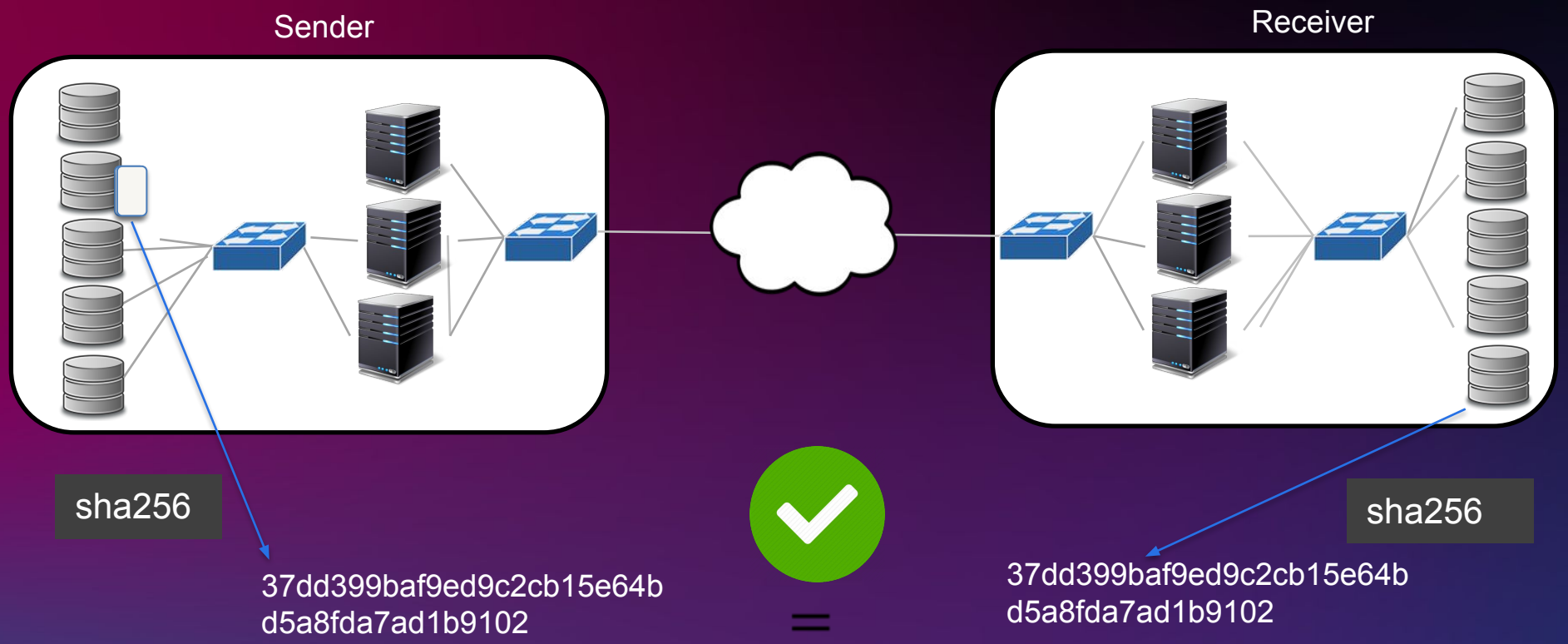
In-Network Caching Assisted Error Recovery For File Transfers

Nagmat Nazarov and Engin Arslan
University of Nevada, Reno

Silent data corruption

- Silent data corruption : It materializes as bit flips in storage (both volatile memory and non-volatile disk) or even within processing cores.
- Wide area file transfers are susceptible to silent data corruptions.
- Data can be corrupted during disk read and write operations.
- 5% of all file transfers in research networks are exposed to silent errors.

End-to-End Integrity Verification



- End- to-end integrity verification minimizes the likelihood of silent data corruption by comparing checksum of files at source and destination servers using secure hash algorithms such as MD5 and SHA1.

Issues with end-to-end integrity verification

- End-to-end integrity verification inevitably creates overheads (extra disk I/O and more computation). Thus, the overall data transfer time increases.
- Load on the network increases and consumes significant network bandwidth.
- Receiver need to get the original file all the way from the sender.

In-Network caching is the solution

- Taking advantage of caching and computing resources deployed inside the network to expedite the error recovery for file transfers.
- Storing files in cache servers in network along the transfer path.
- Lowers the duration of error recovery when available bandwidth between caching site and destination is larger than available bandwidth.
- Selectively choose which network flows to mirror and save in the caching server.

