Distributed Resilience in High-Energy Physics Data Acquisition

Alexander Wolosewicz (IL Tech), Nishanth Shyamkumar (IL Tech), Wesley Ketchum (FNAL), Eli Dart (Esnet / LBNL), Jim Kowalkowski (FNAL), Michael H. L. Wang (FNAL), Nik Sultana (IL Tech)

ILLINOIS TECH
College of Computing

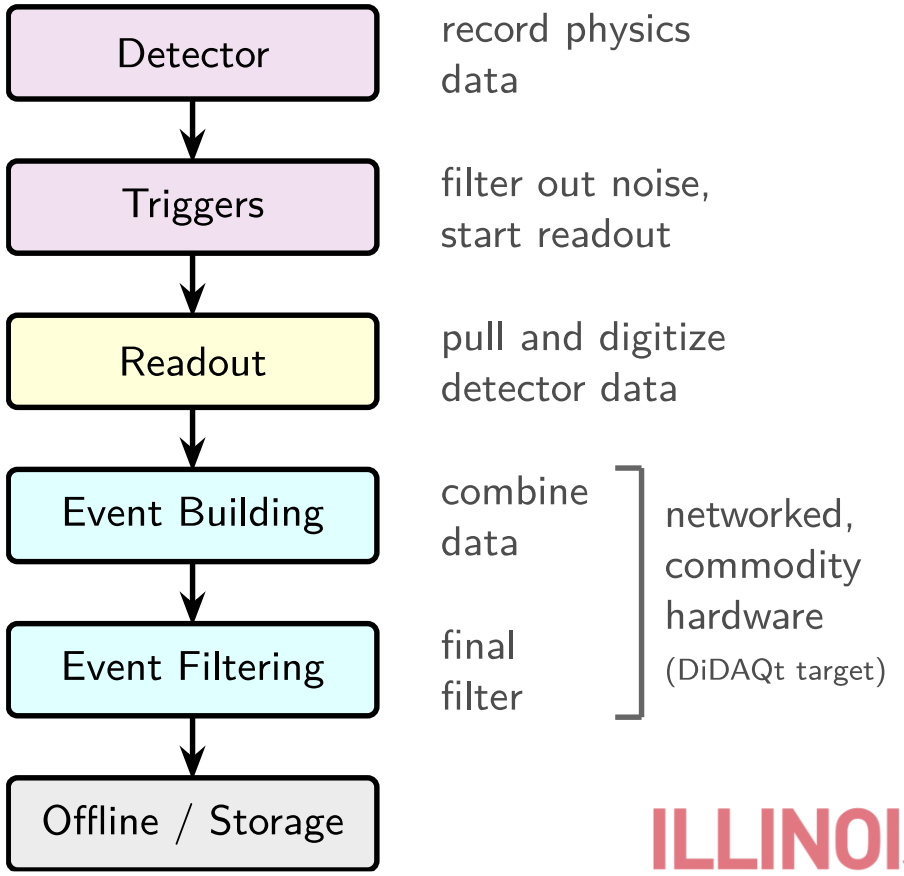
Short Bio

- 3rd year Ph.D. student
