Berkeley Ninja Architecture

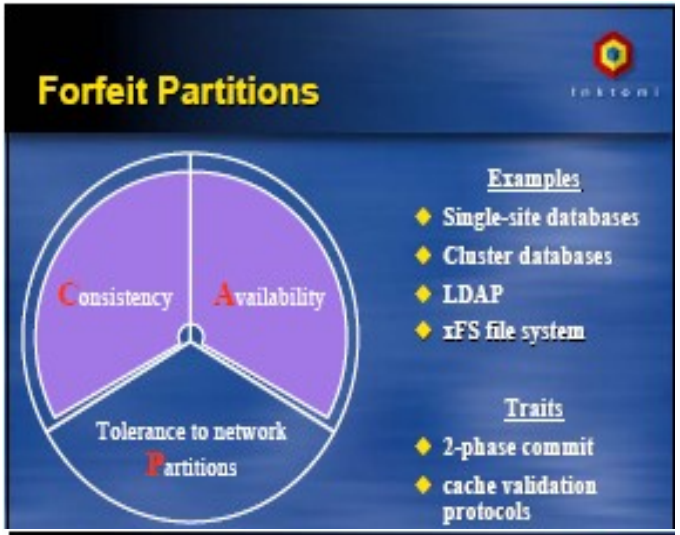
ACID vs BASE

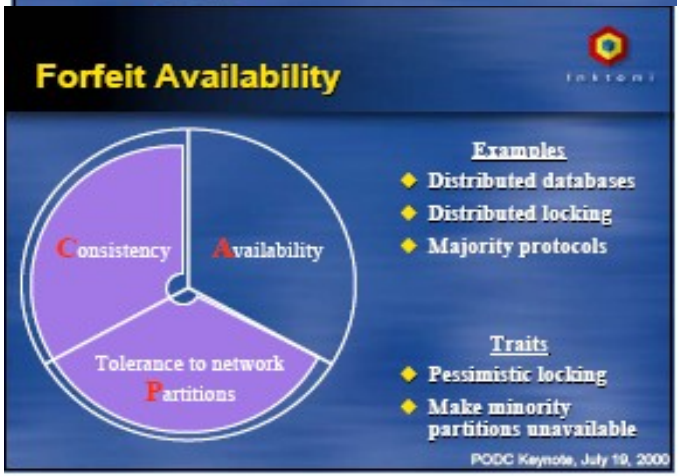
1. Strong Consistency
2. Availability not considered
3. Conservative

--> Traditional databases

1. Weak consistency
2. Availability is a primary design element
3. Aggressive

--> large-scale distributed systems

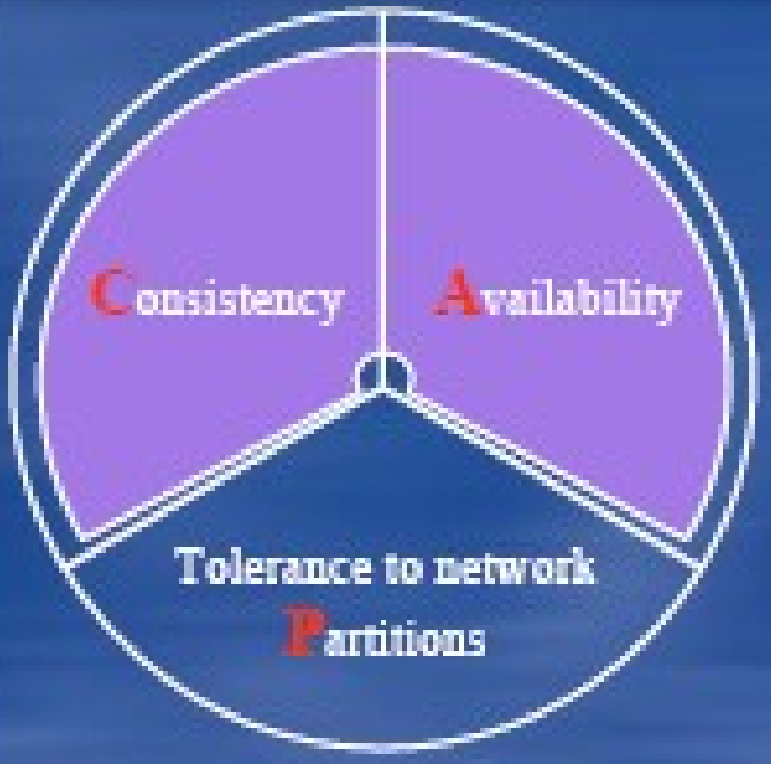




CAP Theorem:

Of the three different qualities(network partitions, consistency, availability), at most *two* of the three qualities can be maintained for any given system.

