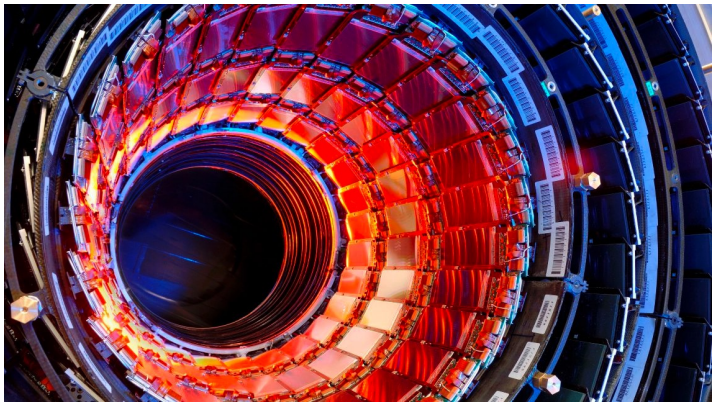
Northeastern

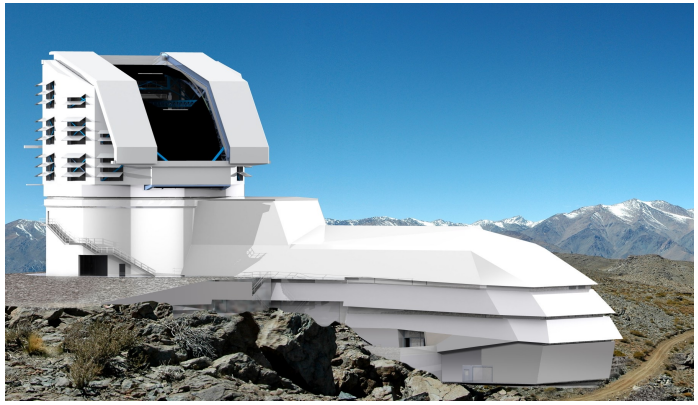
NDN for Data-Intensive Science Experiments (N-DISE)

Edmund Yeh

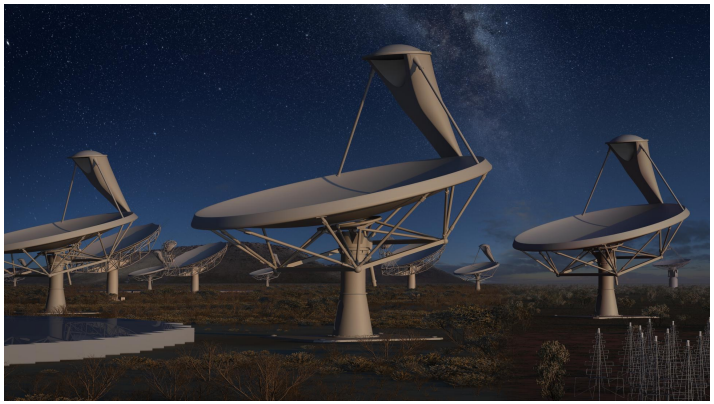
Data-Intensive Science



LHC: Large Hadron Collider



LSST: Large Synoptic Survey Telescope



SKA: Square Kilometer Array

Genomics

NSF N-DISE Project

- NSF CC* N-DISE: NDN for Data Intensive Science Experiments
  ($875K grant, 2020-2022)

- Team:
  - Northeastern (PI: E. Yeh), Caltech (Co-PI: H. Newman), UCLA (Co-PIs: L. Zhang and J. Cong), Tennessee Tech (Co-PI: S. Shannigrahi)
  - In partnership with LHC, genomics collaborators and NDN project team

- Challenges:
  - LHC data volume to grow 10x due to High Luminosity LHC (2027): increased data complexity
  - Human genome data, Earth Biogenome (~ exabyte range)
  - Need to use diverse computation, storage, networking resources

- Approach: build data-centric ecosystem to provide agile, integrated, interoperable, scalable, robust and trustworthy solutions for heterogeneous data-intensive domains

N-DISE Goals

- Deploy, commission first prototype production-ready NDN-based petascale data distribution, caching, access, computation system serving major science programs
- LHC high energy physics program as leading target use case. BioGenome, human genome projects, ATLAS, LSST, SKA as future use cases
- Leverage NDN protocols, high throughput forwarding/caching methods, containerization techniques
- Integrated with SDN methods and FPGA acceleration subsystems
- Deliver LHC data over wide area network at throughputs ~ 100 Gbps
- Dramatically decrease download times by using optimizing caching
- Enhanced WAN testbed with high performance NDN data cache servers

N-DISE Update

- N-DISE Deployment Architecture and NDNc
- WAN Testbed and Throughput Test
- Optimized Caching and Forwarding
- Congestion Control
- FPGA Acceleration

