Black-Box Problem Diagnosis in Parallel File Systems

Michael P. Kasick[1]

Jiaqi Tan[2], Rajeev Gandhi[1], Priya Narasimhan[1]

[1]Carnegie Mellon University [2]DSO National Labs, Singapore

Problem Diagnosis Goals

- To diagnose problems in off-the-shelf parallel file systems
  - Environmental performance problems: disk & network faults
  - Target file systems: PVFS & Lustre
- To develop methods applicable to existing deployments
  - Application transparency: avoid code-level instrumentation
  - Minimal overhead, training, and configuration
  - Support for arbitrary workloads: avoid models, SLOs, etc.

Motivation: Real Problem Anecdotes

- Problems motivated by PVFS developers' experiences
  - From Argonne's Blue Gene/P PVFS cluster

Motivation: Real Problem Anecdotes

- Problems motivated by PVFS developers' experiences
  - From Argonne's Blue Gene/P PVFS cluster
- "Limping-but-alive" server problems
  - No errors reported, can't identify faulty node with logs
  - Single faulty server impacts overall system performance

Motivation: Real Problem Anecdotes

- Problems motivated by PVFS developers' experiences
  - From Argonne's Blue Gene/P PVFS cluster
- "Limping-but-alive" server problems
  - No errors reported, can't identify faulty node with logs
  - Single faulty server impacts overall system performance
- Storage-related problems:
  - Accidental launch of rogue processes, decreases throughput
  - Buggy RAID controller issues patrol reads when not at idle

Motivation: Real Problem Anecdotes

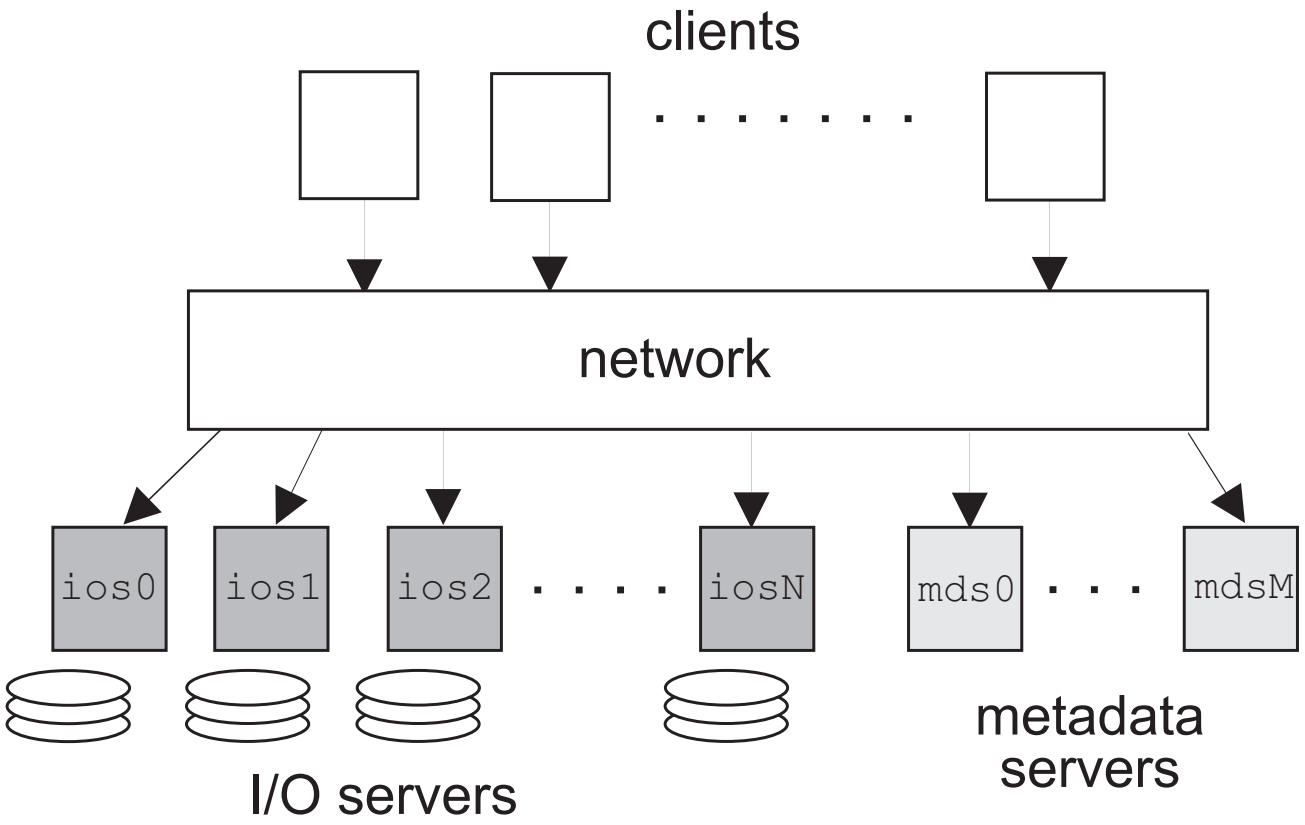
- Problems motivated by PVFS developers' experiences
  - From Argonne's Blue Gene/P PVFS cluster
- "Limping-but-alive" server problems
  - No errors reported, can't identify faulty node with logs
  - Single faulty server impacts overall system performance
- Storage-related problems:
  - Accidental launch of rogue processes, decreases throughput
  - Buggy RAID controller issues patrol reads when not at idle
- Network-related problems:
  - Faulty-switch ports corrupt packets, fail CRC checks
  - Overloaded switches drop packets but pass diagnostic tests

Outline

Target Parallel File Systems

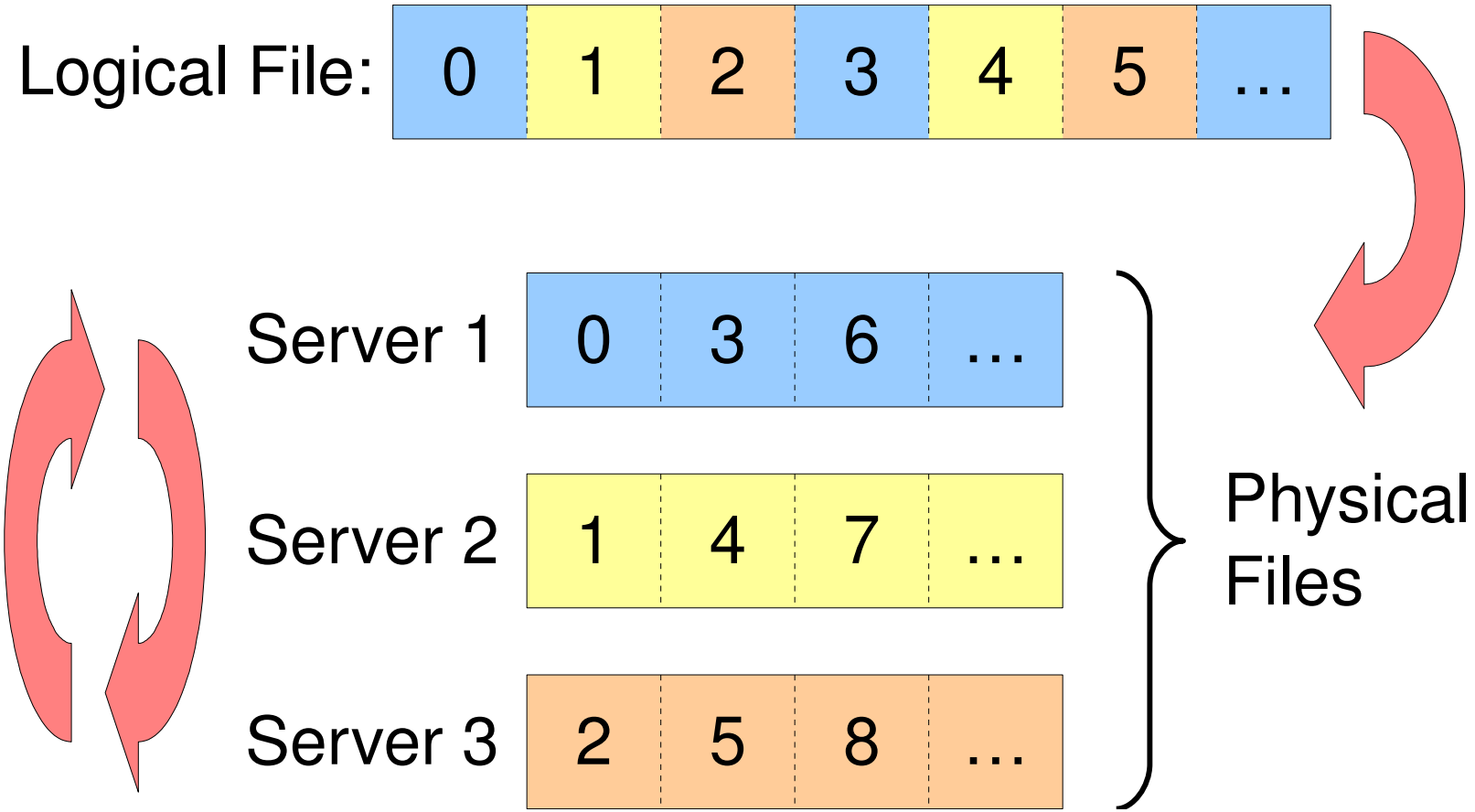
- Aim to support I/O-intensive applications
- Provide high-bandwidth, concurrent access

Parallel File System Architecture



- One or more I/O and metadata servers
- Clients communicate with every server
  - No server-server communication

Parallel File System Data Striping

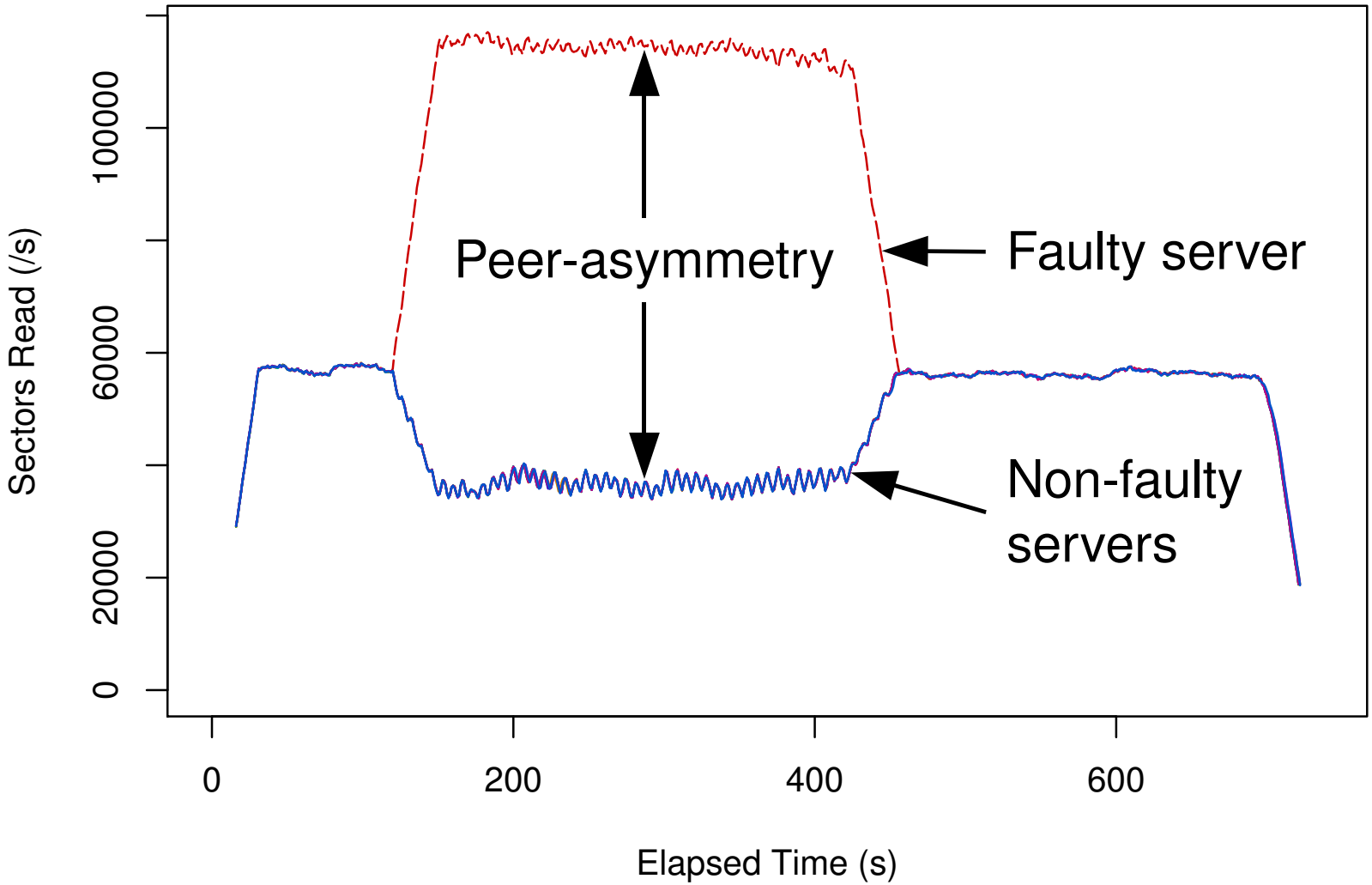


- Client stripes local file into 64 kB–1 MB chunks
- Writes to each I/O server in round-robin order

