A Low-bandwidth Network File System

Athicha Muthitacharoen *et al.*

Presented by Matt Miller

September 12, 2002

Problem Statement

- Network file systems cannot be used over slower WANs due to longer delay, less available bandwidth
- Ad hoc solutions:
  - Copy file to local machine, edit, then upload. Risks update conflicts.
  - Use remote login. Potentially large interactive delays.

Overview of Proposed Solution

- Transfer less data over the network by exploiting inter-file similarities
- Follow close-to-open consistency model
- Efficiently divide files into “chunks” and only transfer chunks the remote machine does not already have

Key Contribution

Proposes file update mechanism which avoids transferring redundant data from different versions of a file over the bottleneck resource while providing some consistency.

- Use hashing to exploit inter-file similarities and save bandwidth
- Provide an efficient scheme to delineate files while not causing massive breakpoint changes for small modifications
- Decrease latency delay by pipelining read and write RPCs
- Provide robustness despite client failures

Related Work

- AFS: Servers provide callbacks to clients when other clients modify a file
- Leases: Similar to AFS, but server is only obliged to report changes for specified amount of time
- rsync: Exploits inter-file similarities when copying directory trees over a network

LBFS Hashing Scheme

- Use SHA-1 scheme, assumes data with equal hash values are the same (extremely low collision probability)
- Delineate a file into chunks with hash values based on data within chunk
- Needs to be resistant to small changes causing completely new chunk values for the file (e.g. inserting a byte at the beginning of the file)

LBFS Hashing Scheme (2)

- Scan entire file while calculating Rabin fingerprints values for overlapping 48-byte windows
- If the last 13 bits of the fingerprint match specified value, allow window to be a breakpoint between two chunks
  - Expected chunk size = 2^{13} = 8 KB
- Place upper and lower bound on chunk sizes to avoid pathological cases

Consistency

- Maintain database correlating hash values to (file, offset, count) tuples. Do not rely on database, always compute hash values to reconstruct file.
- Operate only on whole files
- Close-to-open: When a client has written and closed a file, another client opening the same file will see the new contents

Consistency (2)

- Client receives lease on file
- Server notifies clients of changes to the file while the lease is valid
- When a client opens a file for which the lease has expired, it checks the file attributes for modification times
- If the server version is more recent, the reading protocol takes place