Forfeit Partitions



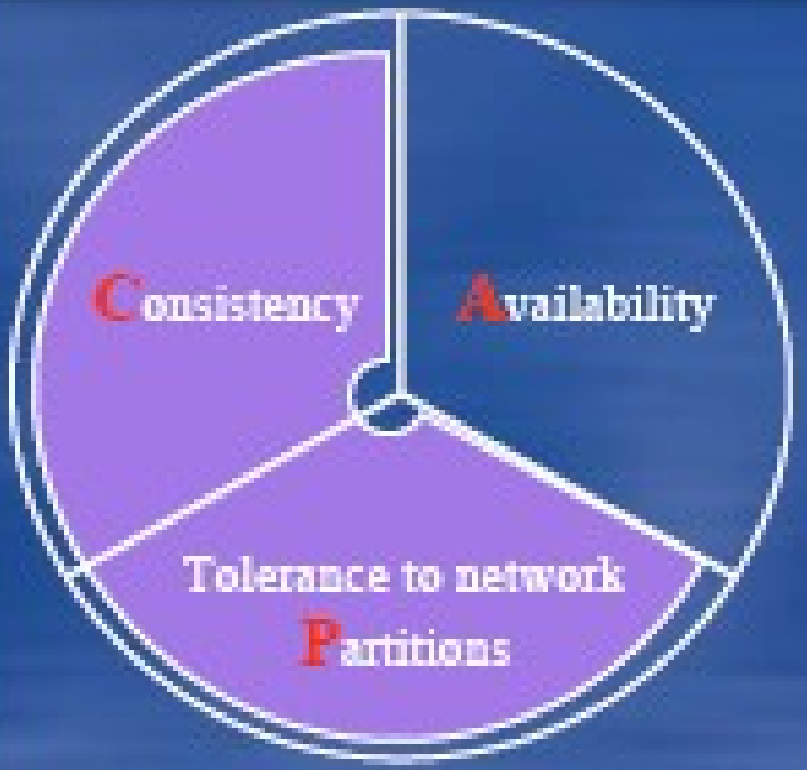
Examples

- Single-site databases
- Cluster databases
- LDAP
- xFS file system

Traits

- 2-phase commit
- cache validation protocols

Forfeit Availability



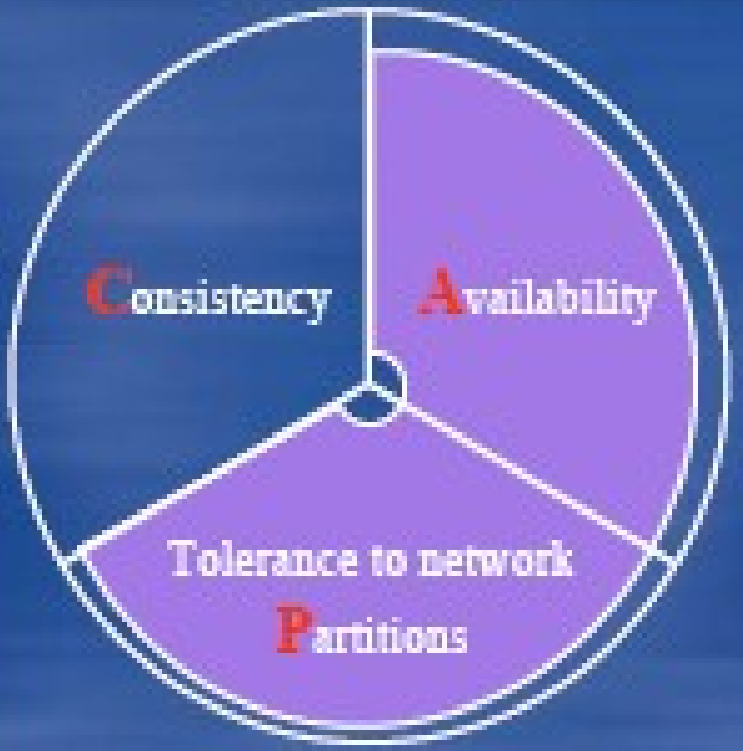
Examples

- Distributed databases
- Distributed locking
- Majority protocols

Traits

- Pessimistic locking
- Make minority partitions unavailable

Forfeit Consistency



Examples

- Coda
- Web cachinge
- DNS

Traits

- expirations/leases
- conflict resolution
- optimistic

Boundary between entities

1. Remote Procedure Calls

    --> The way it is used currently is not sustainable for larger systems

2. Trusting the other side

    --> need to check arguments before executing RPC

3. Multiplexing between many different clients

    --> How this is done effects boundary definition

Key Messages

1. Parallel programming tends to avoids the notion of availability, online evolution, checkpoint/restart (although currently this is changing)
2. For Robustness in distributed systems, we must think probabilistically about system design qualities
3. Message-Passing seems to be most effective solution, as boundaries must be clearly defined.
4. Need to have more support for partial failure, graceful degradation, and parallel I/O

Discussion

1. Do you believe that techniques applied in distributed database community also can apply to large-scale distributed systems? Or does a completely new approach need to be taken?
2. This work was presented in 2000. Do the principles of robustness apply for today's distributed systems?
3. Do you agree with the notion that without clear boundaries, large-scale distributed systems will remain unmaintainable?

Cumulus: A FileSystem Backup to the Cloud

Back-up to the cloud: Thin or thick?

- **Problem: Price back-up attractively:**
  - Minimize cost of storage
  - Minimize cost of shipping data
- **Solution:**
  - Integrated solutions
  - Back-up specific SW on clients and data centers
  - Thick Cloud
- **Portability?**
- **Provider Monoply?**

Cumulus Design Choice

1. Minimal Interface(4 commands)
2. Highly portable
3. Efficient (through simulation)
4. Practicality (Amazon S3 prototype)

A Cloud Computing Design Decision

Software as a Service(thick cloud)

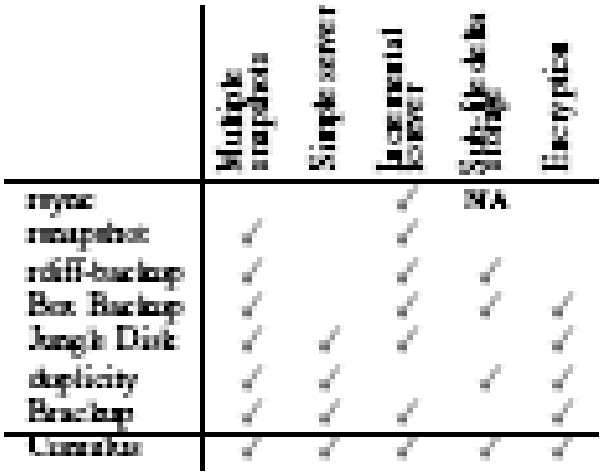
1. Highly specific implies Better Performance
2. Reduced Flexibility

Utility Computing(thin cloud)

1. Abstract
2. Portable
3. Less Efficient

What is the right choice? And is there a right choice?

Comparison of Cumulus to Other Systems



System	Storage	Upload	Operations
Jungle Disk	≈ 2 GB	1.26 GB	30000
	$0.30	$0.126	$0.30
Brackup	1.340 GB	0.760 GB	9027
(default)	$0.201	$0.076	$0.090
Brackup	1.353 GB	0.713 GB	1403
(aggregated)	$0.203	$0.071	$0.014
Cumulus	1.264 GB	0.465 GB	419
	$0.190	$0.047	$0.004

- Simplest backup system that most will be familiar with: tar, gzip
- Others: rsync, rdiff-backup, Box Backup, Jungle Disk, Duplicity, Brackup

--> In contrast to all other systems, Cumulus supports multiple snapshots, simple servers, incremental backups , sub-file disk storage, and encryption.