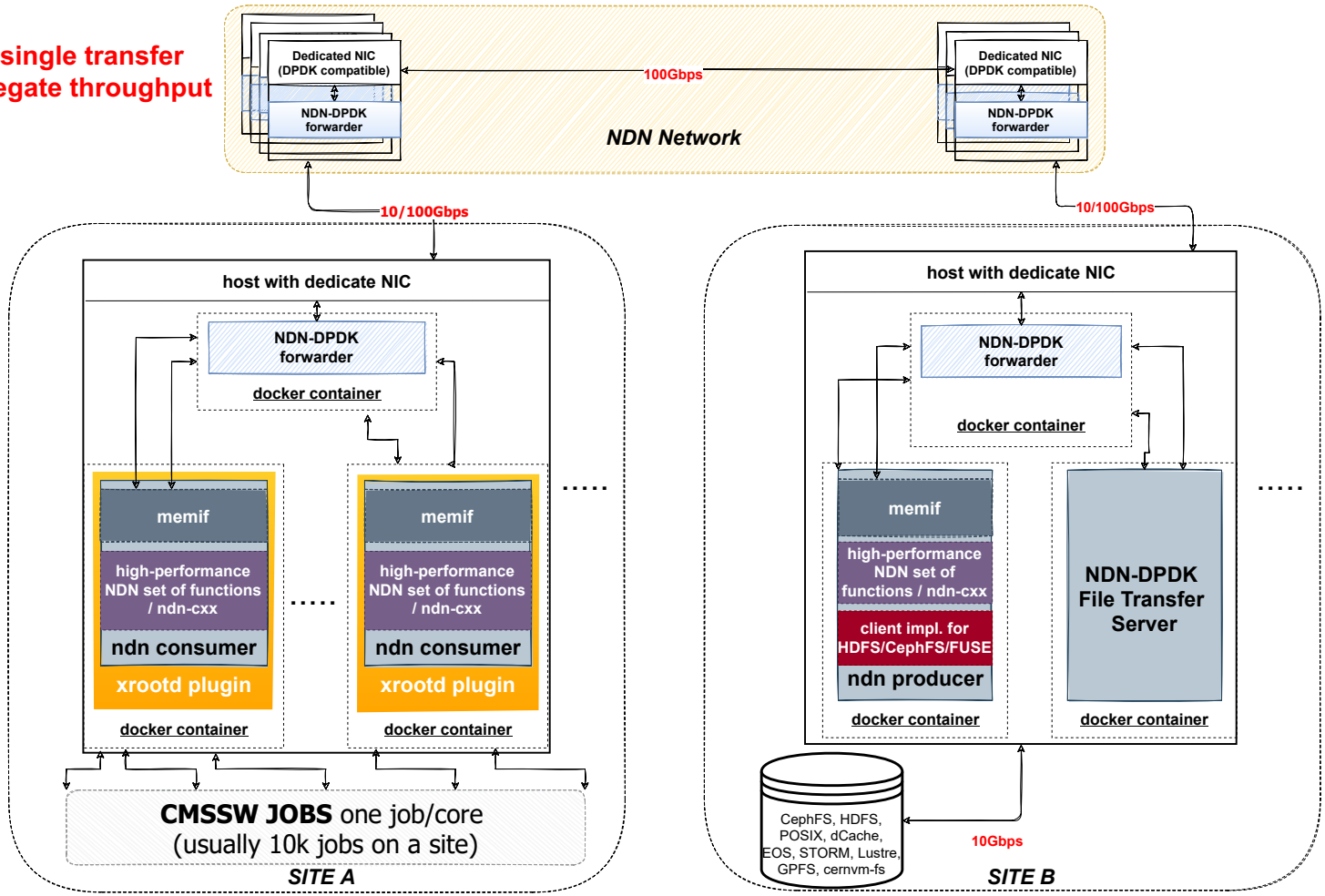
N-DISE Update

- **N-DISE Deployment Architecture and NDNc**
- WAN Testbed and Throughput Test
- Optimized Caching and Forwarding
- Congestion Control
- FPGA Acceleration

N-DISE Deployment Architecture

Performance goals:

- **~10 Gbps for a single transfer**
- **100 Gbps aggregate throughput**



NDNc

- A lightweight integration of ndn-cxx with NDN-DPDK to achieve high throughput performance in scientific applications
  - https://github.com/cmscaltech/sandie-ndn/tree/master/NDNc
  - uses memif shared memory packet interface that provides performance packet transmit and receive between user application and VPP
- GraphQL C++ client able to configure the local NDN-DPDK forwarder (*createFace*, *insertFibEntry* and *delete* JSON mutations available)
- Single threaded memif transport layer to transmit/receive bursts of packets at a time
- Face can transmit/receive one or many *ndn::Blocks* in a single burst
- Offers PIT token support (unavailable in ndn-cxx, needed by NDN-DPDK)
- Uses ndn-cxx library to encode/decode L2 and L3 packets
- Congestion window and retransmission: fixed and AIMD

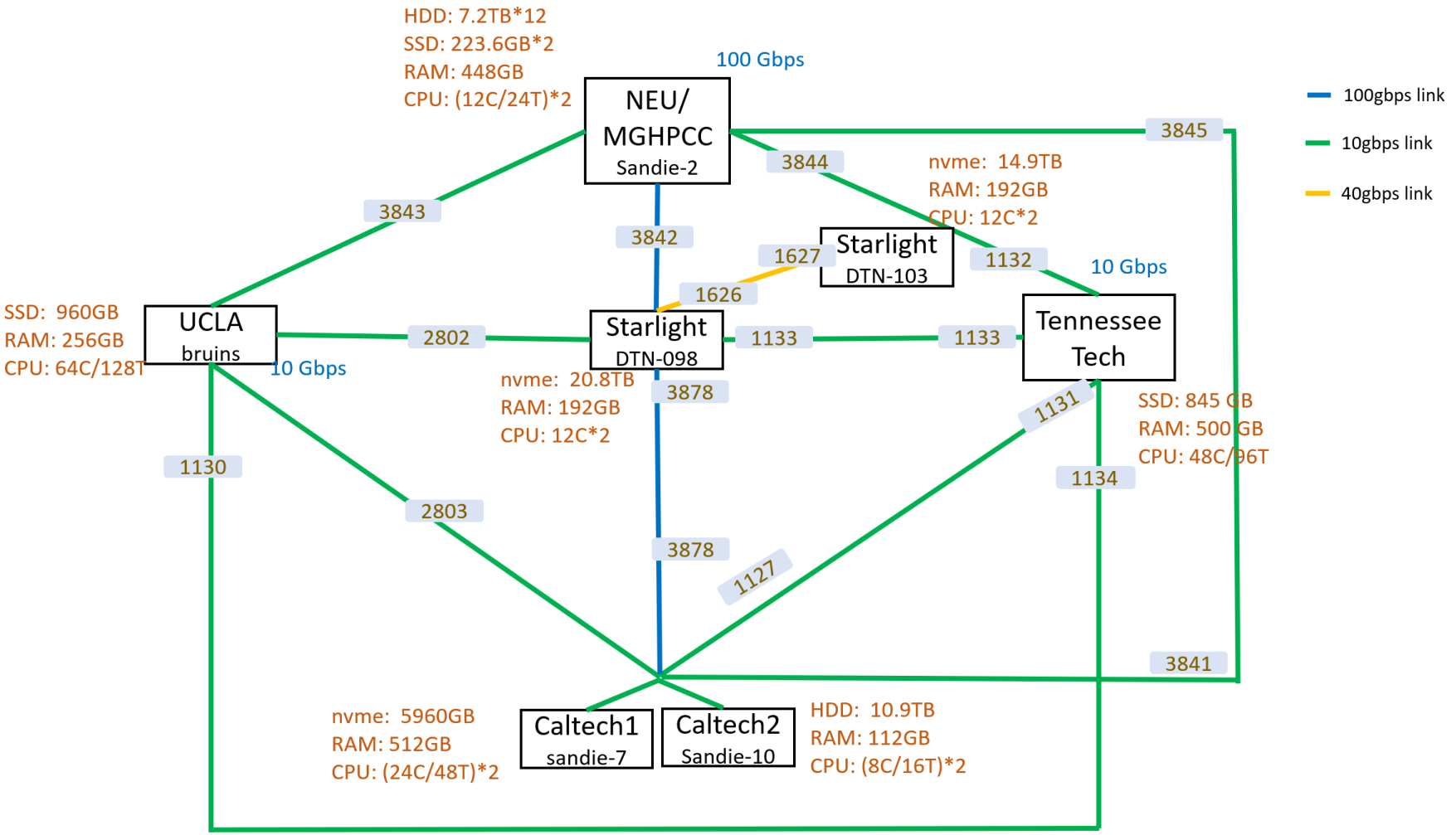
NDNc Future Plans

- Extensive benchmarking to understand the current behaviour and the maximum throughput performance that can be achieved
- Identify possible bottlenecks and fix them
- Add multi-threaded support to memif and pipelines
- Port the NDN XRootD plugin developed during the SANDIE project to NDNc
- Extend the number of consumer applications to HDFS/CephFS/Fuse

N-DISE Update

- N-DISE Deployment Architecture and NDNc
- **WAN Testbed and Throughput Test**
- Optimized Caching and Forwarding
- Congestion Control
- FPGA Acceleration

