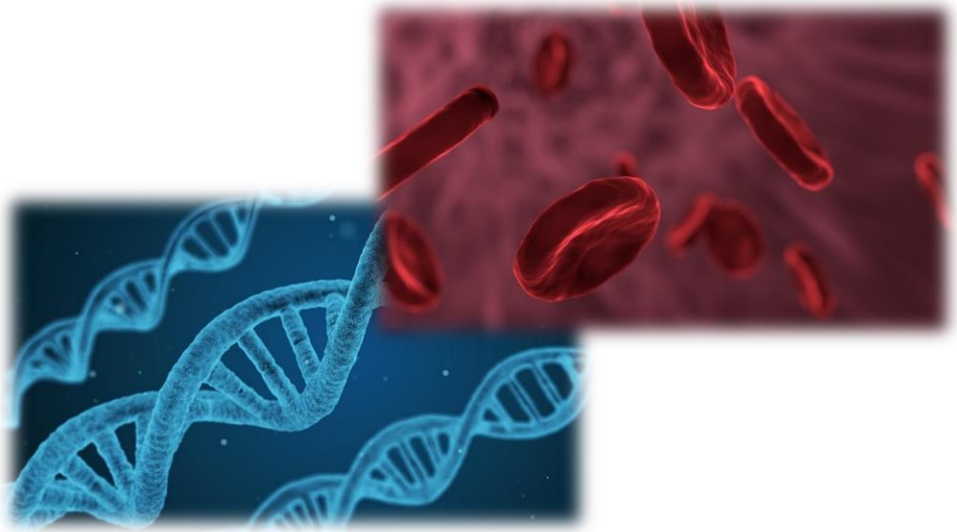
- Target application is changed
  - Existing: Physics, Astronomy, Graphics, Videos
  - Recent: Bio-science, Medical science

- We need fully encrypted transfer
  - for very high-speed cell sorter called “Serendipiter”
  - Bio science is regulated by low
  - Traditionally used “scp” is very slow



Our Goal

- Design a data transfer facility optimal for bio- and medical science
  - to enable fully encrypted storage transfer
  - using ordinary Linux 1 server pair
  - utilizes all available bandwidth on intercontinental network
  - named “Secure Data Reservoir”
- Verify the performance
  - on high-speed long-distance network
  - on real network
  - with pacing technique



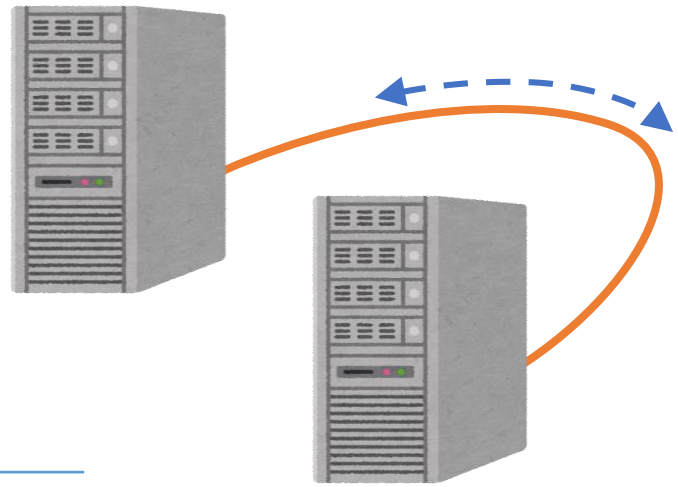
Preliminary Evaluations

- We made several preliminary evaluations
  - for the better design of Secure Data Reservoir

- Evaluations
  - Network performance
  - RAID performance
  - Storage performance
  - Encryption performance

Network Performance

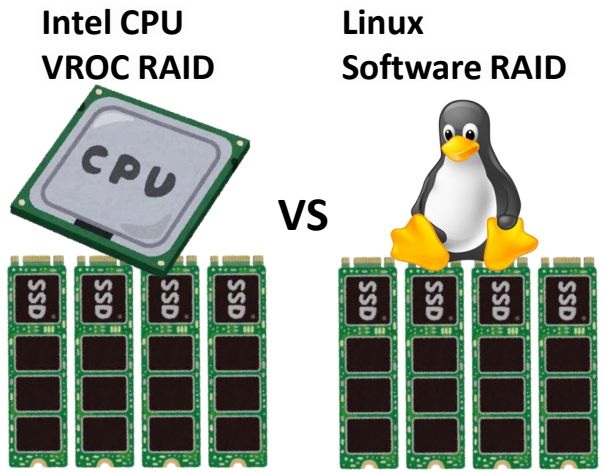
- We plan to use Chelsio 100 Gbps NIC (T62100-LP-CR)
- Confirm basic TCP send/receive performance
  - on back-to-back network
  - using iperf3
- Result
  - Wire-rate performance
  - Receiver CPU load is higher than sender



	Throughput	**CPU load**
Send	99 Gbps	75 %
Receive	99 Gbps	90 %

RAID Performance

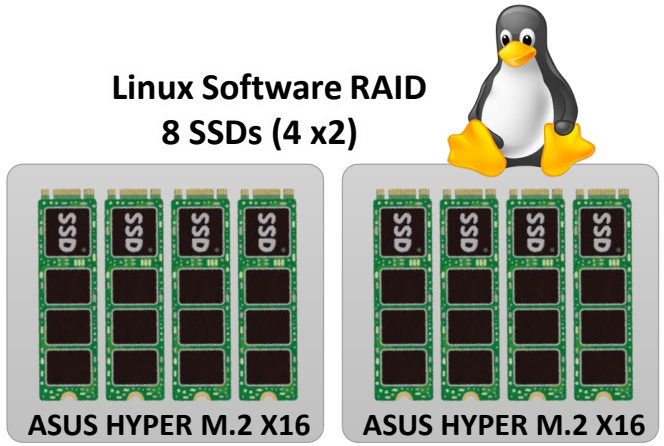
- We plan to use Intel VROC RAID
- Compare Intel VROC RAID and Linux software RAID
  - on same SSD
  - using fio benchmark
- Result
  - Almost no difference
  - We can use any SSDs regardless of VROC support



	Read	CPU load	Write	CPU load
Linux software RAID	6729 MB/s	28.2 %	2059 MB/s	15.9 %
VROC RAID	6751 MB/s	28.3 %	2087 MB/s	16.2 %

Storage Performance

- We plan to use Samusung SSD 960 PRO (NMVe)
- Confirm basic read/write performance
  - using 8 SSDs per 1 server (Linux software RAID)
  - using fio benchmark
- Result
  - Sufficient performance for data transfer on 100 Gbps network
  - Write CPU load is much higher than read

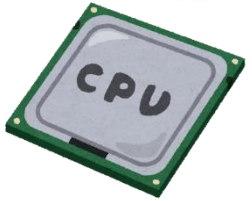


	Throughput	CPU load
Read	180 Gbps	100 %
Write	95 Gbps	100 %

Encryption Performance

- We plan to use Crypto Offload function of Chelsio NIC
- Compare Chelsio Crypto Offload and Intel AES-NI (CPU)
  - using openssl benchmark
- Result
  - Almost no difference
  - Chelsio Crypto Offload does not reduce CPU load