Parallel File Systems: Empirical Insights (I)

- Server behavior is similar for most requests
  - Large requests are striped across all servers
  - Small requests, in aggregate, equally load all servers
- **Hypothesis:** Peer-similarity
  - Fault-free servers exhibit similar performance metrics
  - Faulty servers exhibit dissimilarities in certain metrics
  - Peer-comparison of metrics identifies faulty node

Example: Disk-Hog Fault



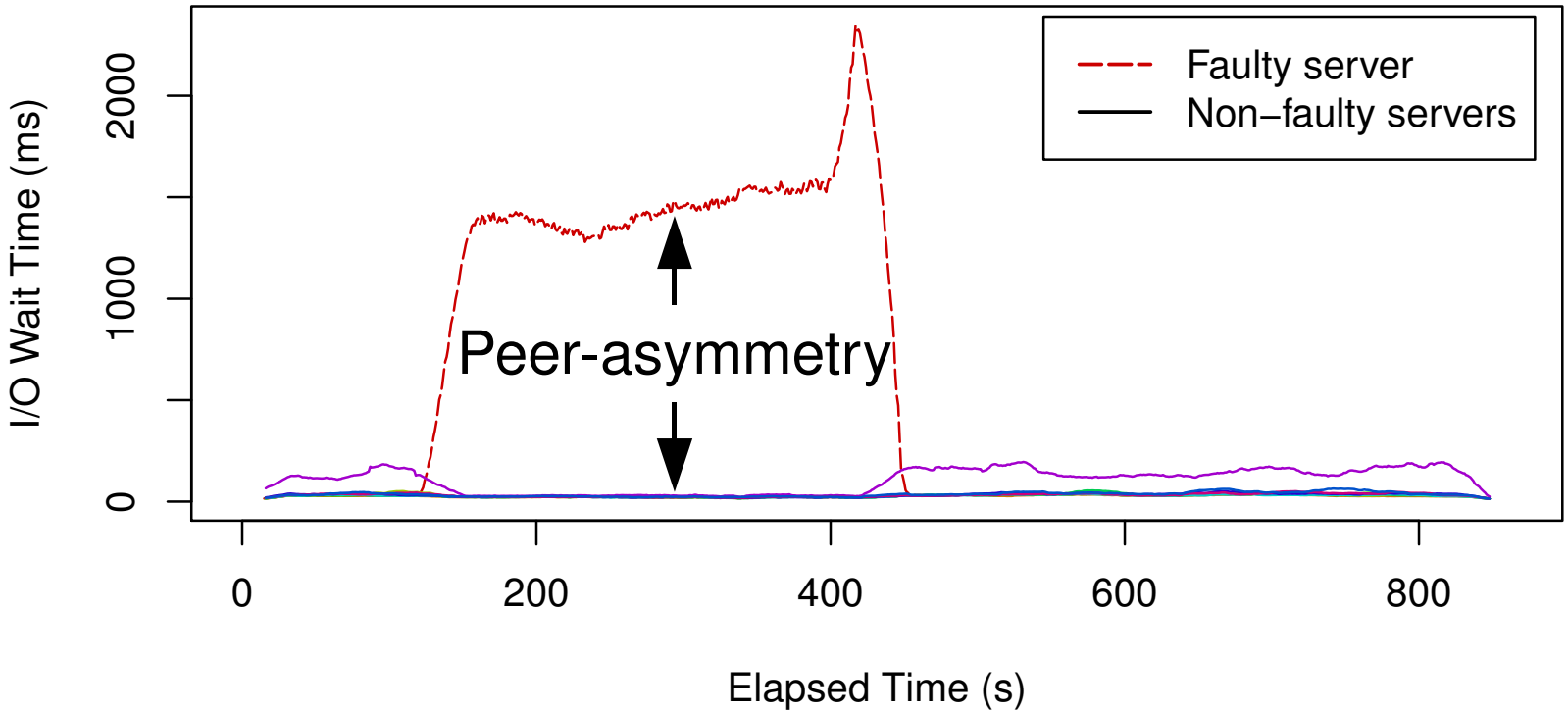
- Strongly motivates peer-comparison approach

Parallel File Systems: Empirical Insights (II)

- Faults manifest asymmetrically only on some metrics
  - Ex: A disk-busy fault manifests . . .

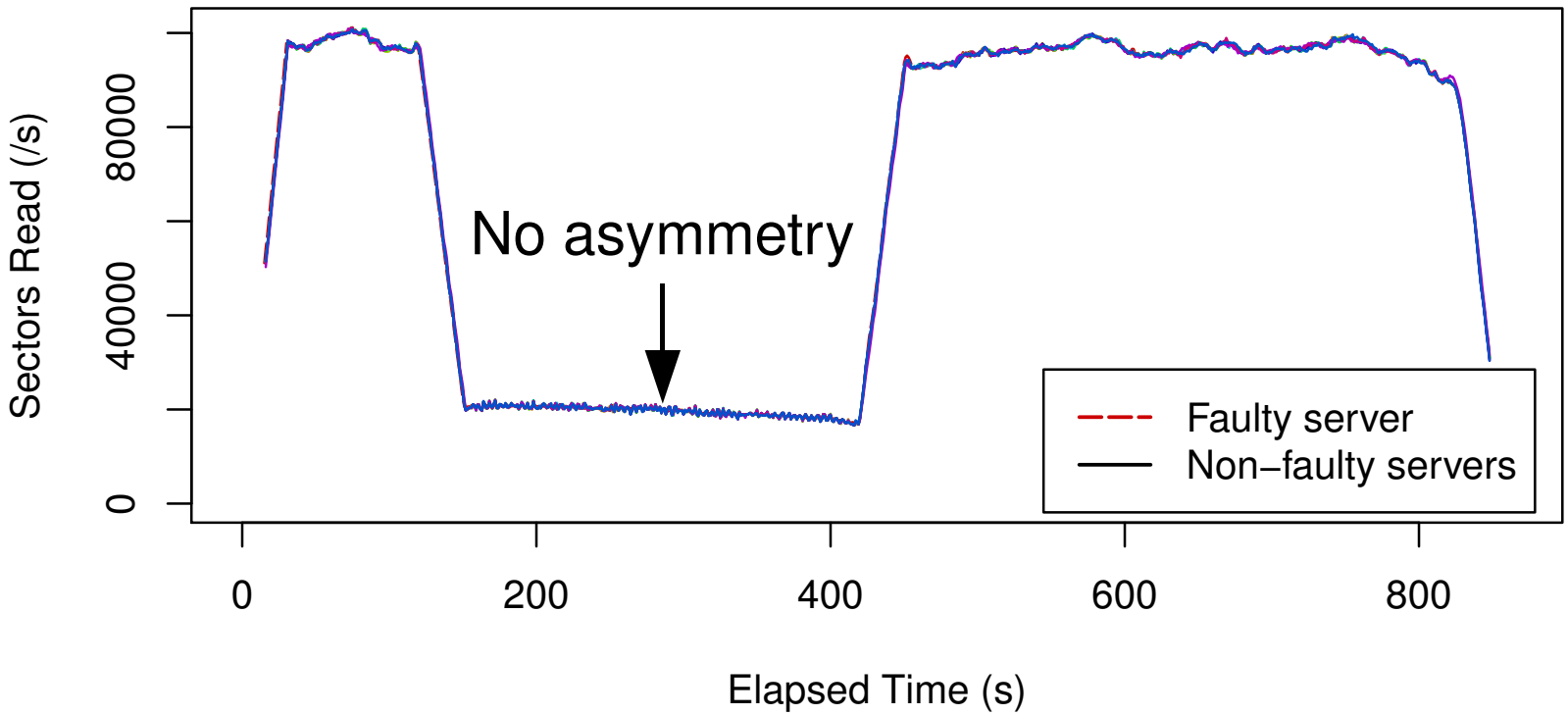
Parallel File Systems: Empirical Insights (II)

- Faults manifest asymmetrically only on some metrics
  - Ex: A disk-busy fault manifests . . .
    - Asymmetrically on latency metrics (↑ on faulty, ↓ on fault-free)



Parallel File Systems: Empirical Insights (II)

- Faults manifest asymmetrically only on some metrics
  - Ex: A disk-busy fault manifests . . .
    - Asymmetrically on latency metrics (↑ on faulty, ↓ on fault-free)
    - Symmetrically on throughput metrics (↓ on all nodes)



Parallel File Systems: Empirical Insights (II)

- Faults manifest asymmetrically only on some metrics
  - Ex: A disk-busy fault manifests . . .
    - Asymmetrically on latency metrics (↑ on faulty, ↓ on fault-free)
    - Symmetrically on throughput metrics (↓ on all nodes)

- Faults distinguishable by which metrics are peer-divergent

Outline

System Model

- Fault Model:
  - Non-fail-stop problems
    - “Limping-but-alive” performance problems
  - Problems affecting storage & network resources
- Assumptions:
  - Hardware is homogeneous, identically configured
  - Workloads are non-pathological (balanced requests)
  - Majority of servers exhibit fault-free behavior

Instrumentation

- Sampling of storage & network performance metrics
  - Sampled from `/proc` once every second
  - Gathered from all server nodes
- Storage-related metrics of interest:
  - Throughput: Bytes read/sec, bytes written/sec
  - Latency: I/O wait time
- Network-related metrics of interest:
  - Throughput: Bytes received/sec, transmitted/sec
  - Congestion: TCP sending congestion window

Workloads

- ddw & ddr (`dd` write & read)
  - Use `dd` to write/read many GB to/from file
  - Large (order MB) I/O requests, saturating workload
- iozonew & iozoner (IOzone write & read)
  - Ran in either write/rewrite or read/reread mode
  - Large I/O requests, workload transitions, `fsync`
- postmark (PostMark)
  - Metadata-heavy, small reads/writes (single server)
  - Simulates email/news servers

Fault Types

- Susceptible resources:
  - Storage: Access contention
  - Network: Congestion, packet loss (faulty hardware)
- Manifestation mechanism:
  - Hog: Introduces new visible workload (server-monitored)
  - Busy/Loss: Alters existing workload (unmonitored)

Fault Types

- Susceptible resources:
  - Storage: Access contention
  - Network: Congestion, packet loss (faulty hardware)
- Manifestation mechanism:
  - Hog: Introduces new visible workload (server-monitored)
  - Busy/Loss: Alters existing workload (unmonitored)

	Storage	Network
Hog	disk-hog	write-network-hog read-network-hog
Busy/Loss	disk-busy	receive-packet-loss send-packet-loss

Experiment Setup

- PVFS cluster configurations:
  - 10 clients, 10 combined I/O & metadata servers
  - 6 clients, 12 combined I/O & metadata servers
- Luster cluster configurations:
  - 10 clients, 10 I/O servers, 1 metadata server
  - 6 clients, 12 I/O servers, 1 metadata server
- Each client runs same workload for $\approx$600 s
- Faults injected on single server for 300 s
- All workload & fault combinations run 10 times