N-DISE WAN Testbed



Northeastern

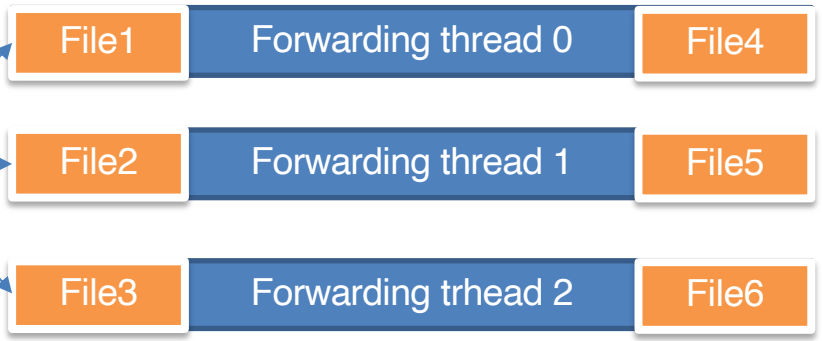
Throughput Test Setup

- 2 servers: Starlight dtn-098 & Starlight dtn-103
- Link capacity: ~40gbps with iperf3 test
- RTT= 29.8ms: Set up a loop at StarLight between Dtn098 and DTN103. Path goes to Canada and then back to Starlight
- NIC: ConnectX-5 on both sides
- CPU: Intel(R) Xeon(R) Gold 6136 CPU @ 3.00GHz
- NDN consumer from the N-DISE project. Allow 2 threads for each consumer application and launch 6 consumers simultaneously.
- Congestion control: fixed window size, 8192 packets
- NDN-DPDK forwarder: 3 forwarding threads and use 4 name prefix components to split the packets flow.
- Use 6 1GB files, and evenly allocate them to the 3 forwarding threads. We cache them at dtn-098 in advance and then request them from dtn-103

File names

File1: /ndnc/ft/120000/F49B50CE-F3BD-E811-8430-008CFA165FD0.root
File2: /ndnc/ft/270000/A883DD63-5EBD-E811-8361-0CC47A57CBF8.root
File3: /ndnc/ft/100000/4E49B01C-D6BC-E811-83FD-B496910A9088.root
File4: /ndnc/ft/270000/54F819E1-A1BD-E811-824B-0CC47AA53D60.root
File5: /ndnc/ft/100000/BCD03B8D-FDBC-E811-85AF-B496910A80F0.root
File6: /ndnc/ft/100000/A6CF62BA-1ABD-E811-8733-008CFA1C645C.root

File allocation



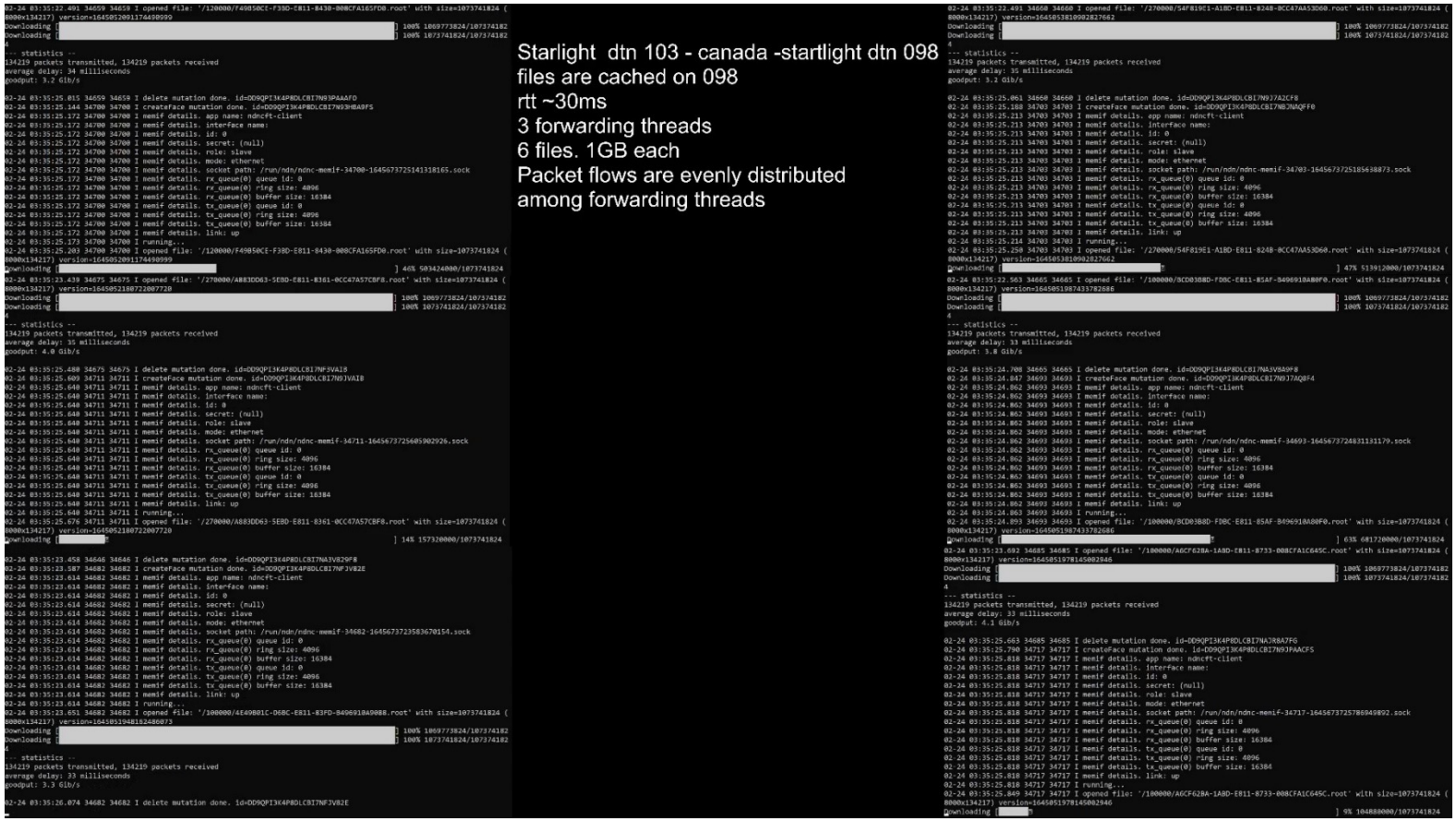
Throughput Test Results

Record for 6 minutes
Averaged throughput(Gbps):