Intel CPU
AES-NI encryption



vs

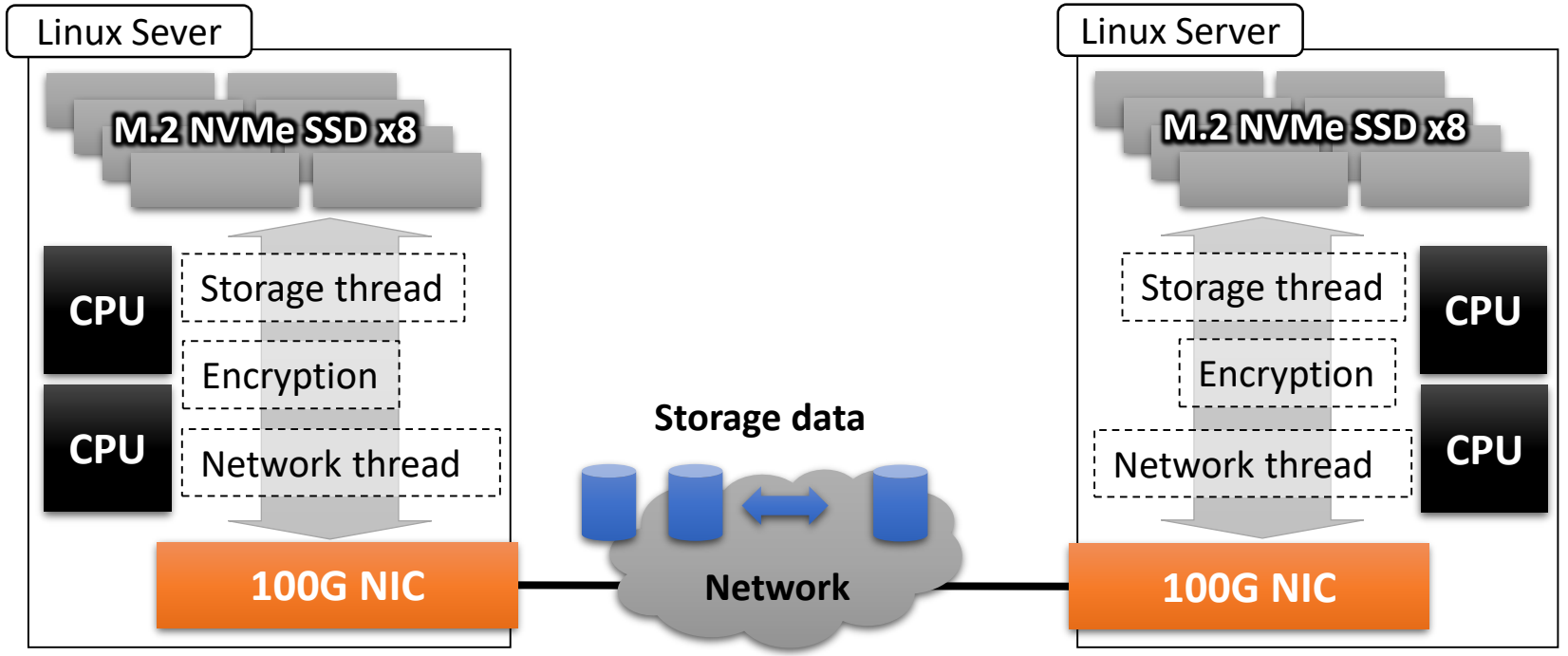
Chelsio
Crypto Offload

	64 bytes	1024 bytes	8192 bytes	CPU load
Intel AES-NI	29.0 GB/s	80.4 GB/s	98.8 GB/s	100 %
Chelsio Crypto Offload	28.6 GB/s	81.2 GB/s	98.9 GB/s	100 %

Design of Secure Data Reservoir

- Design component
  - Chelsio 100G NIC
  - Sumsung SSD 960 PRO (8 SSDs per 1 server)
  - Linux software RAID
  - Intel AES-NI encryption
- Features
  - AES256 encryption
  - Dedicated Storage threads and network threads
  - Pacing technique to stabilize TCP streams

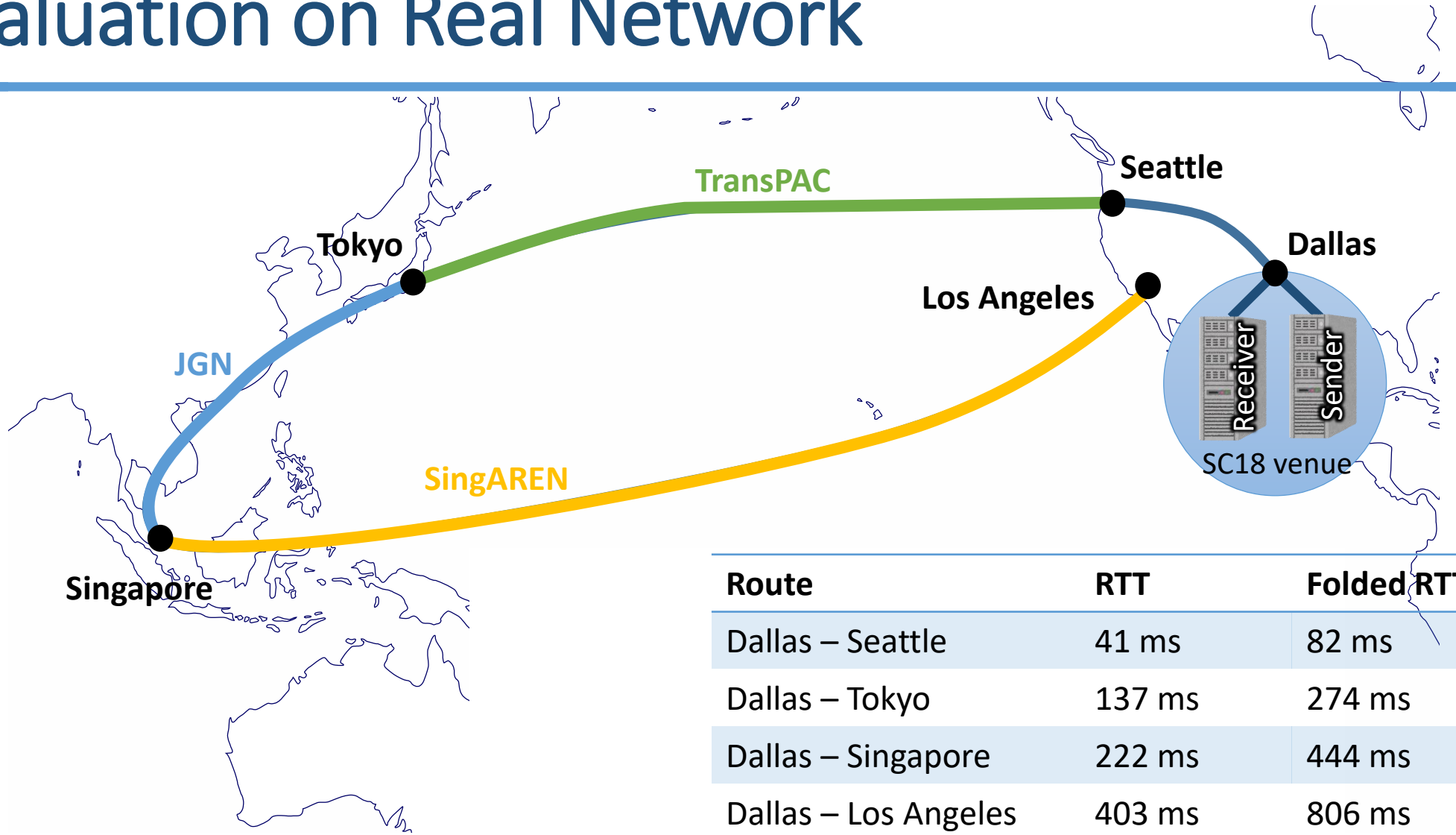
System Configuration



CPU	Intel Xeon Scalable Gold 6144 (8 core, 3.5 GHz) x2
Memory	DDR4-2666 192 GB
Network	Chelsio T62100-LP-CR (100Gbps NIC)
Storage	Samsung SSD 960 PRO (512GB) x8, ASUS HYPER M.2 X16
OS	Linux 4.9.88