Outline

Diagnostic Algorithm

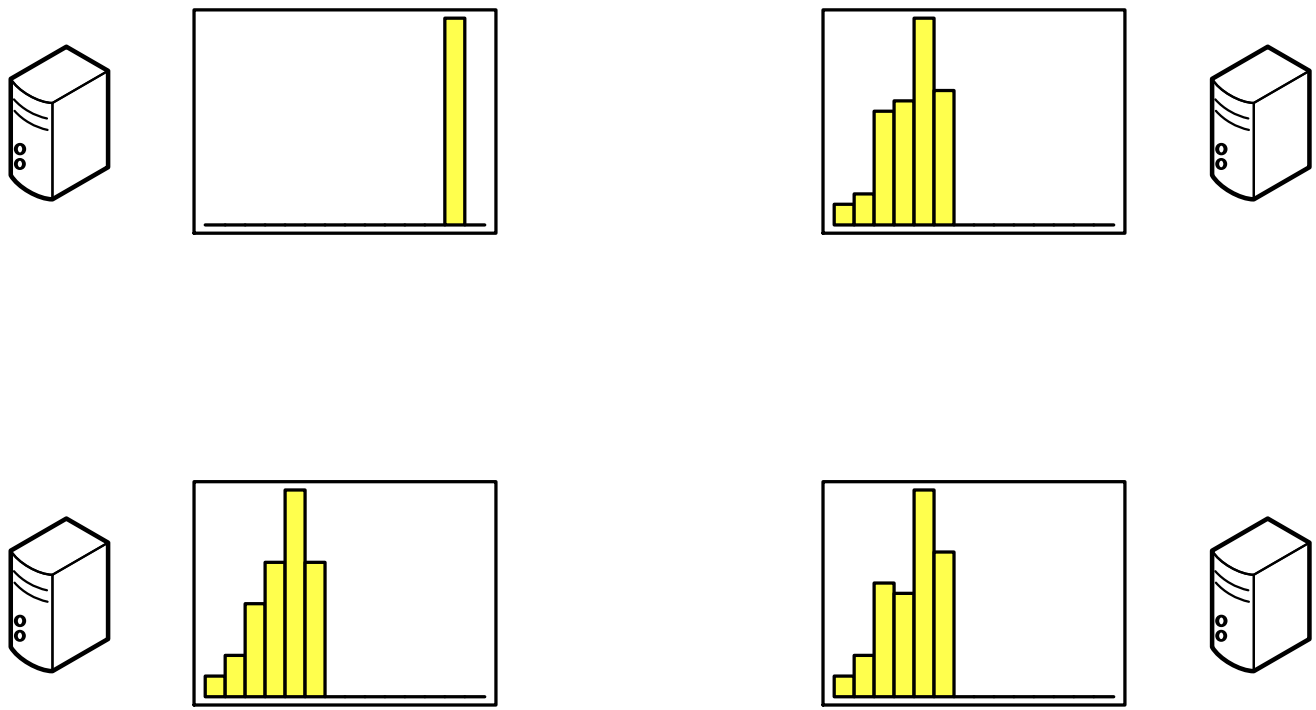
- Phase I: Node Indictment
  - Histogram-based approach (for most metrics)
  - Time series-based approach (congestion window)
  - Both use peer-comparison to indict faulty node
- Phase II: Root-Cause Analysis
  - Ascribes to root cause based on affected metrics

Phase I: Node Indictment (Histogram-Based)

- Peer-compare metric PDFs (histograms) across servers

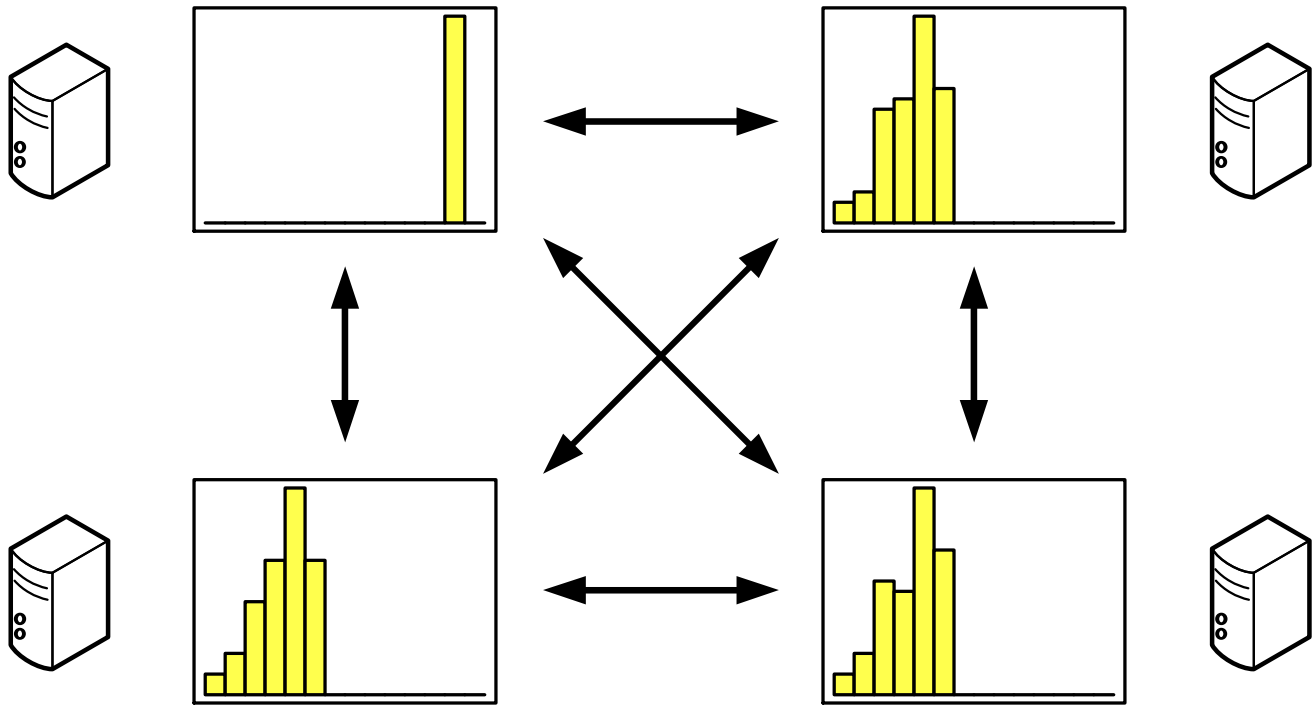
Phase I: Node Indictment (Histogram-Based)

- Peer-compare metric PDFs (histograms) across servers
  - Compute PDF of metric for each server over sliding window



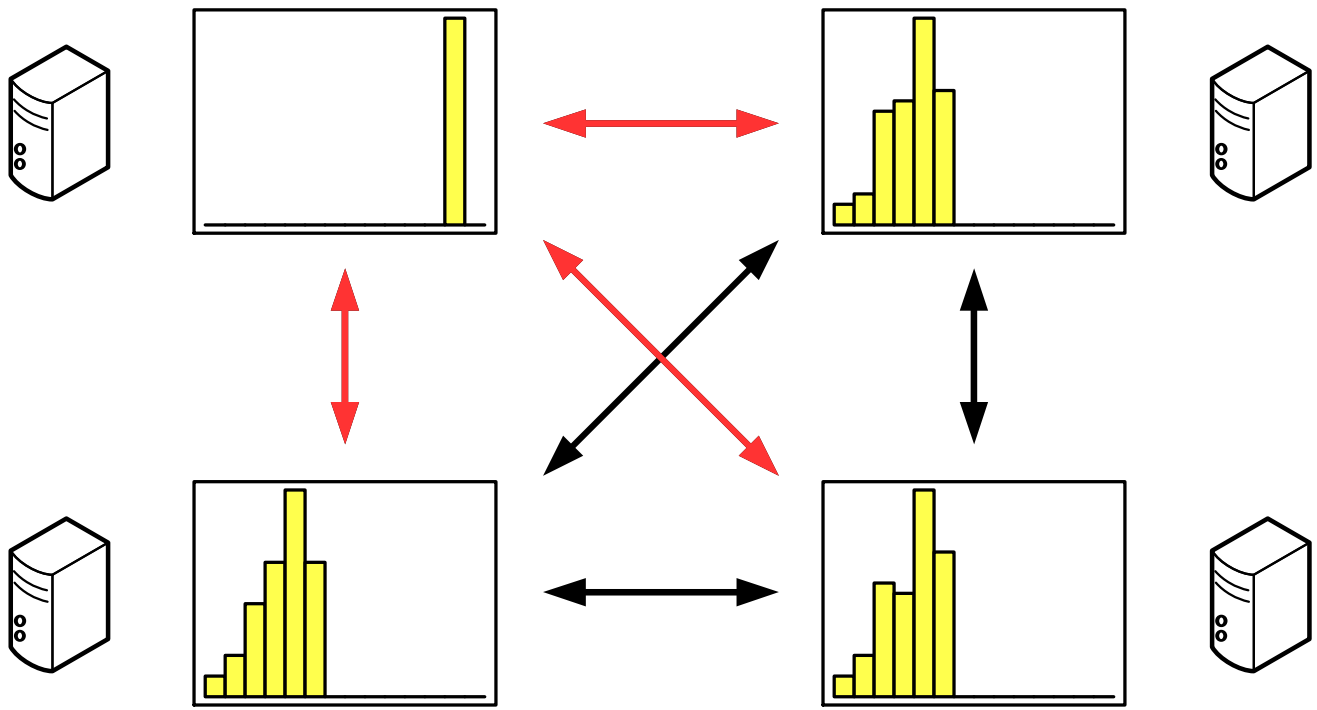
Phase I: Node Indictment (Histogram-Based)

- Peer-compare metric PDFs (histograms) across servers
  - Compute PDF of metric for each server over sliding window
  - Compute Kullback-Leibler divergence for each server pair



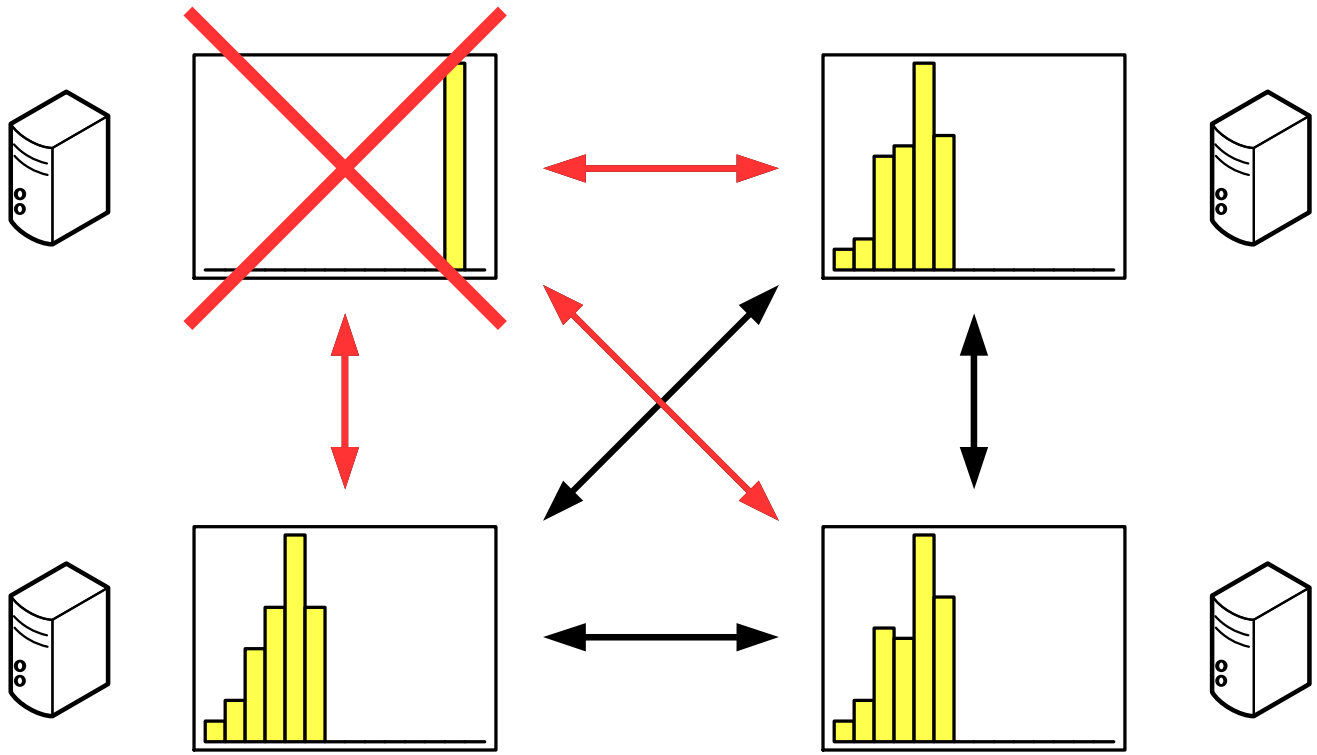
Phase I: Node Indictment (Histogram-Based)

