Towards Redundancy Aware Network Stack for Datacenters

Ali Musa Iftikhar

About me

- Education
  - Undergraduate: LUMS (Pakistan)
  - PhD Student at Tufts (just finished first year)
- Research
  - Advisor: Fahad Dogar
  - Interests: Networks Systems; recent focus: data center networking
  - Current Status: Identified a problem with some potential promising solutions

What am I hoping for?

- Feedback on the problem
  - How important is it? Can it potentially become a thesis?
- Feedback on the initial direction
  - Design
  - Suggestions for evaluation
- Pointers on related work

Importance of Datacenter Application Performance

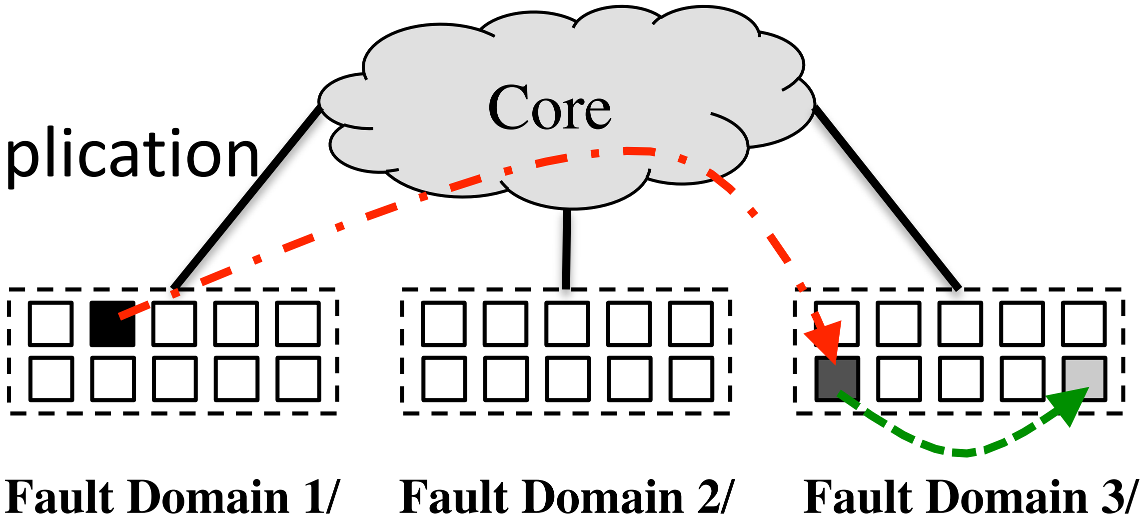
- Datacenters run a wide range of applications
  - Data analytics; user facing services, etc
- Performance matters
  - Low performance leads to fewer users leading to loss in revenue
    - Google demonstrated that slowing down the search results page by 100 to 400 milliseconds reduces the number of searches per user by 0.2% to 0.6%.

Why is this hard?

- Datacenter network is composed of commodity hardware - prone to failures *(Study Gill et al. Sigcomm 11)*
  - Significant impact of failures
    - A benchmark study by L. Ponemon Institute in 2013 shows that the per incident cost of an unplanned outage is likely to exceed $8,000 per minute
- Applications are highly distributed
  - Fan out is large
  - many sequential stages
  - parallelization across 10s-1000s
    - *(Speeding up Distributed Request-Response Workflows, Sigcomm 13)*

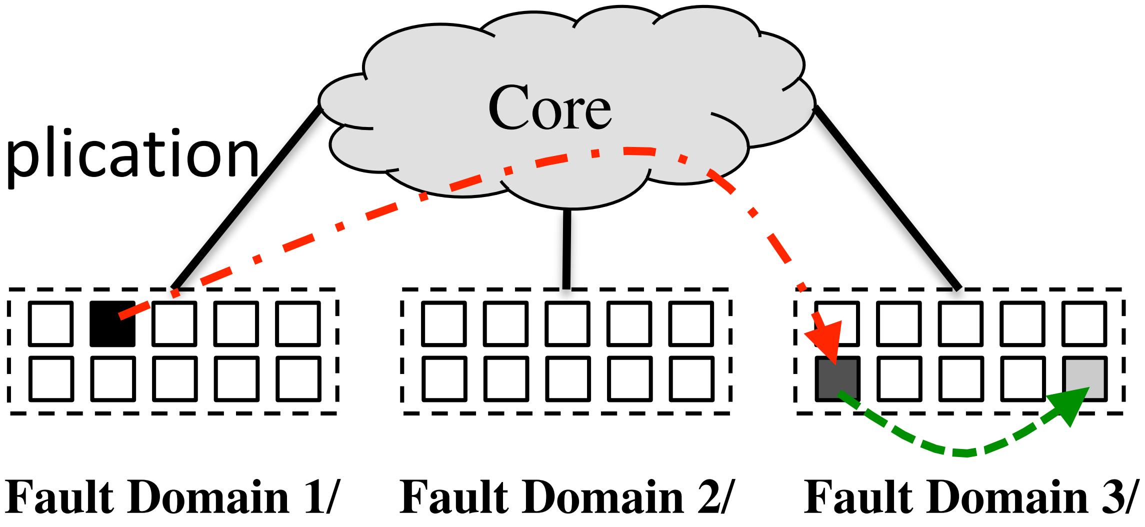
Replication to the rescue

- Most applications use some form of replication
  - Cluster file systems:
    - GFS, HDFS, Cosmos
  - Amazon S3, Windows Azure Storage
  - Facebook's Haystack
- Improves application performance
  - Can prevent loss of data and major disruptions in service
  - Helps in load balancing – reducing load on a single replica



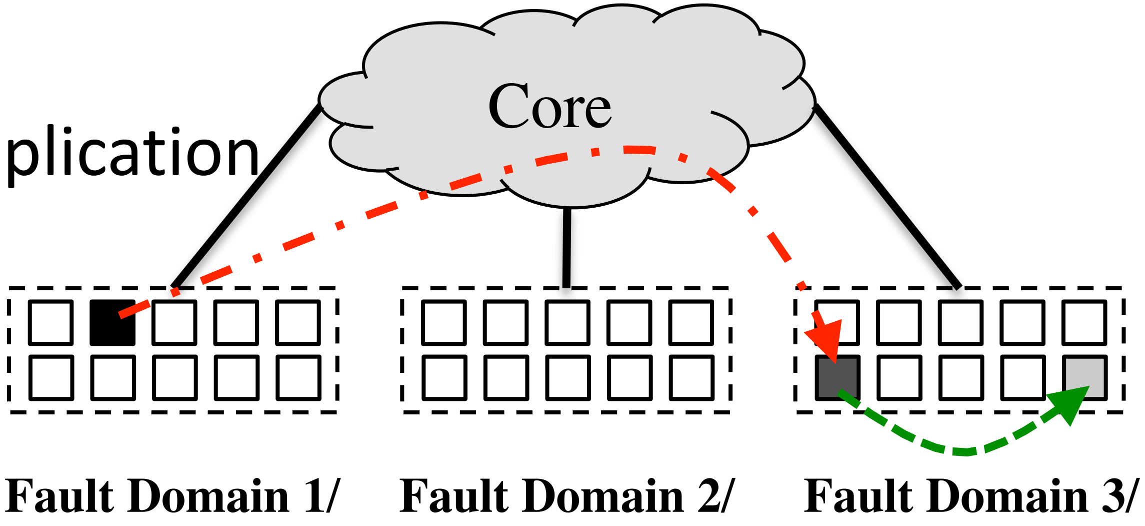
Replication to the rescue

- Most applications use some form of replication
  - Cluster file systems:
    - GFS, HDFS, Cosmos
  - Amazon S3, Windows Azure Storage
  - Facebook's Haystack



- Improves application performance
  - Can prevent loss of data and major disruptions in service
  - Helps in load balancing – reducing load on a single replica
- However this scheme is limited, as the network is unaware of these replicas

Replication to the rescue



- Most applications use some form of replication
  - Cluster file systems:
    - GFS, HDFS, Cosmos
  - Amazon S3, Windows Azure Storage
  - Facebook's Haystack
- Improves application performance
  - Can prevent loss of data and major disruptions in service
  - Helps in load balancing – reducing load on a single replica
- However … of these replicas

We claim that there are potential benefits of making the network replica aware.