東京大学 THE UNIVERSITY OF TOKYO

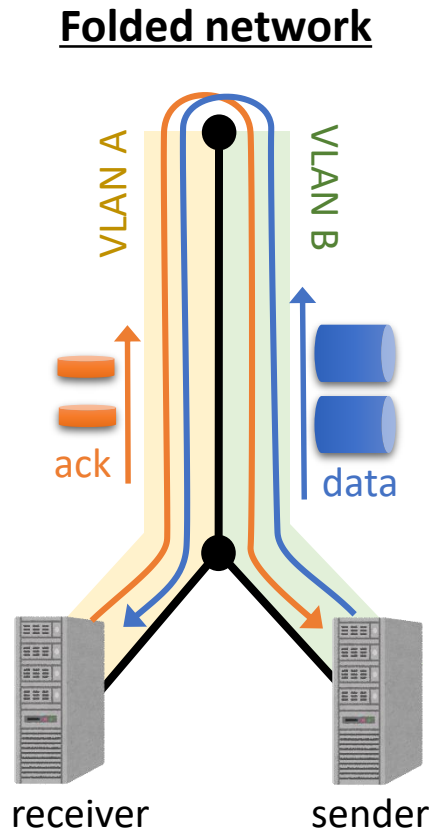
Evaluation on Real Network



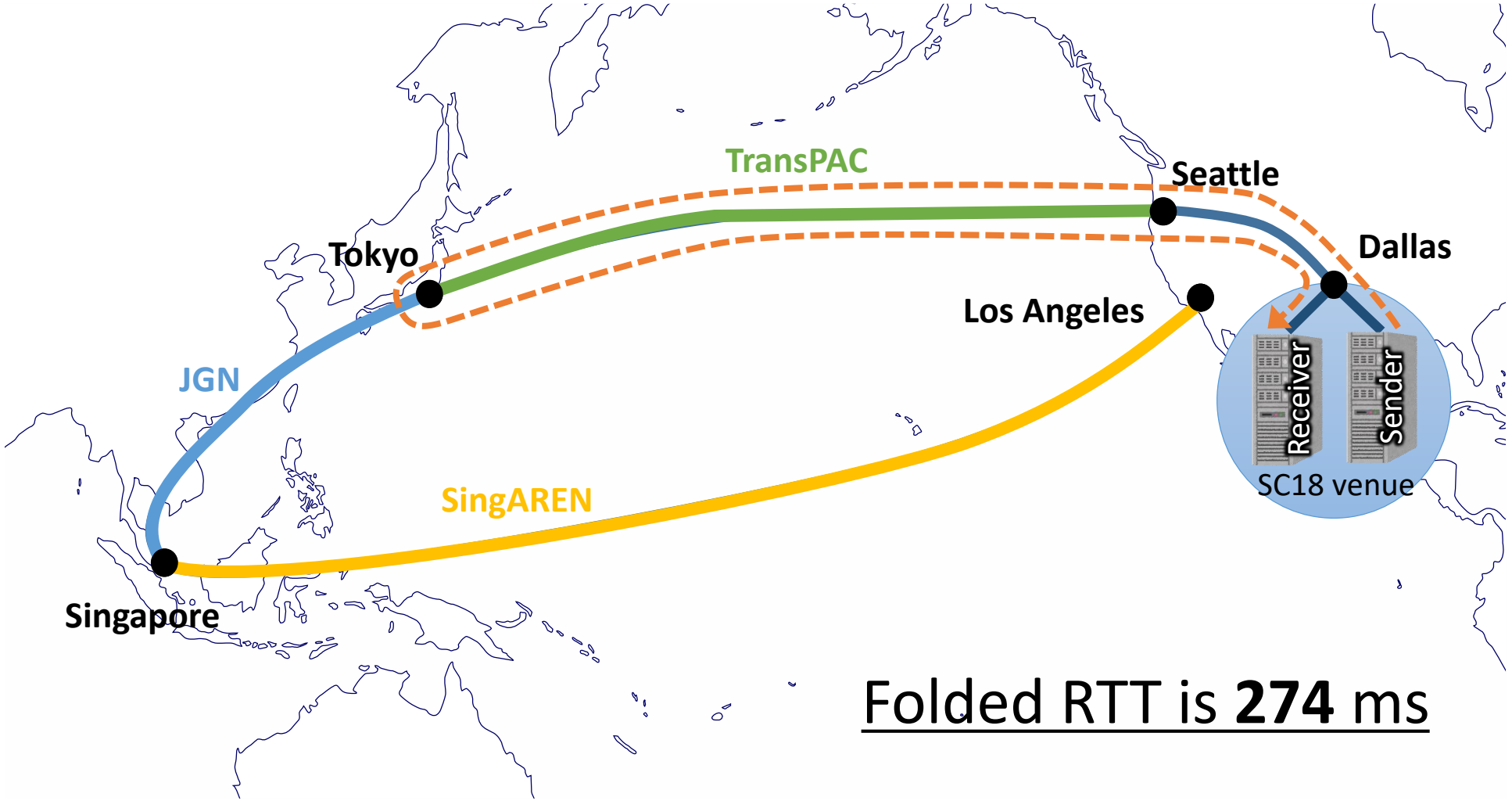
Route	RTT	Folded RTT
Dallas – Seattle	41 ms	82 ms
Dallas – Tokyo	137 ms	274 ms
Dallas – Singapore	222 ms	444 ms
Dallas – Los Angeles	403 ms	806 ms

Experimental Setup

- 2 servers (sender, receiver) placed at SC18 venue
  - Using folded network
- Experiments on 3 routs
  - Dallas – Tokyo
  - Dallas – Singapore
  - Dallas – Los Angeles
- Encrypted memory and Storage transfer
- Sender side traffic pacing technique