- Peer-compare metric PDFs (histograms) across servers
  - Compute PDF of metric for each server over sliding window
  - Compute Kullback-Leibler divergence for each server pair
  - Flag pair anomalous if its divergence exceeds threshold



Phase I: Node Indictment (Histogram-Based)

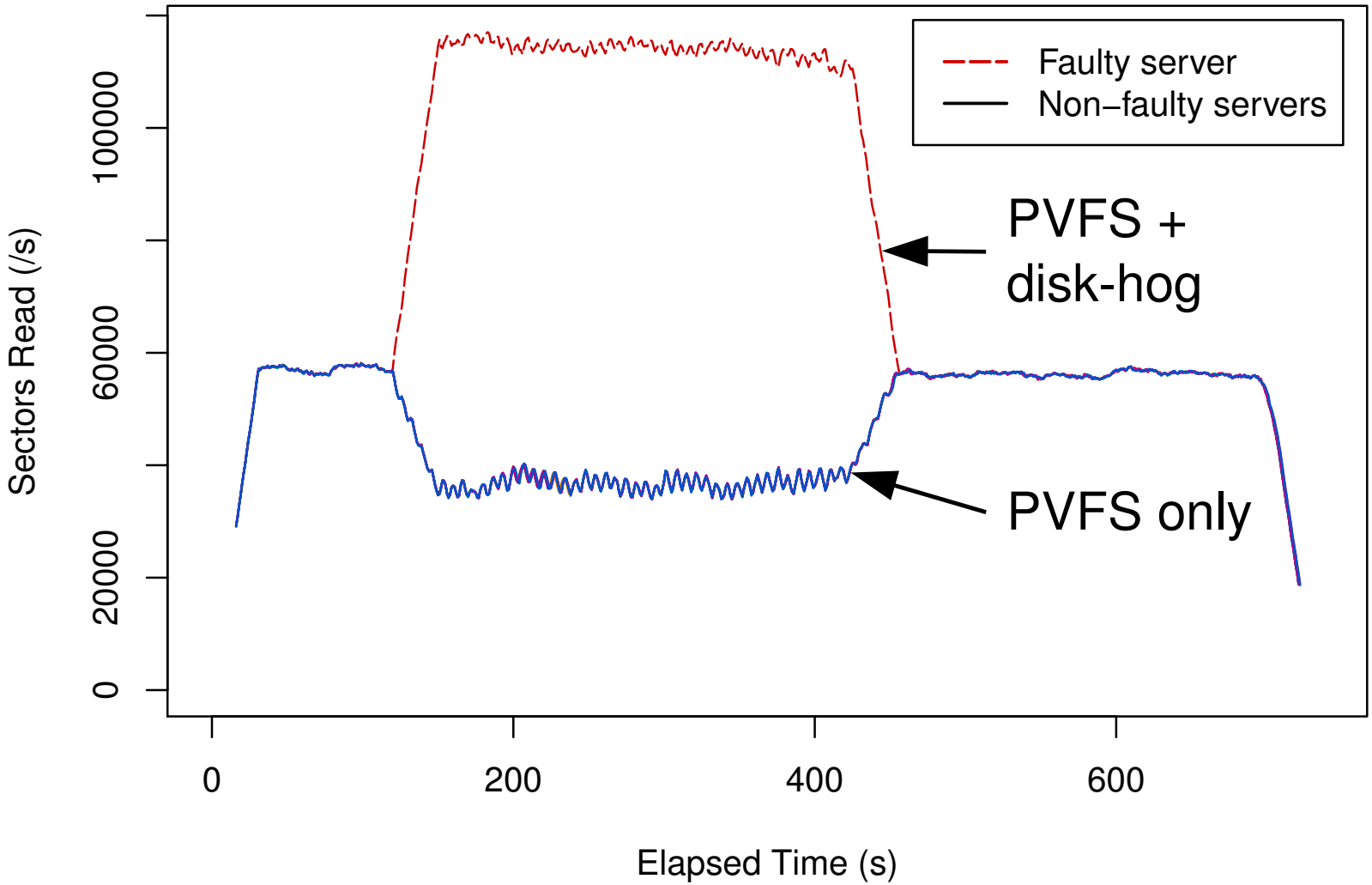
- Peer-compare metric PDFs (histograms) across servers
  - Compute PDF of metric for each server over sliding window
  - Compute Kullback-Leibler divergence for each server pair
  - Flag pair anomalous if its divergence exceeds threshold
  - Flag server if over half of its server pairs are anomalous



Threshold Selection

- Fault-free training session (stress test)
  - Run ddw, ddr, & postmark under fault-free conditions
  - Find minimum threshold that eliminates all anomalies
- Histogram comparison uses per-server thresholds
  - Captures performance profile of each server
  - Important to train on each cluster & file system
- Train on performance-stressing workloads only
  - Metrics deviate most when servers are saturated
  - Less intense workloads have better coupled behavior

Example: PVFS Throughput (Disk-Hog Fault)



- Throughput diverges due to disk-hog on faulty server

Phase II: Root-Cause Analysis

- Build table of metrics & faults affecting them:

Storage Throughput:	Storage Latency:
disk-hog	disk-hog disk-busy
Network Throughput:	**Network Congestion:**
network-hog packet-loss (ACKs only)	network-hog packet-loss

- Derive checklist that maps divergent metrics to cause
  - Infers resource responsible
  - Determines mechanism by which resource faulted

Checklist for Root-Cause Analysis

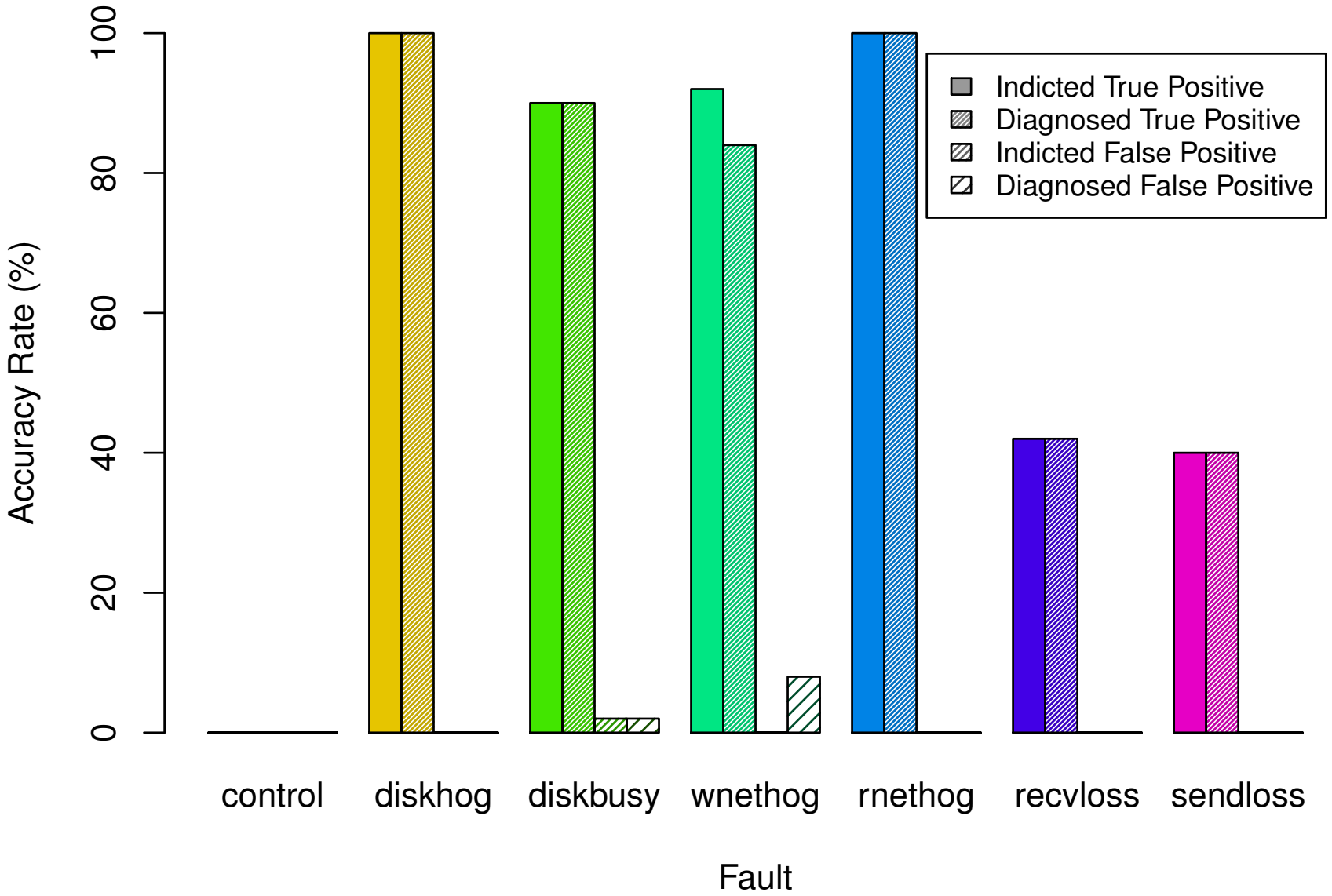
Peer-divergence in . . .

Storage throughput?	Yes: disk-hog fault No: next question
Storage latency?	Yes: disk-busy fault No: . . .
Network throughput?*	Yes: network-hog fault No: . . .
Network congestion?	Yes: packet-loss fault No: no fault discovered

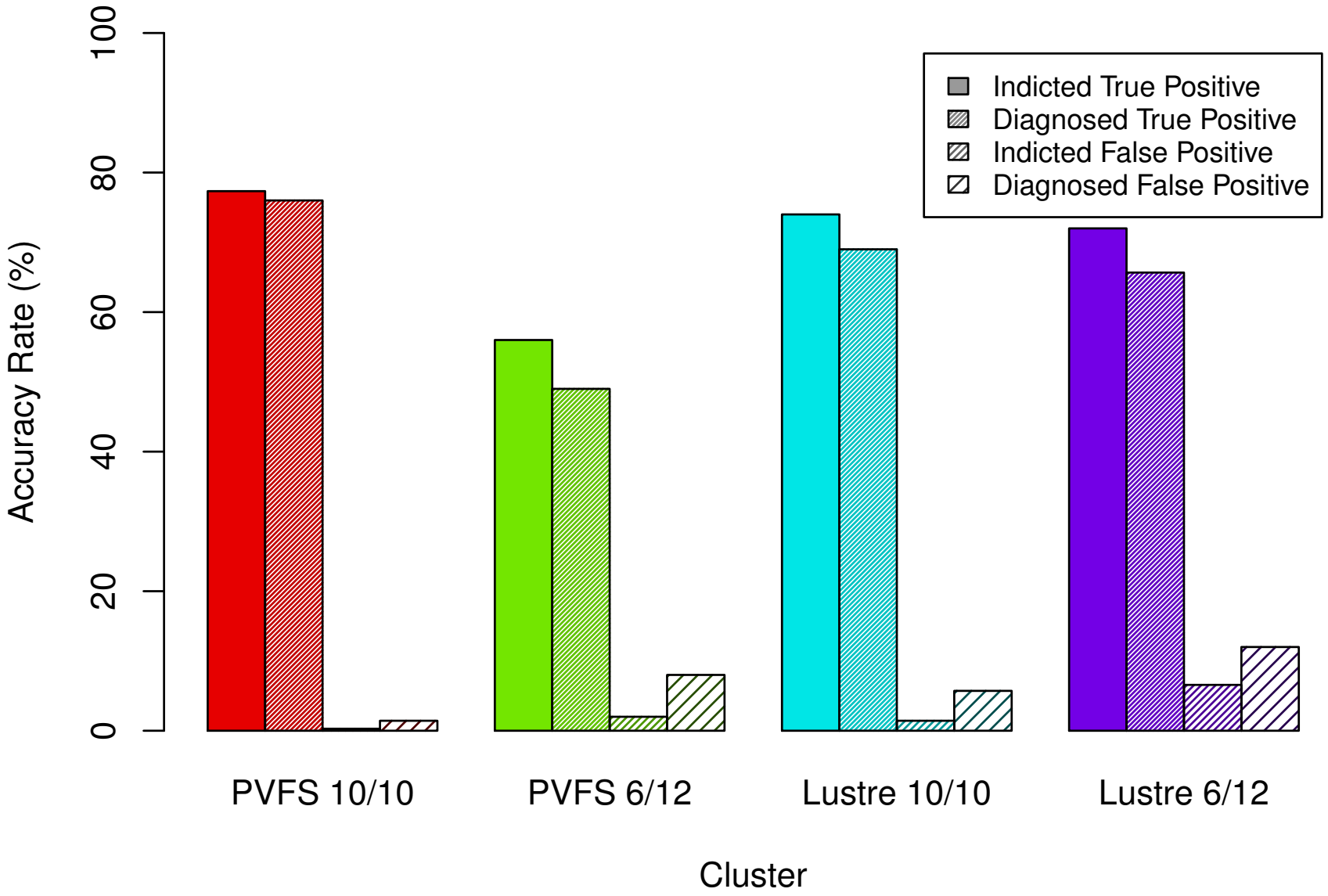
*Must diverge in both receive & transmit, or in absence of congestion