Redundancy Aware Network Stack

- A co-design of applications and the network
- applications share replica information with the network stack (transport and network layer)
- network stack uses redundancy aware mechanisms (eg. routing)
- applications may need to be modified to make full use of the mechanisms

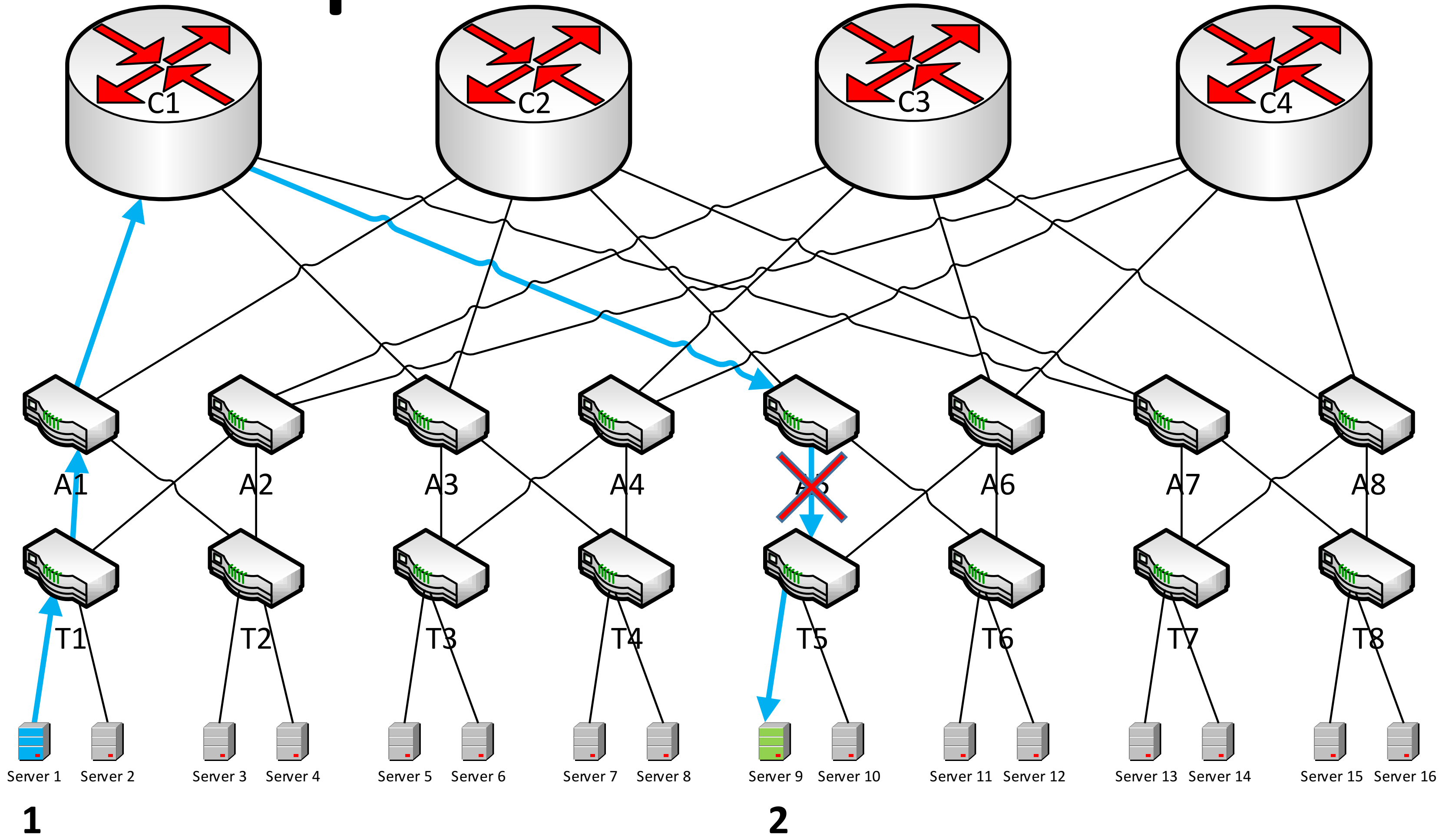
Redundancy Aware Network Stack: Potential Benefits

- **1. Improved replica selection**
  - Accurately choose least congested replicas.
  - Faster adaptive replica selection.
- **2. In-network services**
  - Intelligent erasure coding service to avoid bottlenecks.

Redundancy Aware Network Stack: Potential Benefits

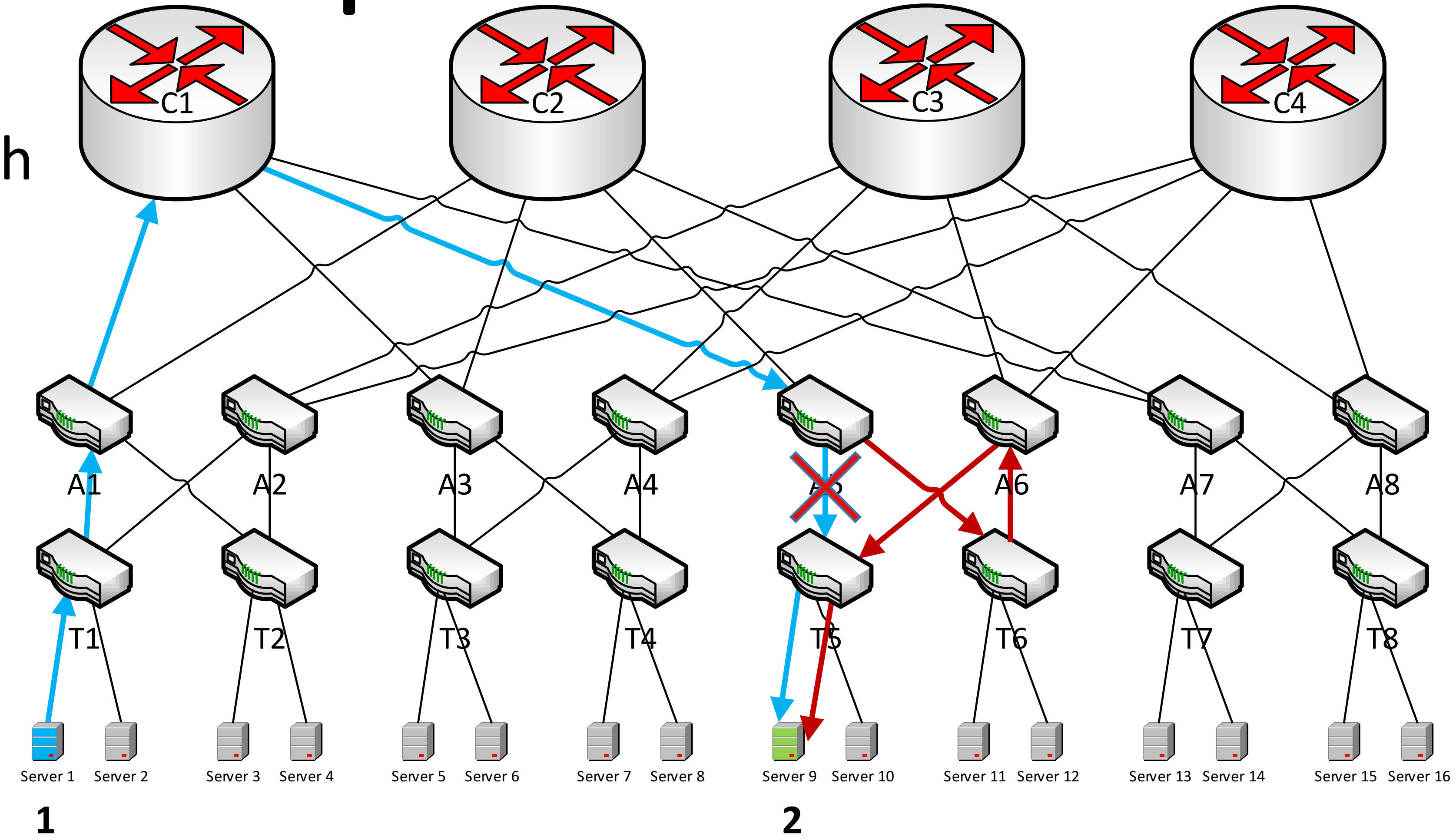
- **1. Improved replica selection**
  - Accurately choose least congested replicas.
  - Faster adaptive replica selection.
- **2. In-network services**
  - Intelligent erasure coding service to avoid bottlenecks.
- **3. Improved failure recovery**
  - Route around failures by using replicas which do not lie along faulty paths.