- Work focused on network debugging and observability

ILLINOIS TECH
College of Computing

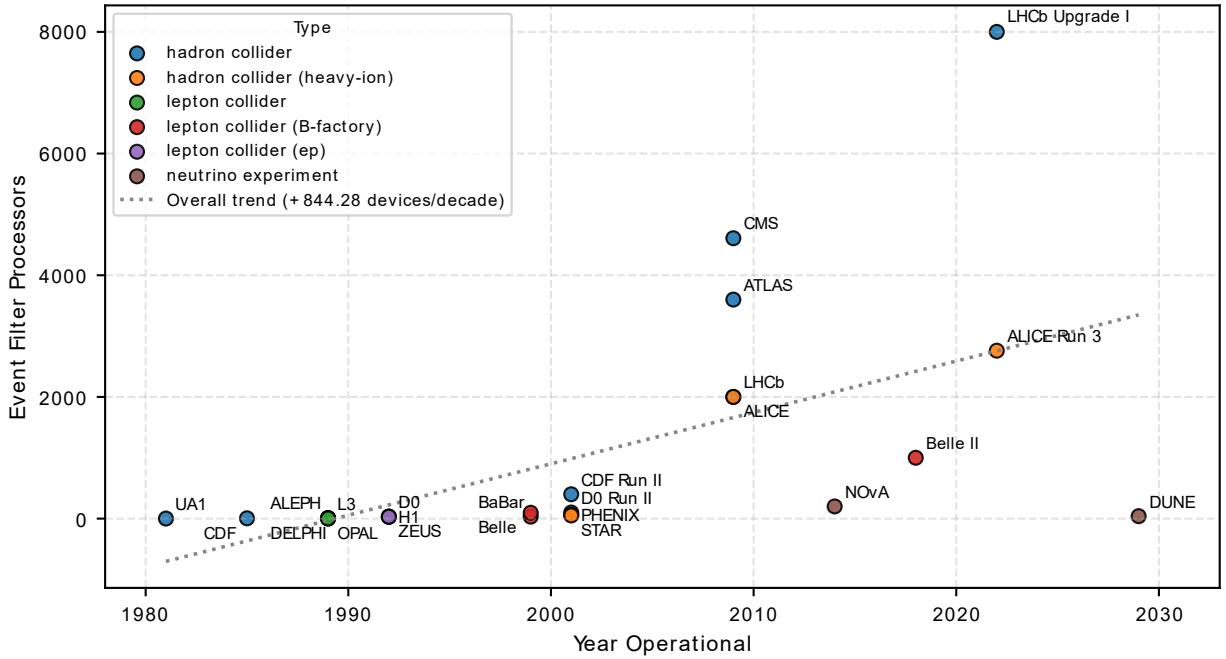
Introduction / Background

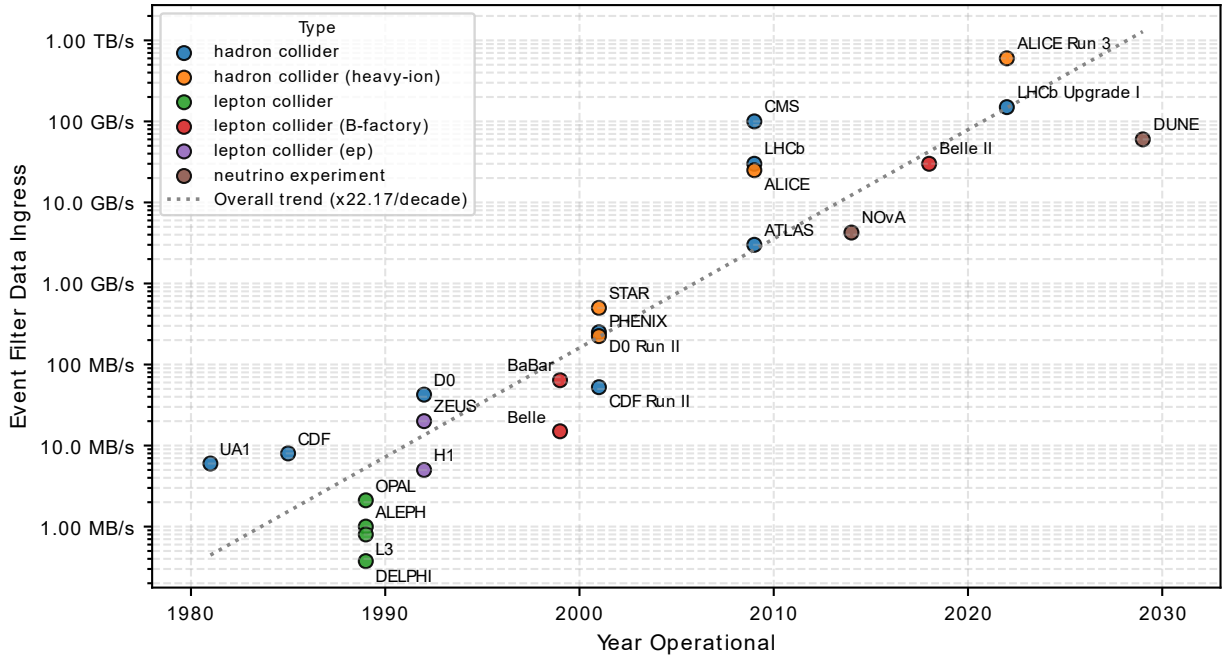
- High-Energy Physics (HEP) Data AcQuisition (DAQ) Networks
  - Responsible for filtering, processing, and storing experiment data
- Staged, unidirectional
- Historically, mix of custom and commodity hardware