Simple User Commands

Get : given a pathname, retrieve the contents of a file from the server

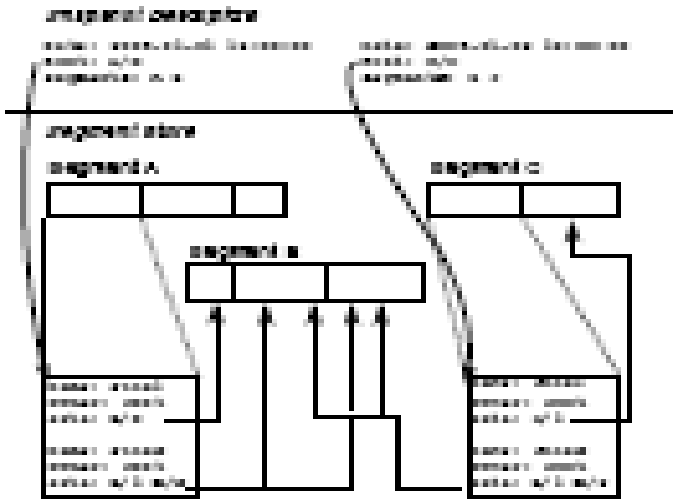
Put: Store the complete file on the server, given its pathname

List: Get the names of files stored on server

Delete: Remove the given file from the server, reclaiming it's space

With these four commands, one can support incremental backups on a wide variety of systems.

Snapshot Storage Format



1. The above illustrates how snapshots are structured on a storage server, using Cumulus.

2. Two different snapshots are taken(on two different days), and each snapshot contains two separate files (labeled file1 and file2)

3. The file1 changes between the two days, while file2 is the same between the two snapshots.

4. The snapshot descriptor contains the date, root, and its corresponding segments.

Cumulus Research Questions

What is the **penalty of using a thin cloud service** with a very simple storage interface compared to a more sophisticated service?

What are the **monetary costs** for using remote backup for two typical usage scenarios? How should remote backup strategies adapt to minimize monetary costs as the ratio of network and storage prices varies?

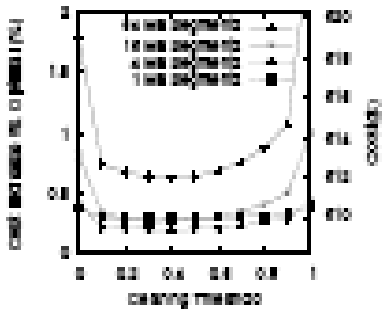
How does our prototype implementation **compare with other backup systems?** What are the additional benefits (e.g., compression, sub-file incrementals) and overheads (e.g., metadata) of an implementation not captured in simulation? What is the performance of using an online service like Amazon S3 for backup?

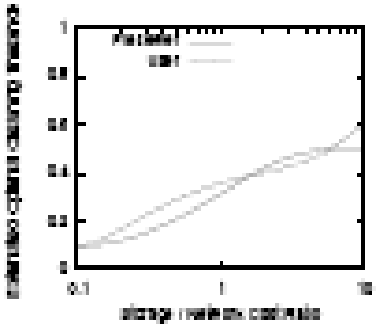
Experimental Setup for Simulation

- Two traces are considered as representative workloads for simulation: file-server and user
- For both workloads, traces contain a daily record of meta-data of all files
- Thin service model is compared to optimal backup, where only the needed storage/transfer is done, and no more.
- There are justifiable reasons that Cumulus does not try to store each file in one segment because of the other design goals it aims for(encryption, compression, etc.)
- Statistics are established for both workloads, as shown below.

	Fileserver	User
Duration (days)	157	223
Entries	26673083	122007
Files	24344167	116426
File Sizes		
Median	0.996 KB	4.4 KB
Average	153 KB	21.4 KB
Maximum	54.1 GB	169 MB
Total	3.47 TB	2.37 GB
Update Rates		
New data/day	9.50 GB	10.3 MB
Changed data/day	805 MB	29.9 MB
Total data/day	10.3 GB	40.2 MB