3.5
3.5
3.4
3.4
3.4
3.5

Total:
20.8 Gbps

Max:
~21.6 Gbps



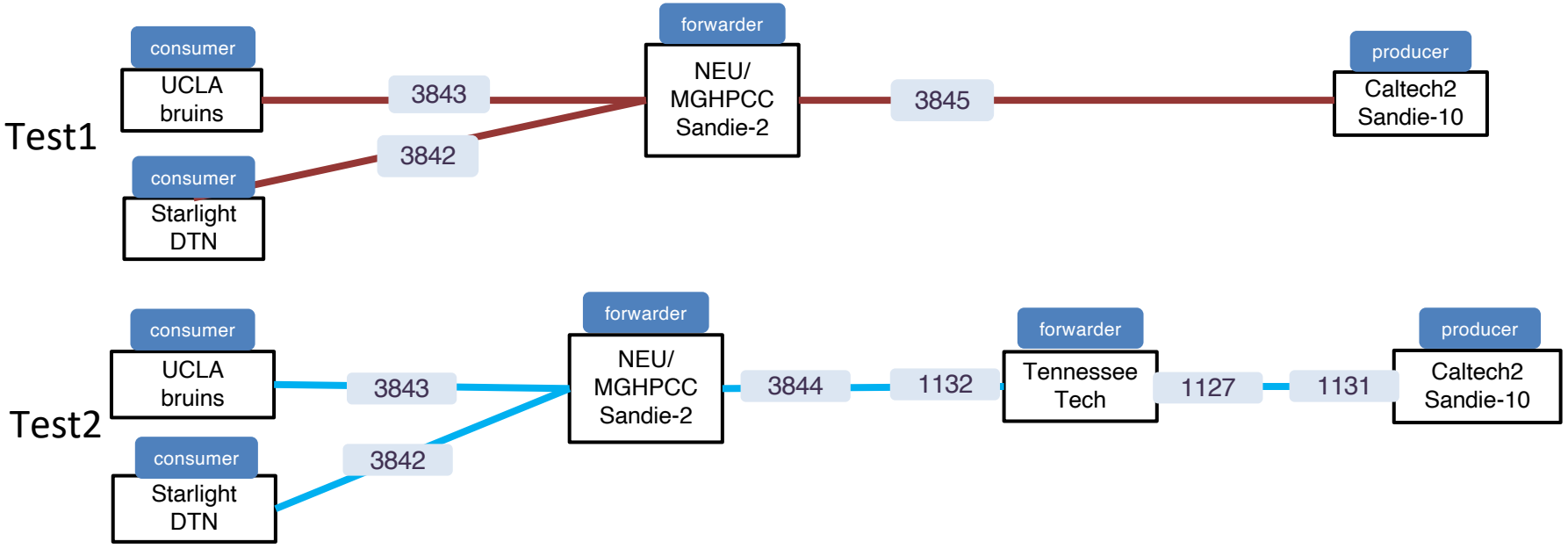
Northeastern

N-DISE Update

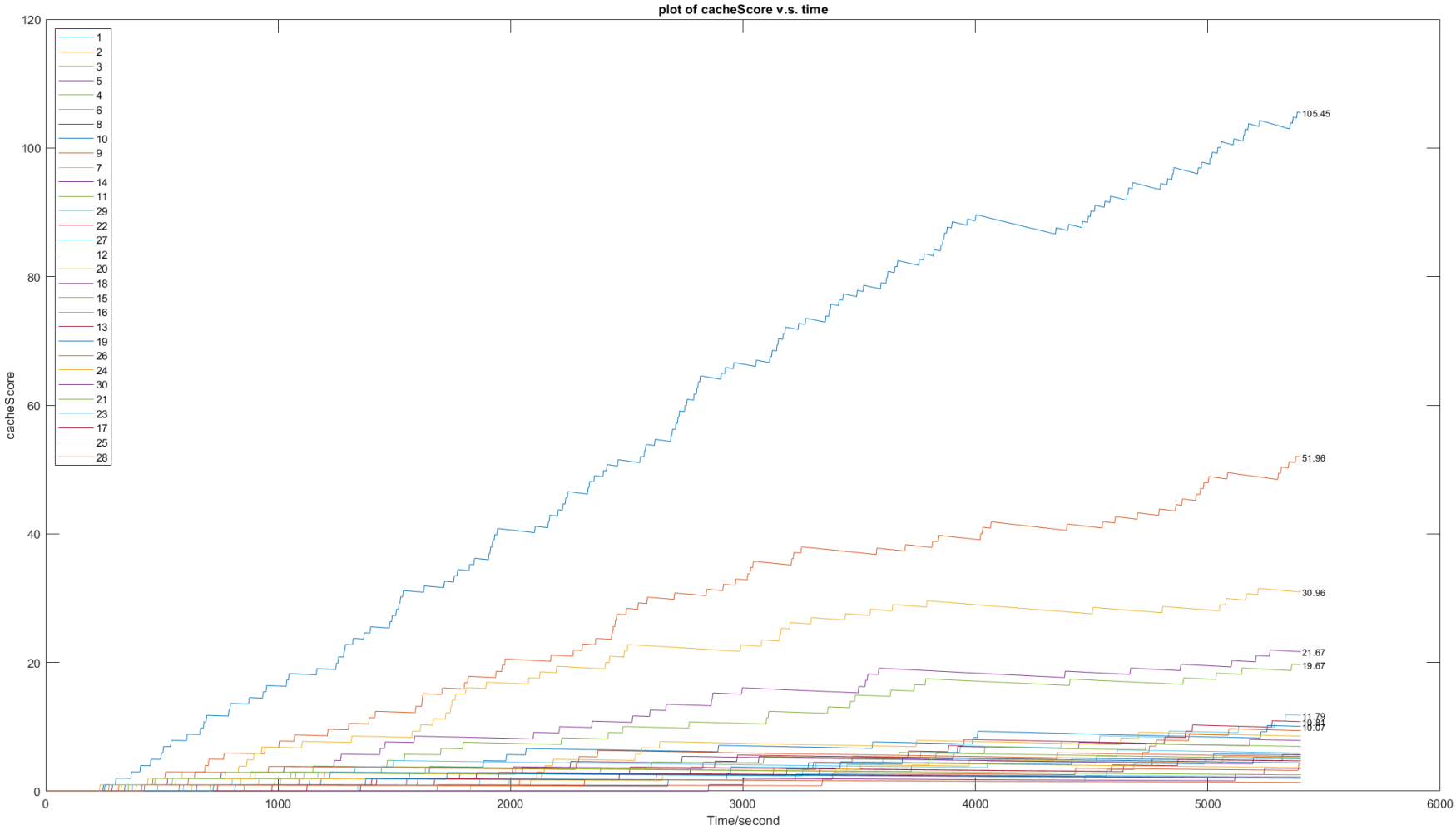
- N-DISE Deployment Architecture and NDNc
- WAN Testbed and Throughput Test
- **Optimized Caching and Forwarding**
- Congestion Control
- FPGA Acceleration

VIP Caching Test

- Test1: 2 consumer 1 forwarder
- Test2: 2 consumer 2 forwarders
- In all tests, we request 30 files based on zipf distribution.
- Each file is 4 GB
- In all tests, only the forwarder nodes can cache files and each node can cache 5 files.