Related Work

Related Work

- Content Distribution Network (CDN) achieve this goal by deploying many cache servers close to end users to store popular content such that they can be delivered to users quickly.
  - Not designed for large scientific datasets
- Named Data Networking (NDN) proposed content-centric networking design to store popular data on network devices (e.g., routers) such that users can access them quickly without the need to contact end hosts.
  - Requires fundamental changes in the network infrastructure

Related Work

- Globus transfer service implements integrity verification for file transfers and overlaps transfer and checksum compute operations to minimize the overhead.
- Divide large files into blocks to improve pipelining (i.e., block-level pipelining) of transfer and compute tasks for mixed-size datasets. (B. Allen et. al . Communications of the ACM'12)

- RIVA - **Robust** Integrity Verification Algorithm - Enhance resilience of end-to-end integrity verification for data transfers by enforcing checksum calculations to read files directly from disk.
- Extreme fault injection technique to reproduce undetected disk write errors.
- Calculate the likelihood of silent data corruption.

In-Network Caching Architecture.

- Files are transferred from the sender to receiver.
- Data packets are mirrored to cache server located near the programmable switch.
- The packets are captured at the mirror site and processed to reconstruct the file.

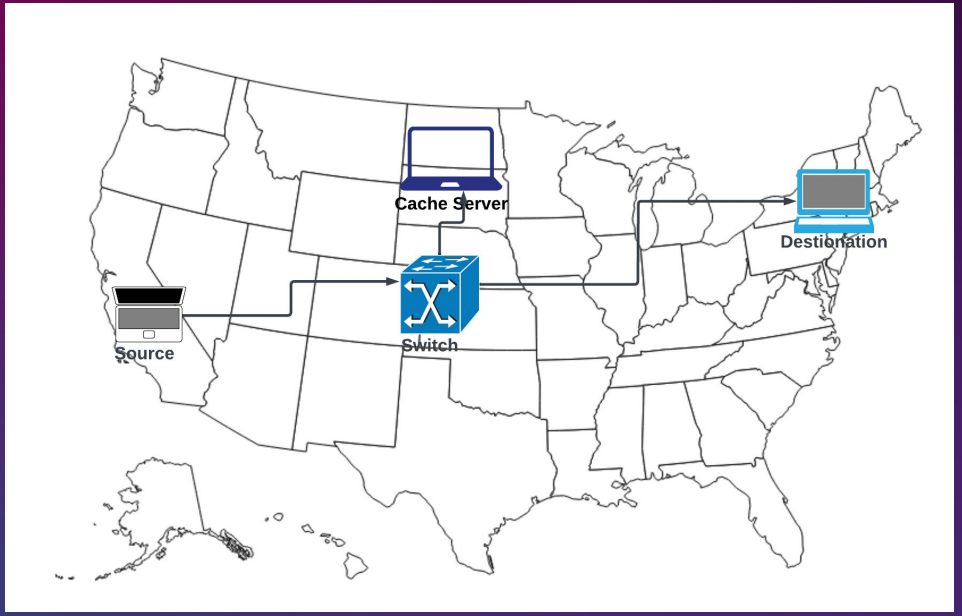


Figure 2: In-network caching architecture

Contributions

- An in-network caching mechanism for file transfers.
- Leverage the programmability of switches to mirror data packets according to certain **flow** and **size**.
- Process the captured packets in the cache server to reconstruct the files.
- Retrieve files from cache servers when an error is detected to minimize error recovery time.
- The proposed method leads to 50% decrease in error recovery process and 20% decrease in total transfer time of file transfers.

Mirroring

- OpenFlow switches are equipped with a fixed set of functionalities which requires new ASIC.
- A packet can be mirrored to a port in addition to the set of targets intended by the packet's source.
- The Mirror ID field in metadata field is used to indicate whether or not a packet will be mirrored.
- The Mirror Buffer steals unused cycles from the Input Pipeline Deparser in order to copy its packets.