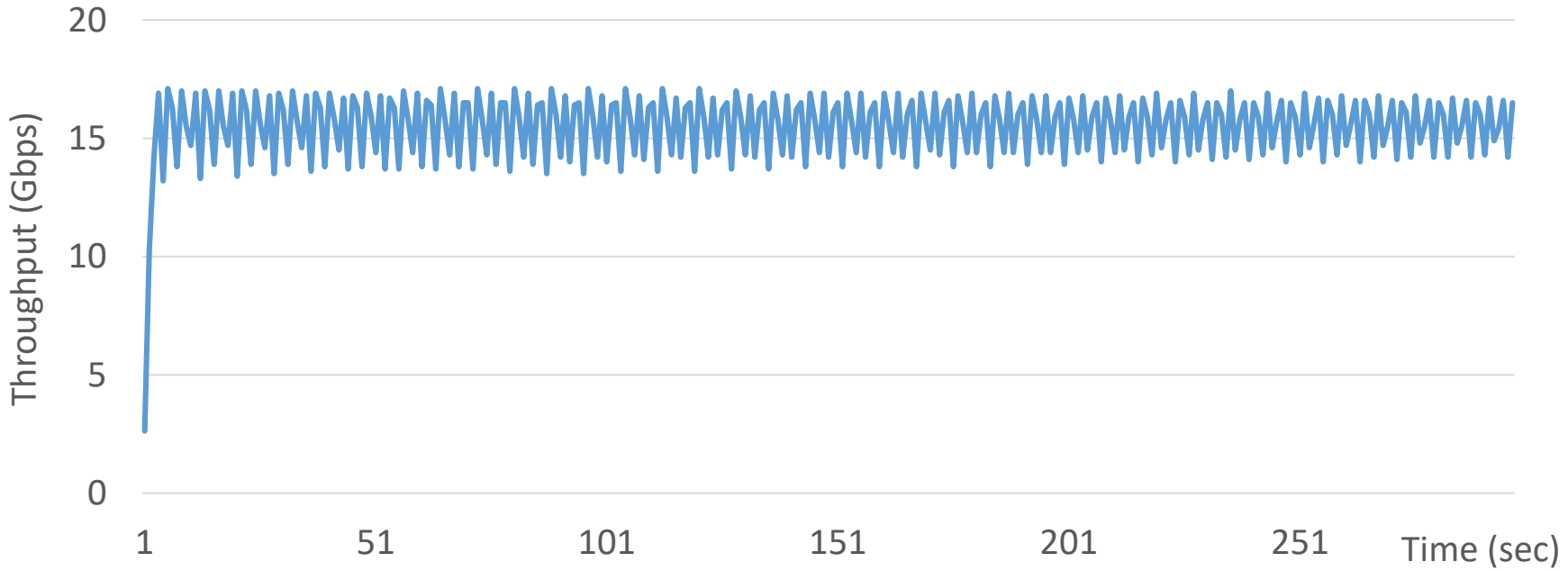


Result: Dallas – Tokyo



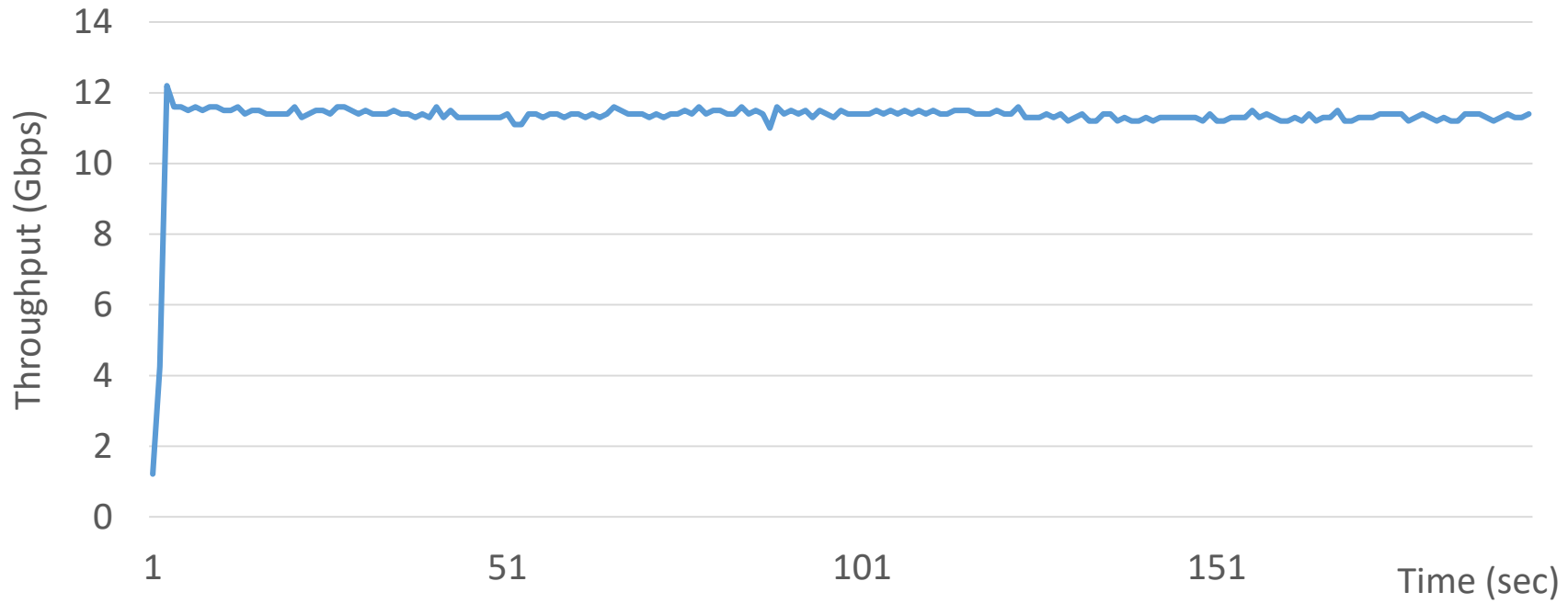
Folded RTT is **274** ms

Result: Dallas – Tokyo (RTT 274 ms)
Encrypted Memory to Memory Transfer



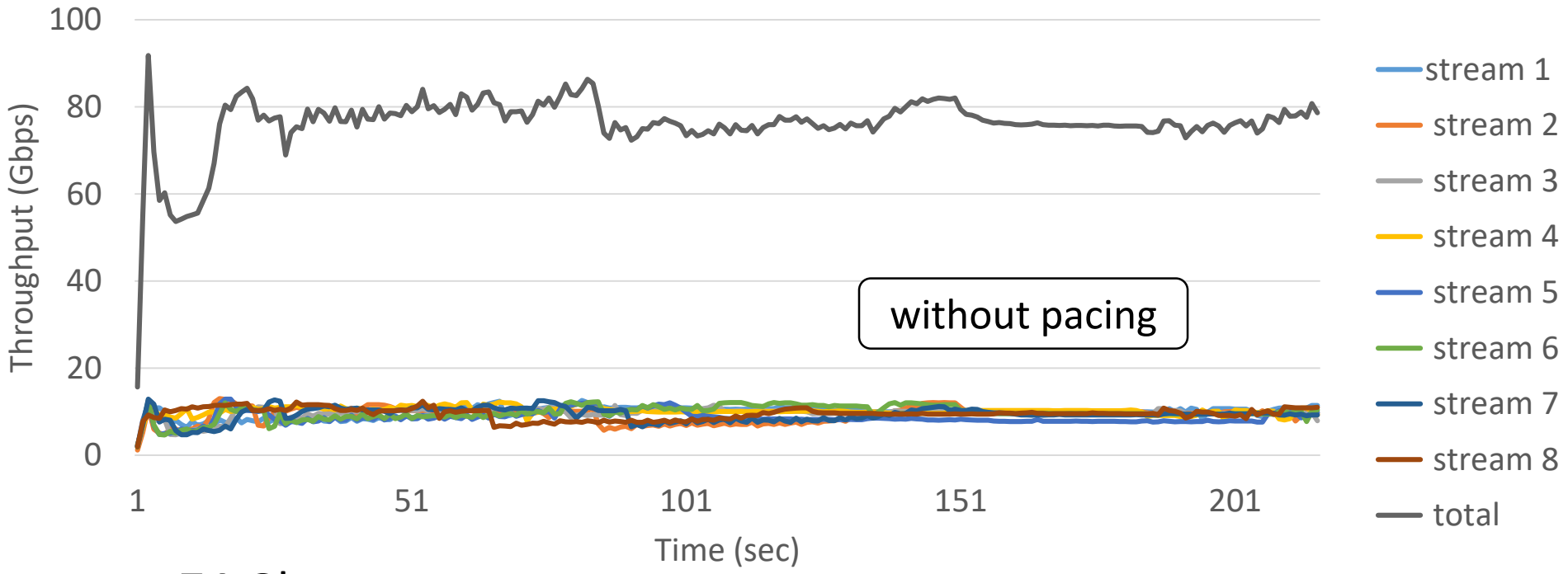
- ~5 seconds to peak throughput
- ~15 Gbps
- Throuput is vibrating due to TCP window size limitation

Result: Dallas – Tokyo (RTT 274 ms)
Encrypted Storage to Storage Transfer



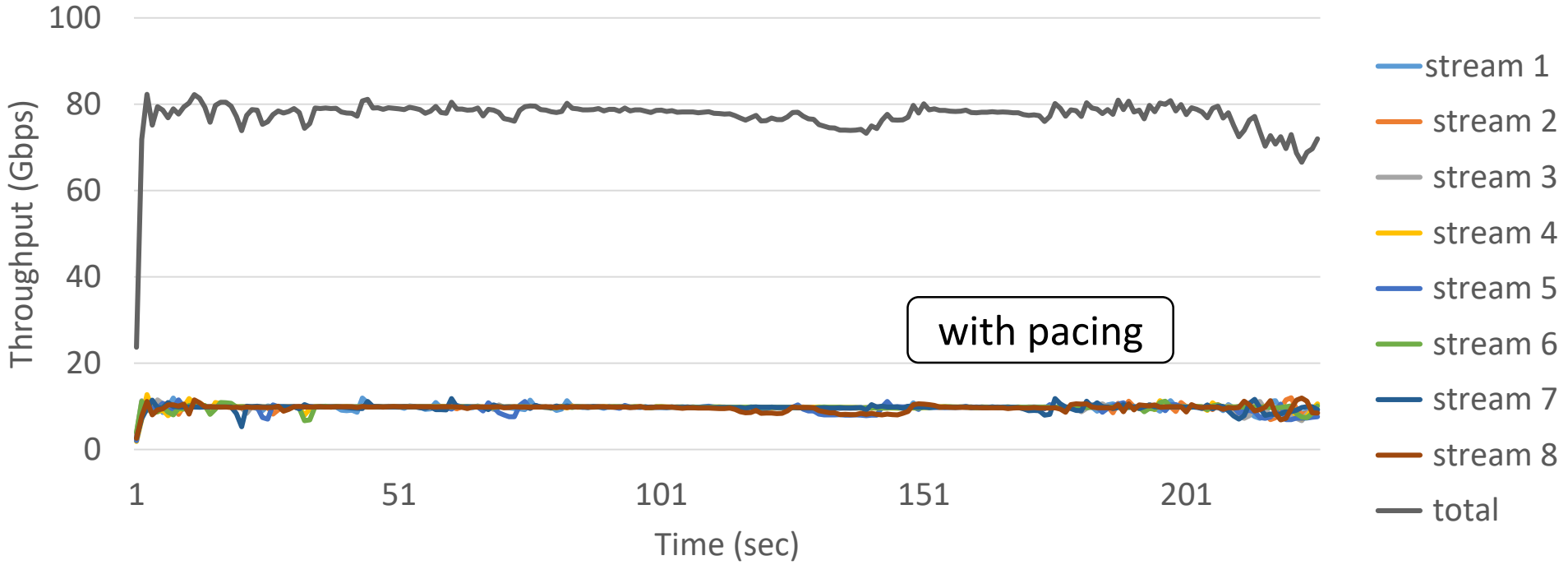
- ~11 Gbps
- Throughput is limited by storage performance