Establishing Cleaning Threshold

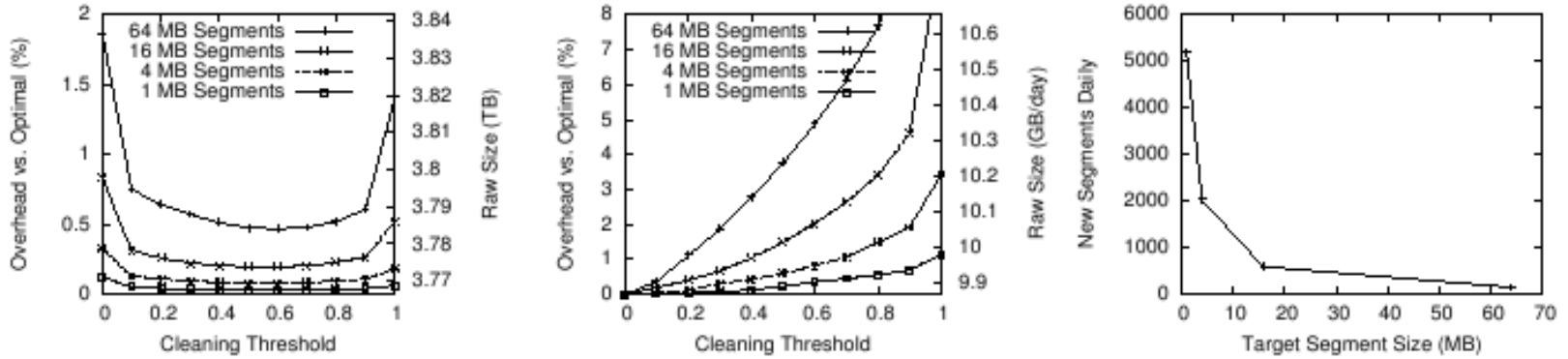




1. As the cost of storage increases, cleaning more aggressively gives advantage

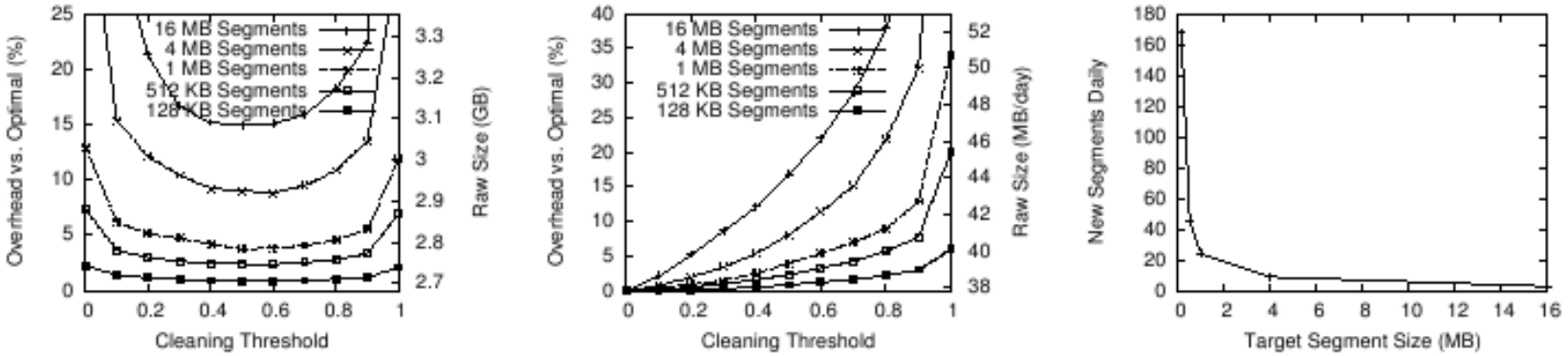
2. Ideal threshold stabilizes at .5 to .6, when storage is 10 times as expensive as network

Cumulus Experimental Simulation



(a) Average daily storage (b) Average daily upload (c) Average segments per day

Figure 3: Overheads for backups in the fileserver trace.



(a) Average daily storage (b) Average daily upload (c) Average segments per day

Figure 4: Overheads for backups in the user trace.

Broader Impact

“Can one build a competitive product economy around a cloud of abstract commodity resources, or do underlying technical reasons ultimately favor an integrated service-oriented architecture?”

→ On one hand, if Cumulus is to be accepted as a general solution for file system backup, many more application must be tested and simulated.

→ On the other hand, the need for standardization in the cloud is very important, and a solution like Cumulus should be adopted as quickly as possible.

Discussion Questions for Cumulus

1. Application-specific solutions vs. general light-weight, portable solutions?
2. Who are the users of Cumulus? Would such a backup tool be easy to pick up for a novice?
3. Is the interface provided adequate? Should there be more functionality?
4. Is the issue of security with backing up data adequately addressed?

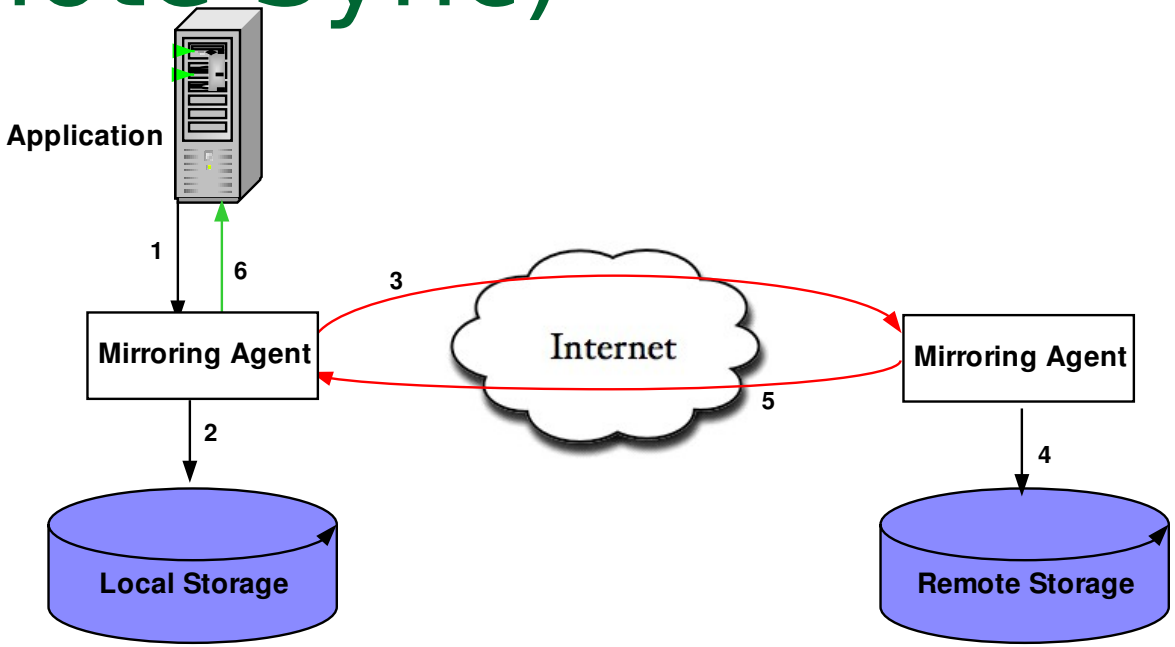
Smoke and Mirrors: Reflecting Files at a Geographically Remote Location Without Loss of Performance

USENIX 09

Why mirror data?