Failure Recovery – Without Replicas



Failure Recovery – Without Replicas

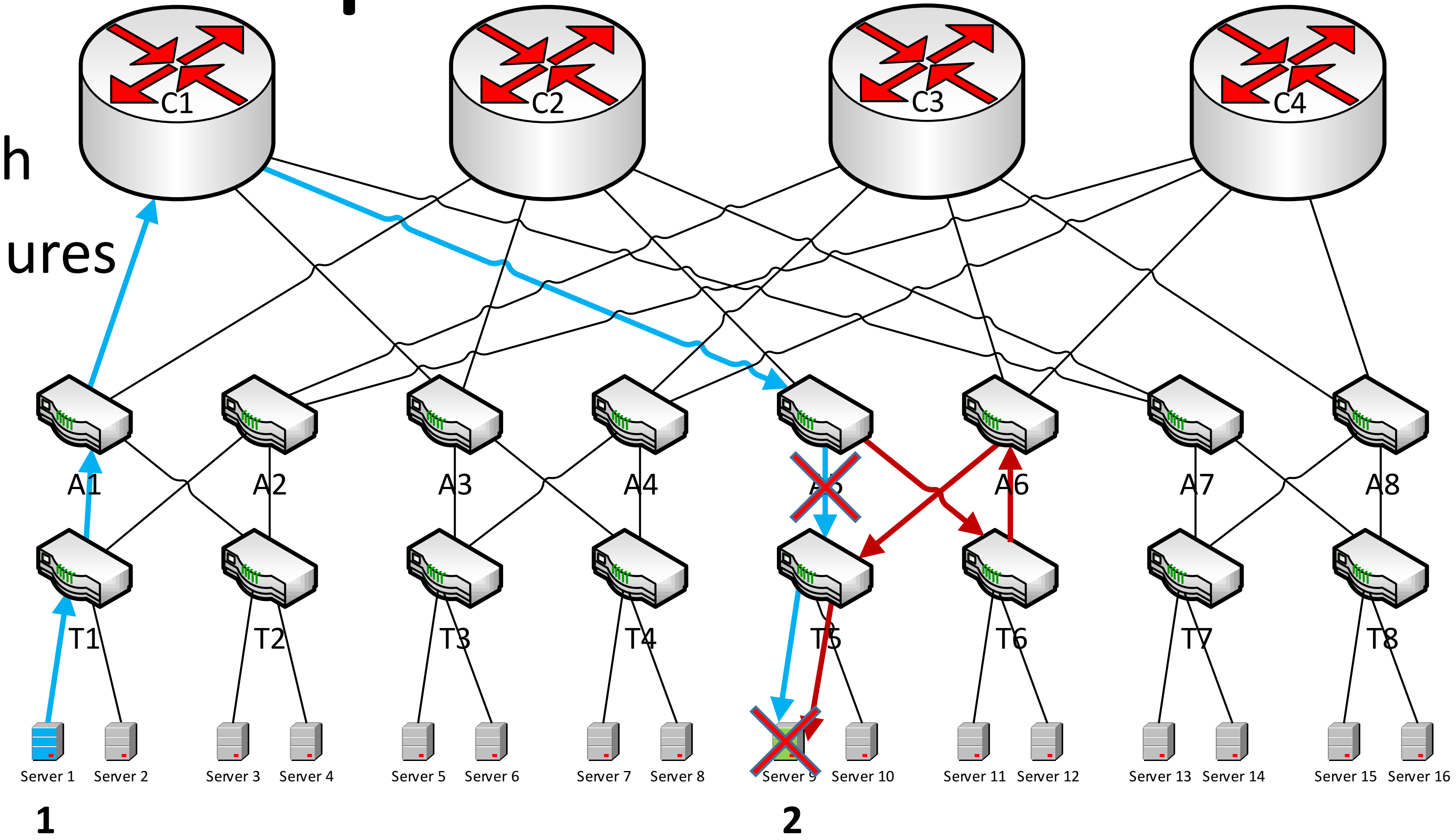
Increased hop-length



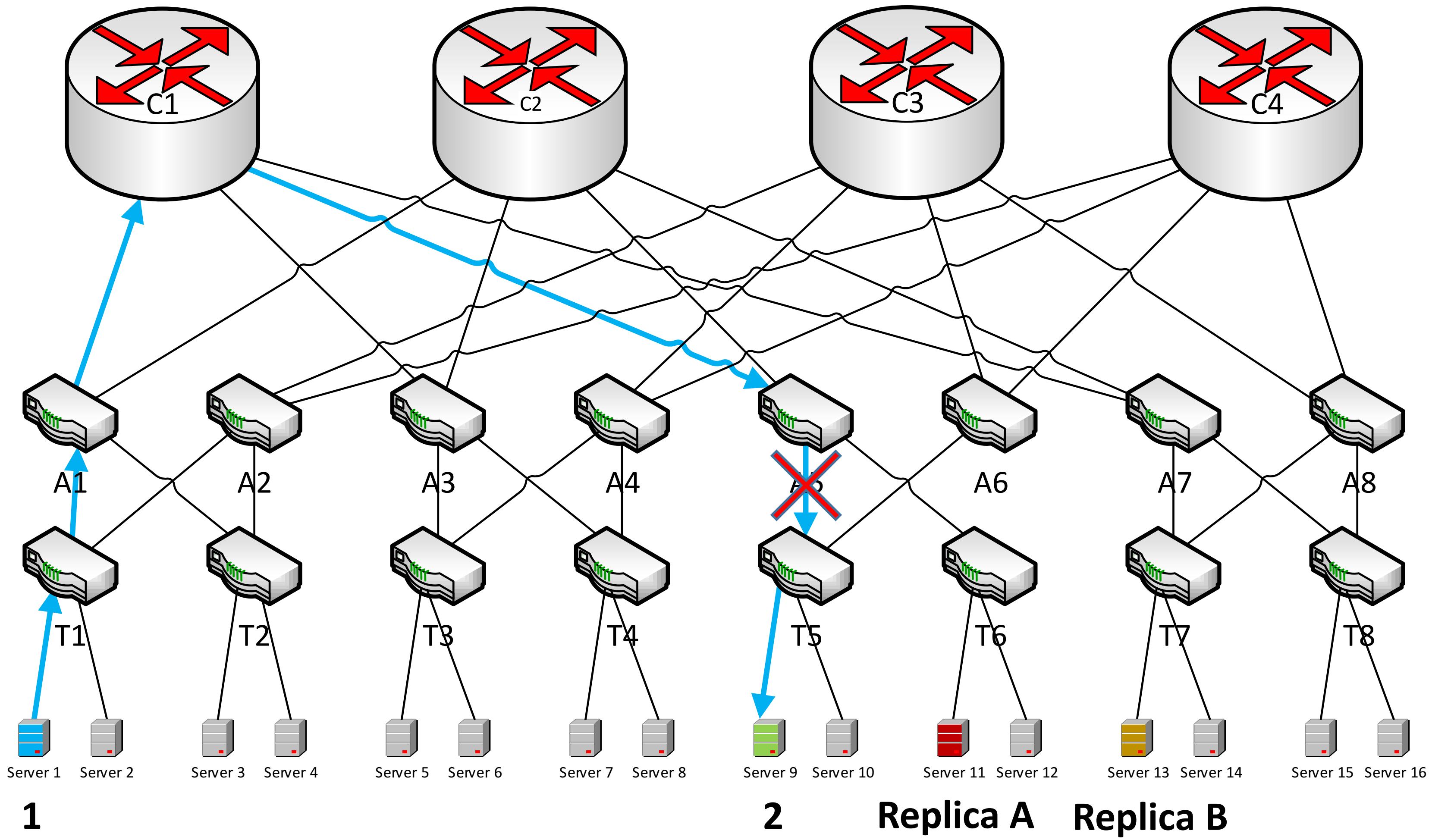
Failure Recovery – Without Replicas

Increased hop-length

Can't tolerate all failures

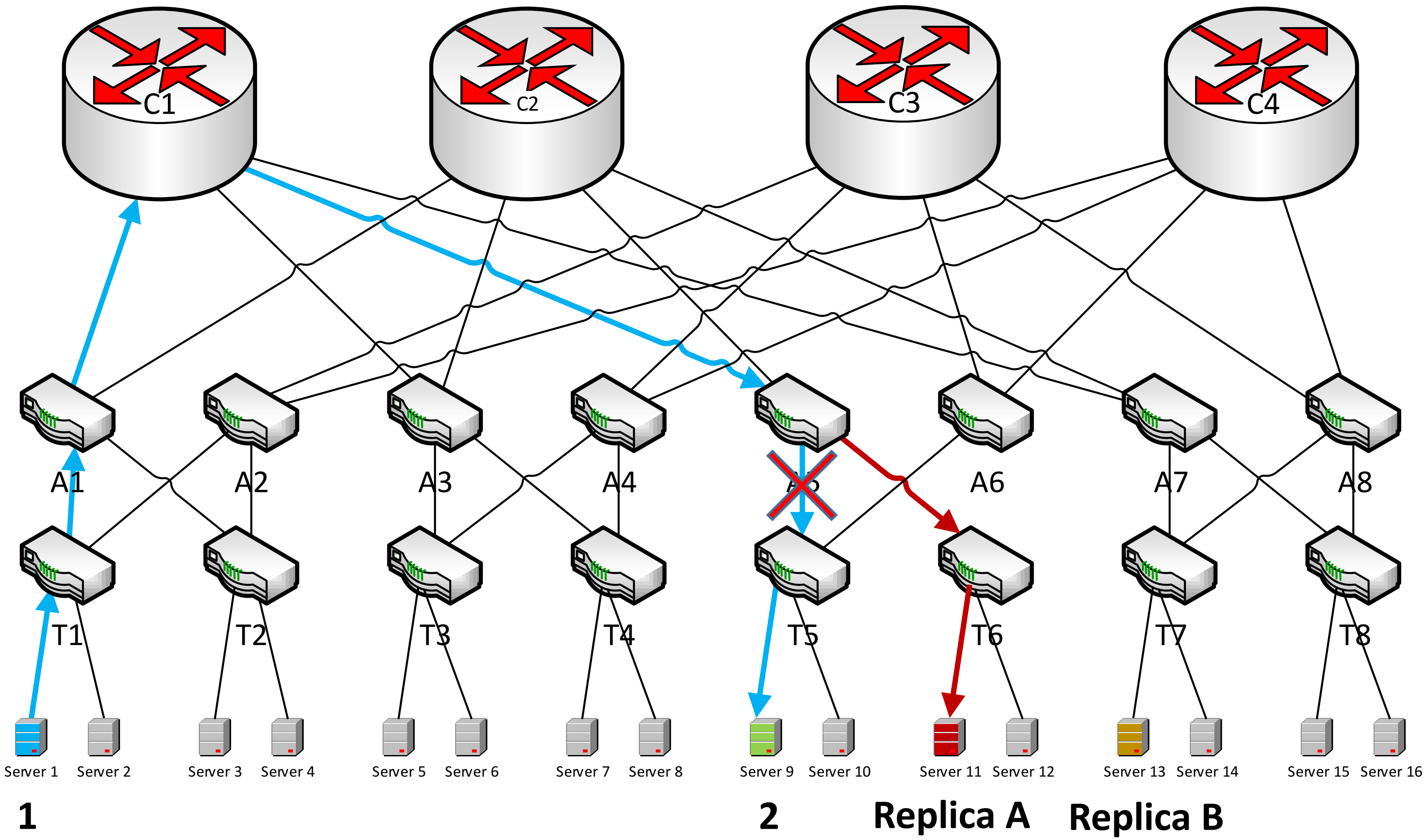