ILLINOIS TECH
College of Computing

The Problem





ILLINOIS TECH
College of Computing

The Problem

- HEP DAQs are growing exponentially over time
- Current state of the practice is that failures are handled manually
  - Systems designed to gracefully degrade, but data can be lost
  - Manual resolution can be very slow
- Historical surveys* show failure rates increase with scale
- Further, data loss becomes more costly (DUNE supernovae data)
- Ultimately, automatic fail-over should be adopted for HEP DAQs

*B. Schroeder and G. A. Gibson, "A Large-Scale Study of Failures in High-Performance Computing Systems," IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 4, pp. 337–350, 2010. [Online]. Available: https://doi.org/10.1109/TDSC.2009.4

*K. M. Uddin Ahmed, M. Alvarez, and M. H. J. Bollen, "Characterizing failure and repair time of servers in a hyper-scale data center," in 2020 IEEE PES Innovative Smart Grid Technologies Europe (ISGT-Europe), 2020, pp. 660–664. [Online]. Available: https://doi.org/10.1109/ISGT-Europe47291.2020.9248891

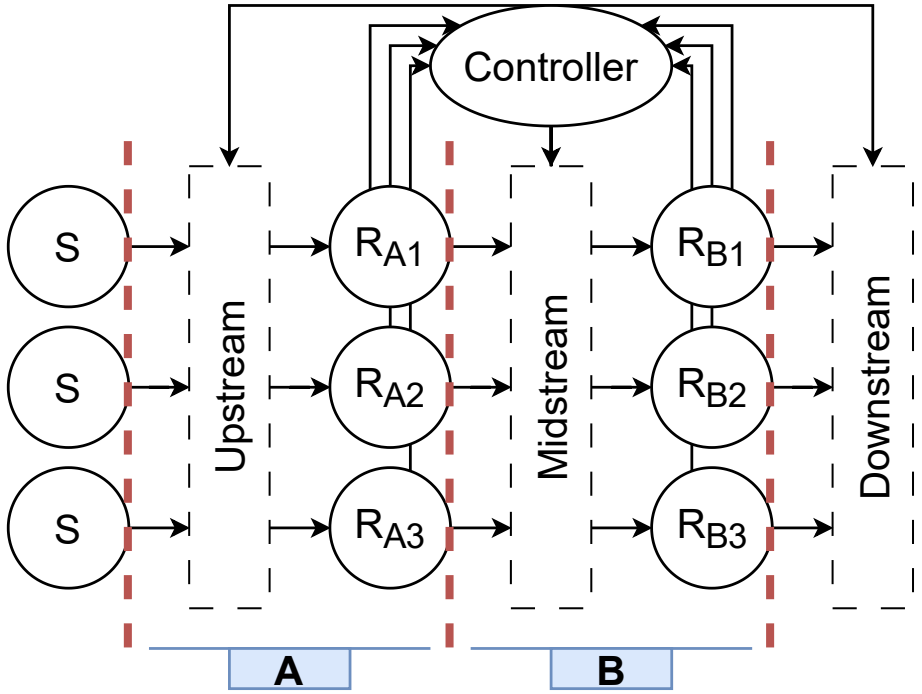
ILLINOIS TECH
College of Computing

Fail-Over in DAQs

- Much work has been done for HPC resiliency, but DAQs have:
  - Unidirectional nature
  - Heterogenous hardware, often a mix of custom and commodity
  - Limited overhead
- So, a solution must be general, light, and support unidirectional traffic