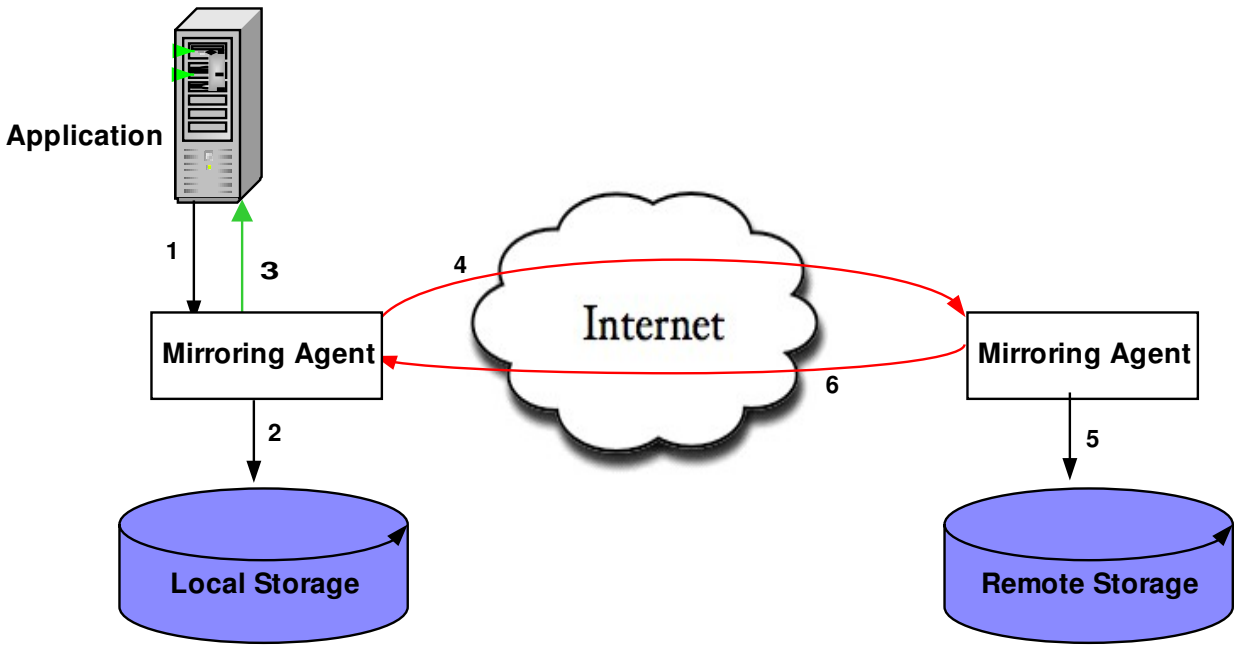
- Faster Access
- Better Availability
- Data protection against loss (Disaster Tolerance)

Synchronous Mirroring (Remote Sync)



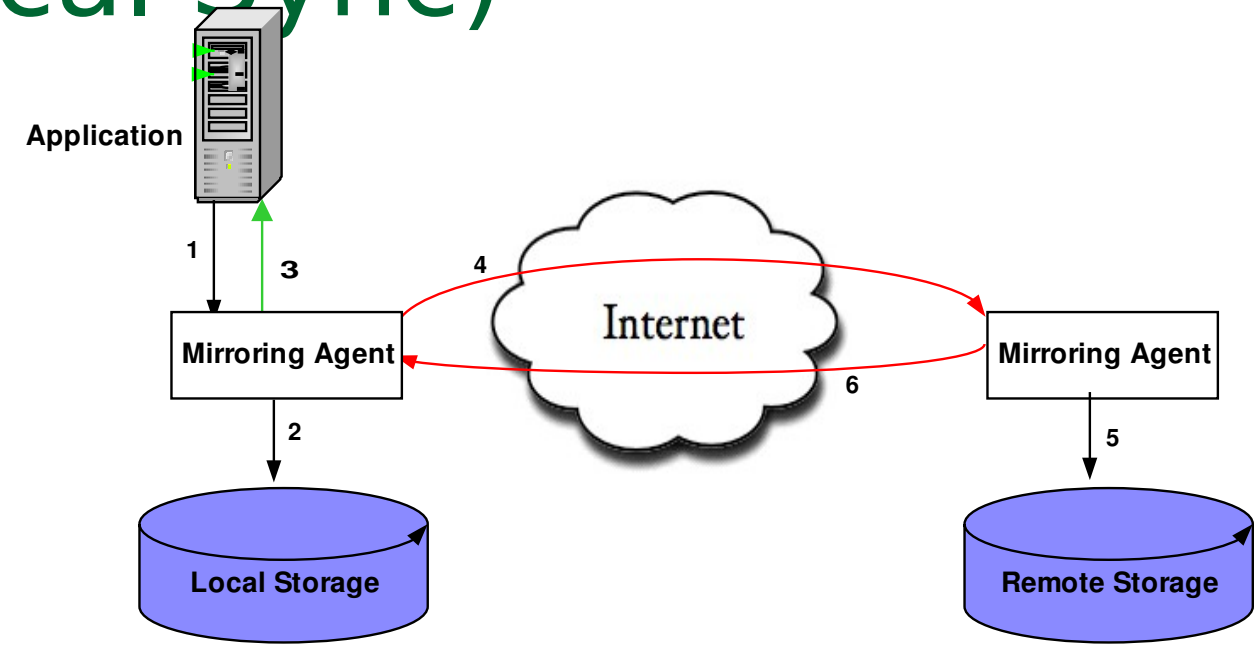
- Reliable
- Slow (Application effectively pauses between step 1 and 6)