Failure Recovery – With Replicas



Failure Recovery – With Replicas

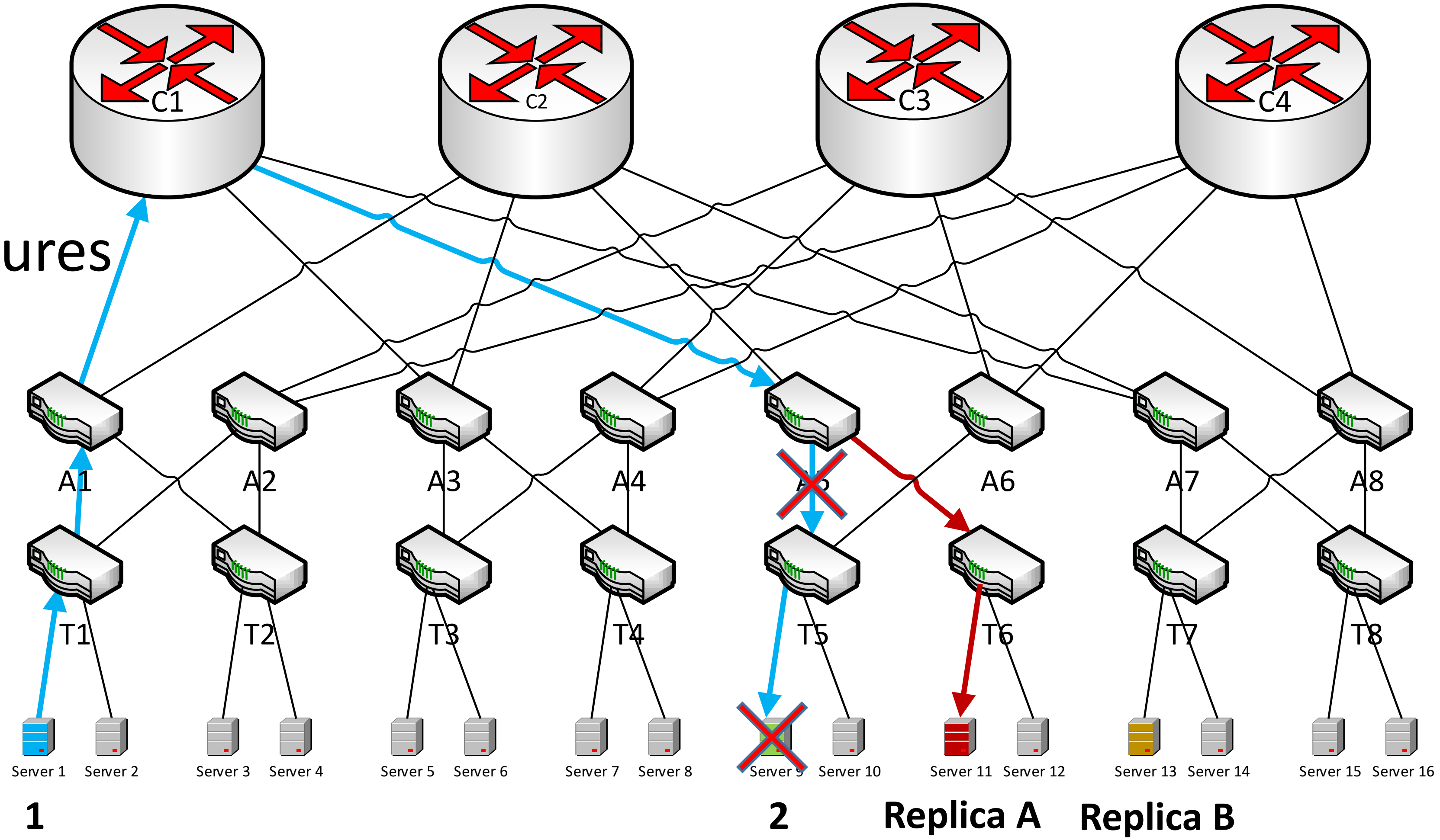
Same hop-length



Failure Recovery – With Replicas

Same hop-length

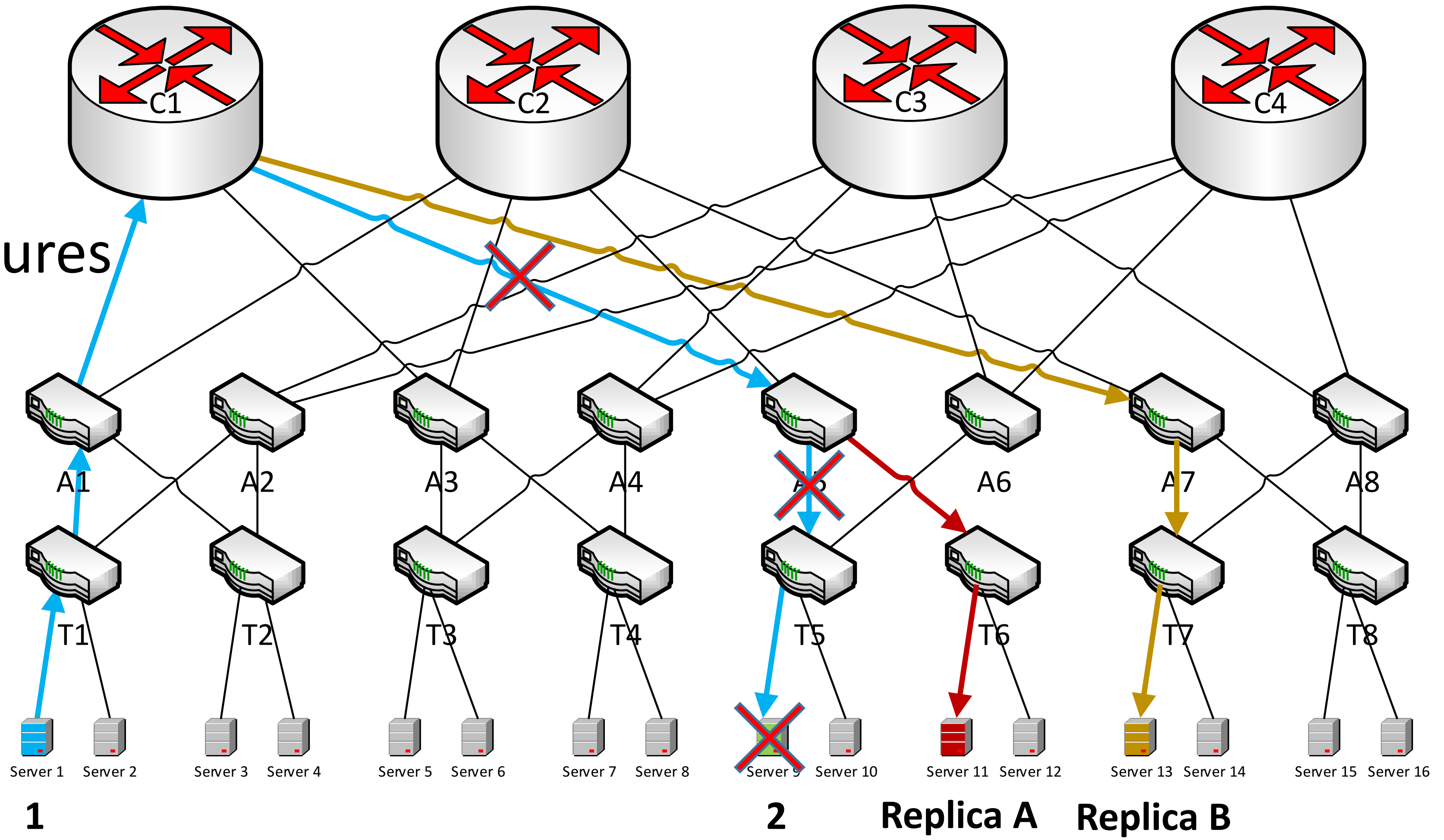
Resilient to most failures



Failure Recovery – With Replicas