DiDAQt

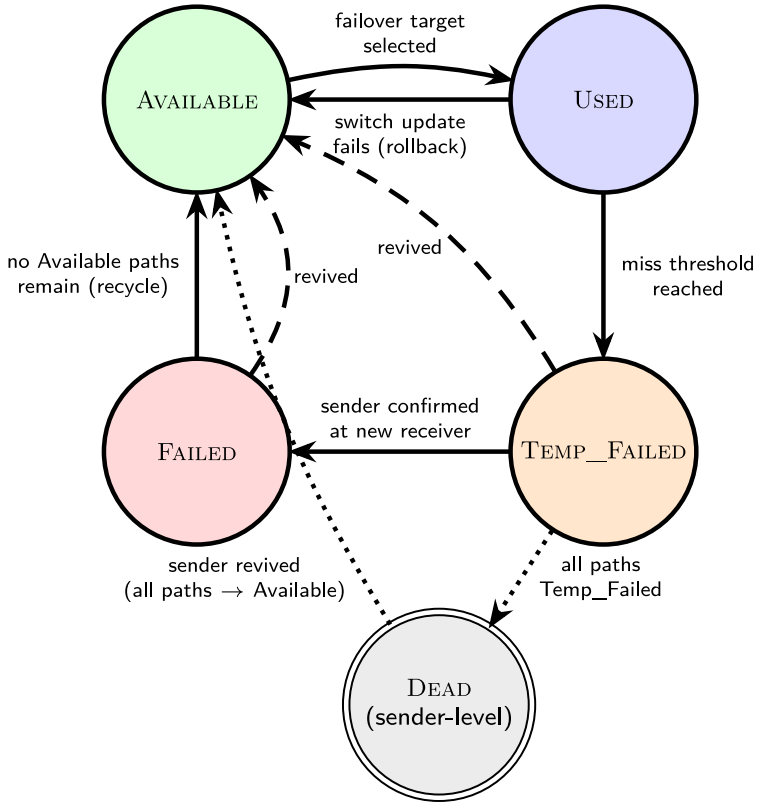
- MPI-inspired API for adding resiliency to controlled networks
- **Ablatives:** a primitive which represents switch configurations defining a source>receiver path
- API enables:
  - **R**eceiver-based failure detection
  - **C**ontroller-based fail-over



ILLINOIS TECH
College of Computing

DiDAQt Operation

- **R** use API to send heartbeats (HBs) indicating they receive healthy traffic
- **C** maintains a network state, when heartbeats vanish, triggers fail-over
- Tries fail-overs until HBs re-appear