Outline

Results: Single Cluster



Results: Aggregate



Results Summary

- True-positives inconsistent across faults
  - Some faults are not observable for all workloads
  - Minimal performance effect where not observable
- True- & false-positives inconsistent across clusters
  - Algorithm sensitive to imprecise thresholds
  - Rank metrics based on degree of dissimilarity
- Strategy is promising in general
- Instrumentation overhead
  - $<$ 1% increase in workload runtime, negligible

Outline

Future Work

- **Analysis:** Improve diagnosis accuracy rates
  - Make analysis robust to imprecise thresholds
- **Real-world data:** Deploy on a production system
  - Validate technique on real workloads, at scale
- **Coverage:** Expand target problem class
  - Other sources of performance & non-performance faults
- **Instrumentation**: Expand instrumentation
  - Additional black-box metrics, request sniffing & tracing

Summary

- Problem diagnosis in parallel file systems
  - Illustrates use of OS-level metrics in diagnosis
  - Leverages peer-comparison to identify faulty nodes
  - Demonstrates root-cause analysis by metrics affected
- Diagnosis method is applicable to existing deployments
  - Instrumentation is minimally invasive, low overhead
  - Fault-free training with stress tests

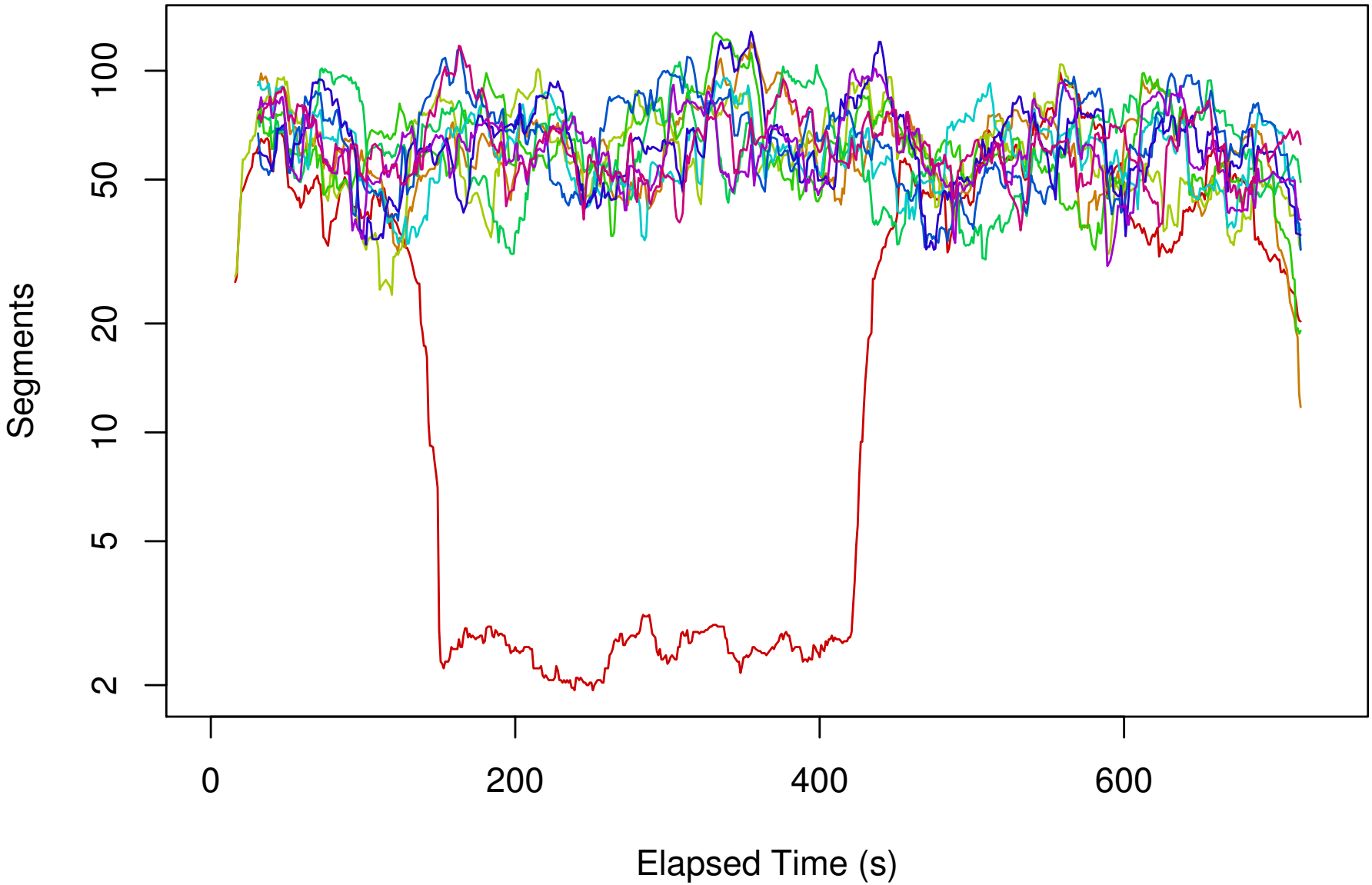
Peer-Comparison Scalability

- Number of comparisons: $\frac{n(n-1)}{2} \Longrightarrow O(n^2)$
- Insight: Don't need to compare one node against all
- Proposed solution:
  - Establish $n-k$ partitions with k servers
  - Perform peer-comparisons among servers in each partition
  - Repartition with a different grouping for each window
- Solution comparisons: $(n-k)\frac{k(k-1)}{2} \Longrightarrow O(n)$

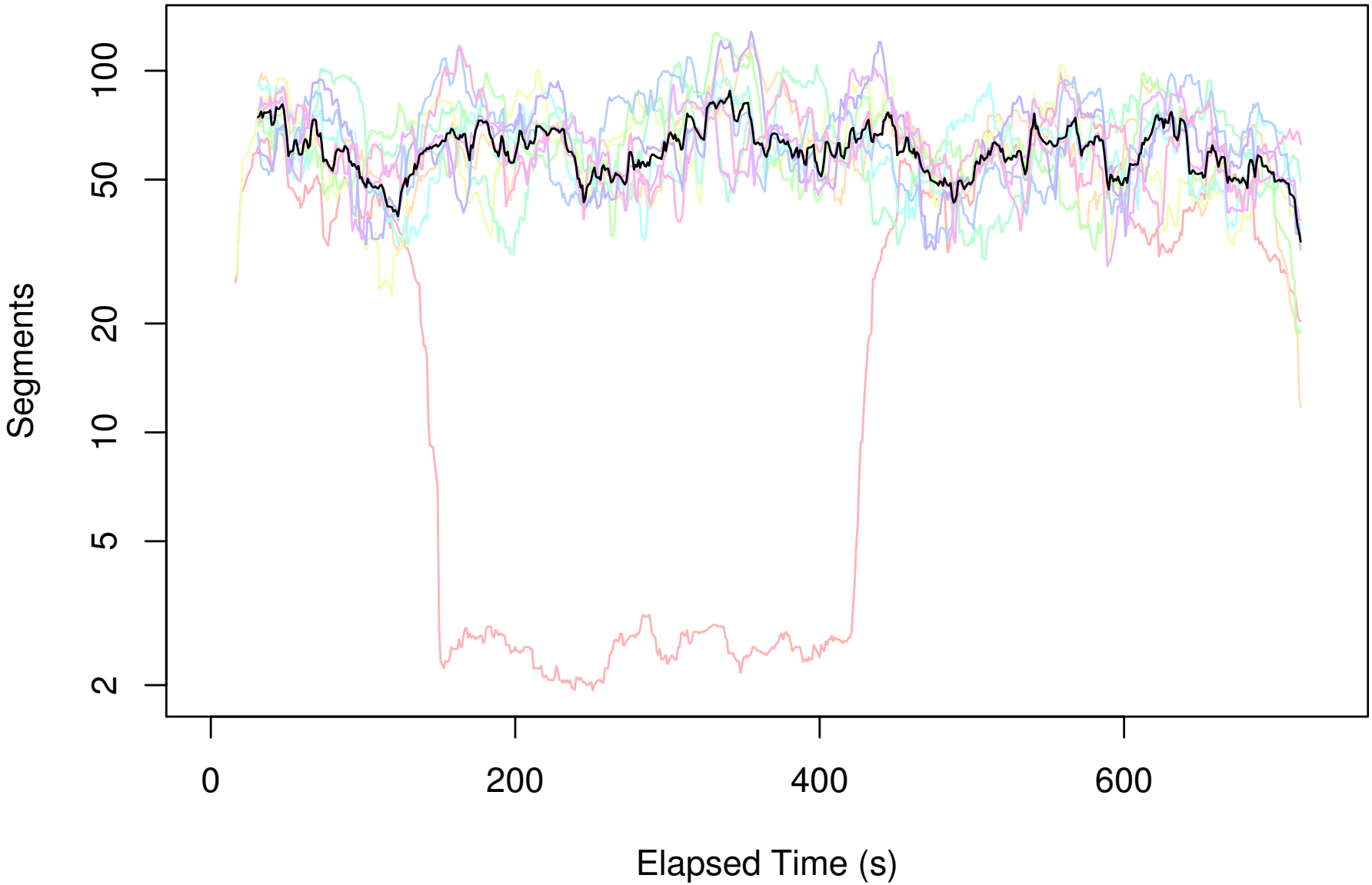
Congestion Window Problem

- No closely-coupled peer behavior
  - cwnd is intentionally noisy under normal conditions
  - Synchronized connections can't fully use link capacity
  - Can't compare histograms, too much variance
- Congestion window packet-loss heuristic:
  - TCP responds to packet-loss by halving cwnd
  - Exponential decay after multiple loss events
  - Log scale: Each loss results in linear cwnd decrease