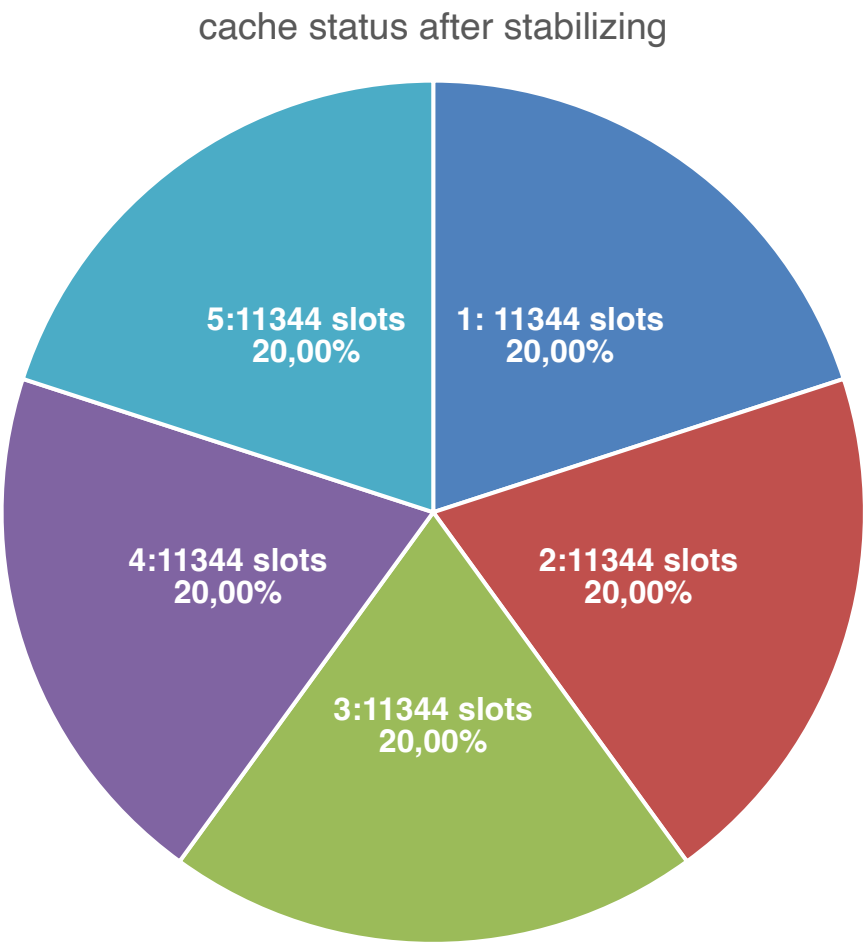


Test1: Cache Scores at Forwarder Node

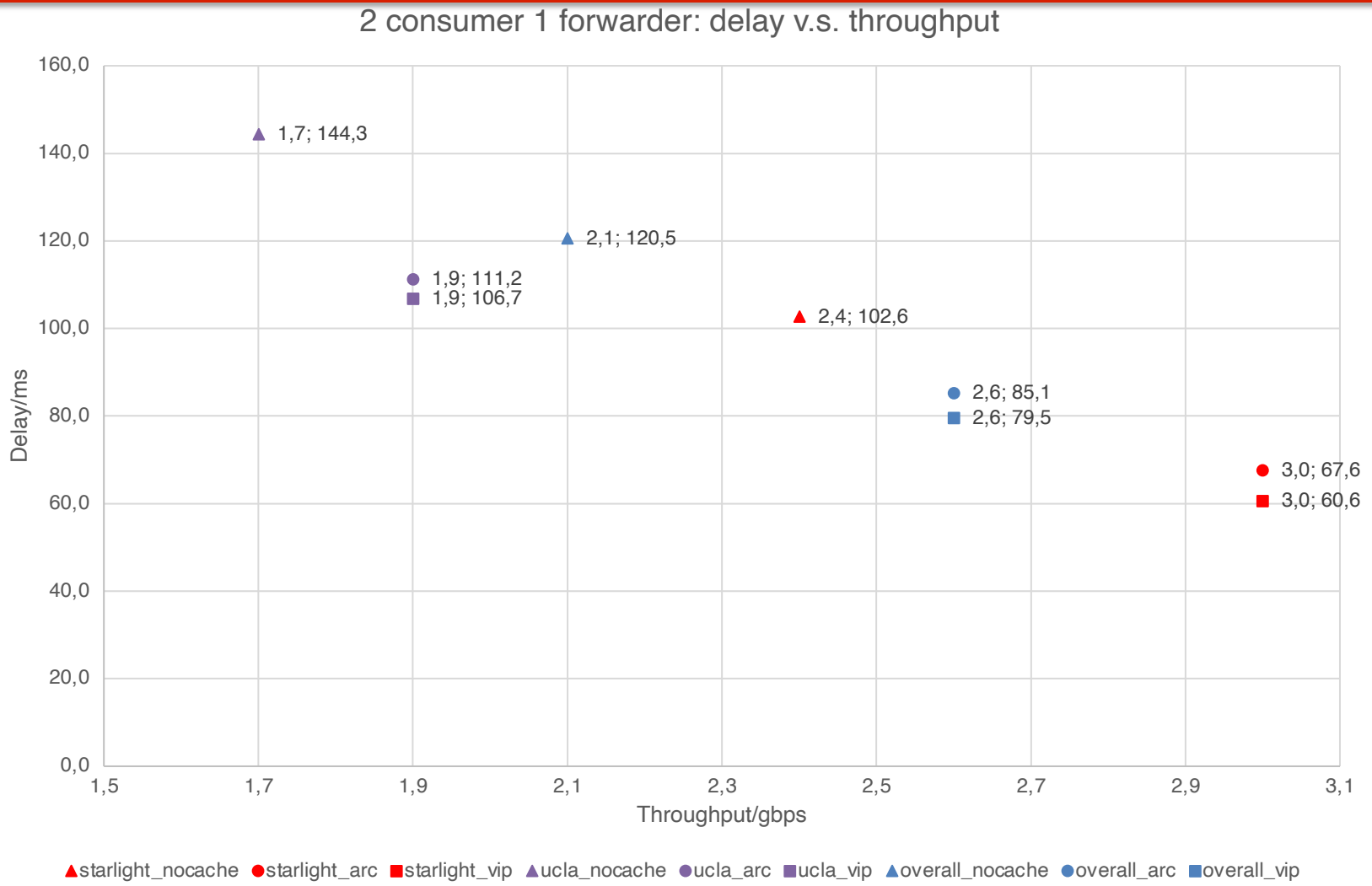


Northeastern

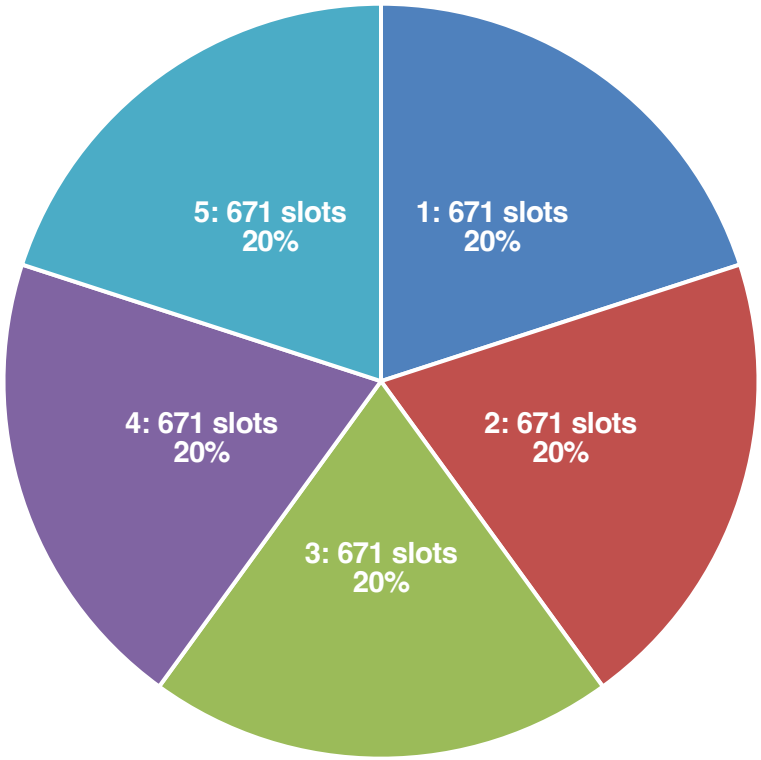
Test1: Cache Status at Forwarder Node



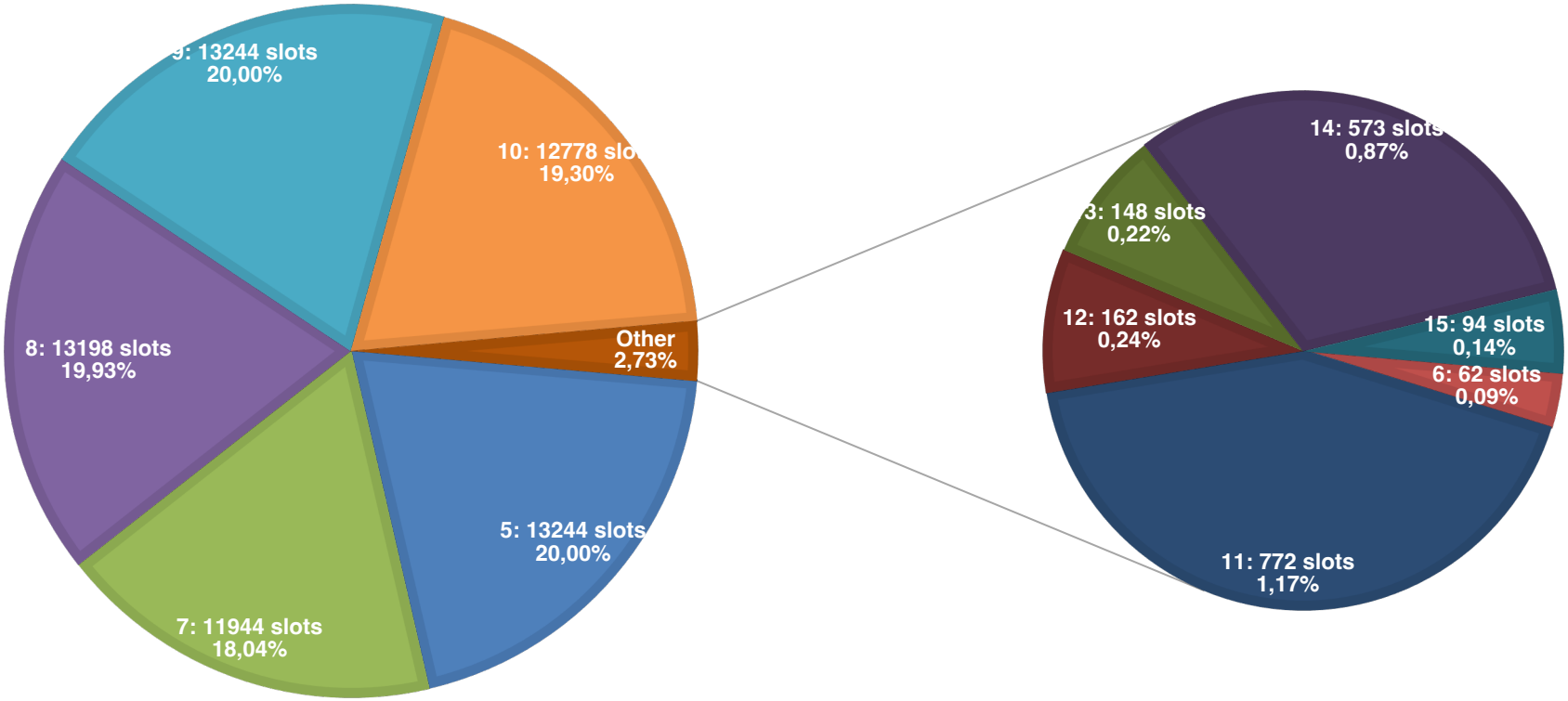
Test1: Delay/Throughput Results



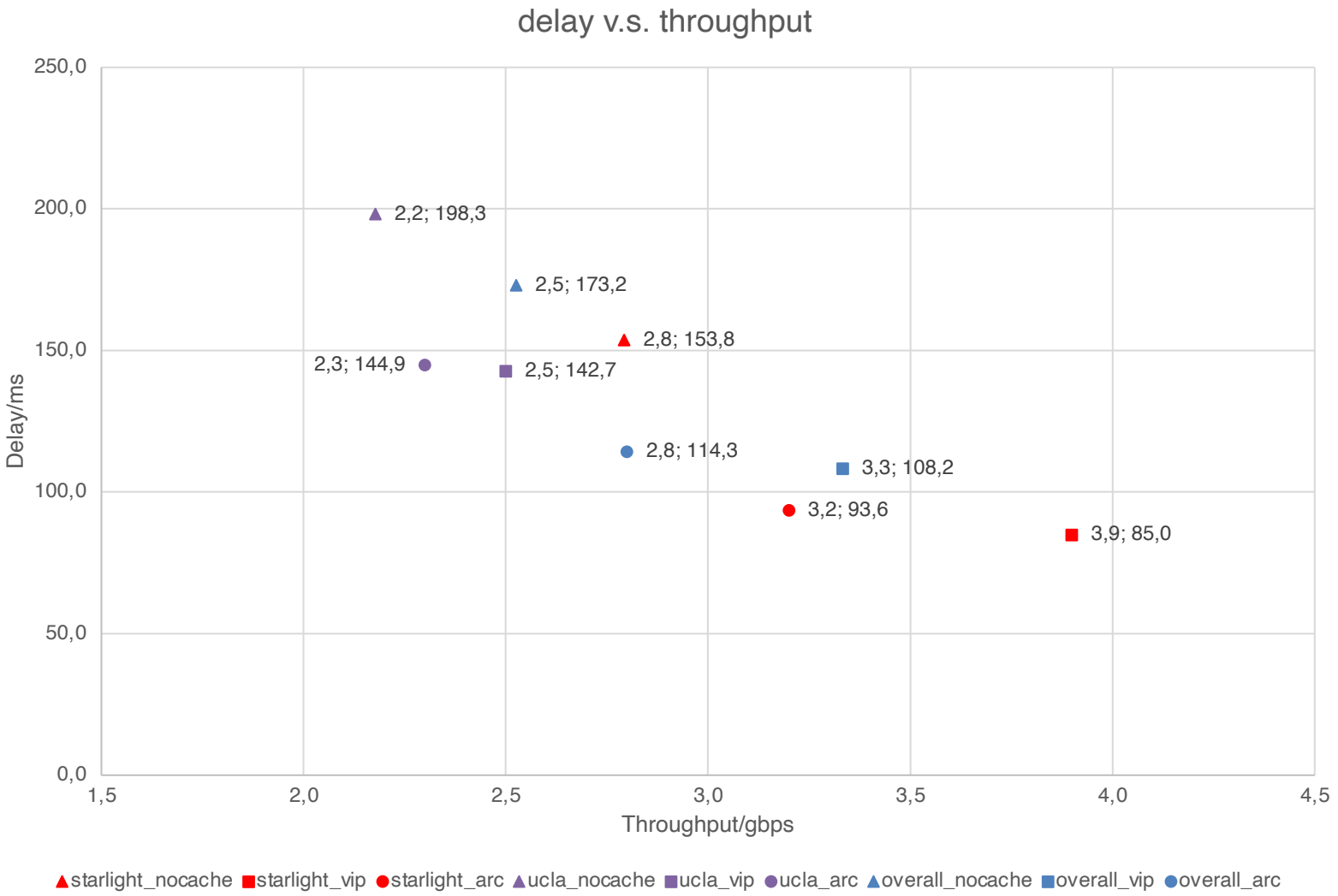
Test2: Cache Status at NEU Forwarder



Test2: Cache Status at Tenn Tech Forwarder



Test2: Delay/Throughput Results

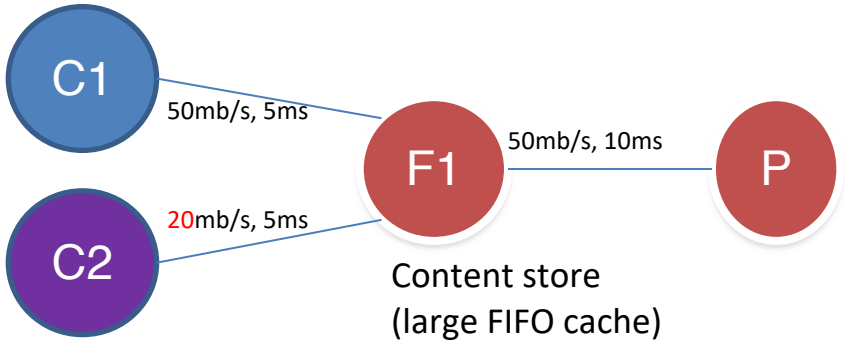