Same hop-length

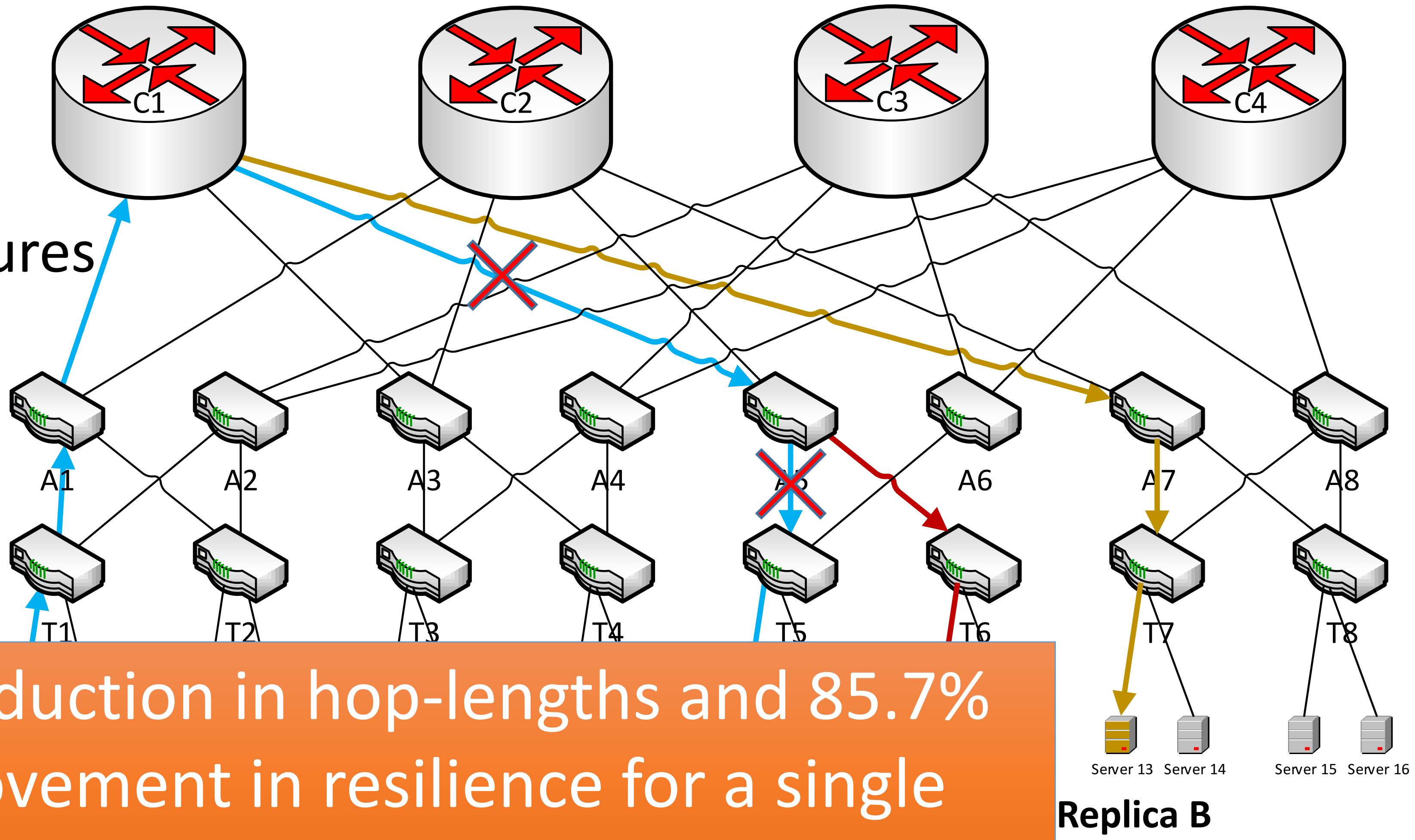
Resilient to most failures



Failure Recovery – With Replicas

Same hop-length

Resilient to most failures



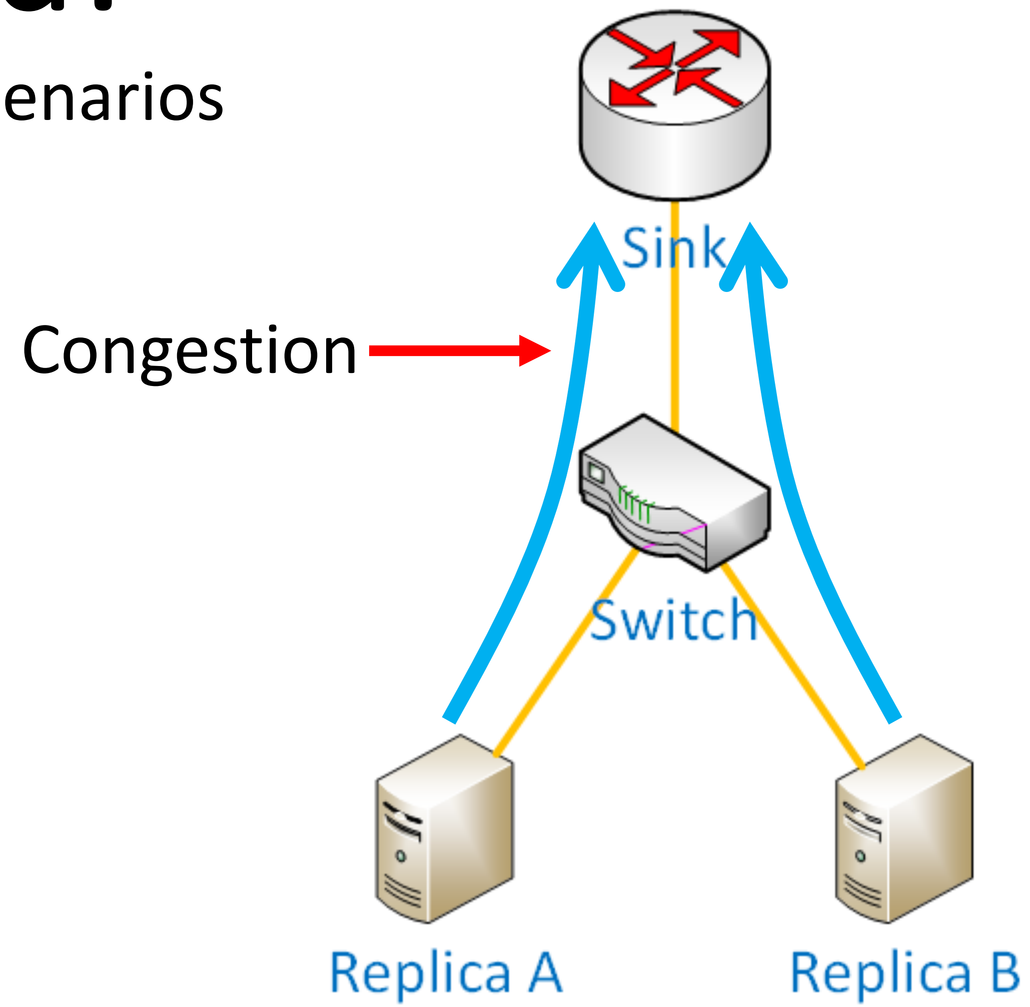
25% reduction in hop-lengths and 85.7% improvement in resilience for a single failure.