Reading Protocol

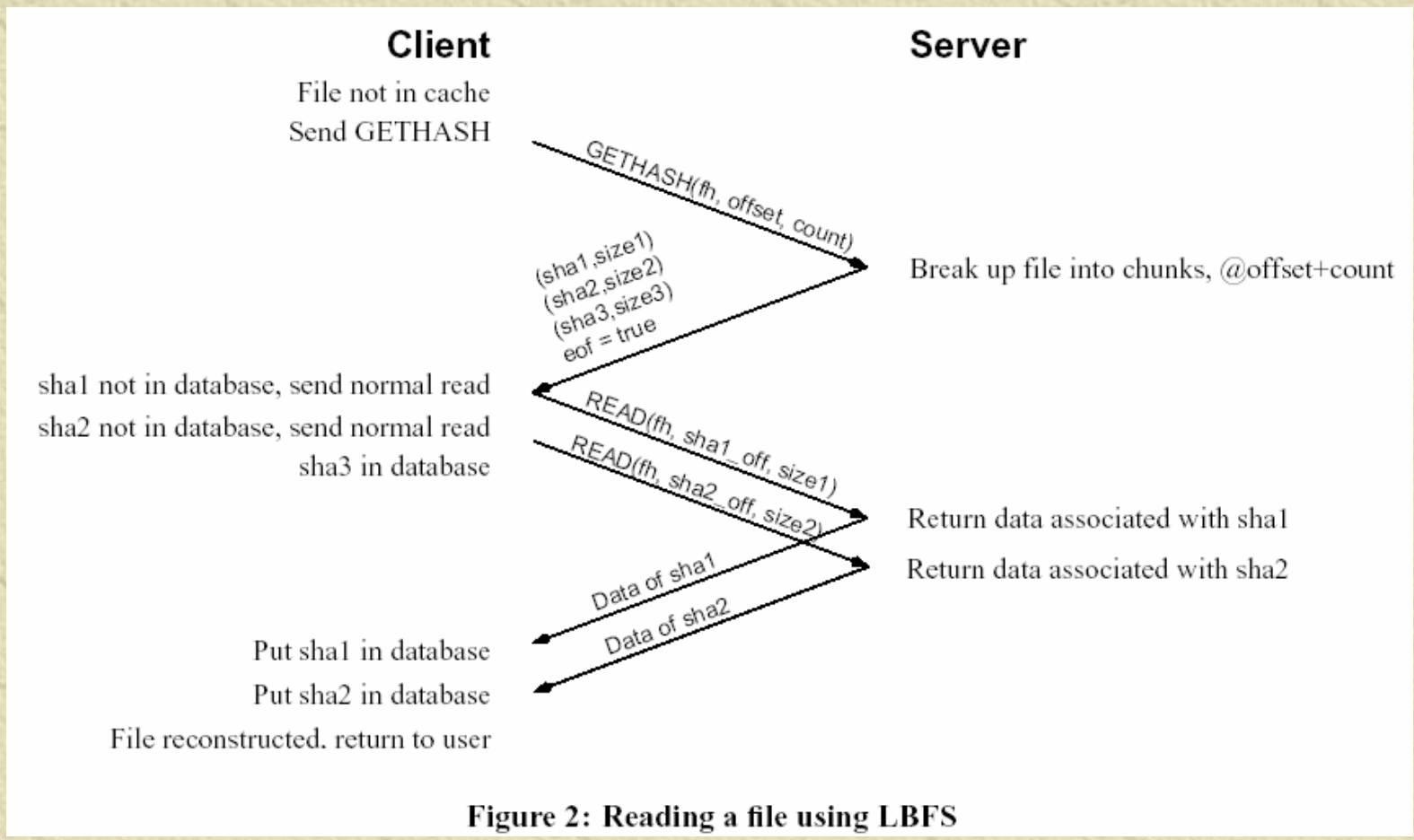

Figure 2: Reading a file using LBFS

Writing Protocol

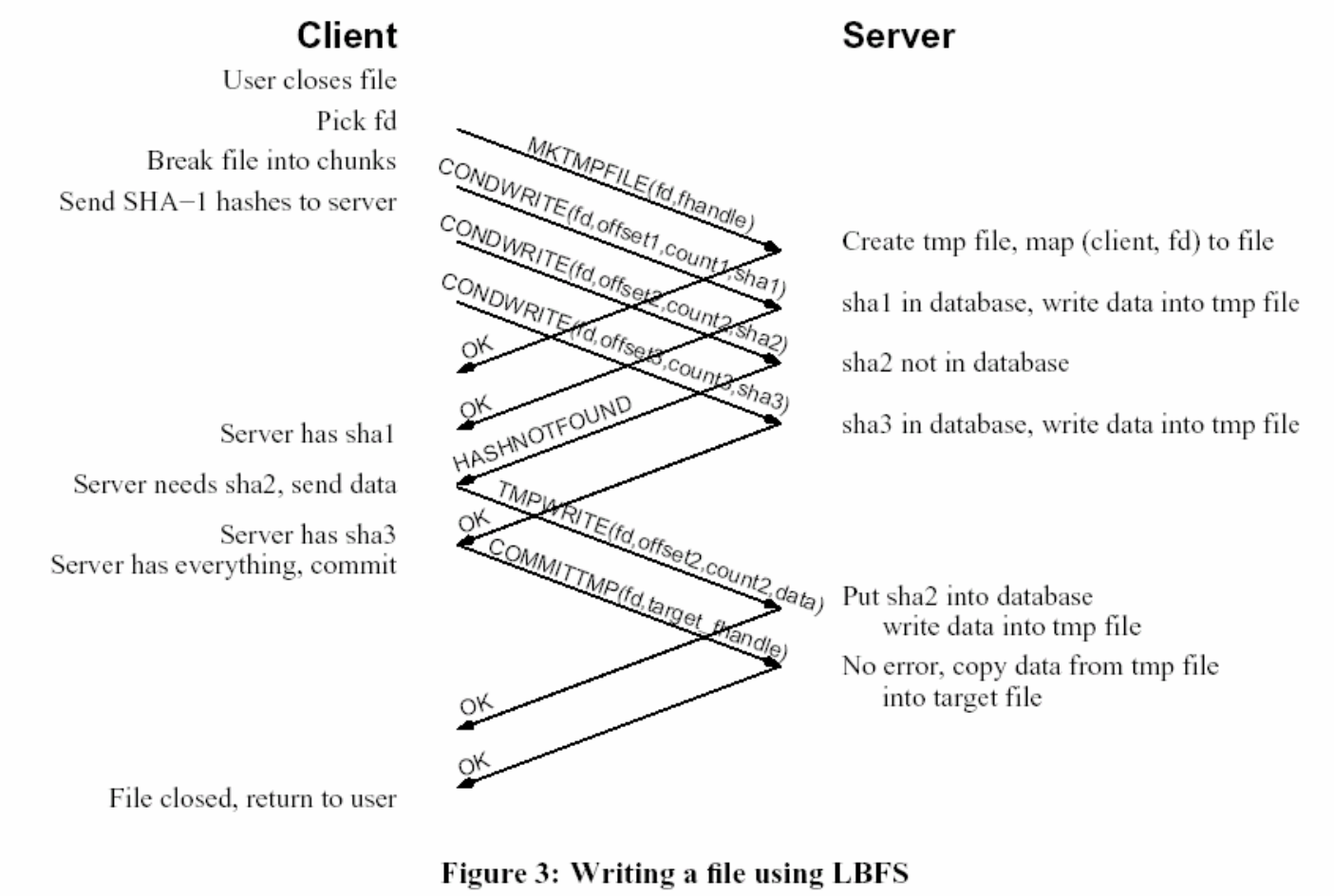

Figure 3: Writing a file using LBFS

Implementation

- Client uses xfs file system, file access is done via NFS, asynchronous RPC over TCP is used for communication
- All RPC traffic is gzip'd
- Unix semantics for i-nodes leads to inefficiency and possible inconsistency during server crash

Evaluation

- The window size for computing the Rabin fingerprint does not seem to have much impact on data sharing
- Expected chunk sizes were close to expected value
- Some small changes do remove much commonality between revisions (e.g. renumbering all the pages of a document)

Bandwidth Utilization

- Tests done on native file system (NFS or CIFS), AFS, just Leases+gzip, and LBFS
- Tests involve editing MS Word document, recompiling new version of emacs and changing the source tree between perl versions