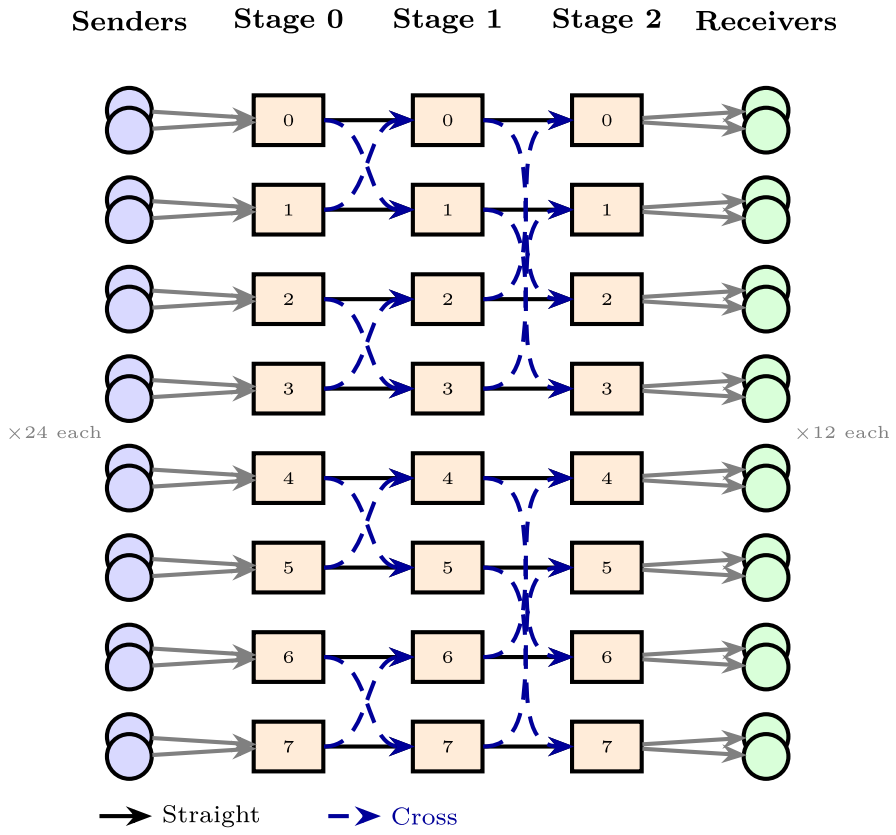
ILLINOIS TECH
College of Computing

DiDAQt API

Function	Purpose
init_ctx(r_id, &ctx)	Creates a DiDAQtcontext on a **R**eceiver that tracks heartbeat intervals and IDs.
set_controller(ctx, ip, port)	Sets the network information for the controller in the context.
start(ctx)	Launch the DiDAQtthread for the given context.
stop(ctx)	Stop the DiDAQtthread for the given context.
schedule_heartbeat(s_id, ctx)	On the next heartbeat interval, indicate that the path with a given **S**ender (`s_id`) is healthy.
deschedule_heartbeat(s_id, ctx)	On the next heartbeat interval, overwrite and prevent scheduling this **S** as healthy.
init_ctx(&ctx)	Creates a DiDAQtcontext on a **C**ontroller for tracking path statuses.
set_grace_period(ctx, grace_ns)	Set the time this context allows for a fail-over to complete before recognizing it as failed.
set_miss_threshold(ctx, threshold)	Set how many consecutive heartbeats a sender must be absent from until the controller triggers a fail-over.
set_event_callback(ctx, fn, user_data)	On an event (fail-over, sender confirmed post-failover, dead sender, revived sender), `fn` is called with the given `user_data`.
process_topology(file, ctx)	Provide topology information to a context and run initial static analysis.
register_handler(ctx, switch_type, fn, user_data)	The context will call `fn` with the `s_id` of the failed **S**, the current ingress and egress ports, the future ingress and egress ports, and the given `user_data` to change the path through a node of `switch_type`.
process_heartbeats(buf, len, ctx)	Provide a received heartbeat packet to be processed by the context.
get_path_statuses(ctx, &out, &count)	Provide a snapshot of the current path statuses, writing the number to `count` and the data to `out`.
set_path_status(ctx, path_id, status)	Manually set a path status.
revive_sender(ctx, s_id)	**S** which fail on every path are marked as failed. This restores one with the given `s_id` to be checked in heartbeat processing.