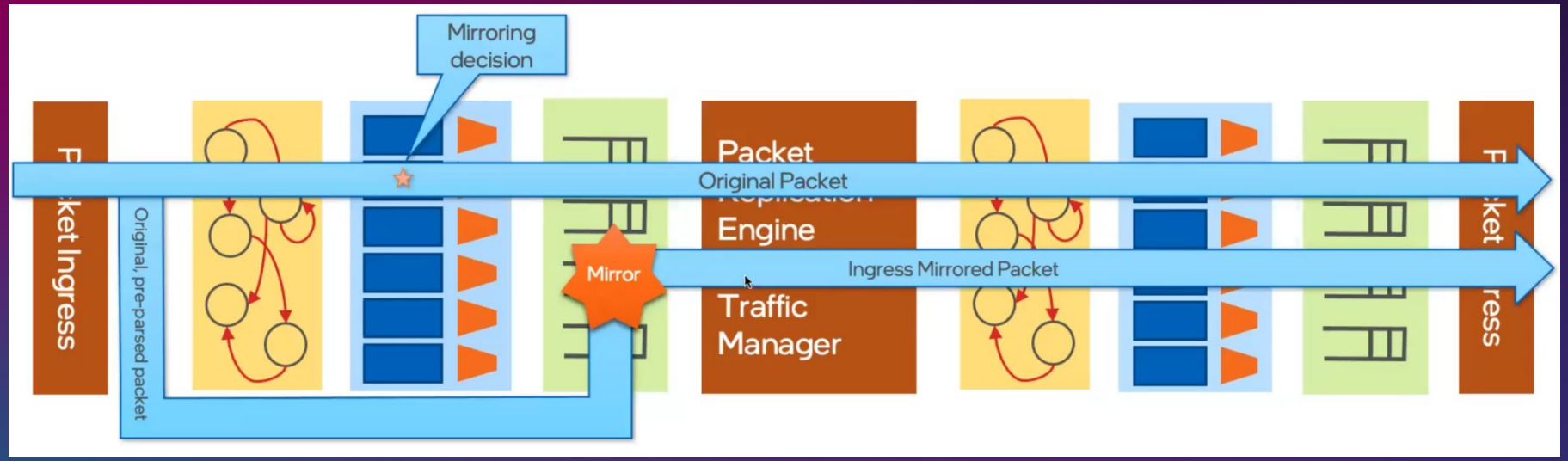


Figure 3: Ingress Mirroring mechanism in Intel Tofino 1.

Flow and **Size** based Mirroring

- Implemented **Flow** and **Size** Based Mirroring.
- Intel Tofino1 Supports up-to 7 Mirror Sessions.
- IP based flow instead of Port based flow. (Host A to Host D are mirrored based on flows(IP))
- We exploited **Registers externs** to limit by size in packets.

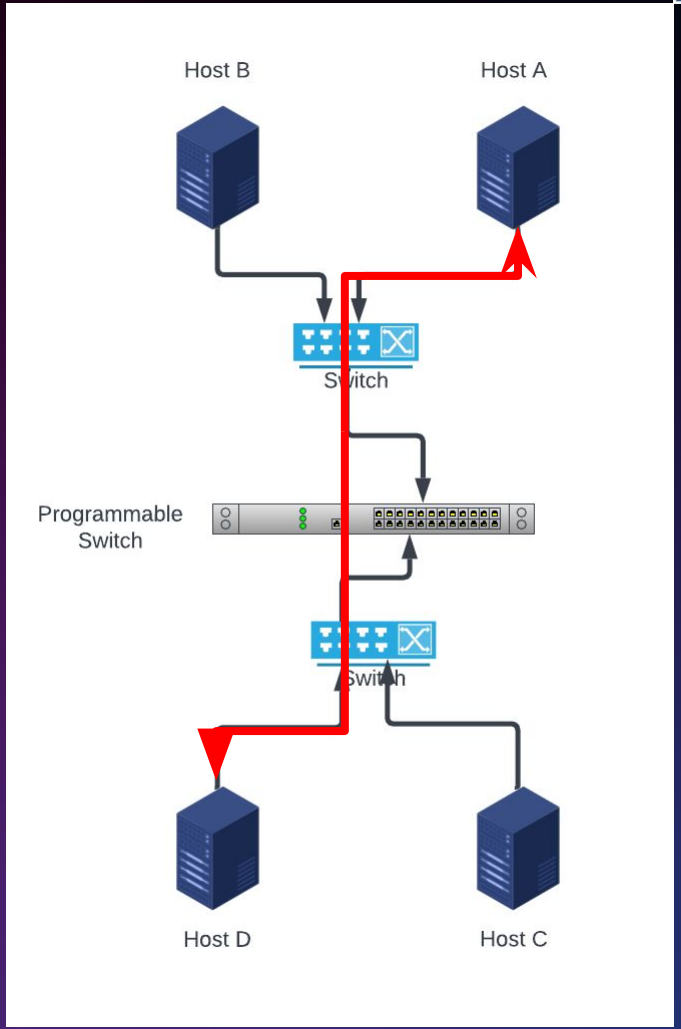


Figure 4: Flow based mirroring mechanism

How to reconstruct the file at mirror site?