Result: Dallas – Tokyo (RTT 274 ms)
Encrypted Storage to Storage Transfer x8



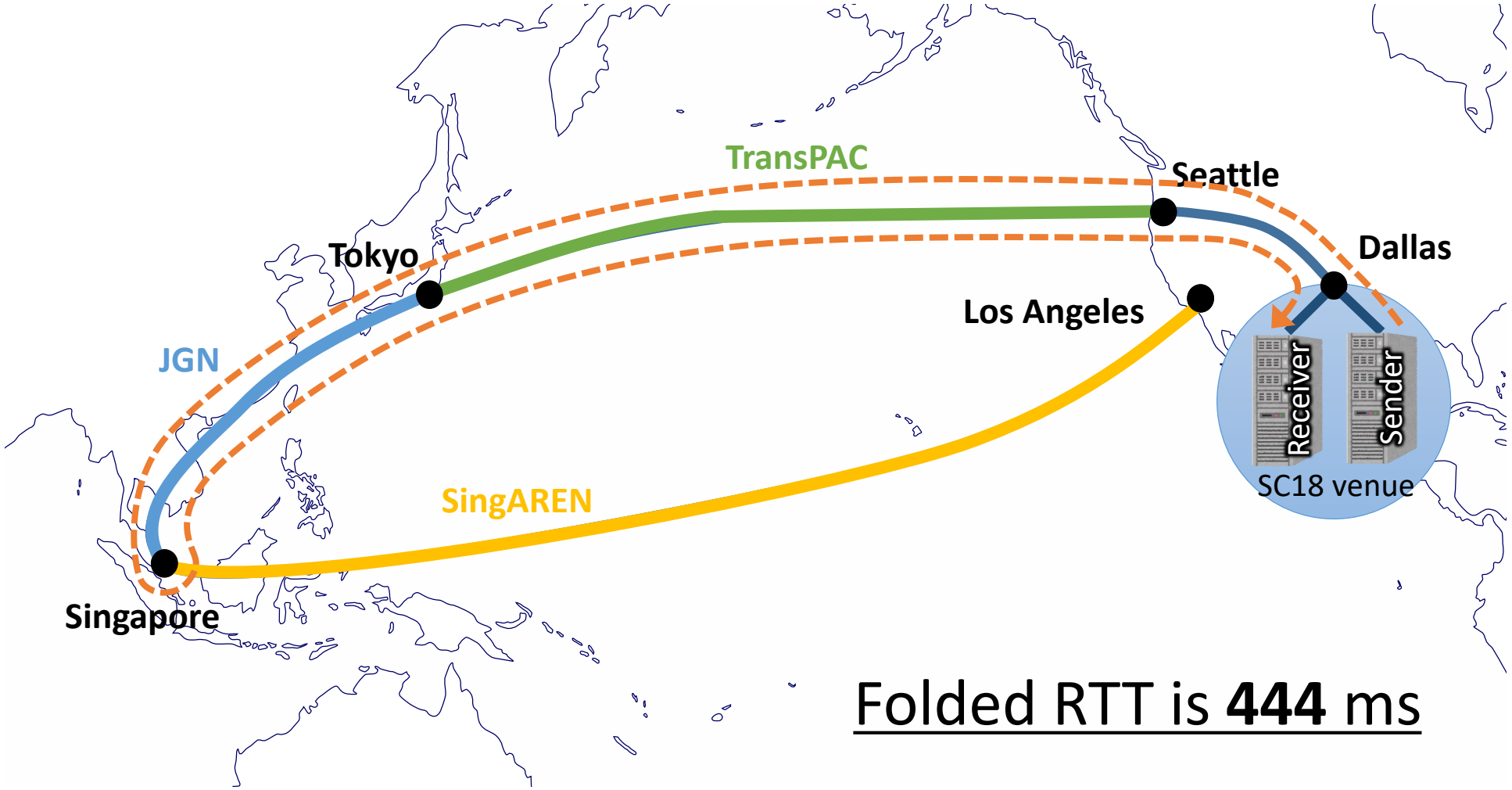
- ~74 Gbps
- Throughput is not stable
- Difference between streams is large

Result: Dallas – Tokyo (RTT 274 ms)
Encrypted Storage to Storage Transfer x8



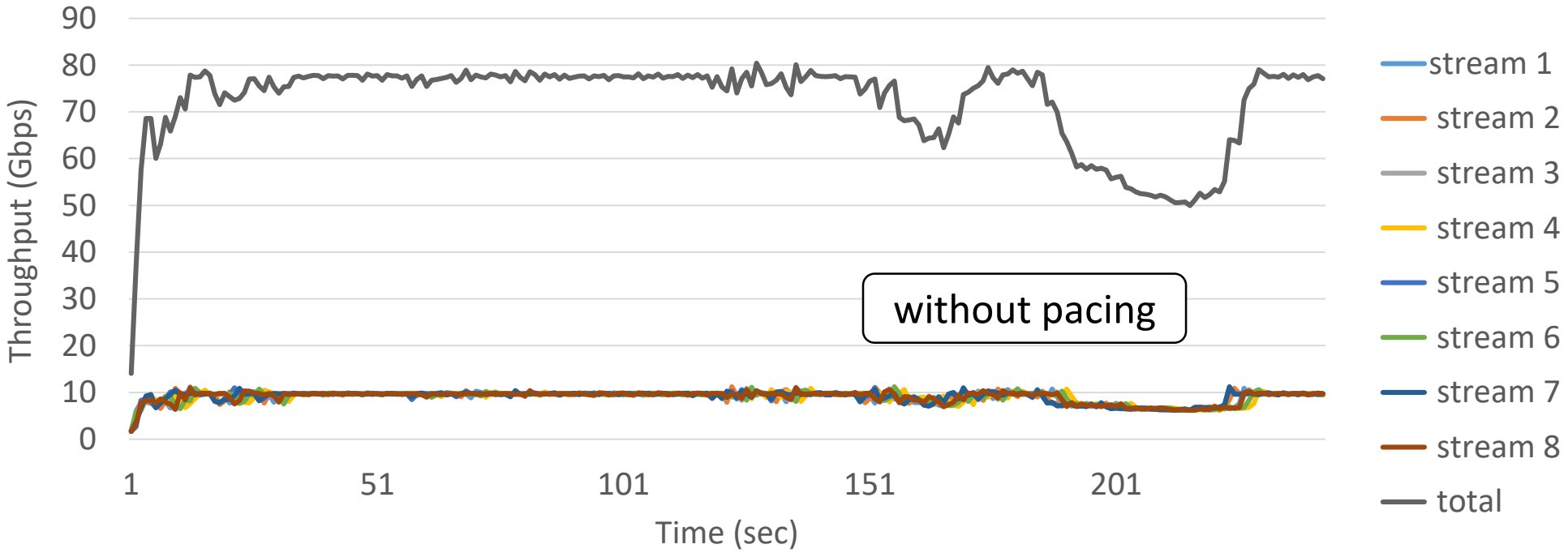
- ~76 Gbps
- Pacing technique improve stability
- Difference between streams decreases