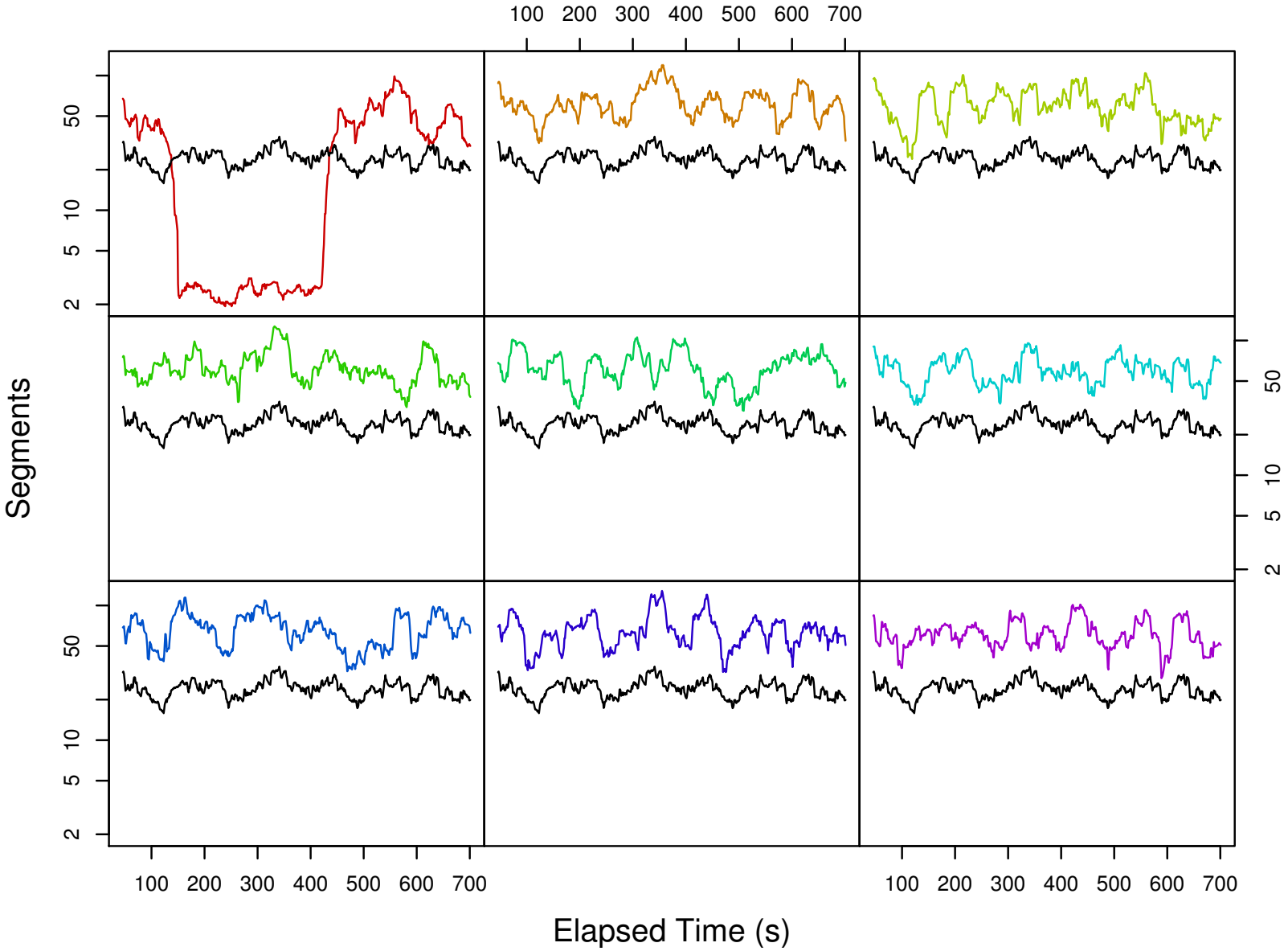
Time Series Comparison Example



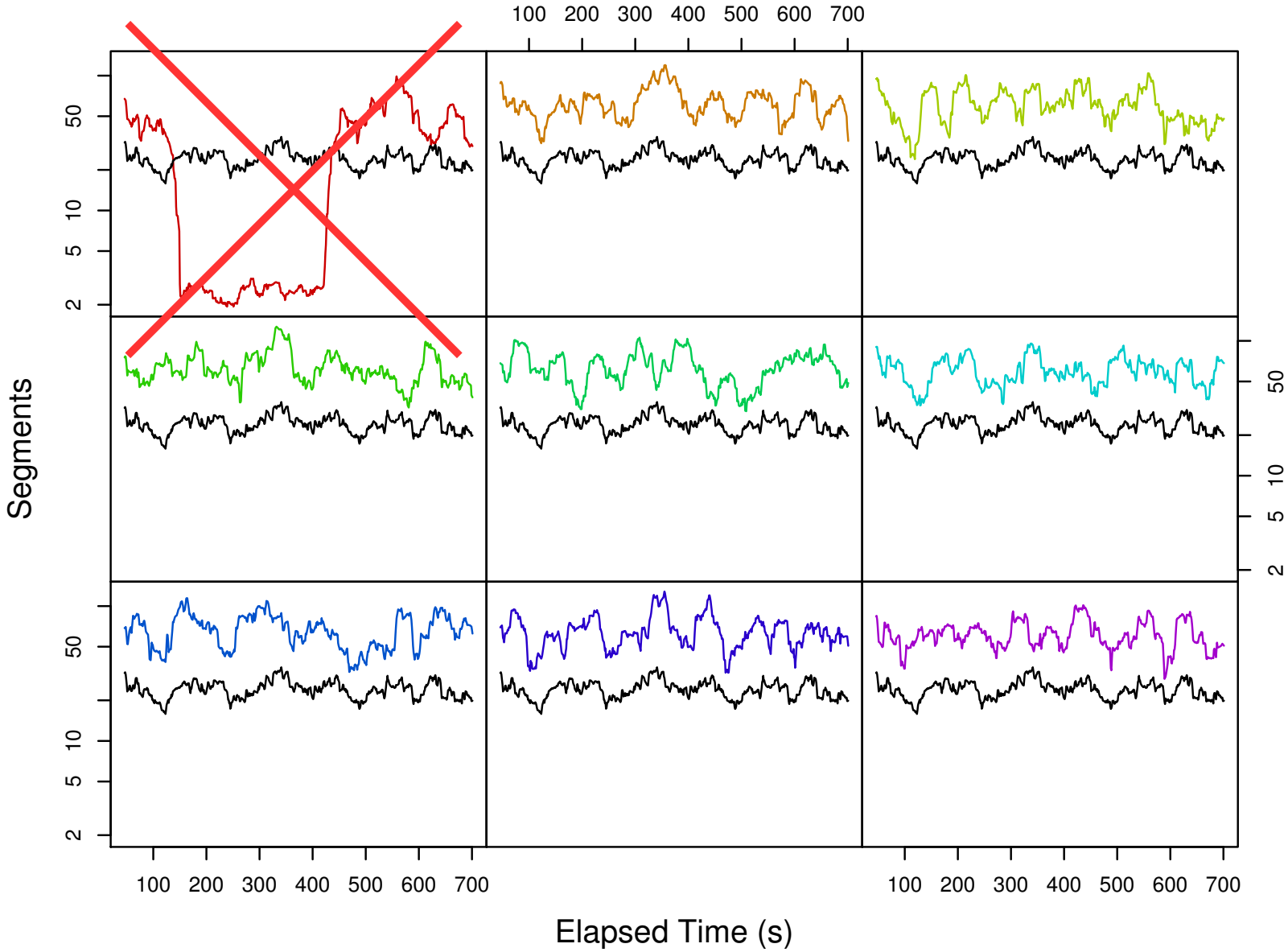
Time Series Comparison Example



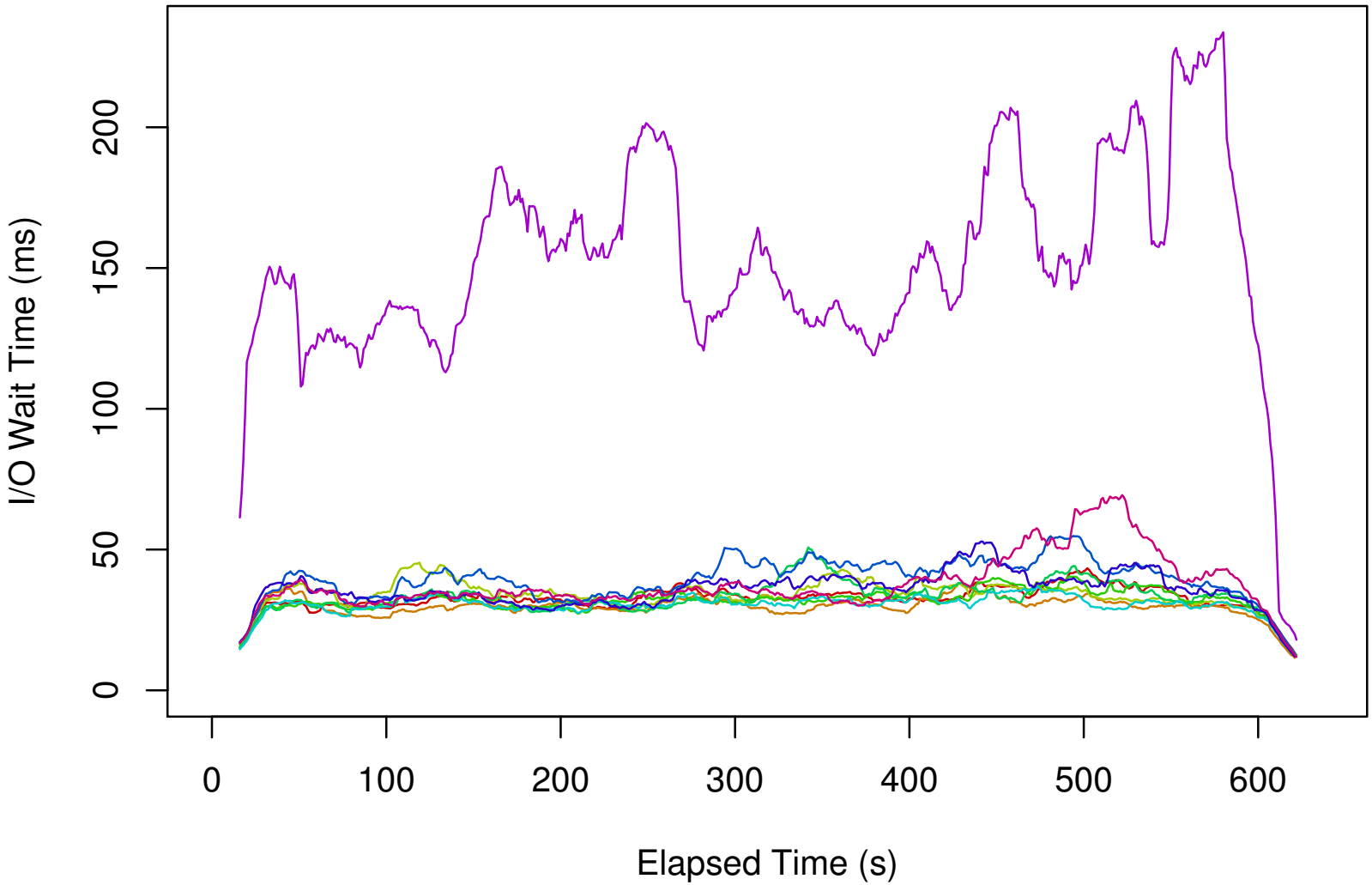
Time Series Comparison Example



Time Series Comparison Example

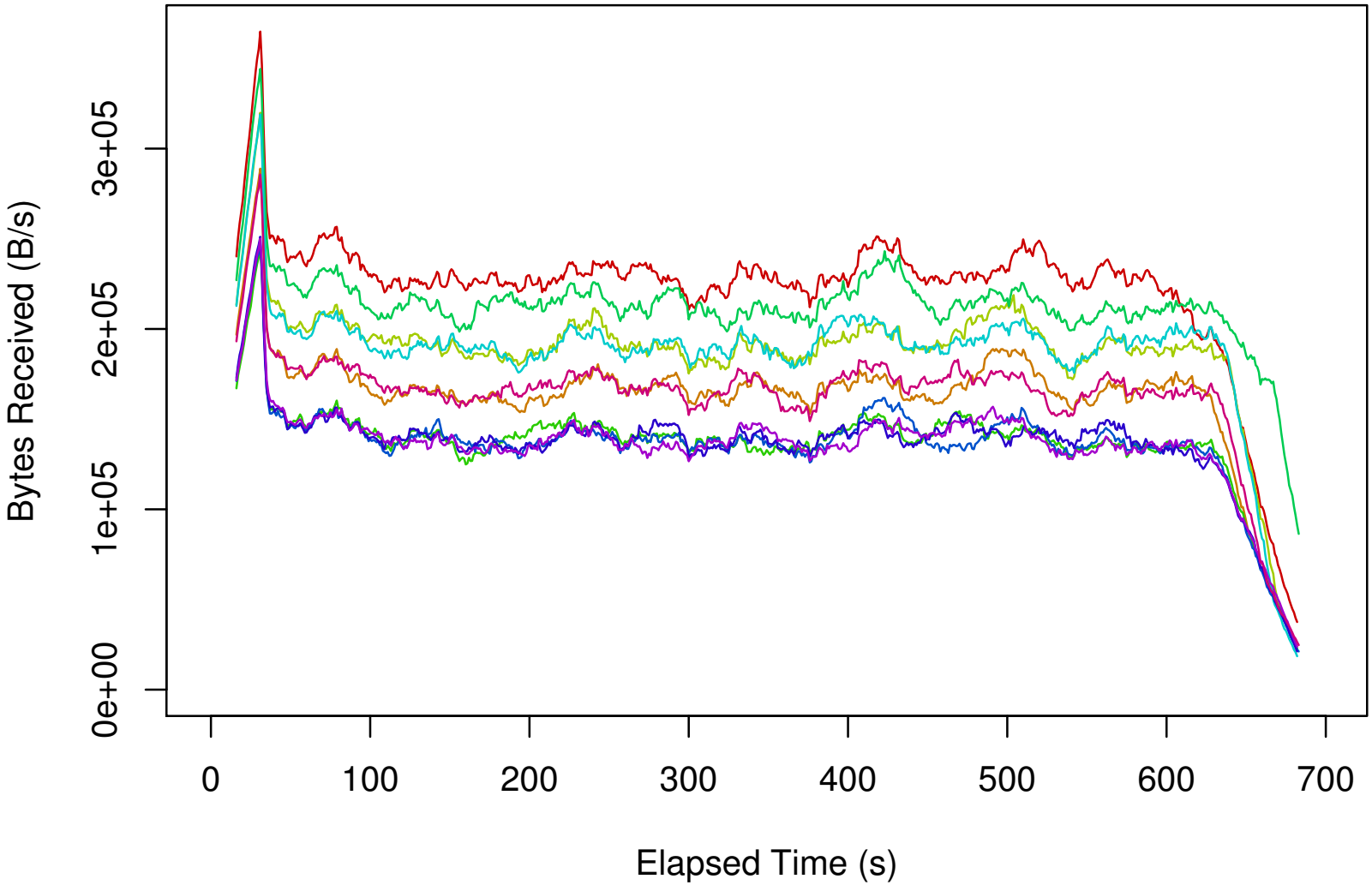


Heterogeneous Hardware (ddr)