Bandwidth Utilization Graph

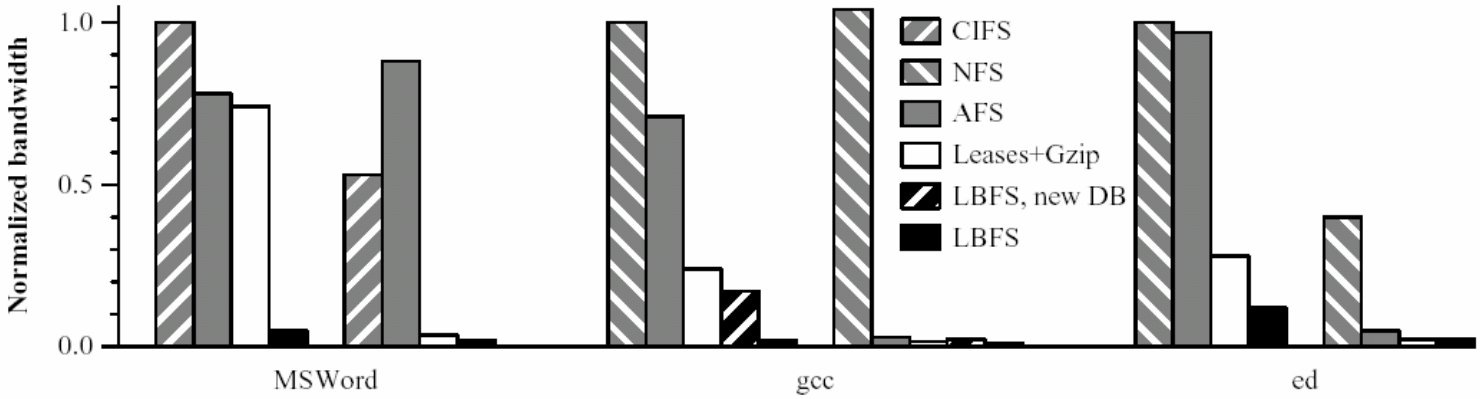

Figure 6: Normalized bandwidth consumed by three workloads. The first four bars of each workload show upstream bandwidth, the second four downstream bandwidth. The results are normalized against the upstream bandwidth of CIFS or NFS.

Other Performance Results

- Reduces application execution time significantly
- Execution faster than AFS and Leases+gzip as bandwidth decreases
- Performs better than AFS independent of RTT and similarly to Leases+gzip
- All perform about the same in the presence of a lossy link

Conclusion

- LBFS avoids transmitting redundant data to a remote machine
- Provides method to delineate files in way which is resistant to the propagation of a small modification changing breakpoints
- Performs much better than the competition

Discussion

- Why does AFS use more downstream bandwidth than the native file system in Figure 6?
- Consistency model can be violated. Would server blocking all but one client from writing be better?
- Tests are in very biased conditions. How about testing a mix where only *k* of the *n* files were previously on the client?
- Scalability is an unaddressed issue. LBFS requires server to do non-trivial computation for each client file access.

Discussion (2)

- Would most users accept the consistency model, or is it too weak? Could schemes to merge writes be added?
- Could the scheme be changed to allow block caching rather than entire files?
- Any more elegant solutions to the static i-number problem?

Consistency Model Problem

"If multiple clients are writing the same file, then the last one to close the file will win and overwrite changes from the others."