Result: Dallas – Singapore



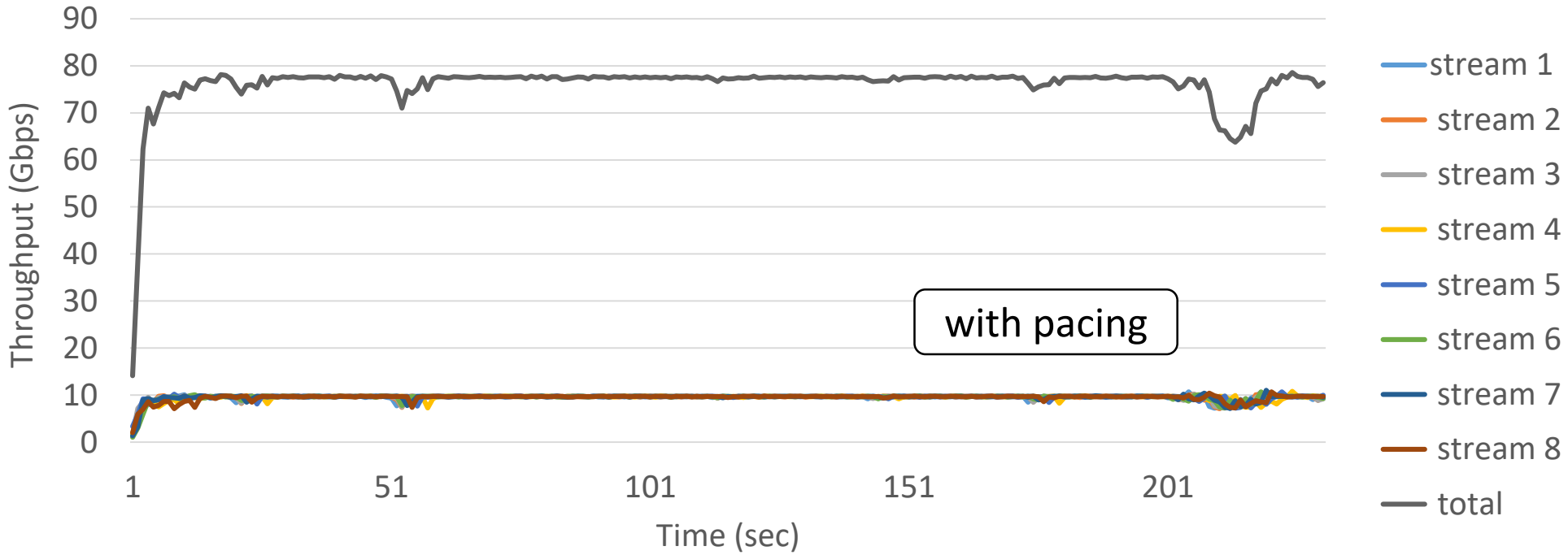
Folded RTT is **444** ms

Result: Dallas – Singapore (RTT 444 ms)
Encrypted Storage to Storage Transfer x8



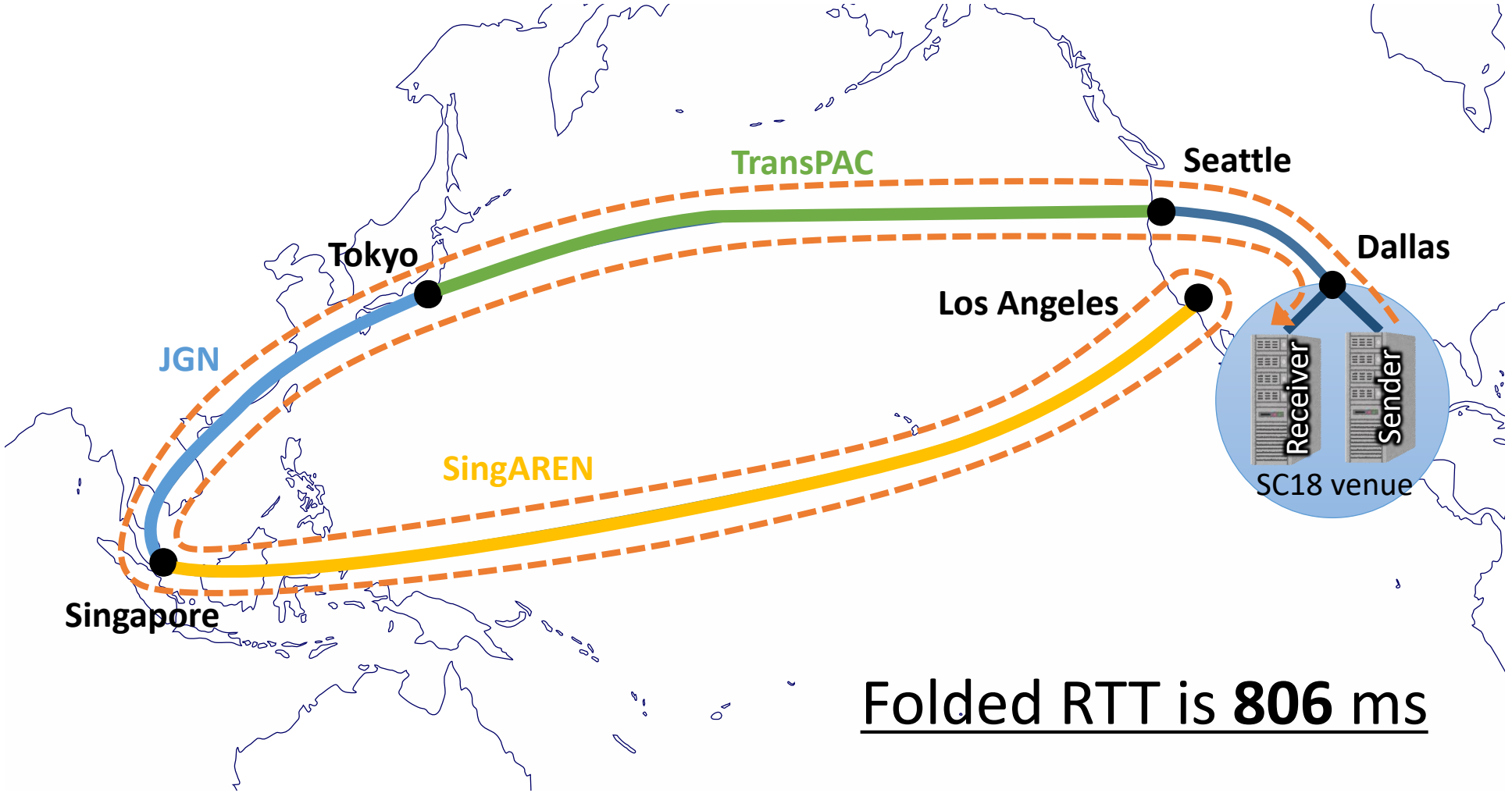
- ~70 Gbps
- Throughput is not stable
- Throughput degradation due to retransmission

Result: Dallas – Singapore (RTT 444 ms)
Encrypted Storage to Storage Transfer x8



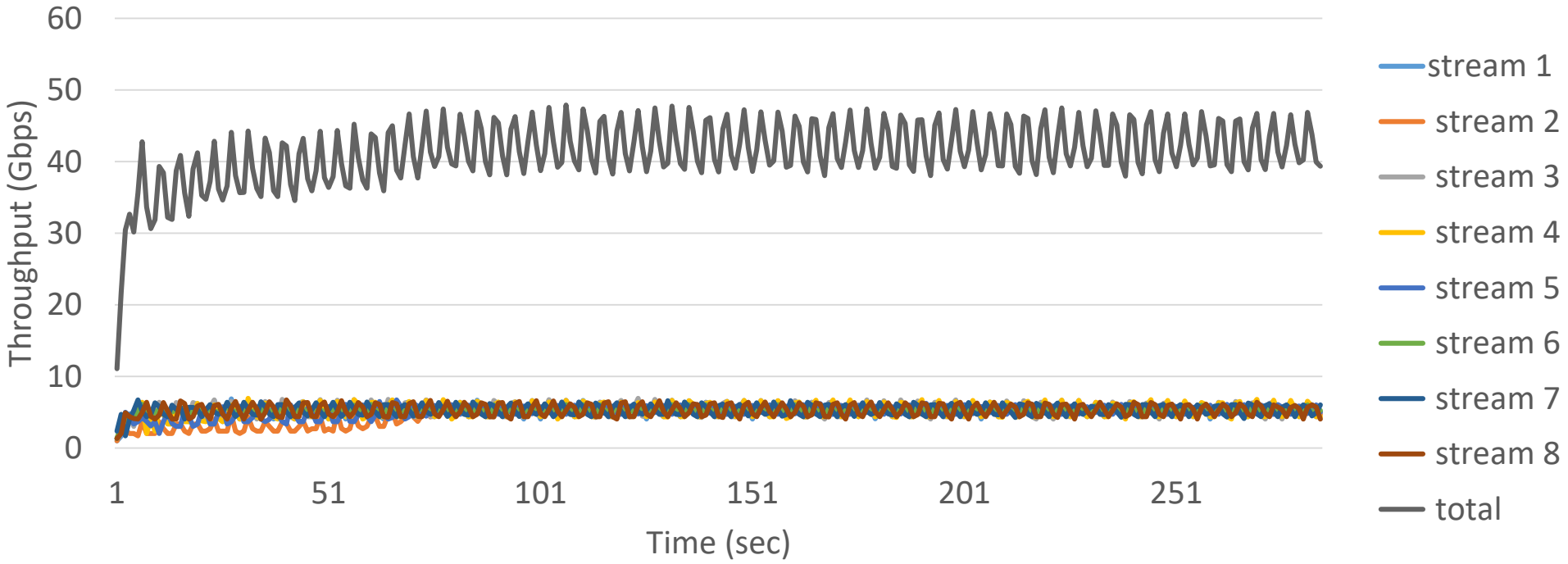
- ~74 Gbps
- Pacing technique improve stability
- Difference between streams decreases