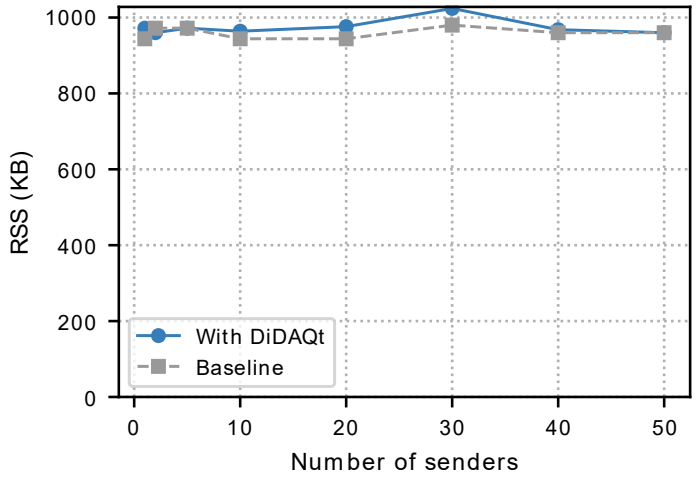
ILLINOIS TECH
College of Computing

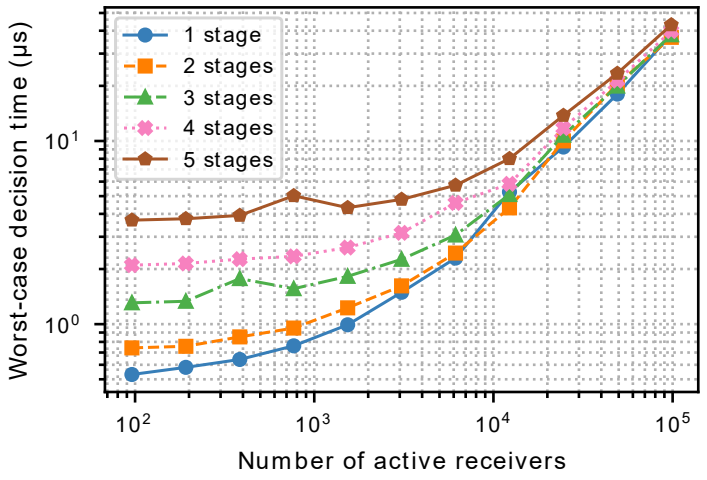
Evaluation

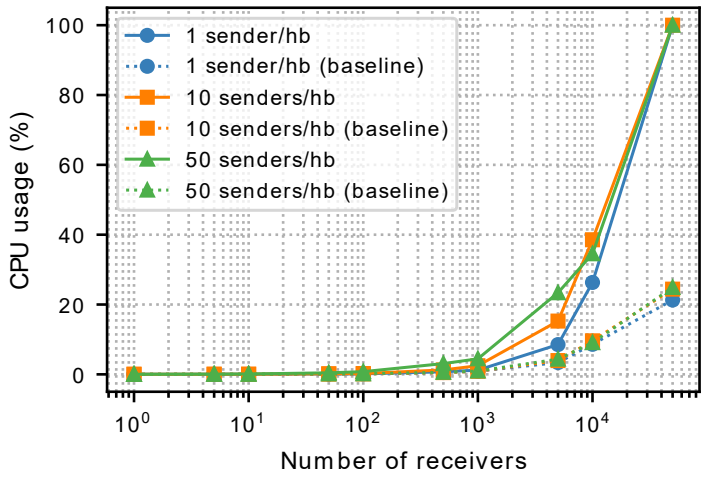


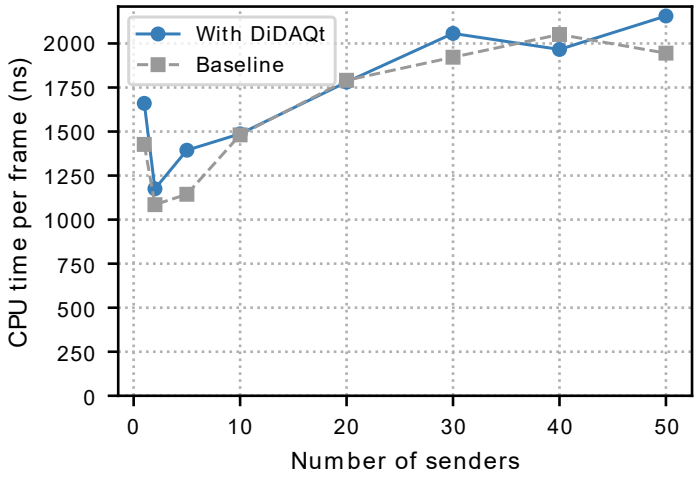
- Butterfly topology for simulation
  - Easy to stress the system
  - Very high redundancy
  - Many paths
- FABRIC Testbed
- Virtual Machines with 8 CPU cores, 32 GB RAM, Ubuntu 22.04