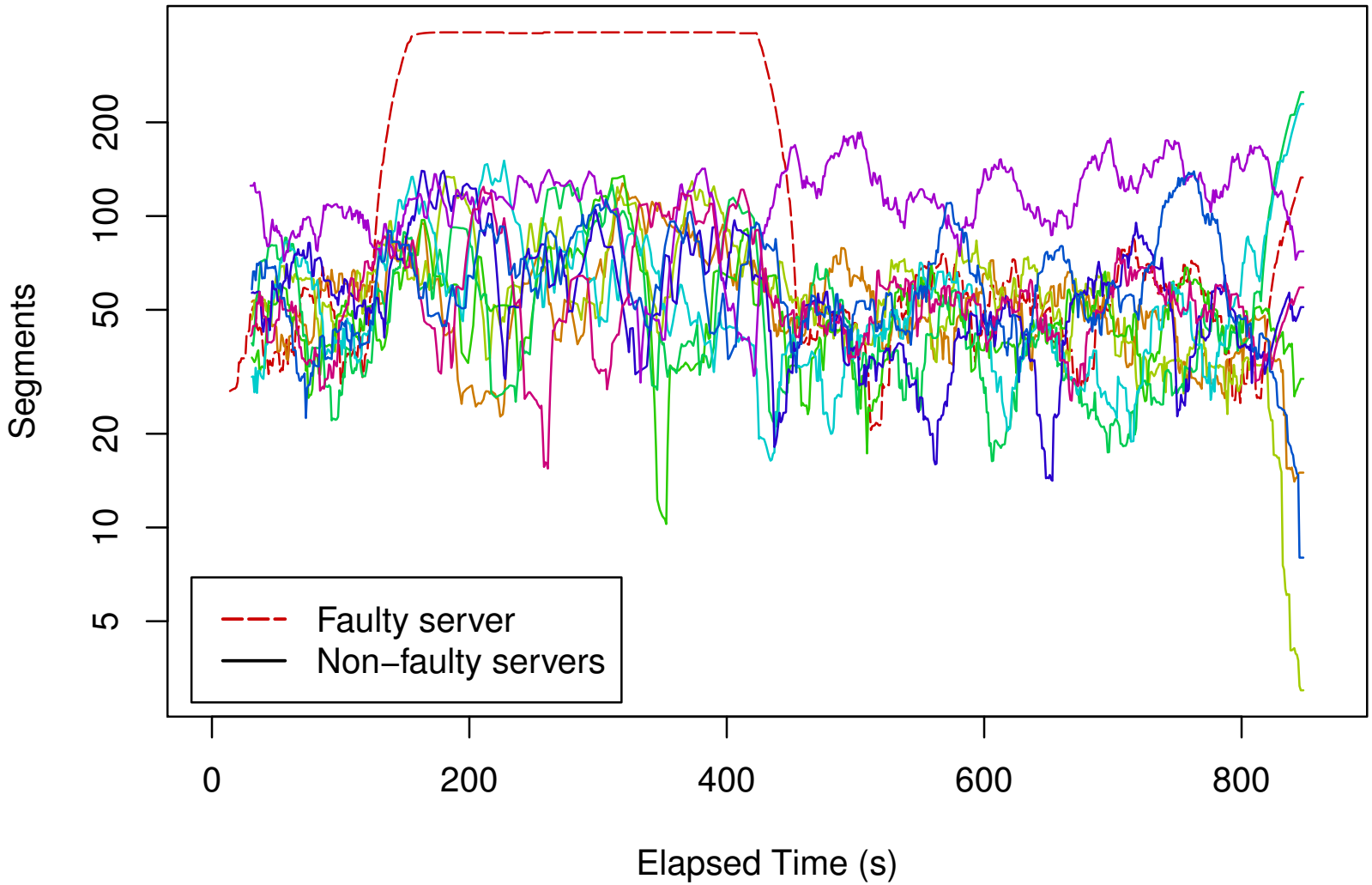
- Disks are same model, have different performance profiles

Load Imbalances (postmark)



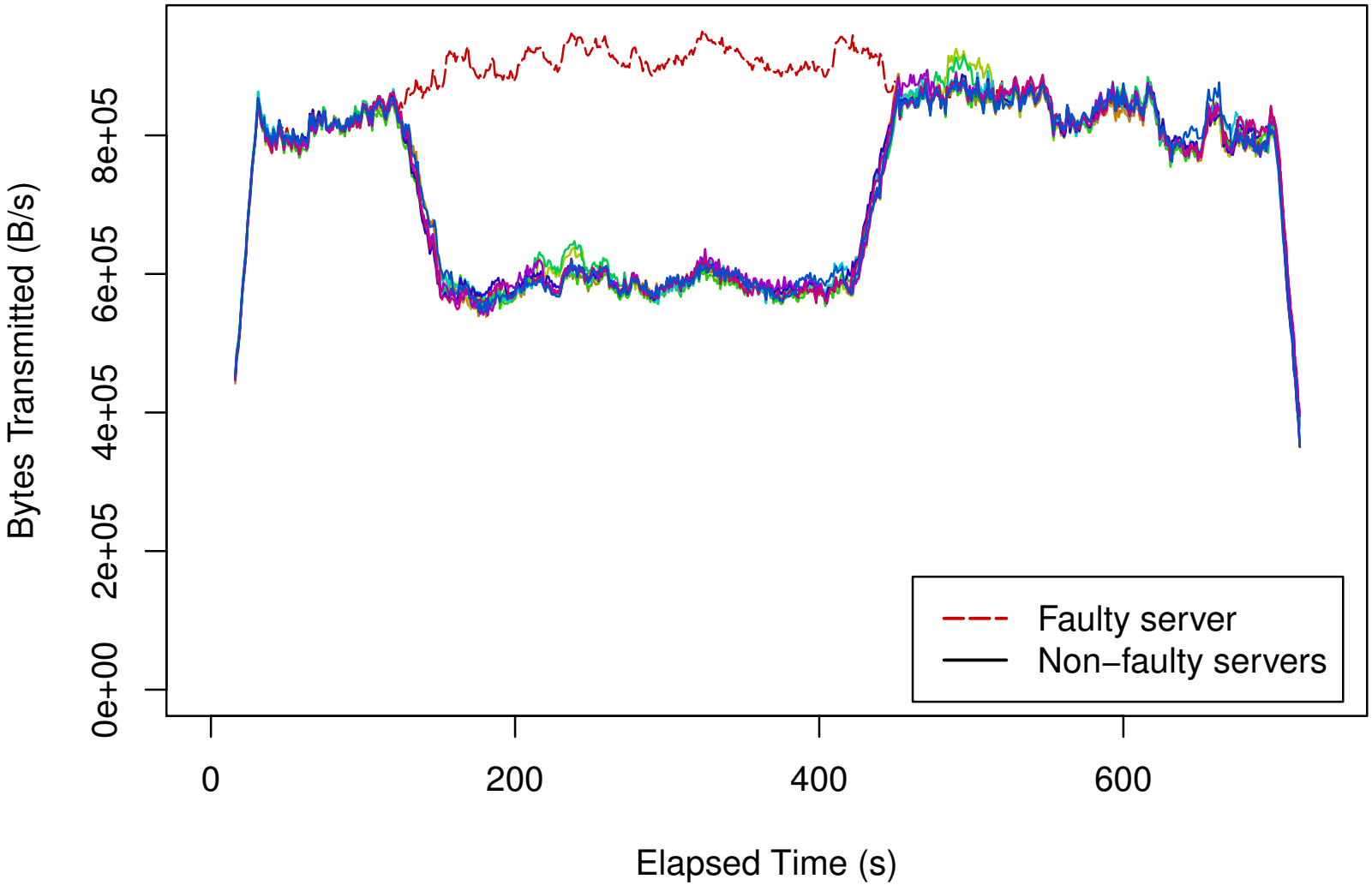
- "/" on one metadata server, all path lookups go there

Cross-Resource Influence (ddr)



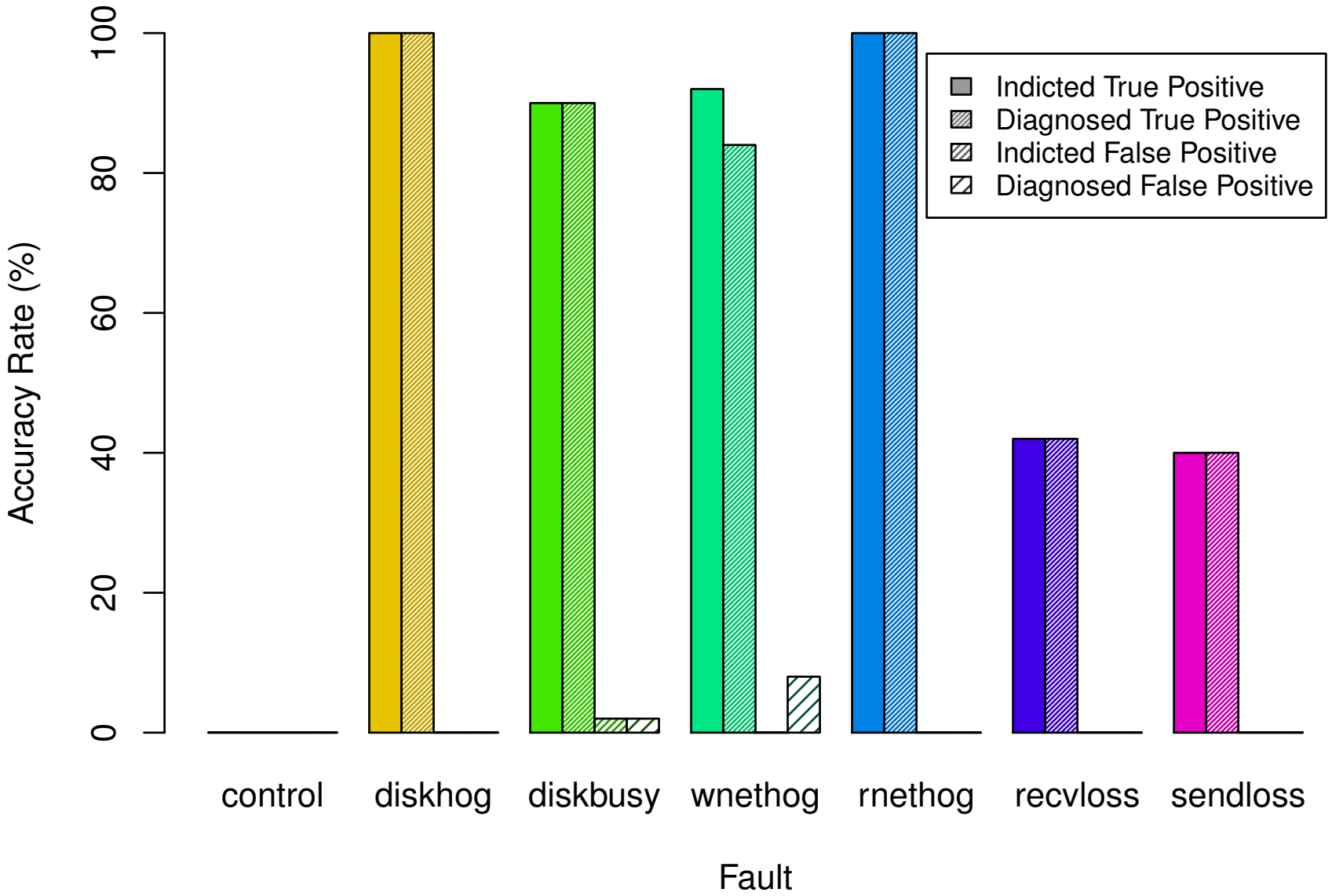
- Disk-busy effect on server cwnd, unintentional sync