Result: Dallas – Los Angeles



Folded RTT is **806** ms

Result: Dallas – Los Angeles (RTT 806 ms)
Encrypted Storage to Storage Transfer x8

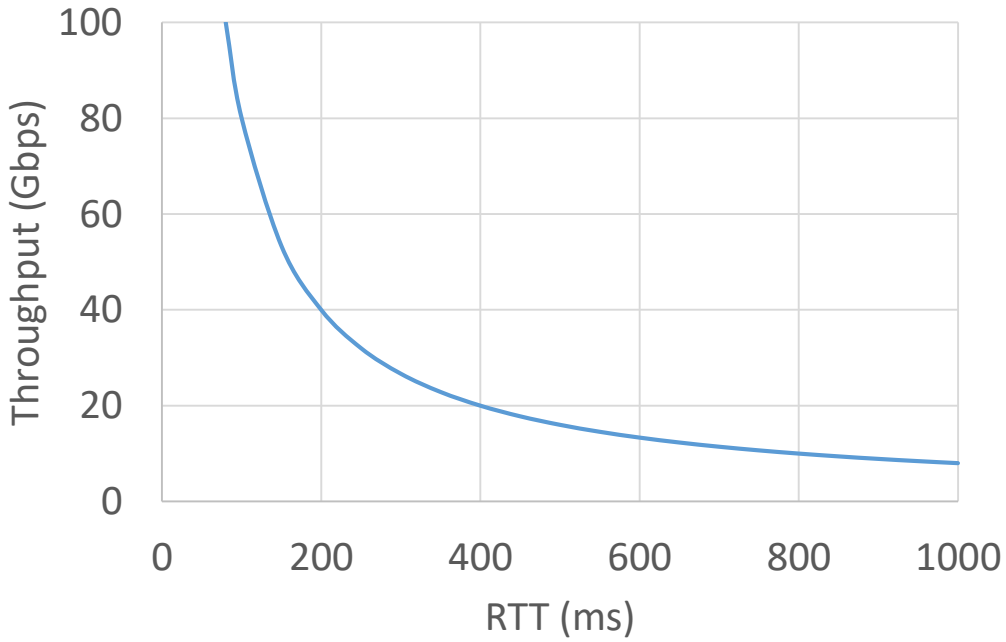


- ~40 Gbps
- Throuput is vibrating due to TCP window size limitation

TCP window problem

- Standard TCP windows size is up to 1 GB
  - It is too small for very long-distance high-speed network
  - like Dallas – Los Angeles route

- We developed LFTCP protocol
  - overcomes TCP window size limitation
  - out of the scope of this presentation



Result Summary
Total of 8 streams encrypted storage transfer

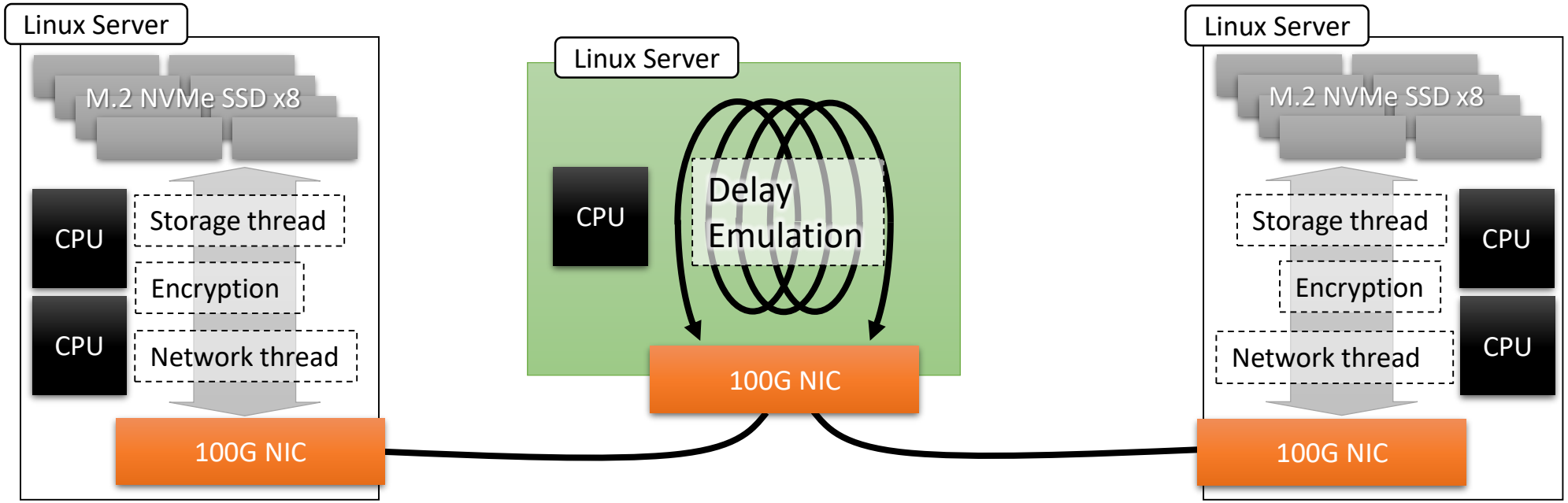
- Pacing technique improve stability
  - Difference between peak and average throughput decreases
- Dallas – Los Angeles is too long
  - Total throughput is limited by TCP window size

Route	RTT	Peak Throughput	Average Throughput
Dallas – Tokyo (without pacing)	274 ms	90.5 Gbps	74.3 Gbps
Dallas – Tokyo (with pacing)	274 ms	83.6 Gbps	76.3 Gbps
Dallas – Singapore (without pacing)	444 ms	80.6 Gbps	70.2 Gbps
Dallas – Singapore (with pacing)	444 ms	79.6 Gbps	74.1 Gbps
Dallas – Los Angeles	806 ms	50.6 Gbps	39.7 Gbps

Comparison with Emulated Delay

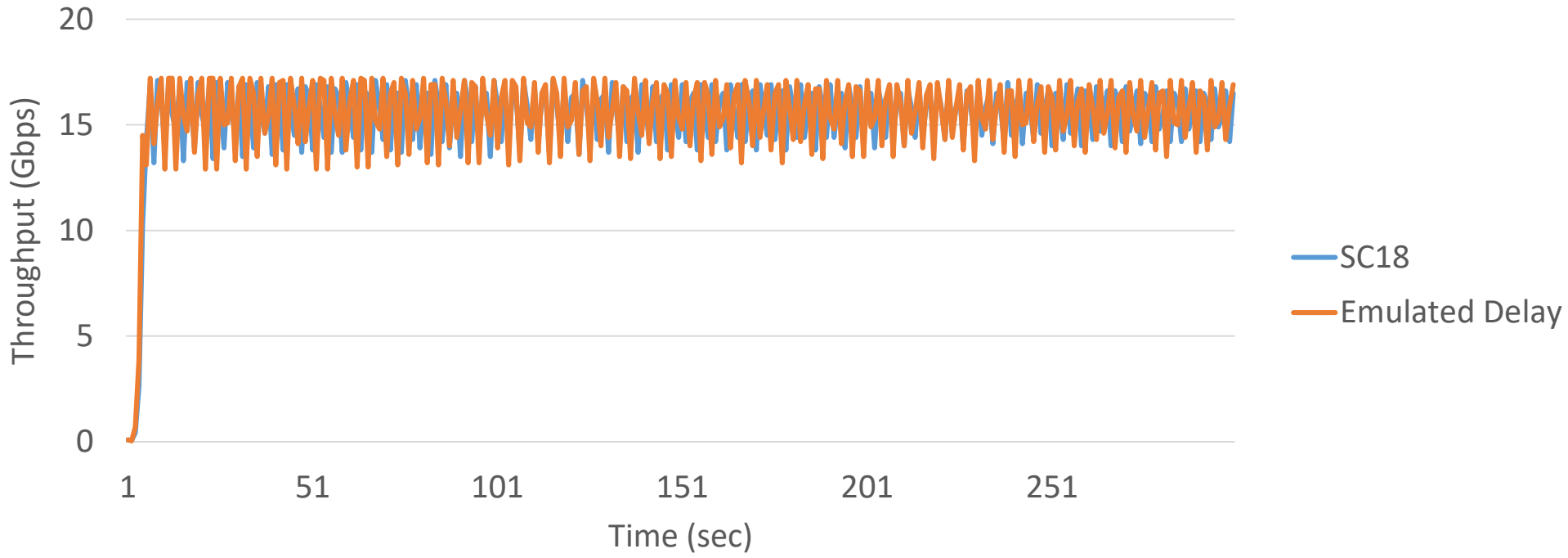
- Compare the result
  - on SC18 network (real network)
  - on emulated delay network
- Memory and storage transfer (1 stream)
  - Due to delay emulation performance
- Network configuration is not exactly same