Redundancy Aware Network Stack: Potential Benefits

- **1. Improved replica selection**
  - Accurately choose least congested replicas.
  - Faster adaptive replica selection.
- **2. In-network services**
  - Intelligent erasure coding service to avoid bottlenecks.
- **3. Improved failure recovery**
  - Route around failures by using replicas which do not lie along faulty paths.
- **4. Reduced overhead of duplicate requests**
  - Initiate duplicate requests to all of the available replicas

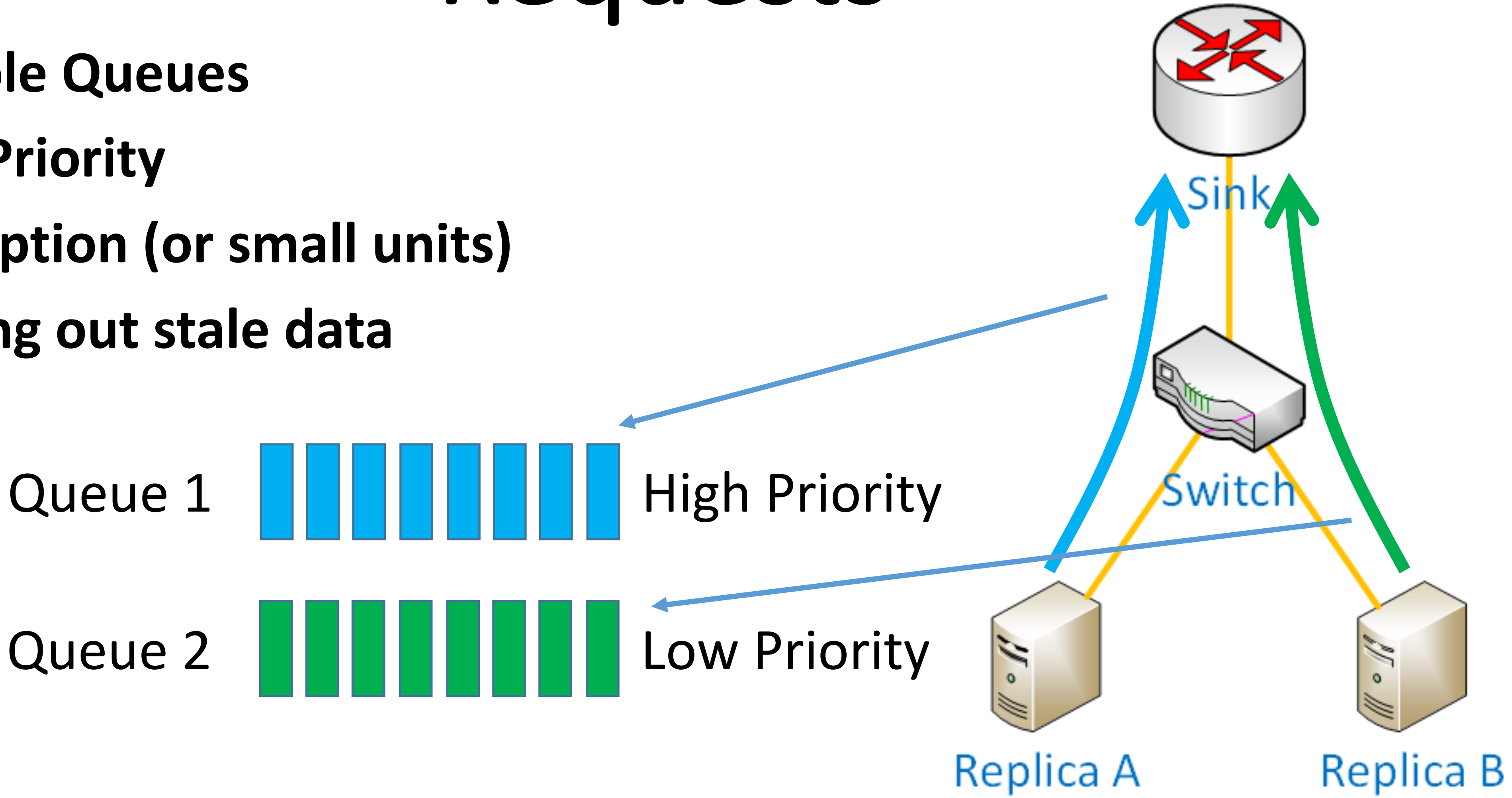
Duplicate Requests: Double the load!