Semi-synchronous Mirroring



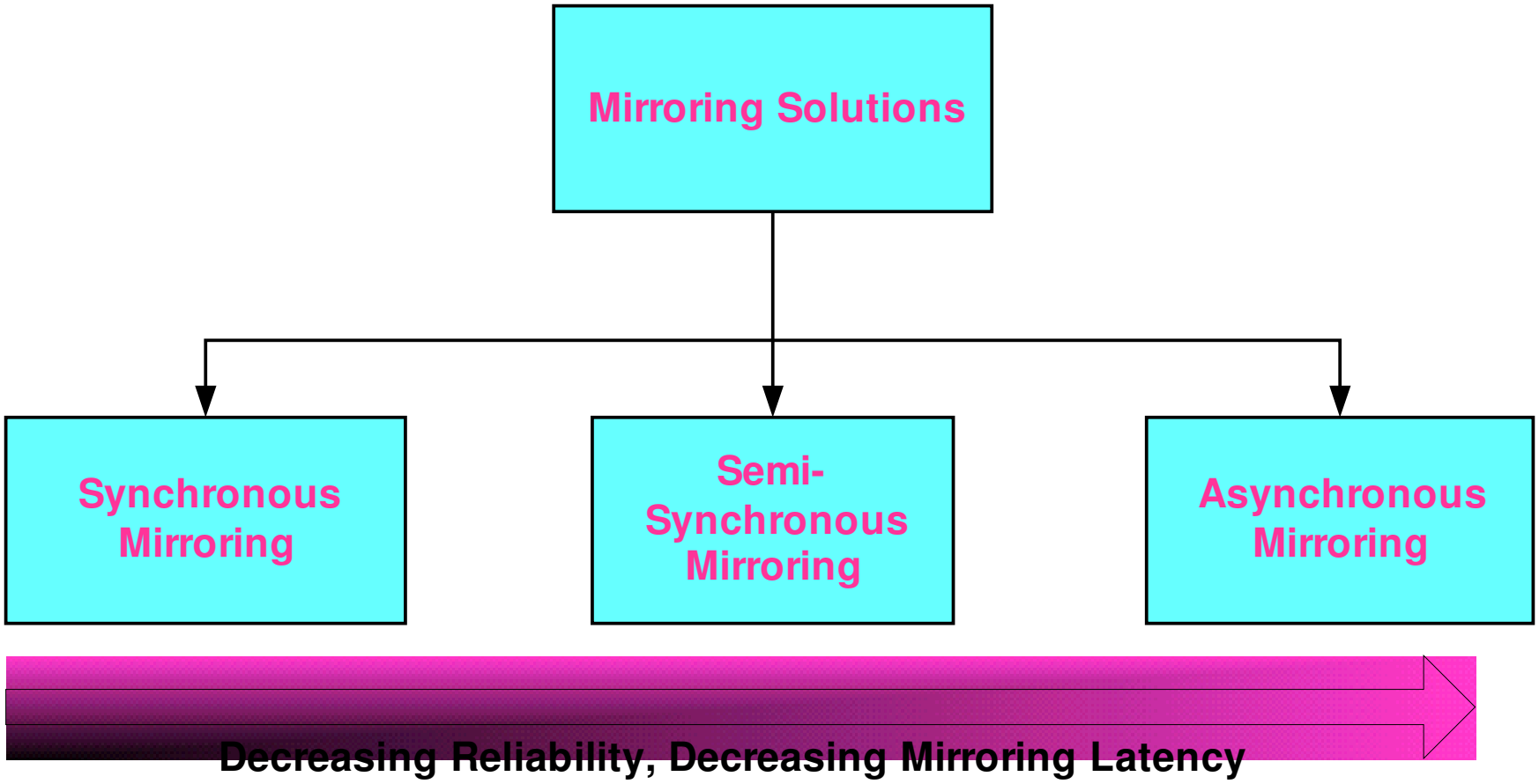
- Faster
- Less Reliable

Asynchronous Mirroring (Local Sync)

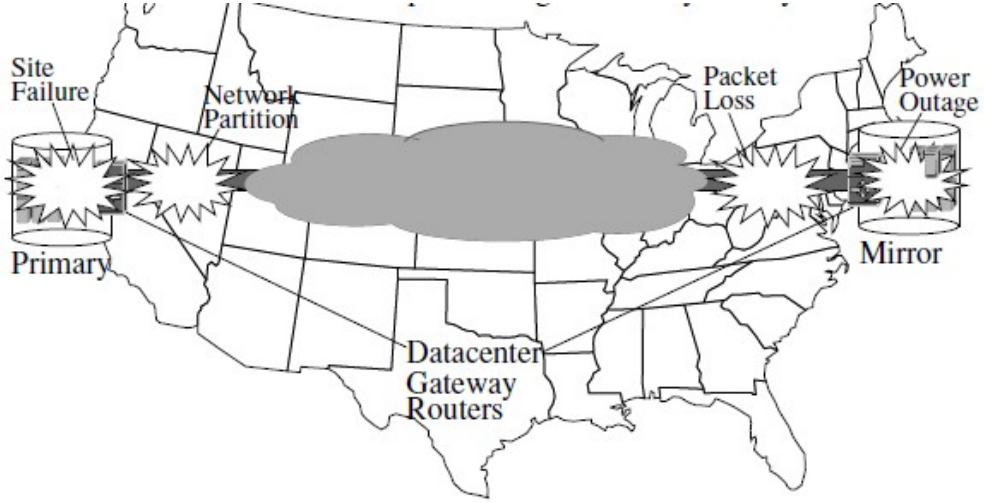


- Faster
- Least Reliable

Mirroring Options:



Failure Model



- Can occur at any level
- Simultaneous or in sequence (rolling disaster)
- Network elements can drop packets