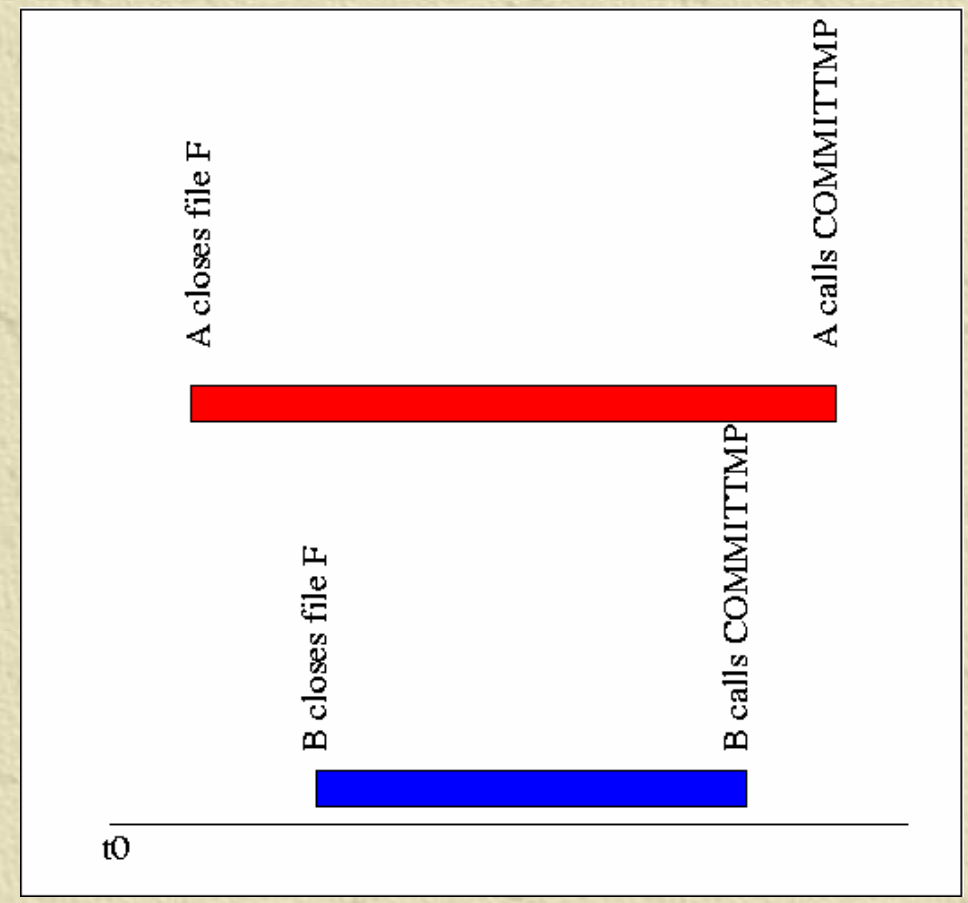

Figure 1

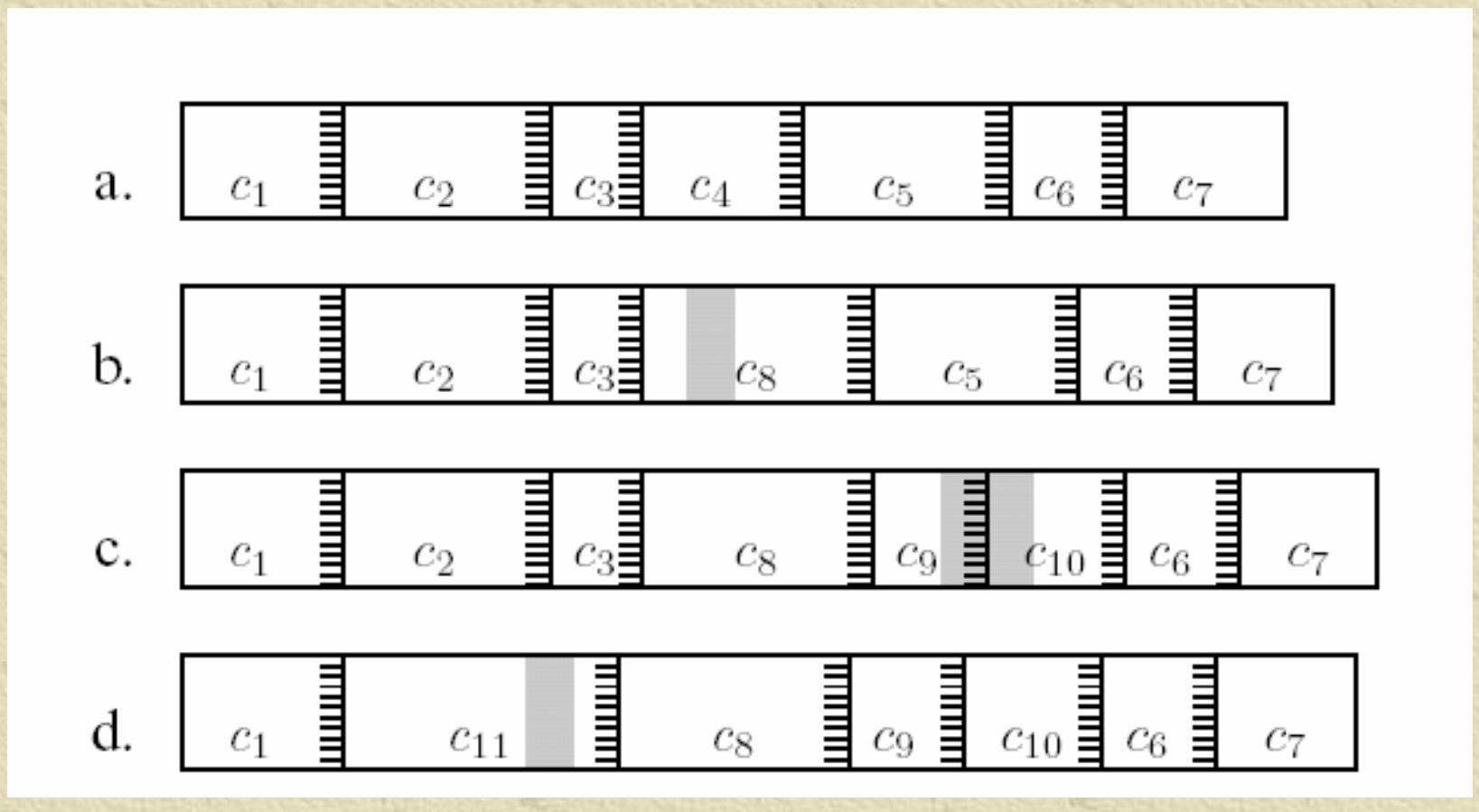
a.
c_1 c_2 c_3 c_4 c_5 c_6 c_7
b.
c_1 c_2 c_3 c_8 c_5 c_6 c_7
c.
c_1 c_2 c_3 c_8 c_9 c_{10} c_6 c_7
d.
c_1 c_{11} c_8 c_9 c_{10} c_6 c_7