- Caters to the most unpredictable scenarios

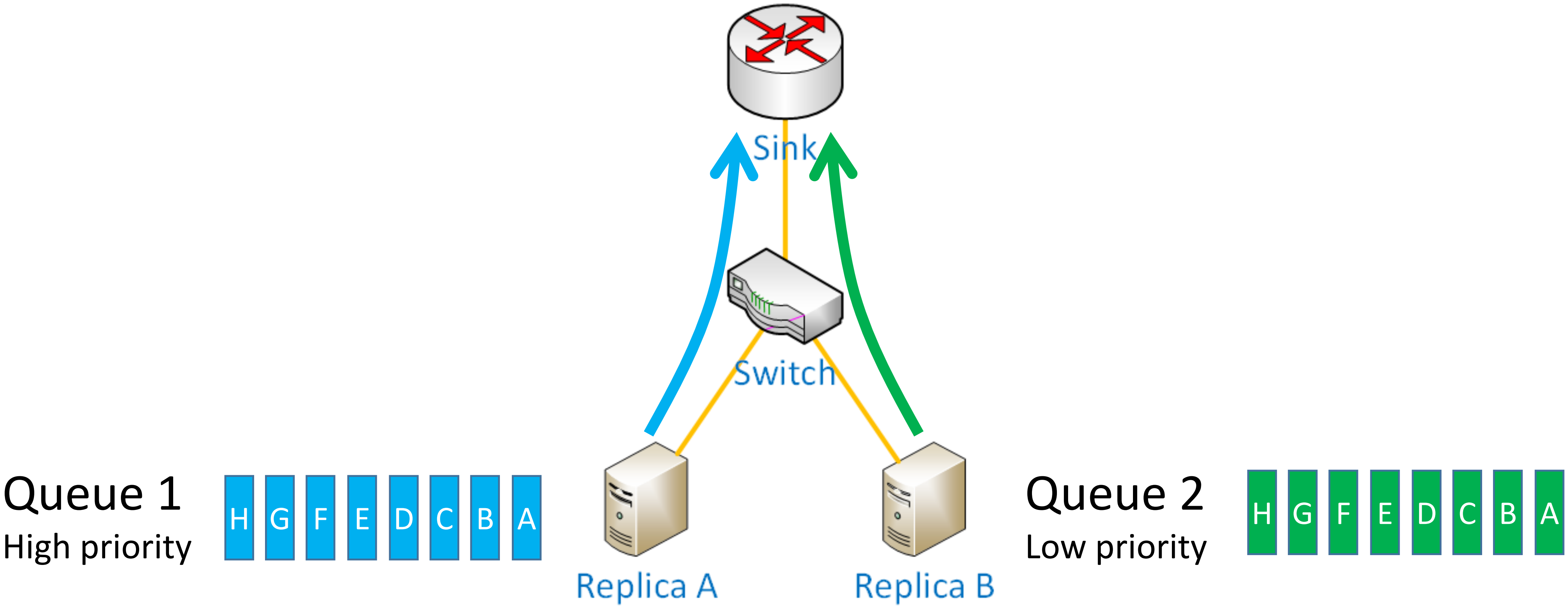


Requirements for Duplicate Requests

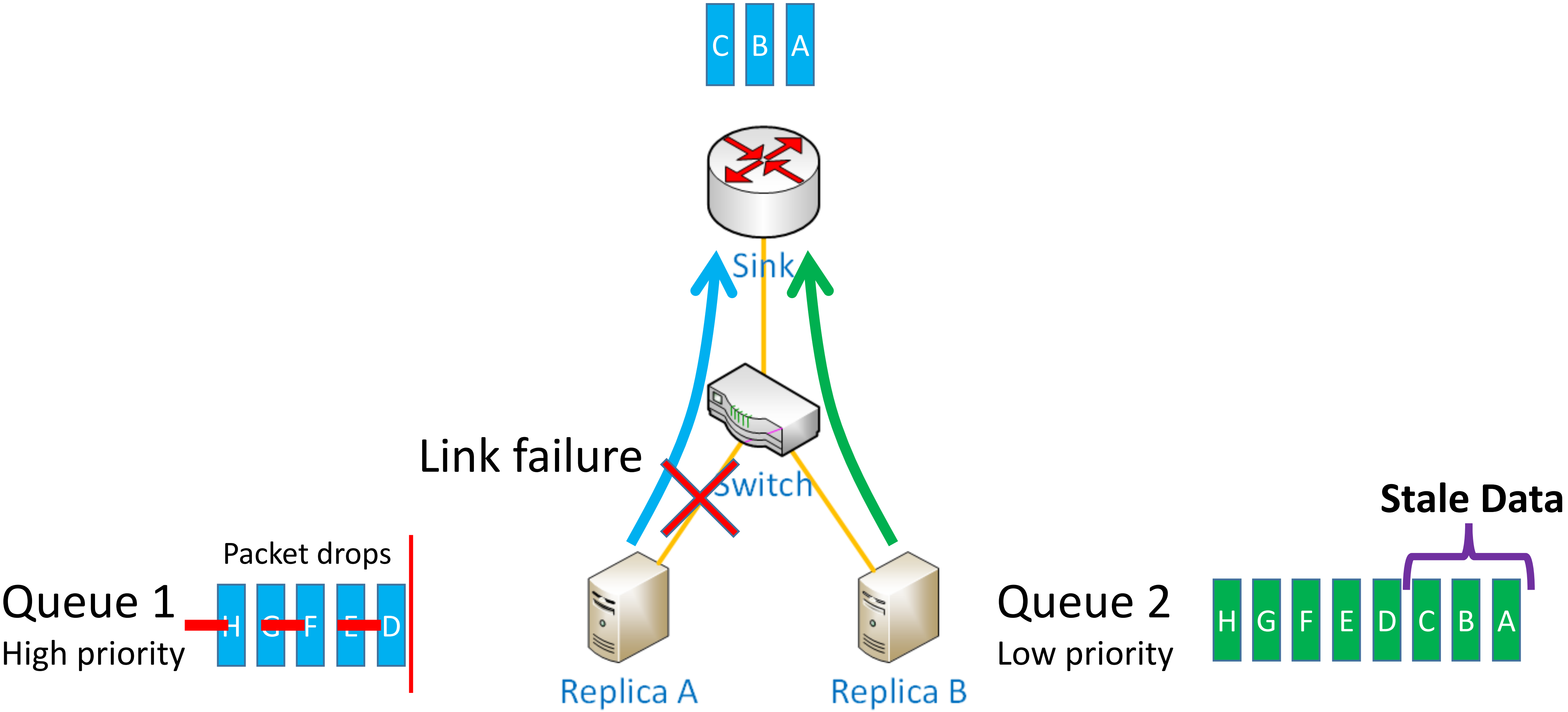
- **Multiple Queues**
- **Strict Priority**
- **Preemption (or small units)**
- **Flushing out stale data**



Flushing out Stale Data

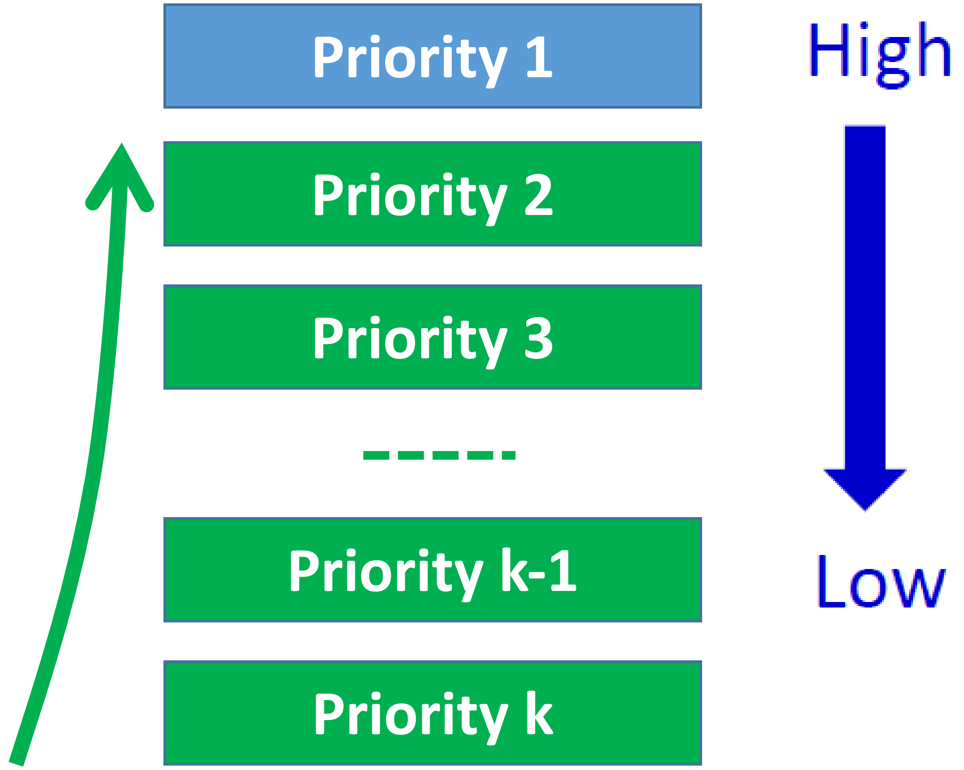


Flushing out Stale Data



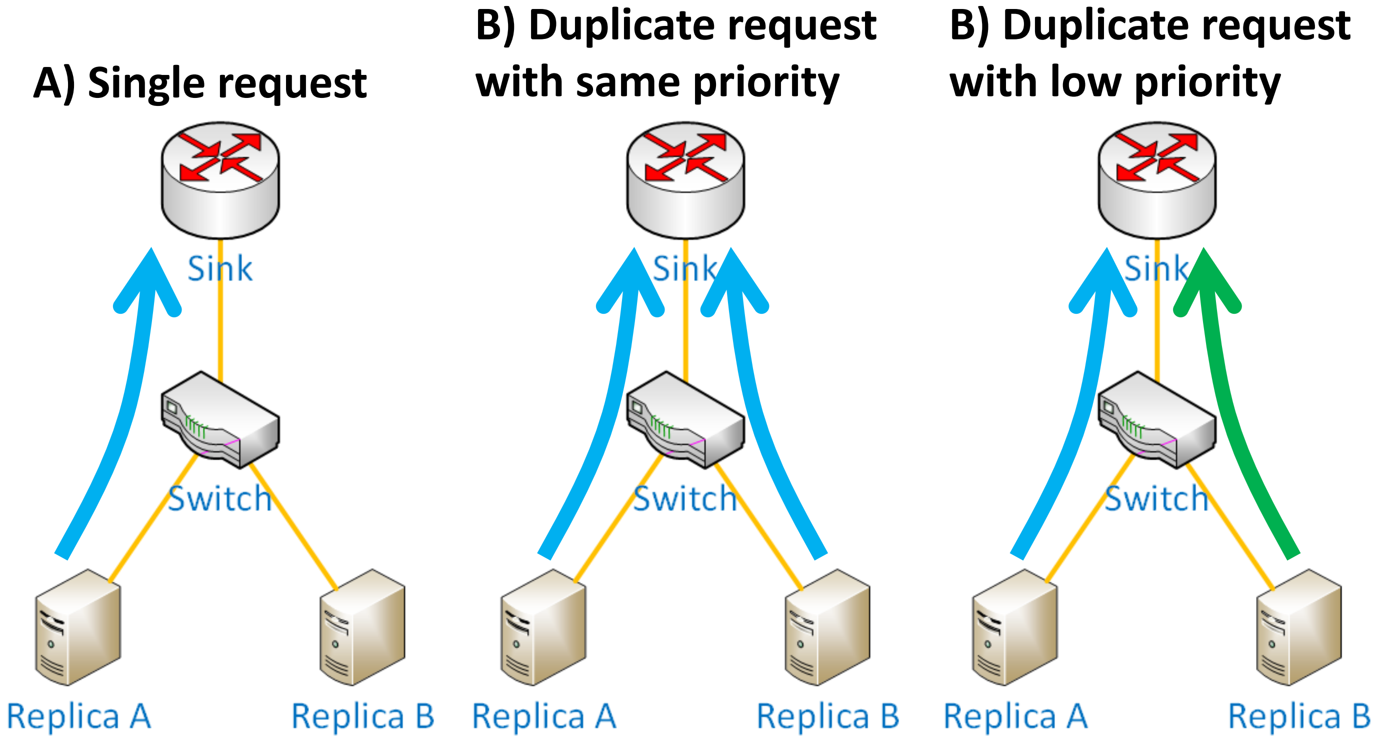
Food for thought: Multiple Priorities

- Typical queues: FIFO *(Pias, Hotnets 14)*
- Can filling queues bottom up to emulate LIFO help?