N-DISE Update

- N-DISE Deployment Architecture and NDNc
- WAN Testbed and Throughput Test
- Optimized Caching and Forwarding
- **Congestion Control**
- FPGA Acceleration

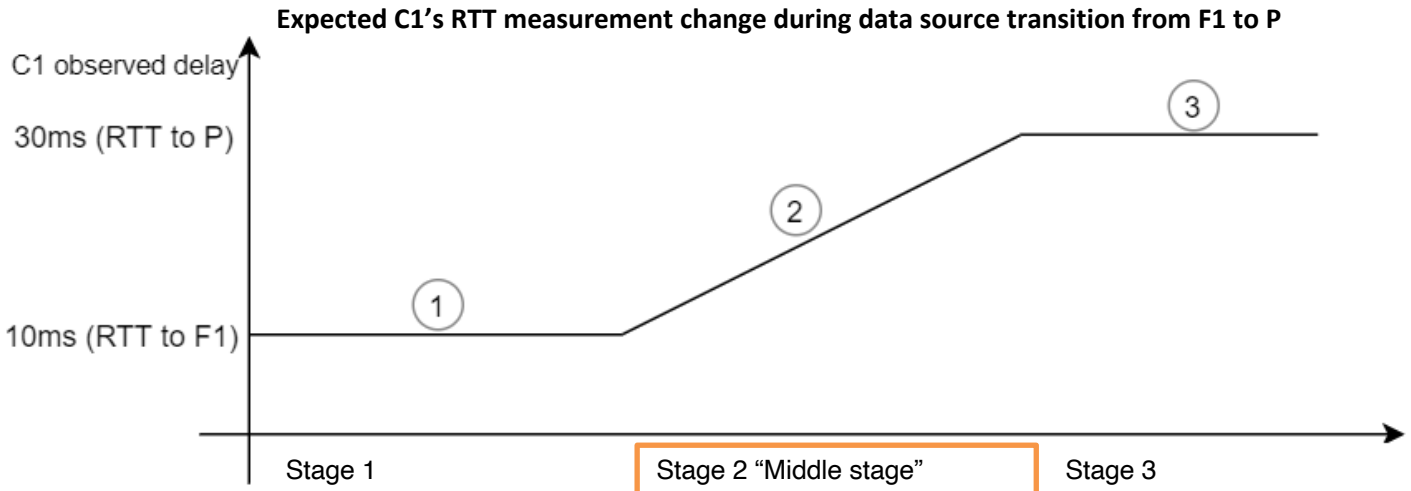
Impact of caching on congestion control

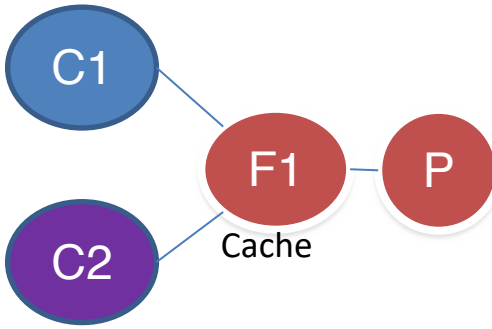
- Multiple NDN consumers fetch data from the same producer: observe impact of caching and interest aggregation on consumer congestion control mechanism
- Scenario:
  - Consumers C1,C2 fetching same segmented object. C2 started first



- Expectation: C1 will initially be satisfied by cache and later catch up with C2 and be satisfied by producer. (note C1 has higher bandwidth to producer)
- Observed in simulation:
  - C1 may never catch up with C2 or be satisfied by producer. Instead, it will continue be satisfied by cache and C2 solicited data in steady state
  - C1 is receiving data at ~20mb/s, much slower than its bandwidth to producer