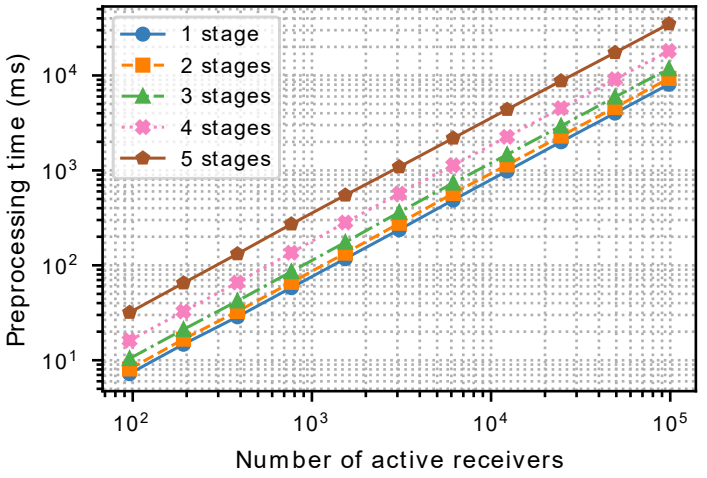
ILLINOIS TECH
College of Computing

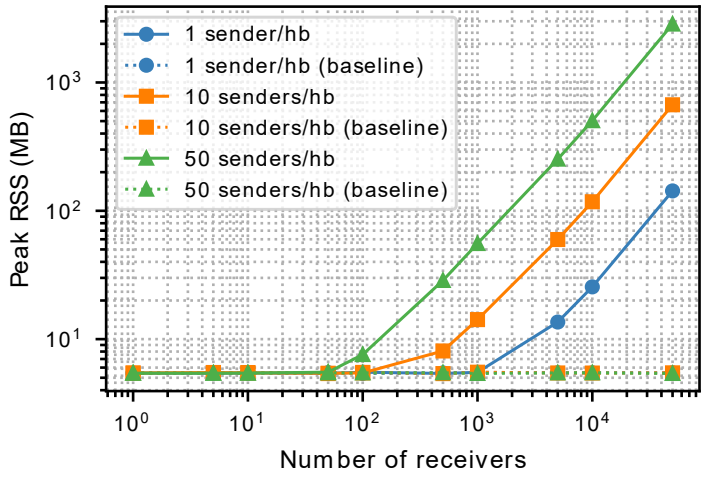






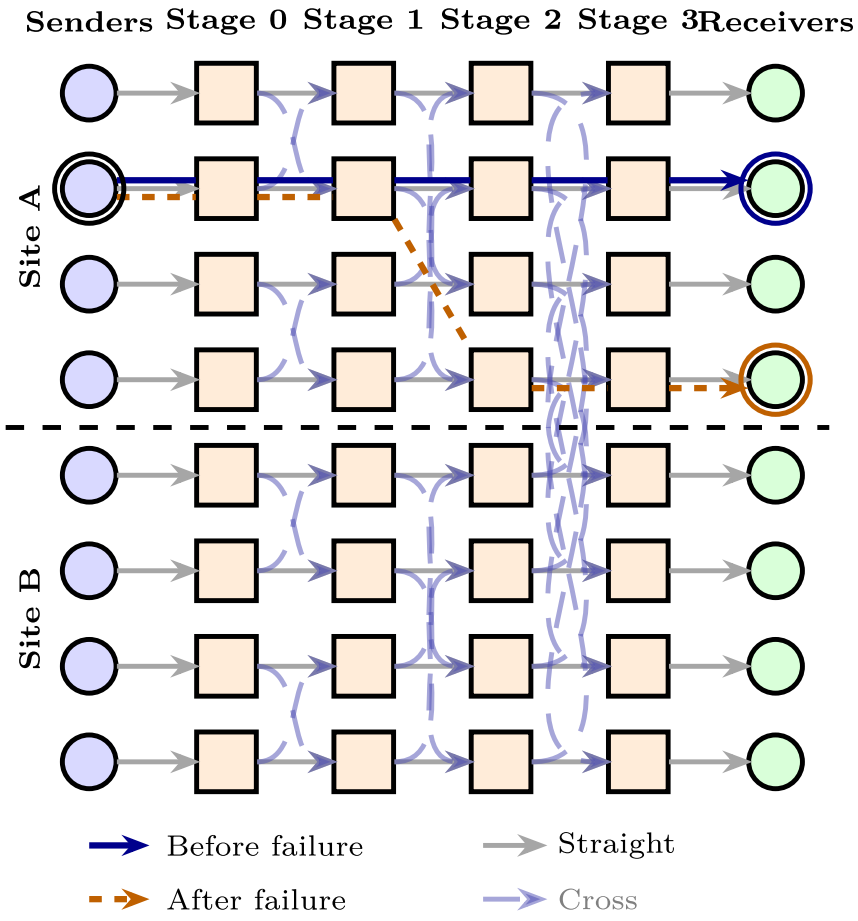






ILLINOIS TECH
College of Computing

Evaluation



- Large sample topology (48 in-band, 122 total VMs), low bandwidth (~100 KB/s), cross-(continent)-sites
- After failing switch, traffic resolves in 3.4s, which requires 2 fail-overs
- With a secondary controller located in Site B, traffic resolves in 3.67s

ILLINOIS TECH
College of Computing

Conclusion

- DiDAQt: Open-source C library designed to provide automated resiliency in HEP DAQs, a critical gap in network resiliency research
- Successfully executes fail-overs across redundant controllers and in the case of multiple failures
- Evaluation demonstrates it is practical for current and future DAQs
- **Future work:** coordination of multiple controllers, integration with a full-scale DAQ workload, optimization in alternative path choices

ILLINOIS TECH
College of Computing