Emulated Delay Setup



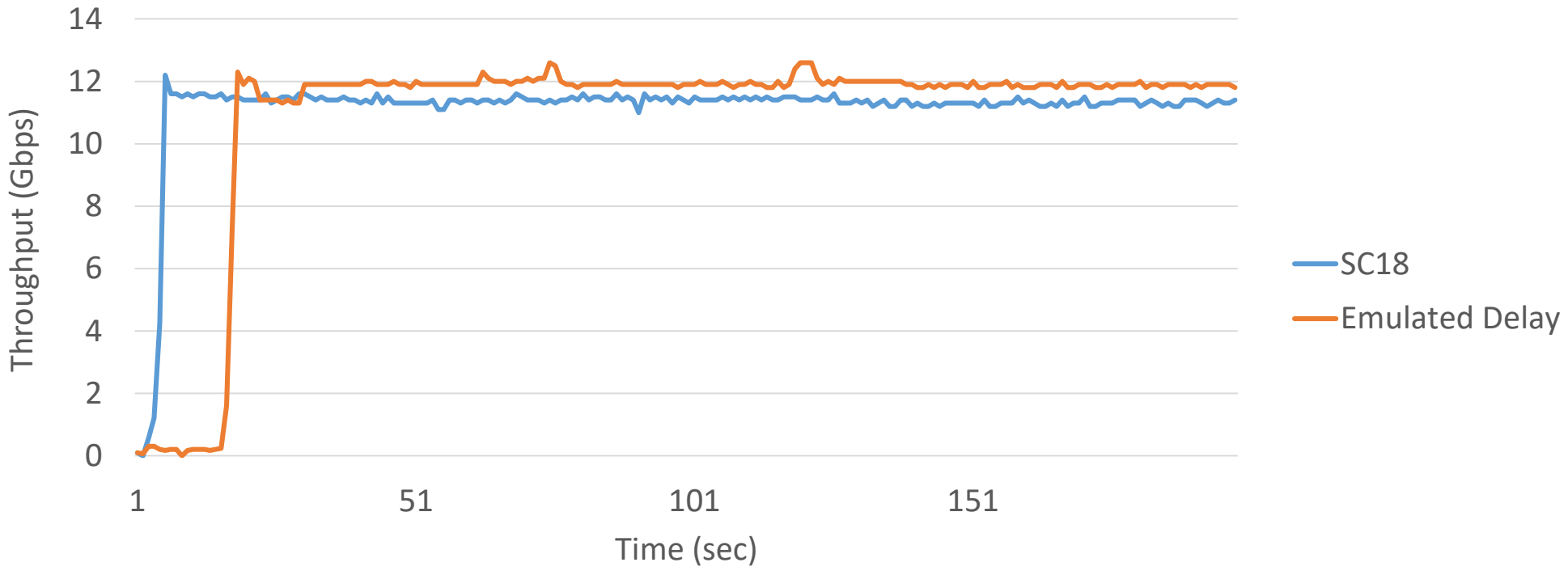
- Delay is emulated by Linux server
- Maximum throughput is ~70 Gbps
  - depends on traffic pattern

Result: RTT 275 ms (Dallas – Tokyo)
Encrypted Memory to Memory Transfer



- Same behavior

Result: RTT 275 ms (Dallas – Tokyo)
Encrypted Storage to Storage Transfer



- Average throughput is nearly same
- Behavior of start is different

Discussion about Emulated Delay

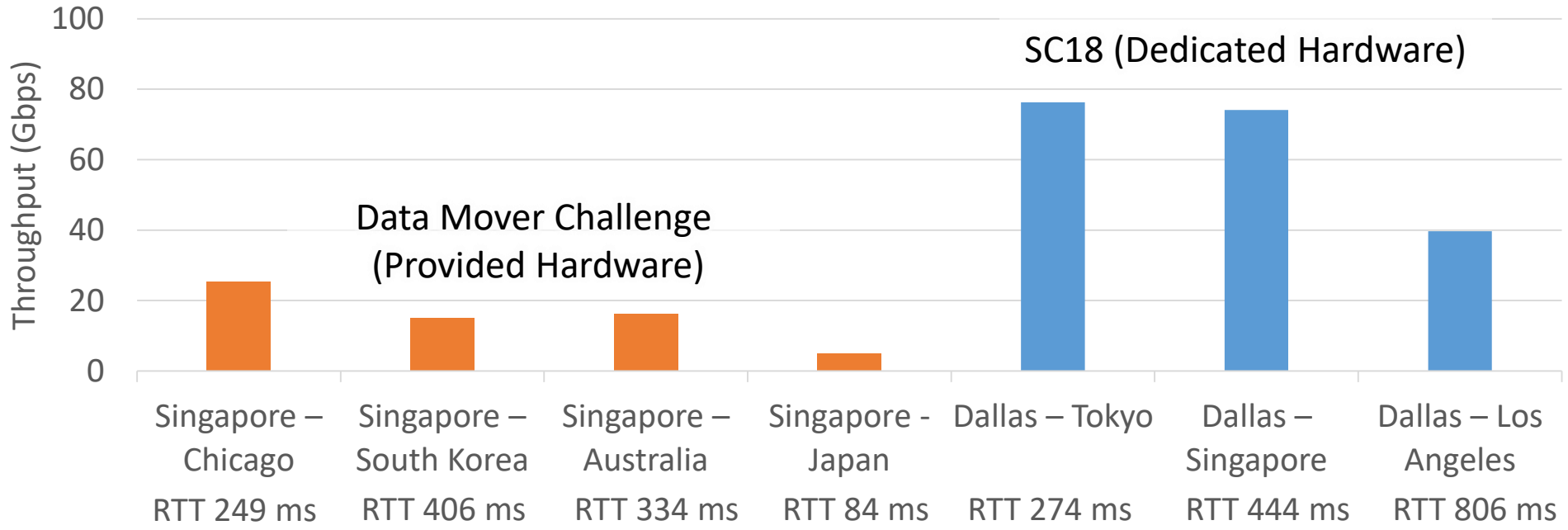
- Emulated delay works with 1 stream transfer
  - Large traffic causes significant performance degradation
  - Pacing technique may be useful
- Average throughput is same
- Behavior of start is different
- Impact is large on storage transfer

Conclusion

- We designed “Secure Data Reservoir”
- We achieved 70 Gbps fully encrypted data transfer
  - on real intercontinental network
- Pacing technique improves stability
  - CPU load on receiver side is higher than sender side
- Throughput may be limited by TCP window size
  - LFTCP protocol overcomes this limitation

Current Work

- We participate **Move That Data!** Data Mover Challenge at SCA19.
- Hardware is critical to performance



Future Work

- Apply Secure Data Reservoir to other large-scale medical applications
  - Currently for “Serendipiter” (very high-speed cell sorter)
- Improvement User Interface
  - Current software is very experimental
- Further performance improvement for 400 Gbps network

Any question?

Thanks to

This work was supported by JSPS KAKENHI Grant Number JP16H01714.