Impact of caching on congestion control

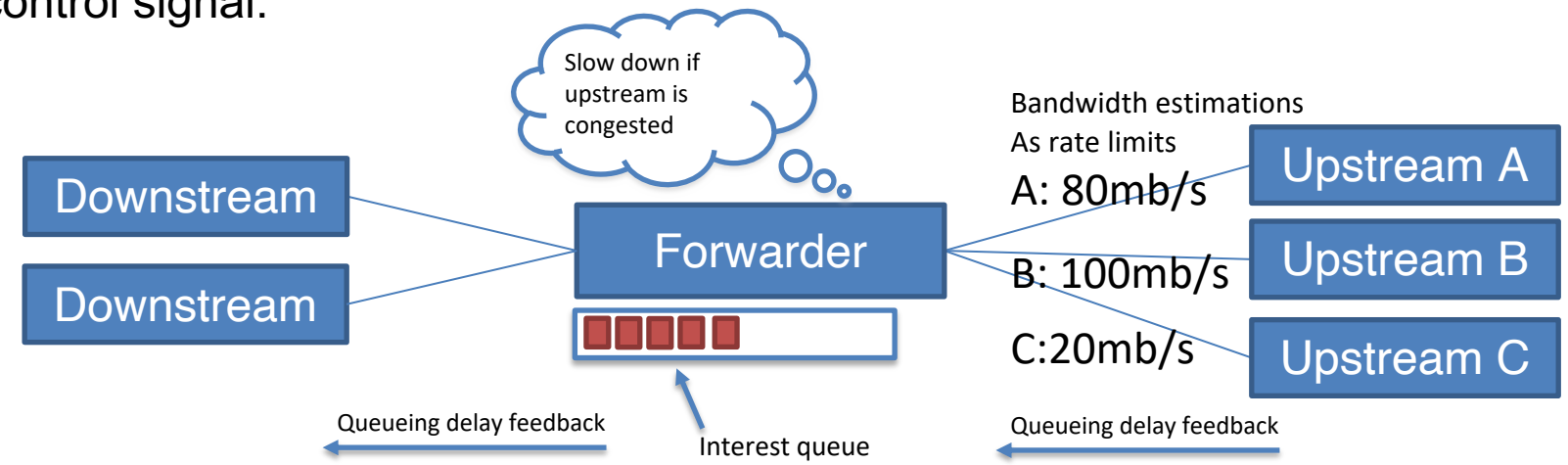




- C1 can fully utilize bandwidth in stage 3. However, C1 may misinterpret its local measurements as congestion in stage 2 and give up bandwidth before stage 3.
- Observation: consumers' local congestion control measurements must be resilient to RTT variations.

Multipath Interest Forwarding

- NDN's stateful forwarding → loop detection → freely utilize multiple paths
- Goal: enable each node to split interests among multiple next hop to maximize over all throughput
- Design: A hop-by-hop congestion control design using queueing delay as control signal.



- Simulated a proof-of-concept implementation in ndnSIM
  - With single consumer: achieved high utilization of multipaths
- Continue on improving queueing delay and stability
- Investigating the integration of multipath forwarding with caching

N-DISE Update

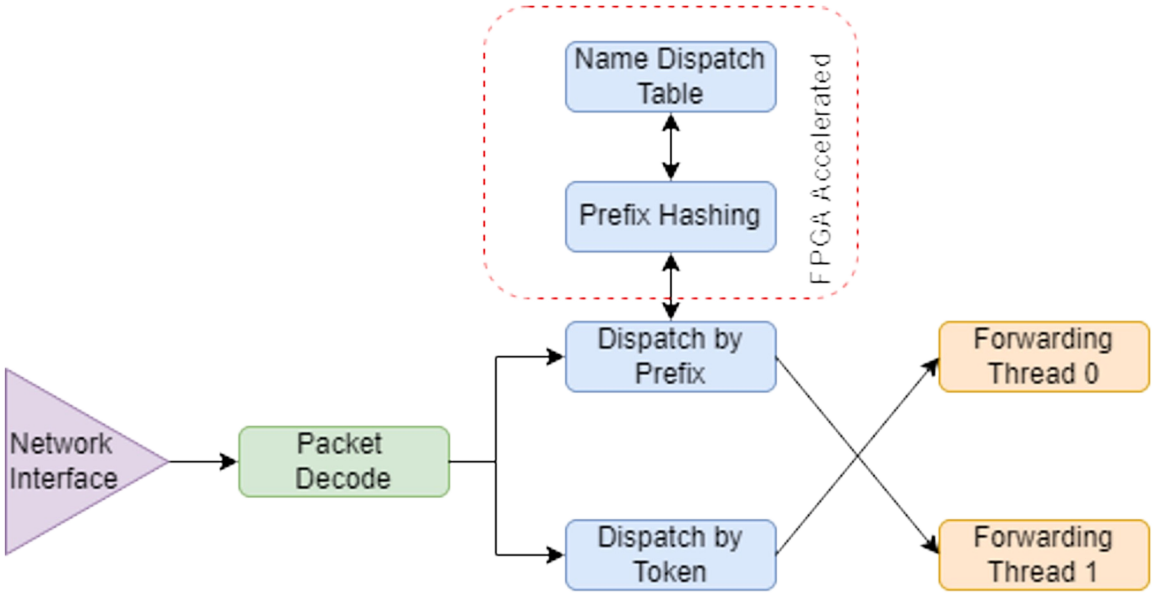
- N-DISE Deployment Architecture and NDNc
- WAN Testbed and Throughput Test
- Optimized Caching and Forwarding
- Congestion Control
- **FPGA Acceleration**

FPGA Acceleration

- FPGAs are used in a wide variety of applications
  - Machine learning
  - Networking
    - IP longest prefix matching
    - Packet inspection for firewalls
  - and many more!
- What makes FPGAs fast?
  - Pipelining of tasks
    - Ex: Each cycle can start an IP lookup
  - Parallel processing
    - Ex: Multiple interfaces have its own processing block instead of sharing one

FPGA for NDN

- Goal is to use a FPGA in forwarder input stage
  - Components in the interest name are hashed
  - There can be many components in the name
    - Hashes are computationally expensive



FPGA Progress

- FPGA Milestone:
  - Hashing of prefixes in named components
  - Table lookup for thread dispatching
  - Preliminary results show 4x improvement over CPU
    - Compute hash and lookup in a standalone environment
- In Progress:
  - Integration with NDN-DPDK
- To Do:
  - Acceleration for PIT+CS table
    - ex: Longest prefix match to find which interface to forward

Conclusions

- Data-intensive science applications require fundamental network/systems solutions to address common needs
- NDN provides data-centric system support through whole data lifecycle -
  - natural fit for LHC, genomics and other data-intensive applications
- High performance N-DISE deployment architecture with containerization and integration with NDN-DPDK using NDNc
- Established high-performance N-DISE WAN testbed
- Obtained throughput ~ 21 Gbps over WAN testbed
- VIP optimized caching and forwarding yields simultaneous delay and throughput improvement over WAN testbed

Conclusions

- Development of hop-by-hop congestion control based on queueing delay: interactions with caching
- FPGA acceleration of hashing of name prefixes and table lookup for thread dispatching shows 4X improvement over CPU
- Working toward first prototype production-ready NDN system: integration with SDN, FPGA, containerization
- Seeking long-term collaboration with domain science, networking and computer systems communities