- Cache servers receives packets from programmable switch in raw format.
- High-performance data capturing and processing pipeline to reconstruct files from mirror traffic.
- Tcpdump to capture mirrored packets on the cache server.
- dpkt can be used to process raw packets as they are being captured by tcpdump to avoid incurring additional delay.

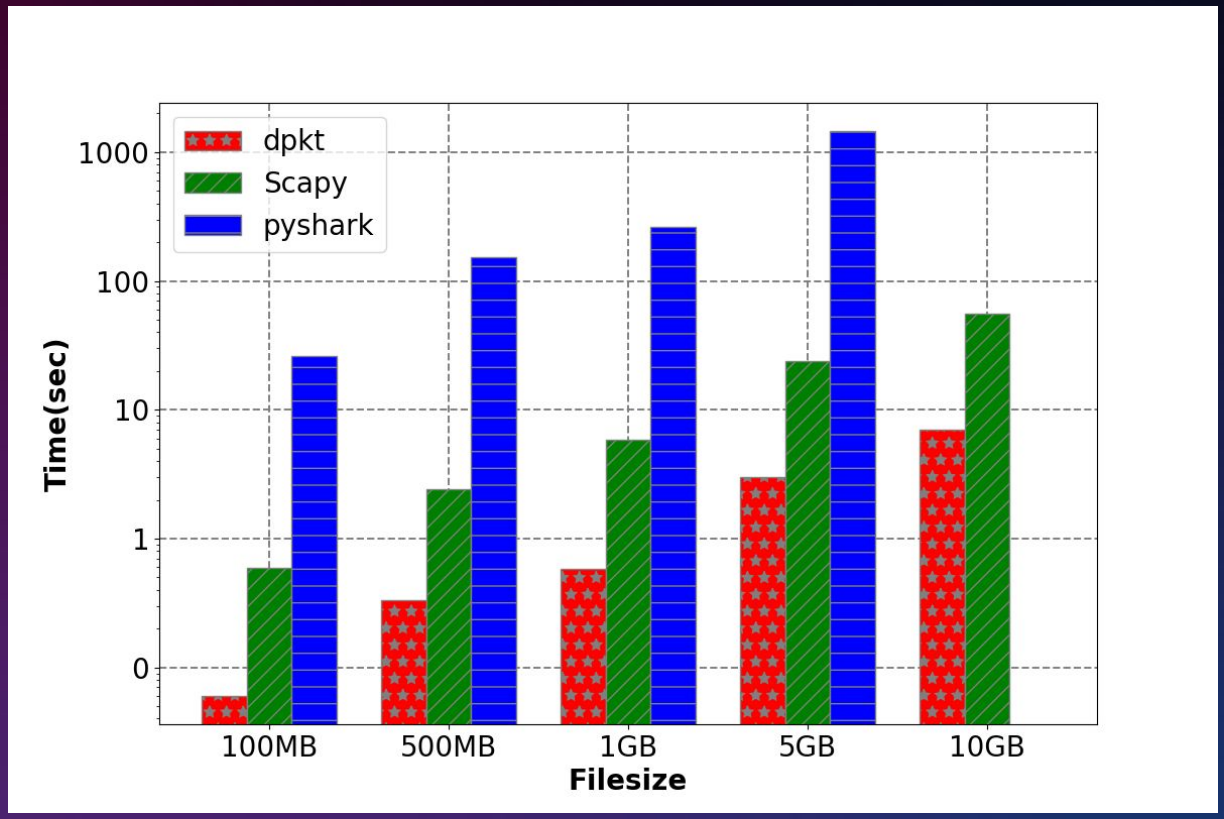


Figure 5: Packet processing libraries by speed.

Experimental Results

Evaluation

- Hardware (EdgeCore Wedge100BF-32QS)
  - 32 × 100G ports
  - 16 Core Intel x86 Broadwell-DE
  - Pentium D-1517 processor
  - 128 GiB SSD storage.

- The source, destination, and cache servers are connected to the EdgeCore switch with 1G, 10G, and 10G interfaces.

- Workloads : Files from 100 MiB to 20 GiB.