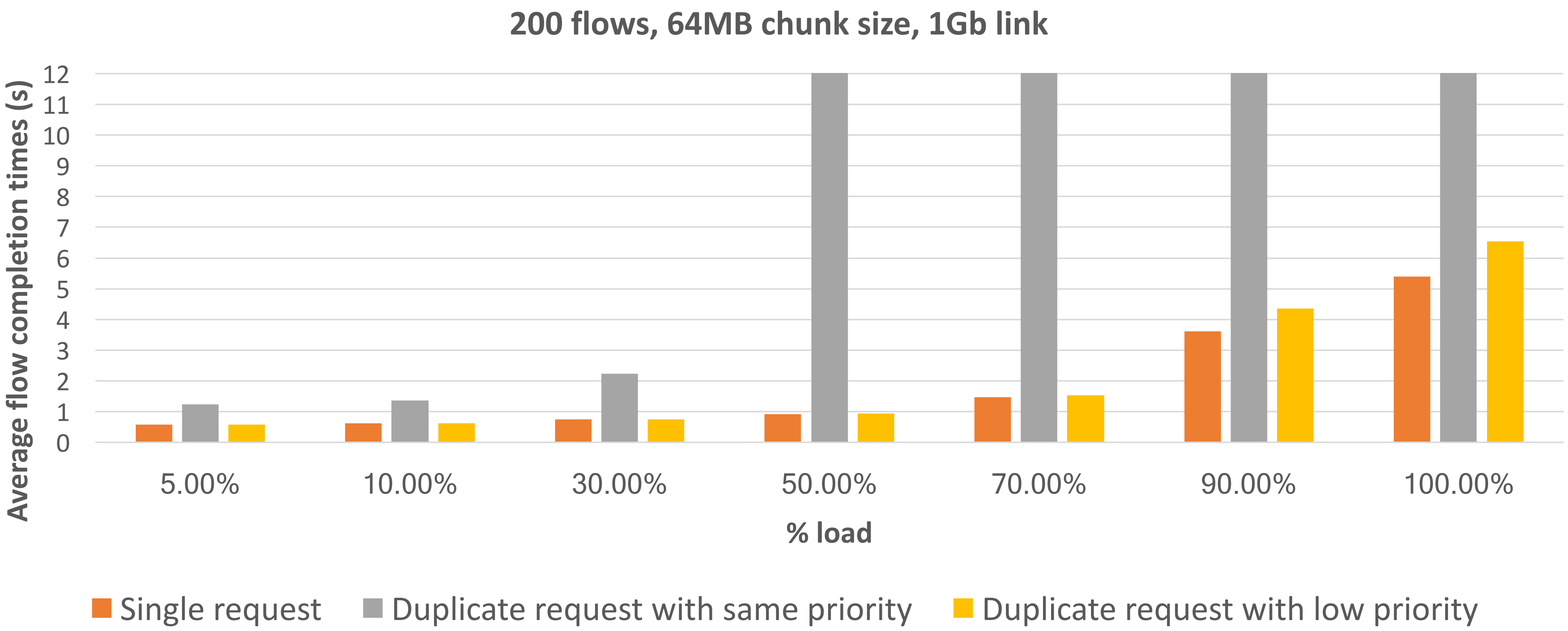


Initial Simulations: Setup

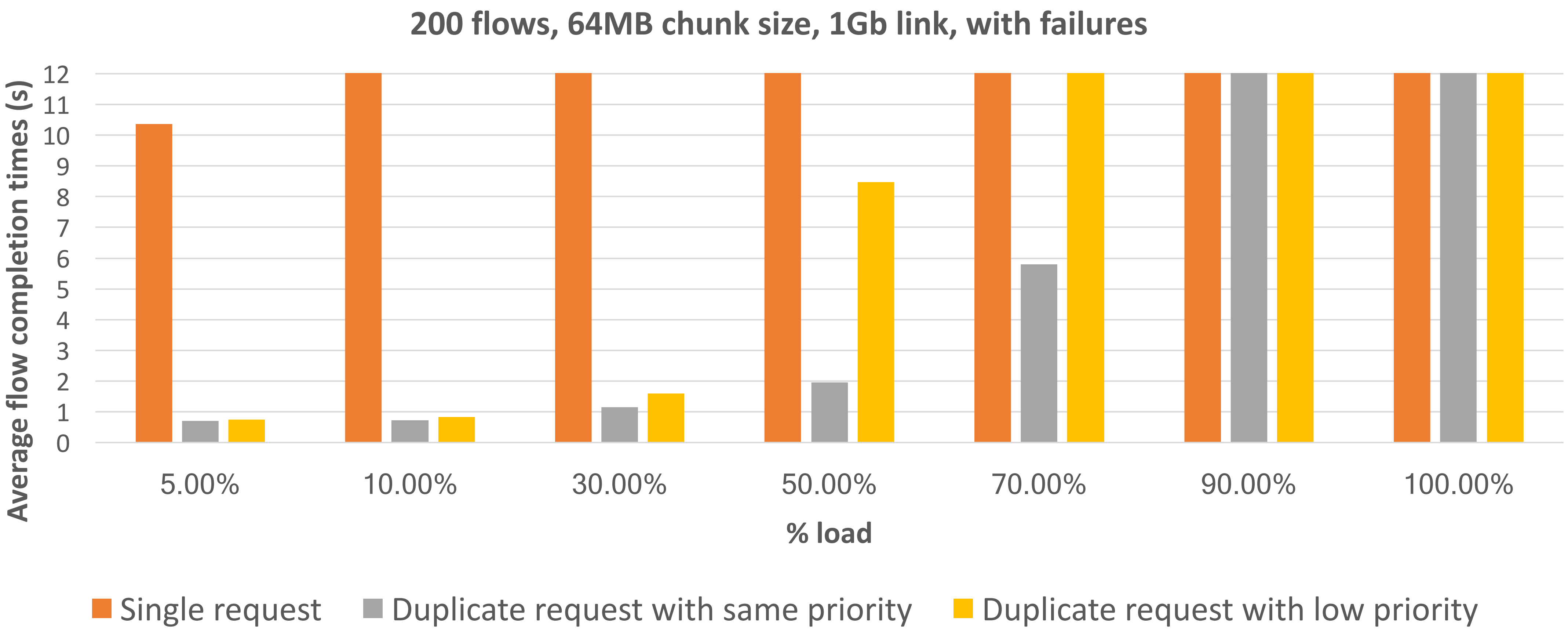
- NS-2 simulator
  - Varying loads
  - Metric: aggregate FCTs
  - Failures on Replica A



Initial Simulations: Results



Initial Simulations: Results



Redundancy Aware Network Stack: Potential Benefits

- **1. Improved replica selection**
  - Accurately choose least congested replicas.
  - Faster adaptive replica selection.
- **2. In-network services**
  - Intelligent erasure coding service to avoid bottlenecks.
- **3. Improved failure recovery**
  - Route around failures by using replicas which do not lie along faulty paths.
- **4. Reduced overhead of duplicate requests**
  - Initiate duplicate requests to all of the available replicas

Related work

- Replica selection:
  - *(Sinbad, Sigcomm 13)*
  - *(C3, Nsdi 15)*
- Fault tolerance in DCNs:
  - *(F10, NSDI 13) (Aspen Trees, CoNext 13) (Conga, Sigcomm 14)*
- Redundant requests:
  - *(Low latency via Redundancy, CoNext 13)*

- None of these talk about a redundancy aware network stack.

Plans forward

- Failure recovery:
  - Open flow for dynamic routing
  - Deal with multiple failures
  - Partial data
- Duplicate requests:
  - Evaluation on HDFS, Cassandra, Memcached
  - Develop a transport protocol to provide support

Broader scope

- Expressive interface between network and application layer
  - Graph based interface
  - Applications express their workflows to the network
- Redundancy aware network mechanisms:
  - Failure recovery, routing and scheduling
- Modified cloud applications
  - Providing complementary support to the modified network mechanisms
  - Duplicate aware scheduling at the application level

What am I hoping for?

- Feedback on the problem
  - How important is it? Can it potentially become a thesis?
- Feedback on the initial direction
  - Design
  - Suggestions for evaluation
- Pointers on related work