Figure 7

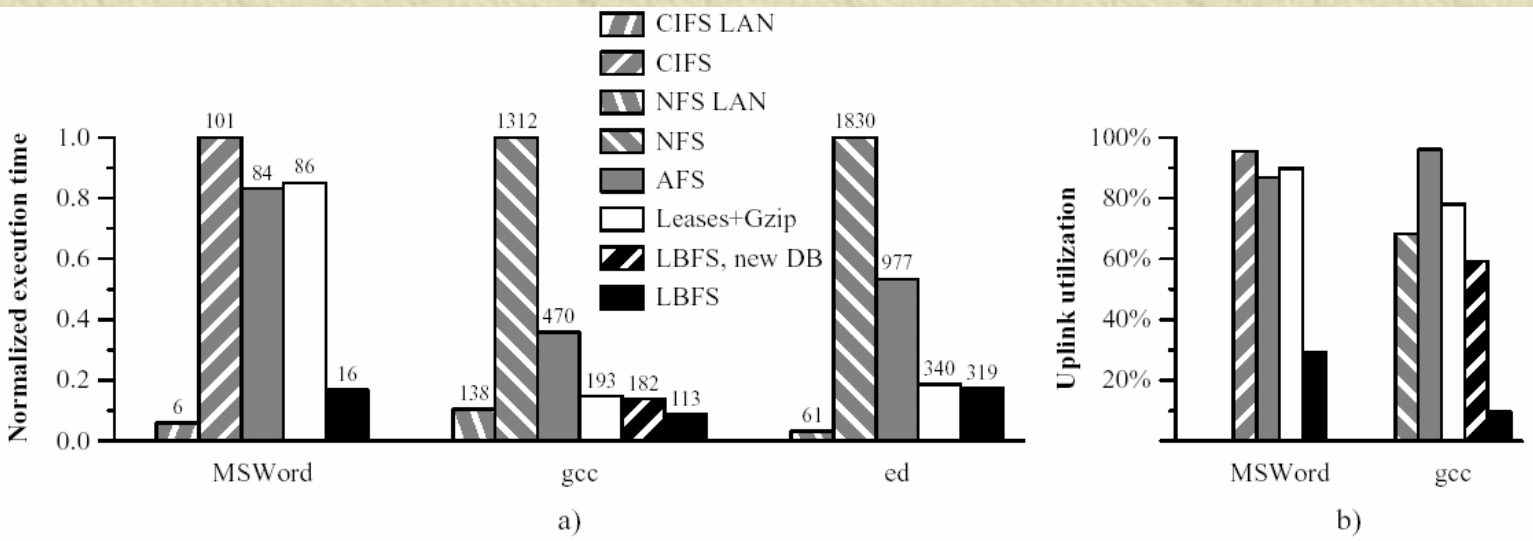

Figure 7: a) Normalized application performance on top of several file systems over a cable modem link with 384 Kbit/sec uplink and 1.5 Mbit/sec downlink. Execution times are normalized against CIFS or NFS results. Execution times in seconds appear on top of the bars. b) Uplink bandwidth utilization of the MSWord and gcc benchmarks.