Delayed ACKs (ddw)

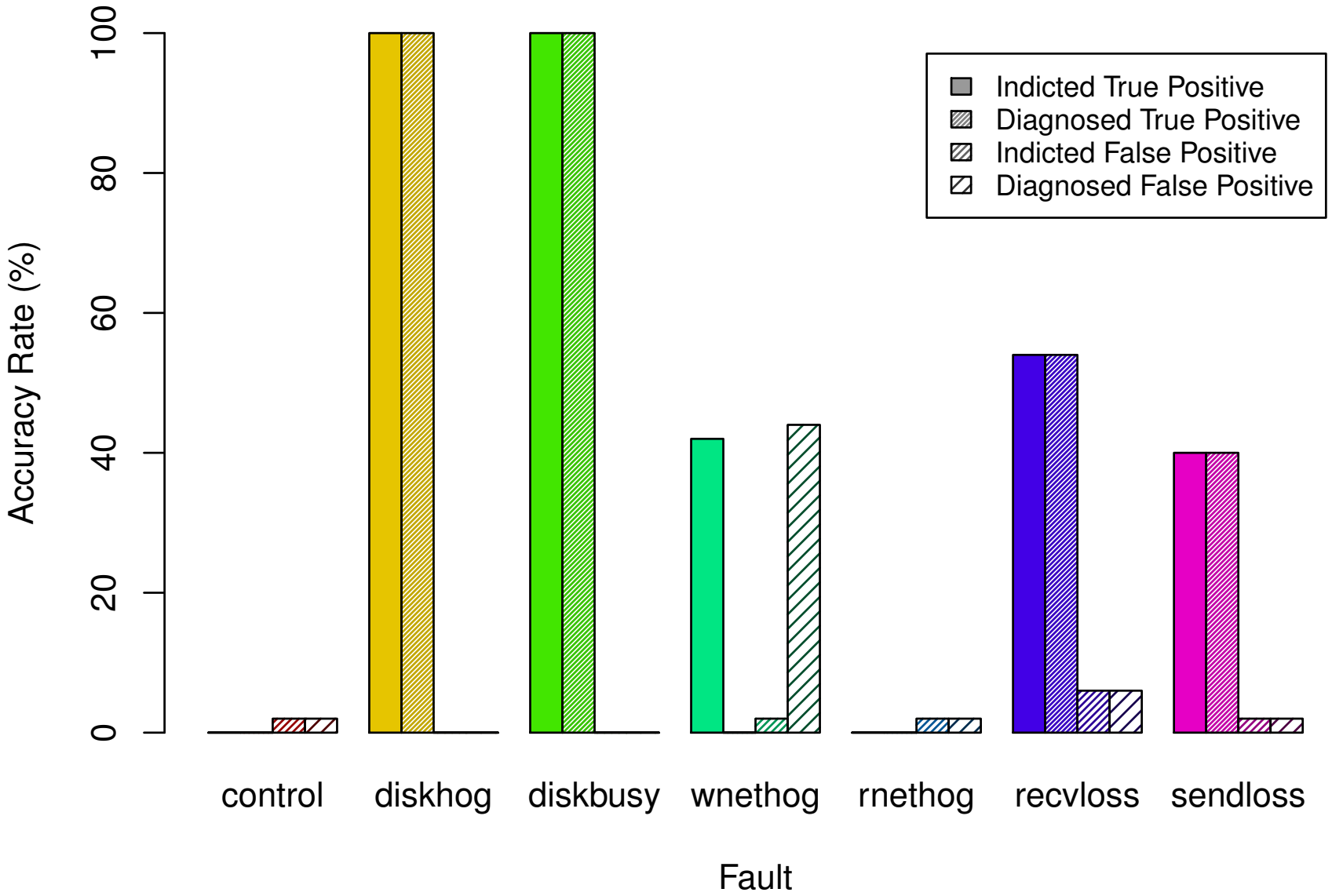


- Packet-loss fault may also deviate network throughput

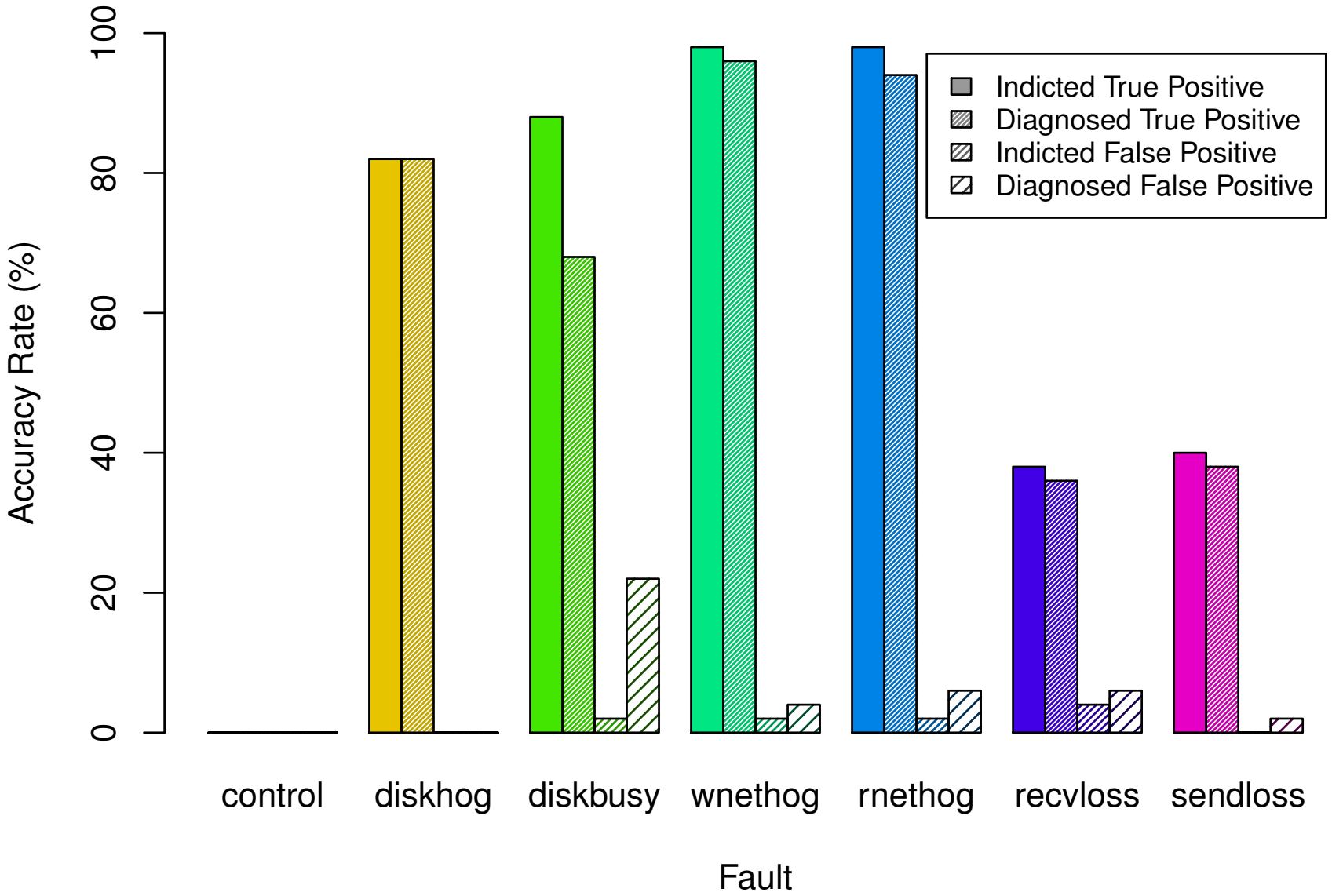
Results: PVFS 10/10 Cluster



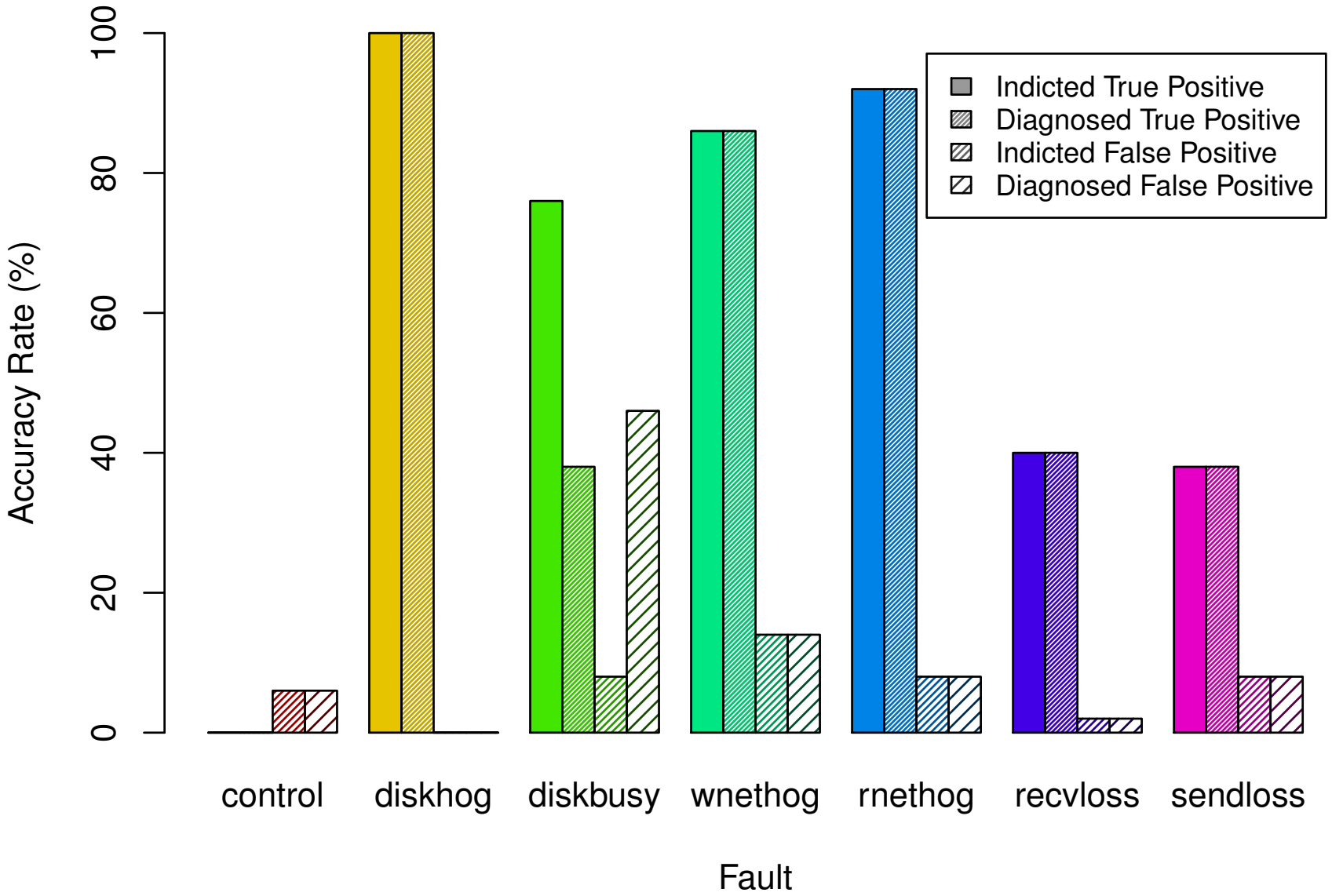
Results: PVFS 6/12 Cluster



Results: Lustre 10/10 Cluster



Results: Lustre 6/12 Cluster



Results: Aggregate