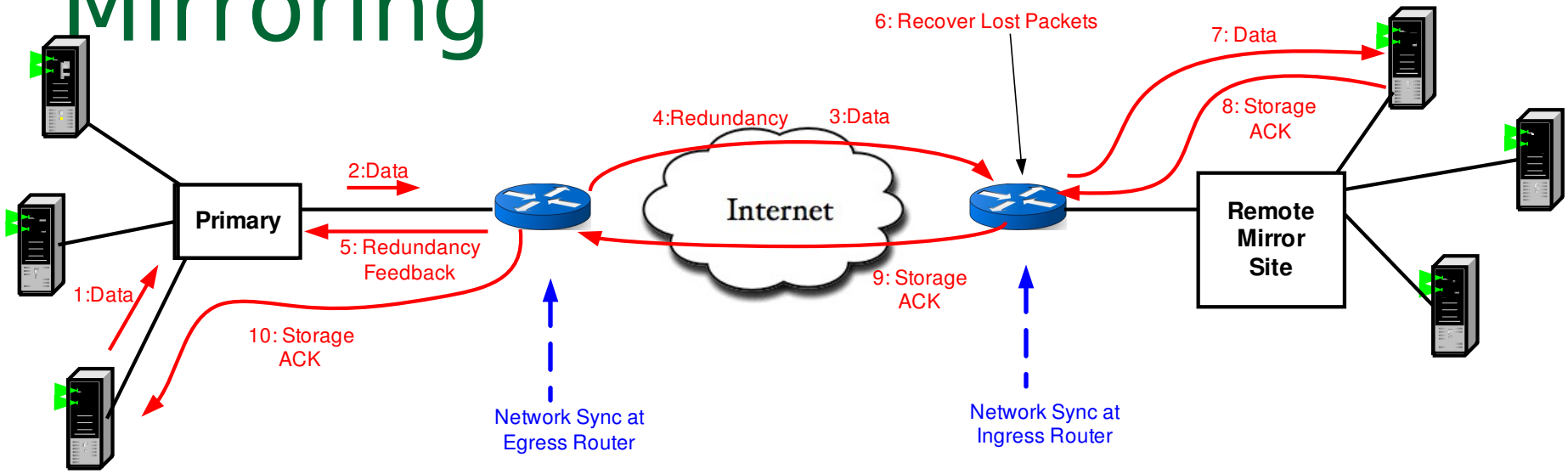
Data Loss Model

Failure	Synchronous Mirroring	Semi-Synchronous Mirroring	Asynchronous Mirroring
Primary only	No Loss	No Loss	Data Loss
Primary and Packet Loss on Link	No Loss	Data Loss	Data Loss
Primary and Mirror	Data Loss	Data Loss	Data Loss

Network Sync Remote Mirroring

- Proactively send error recovery data
- Expose status of data to the application

Network Sync Remote Mirroring



Mirror Solution	Mirror Update	Mirror-ack Receive	Mirror-ack Latency	Rolling Disaster			
				Local-only Failure	Local + Pckt Loss	Local + NW Partition	Local+Mirror Failure
Local-sync	Async- or Semi-sync	N/A	N/A	Loss	Loss	Loss	Loss
Remote-sync	Synchronous	Storage-level ack (7)	WAN RTT	No loss	No loss	No loss	Maybe loss
Network-sync	Semi-sync	nw-sync feedback (3)	≈ Local ping	≈ No loss	≈ No loss	Loss	Loss

Table 1: Comparison of Mirroring Protocols.

Smoke and Mirror File System (SMFS)

- A distributed log-structured file system
- Clients interact with file server
- File server interacts with storage servers
- create(), append(), free() operations mirrored

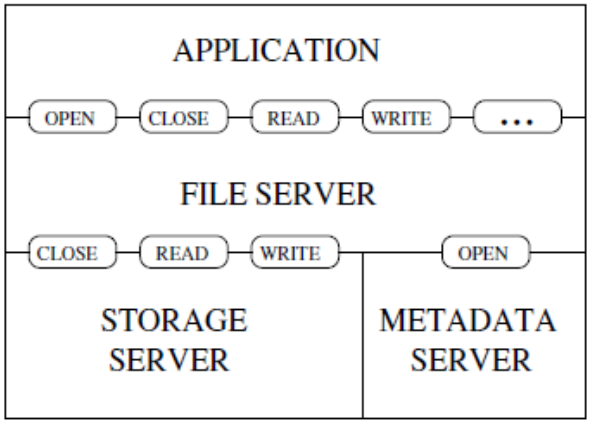


Figure 5: File System Architecture: Applications communicate with the *file server* through (possibly) a NFS interface. The file server in turn communicates with the *metadata server* through the `create()` function call. The metadata server allocates space for the newly created log on storage servers that it selects. The file server then interacts directly with the *storage server* for `append()`, `read()`, and `free()` operations.

Experimental Set-up

- Emulab
- Two clusters of 8 machines each (Primary and Remote)
- Separated by WAN 50-200ms RTT and 1Gbps
- Workload of upto 64 testers
  - Tester is an individual application with only one outstanding request at a time

Evaluation Metrics

- Data Loss
- Latency
- Throughput

Experimental Configurations

- Local-sync
- Remote-sync
- Network-sync
- Local-sync+FEC
- Remote-sync+FEC

Results: Data Loss

- Wide area link failure
- Primary site crash
- Loss rate increased for 0.5sec before disaster