Figures 8 and 9

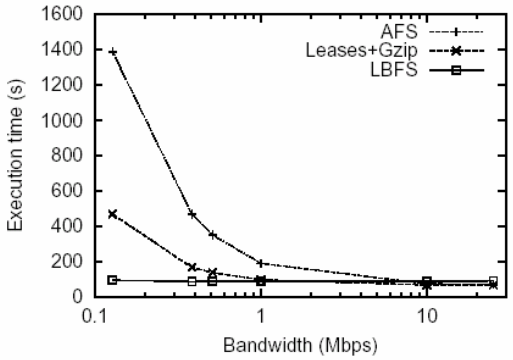

Figure 8: Performance of the gcc workload over various bandwidths with a fixed round-trip time of 10 ms.

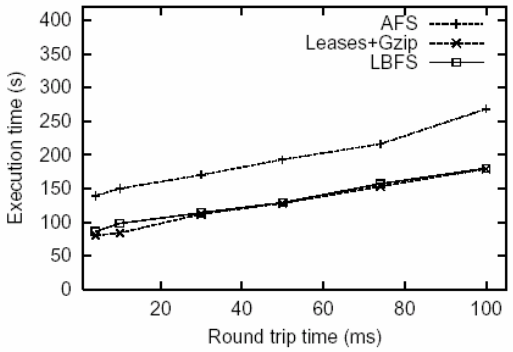

Figure 9: Performance of the gcc workload over a range of round-trip times with fixed 1.5 Mbit/sec symmetric links.

Figure 10

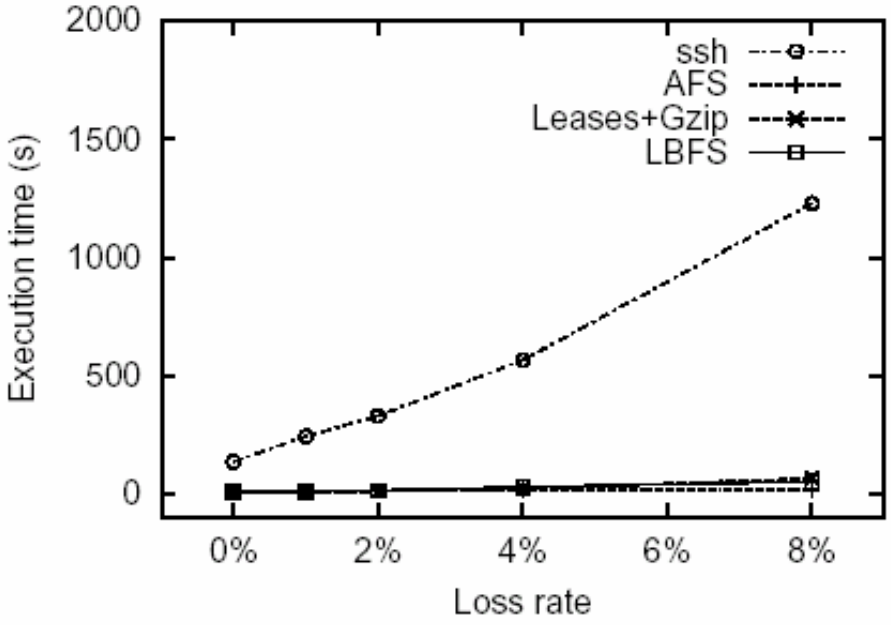

Figure 10: Performance of a shortened `ed` benchmark over various loss rates, on a network with fixed 1.5 Mbit/sec symmetric links and a fixed round-trip time of 10 ms.