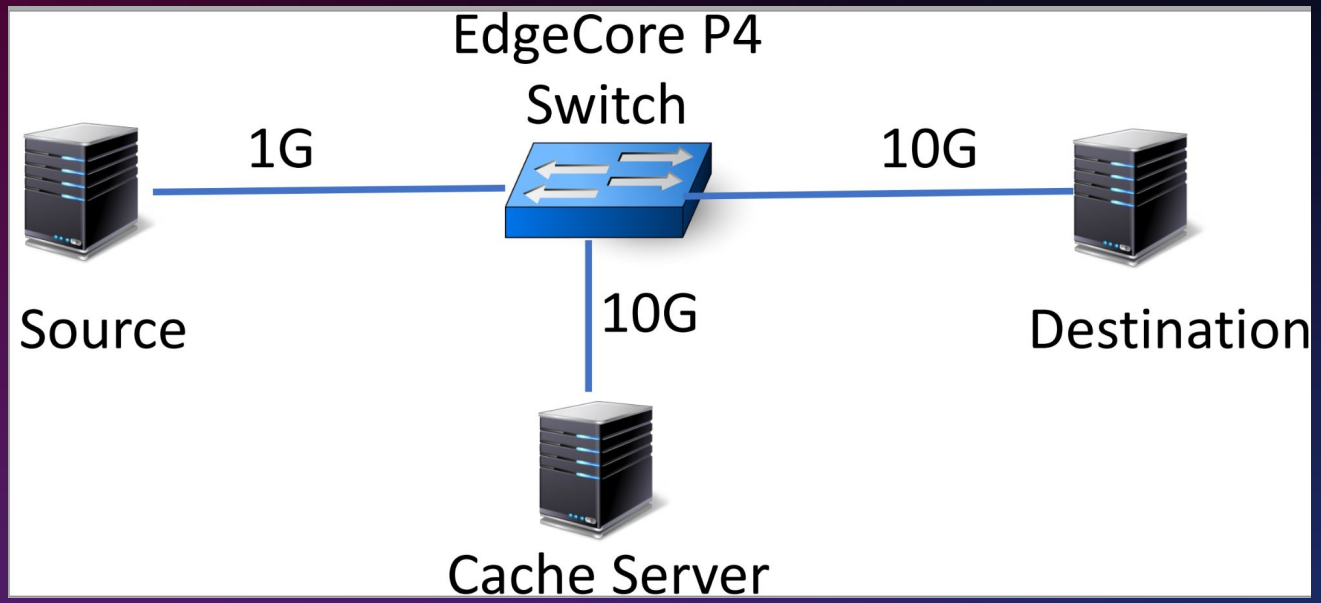


Figure 6: Experimental Setup.

Evaluation

- First transfer files from source to destination and in the case of checksum mismatch, we resend files from the source to destination to recover from silent errors.
- MD5 to calculate file hashes and run integrity check between source and destination nodes.
- For all file sizes, recovery time is shortened by around 50%.

File Size	Regular	In Network Caching	Improvement (%)
100MB	2.08	1.66	55
1GB	21.29	16.31	48
5GB	105.47	81.56	50
10GB	233.43	176.45	46
20GB	468.89	357.27	47

Table 1 : COMPARISON OF ERROR RECOVERY TIMES WITH AND WITHOUT (REGULAR) IN NETWORK CACHING IN THE PRESENCE OF SILENT ERRORS.

Evaluation

- Total transfer time is reduced by 20% with the help of in-network caching.
- Although the bandwidth between the cache server and destination node is 10x higher than the bandwidth between the source and destination nodes, we only observe 2x improvement in transfer times.

File Size	Regular	In Network Caching	Improvement (%)
100MB	2.08	1.66	20
1GB	21.29	16.31	23
5GB	105.47	81.56	22
10GB	233.43	176.45	24
20GB	468.89	357.27	23

Table 2 : COMPARISON OF TOTAL TRANSFER TIMES WITH AND WITHOUT (REGULAR) IN NETWORK CACHING IN THE PRESENCE OF SILENT ERRORS

Conclusion

- Silent error detection is critical for file transfers to avoid permanent data losses.
- If an error is detected, then files can be retrieved from a nearby cache server instead of downloading them from the original source.
- Programmable devices provide a configurable traffic mirroring scheme.
- The preliminary results show that in network caching reduces error recovery times by 50% and total transfer times by 20% for varying file sizes.

Future Work

- Extend the proposed method with more automation such that clients can demand only certain files to be cached instead of all files.
- Cache servers can register themselves to programmable devices to support the implementation of multiple cache servers.
- Explore various cache eviction policies to avoid overwhelming cache servers' storage space with stale data.

Thanks for attention!

- Please send your questions to nnazarov@unr.edu
- Lab Website : https://www.cse.unr.edu/~earslan/