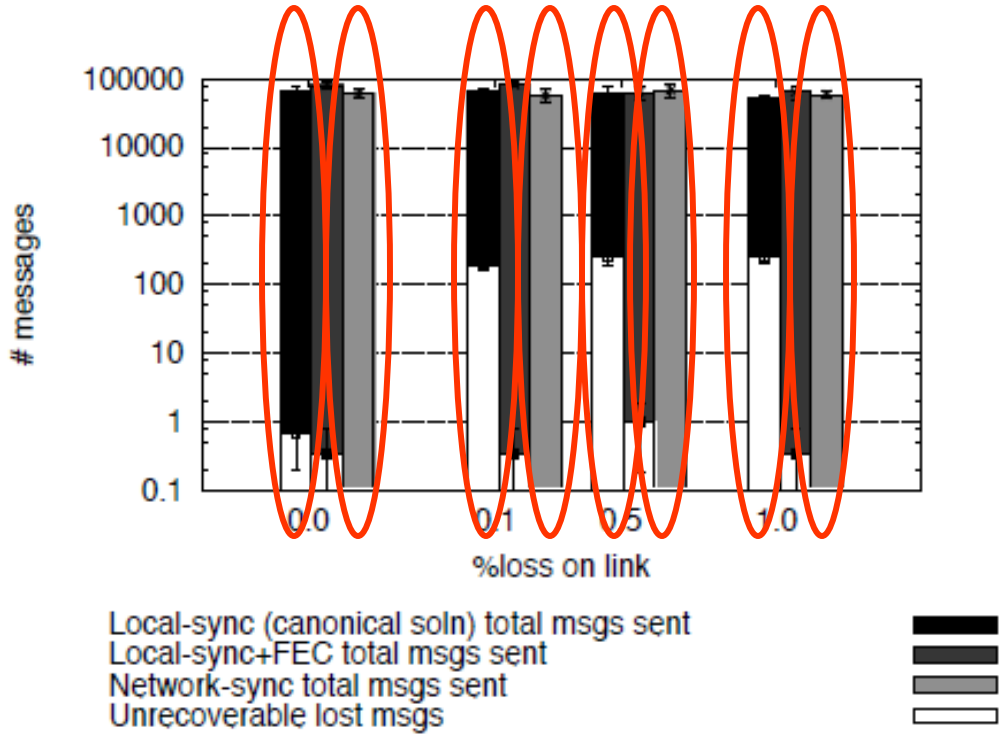


Figure 6: Data loss as a result of disaster and wide-area link failure, varying link loss (50ms one-way latency and FEC params $(r,c) = (8,3)$).

Results: Varying the level of Redundancy

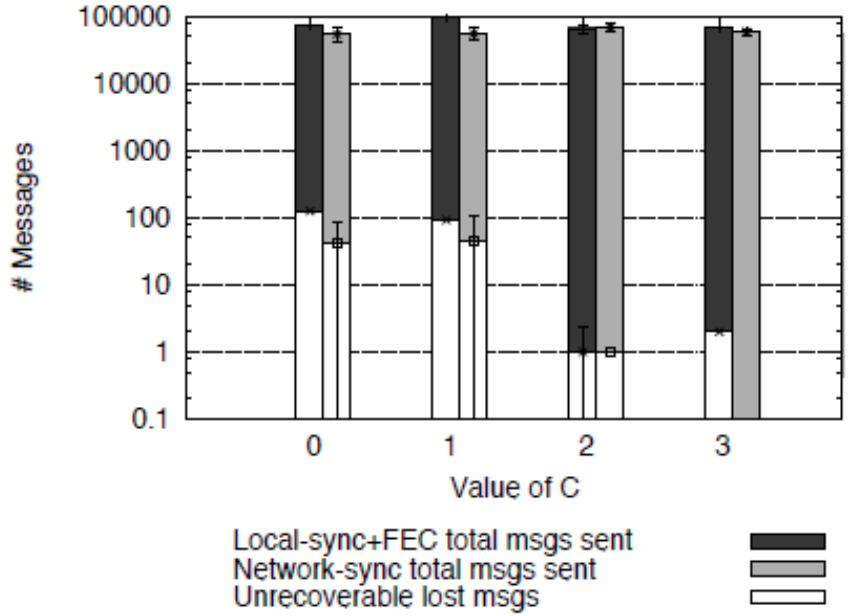


Figure 7: Data loss as a result of disaster and wide-area link failure, varying FEC param c (50ms one-way latency, 1% link loss).

Results: Throughput

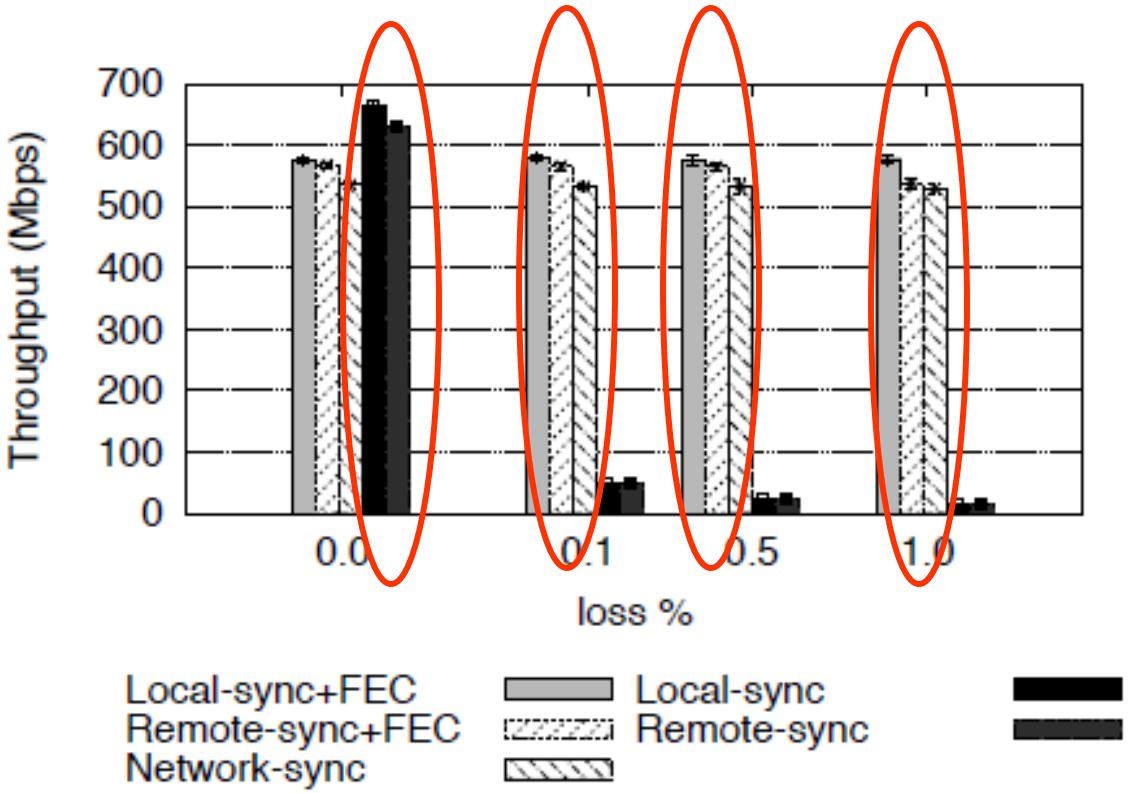


Figure 9: Effect of varying wide-area one-way link *loss* on Aggregate Throughput.

Discussion

- Solution is still imperfect
- What if there are multiple remote sites to choose from?
- Split data across different sites?