Experimentally Evaluating In-Place Delta Reconstruction

Randal Burns, *Johns Hopkins University*

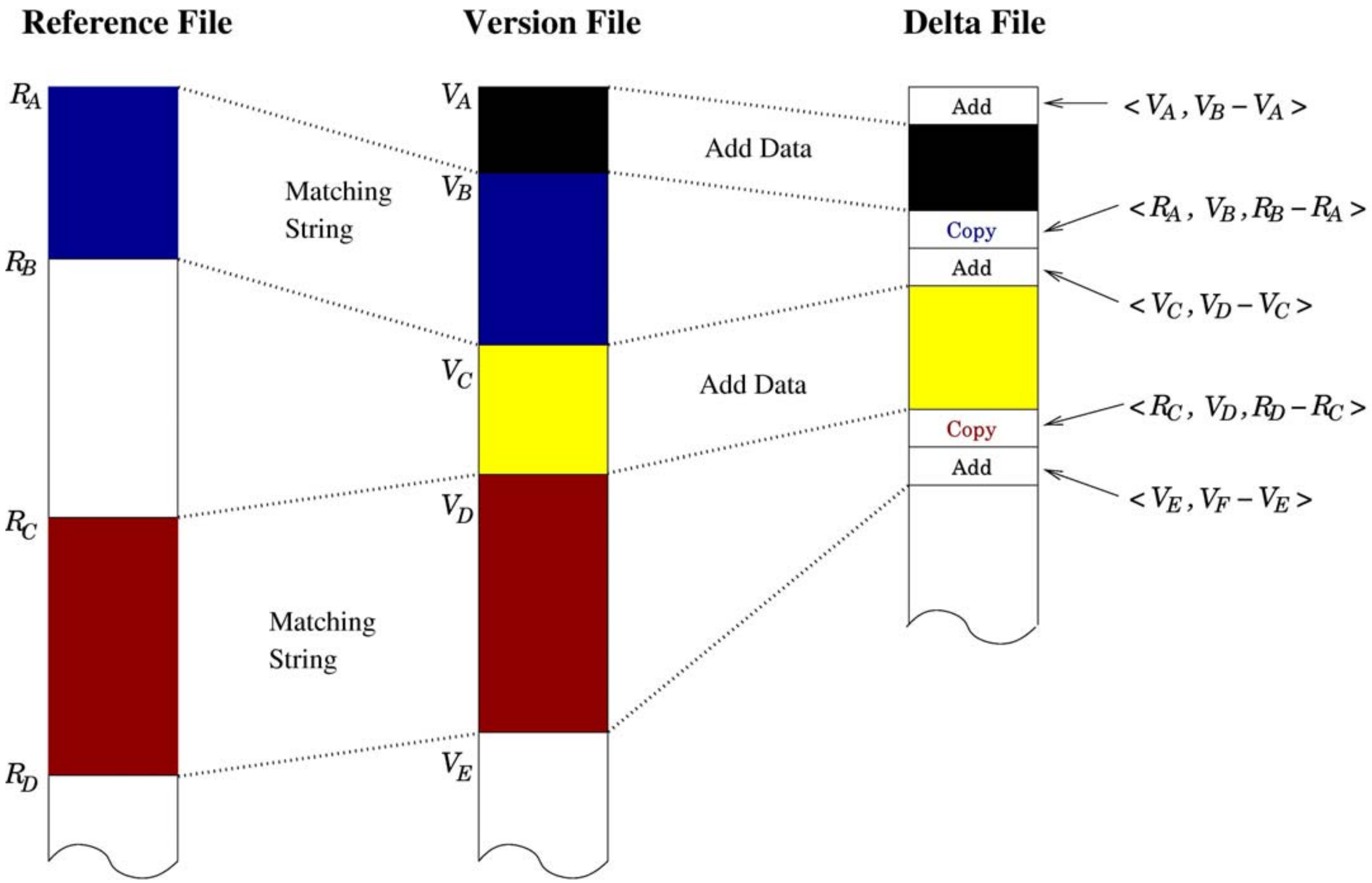
Larry Stockmeyer, *IBM Almaden Research Center*

Darrell D. E. Long, *University of California, Santa Cruz*

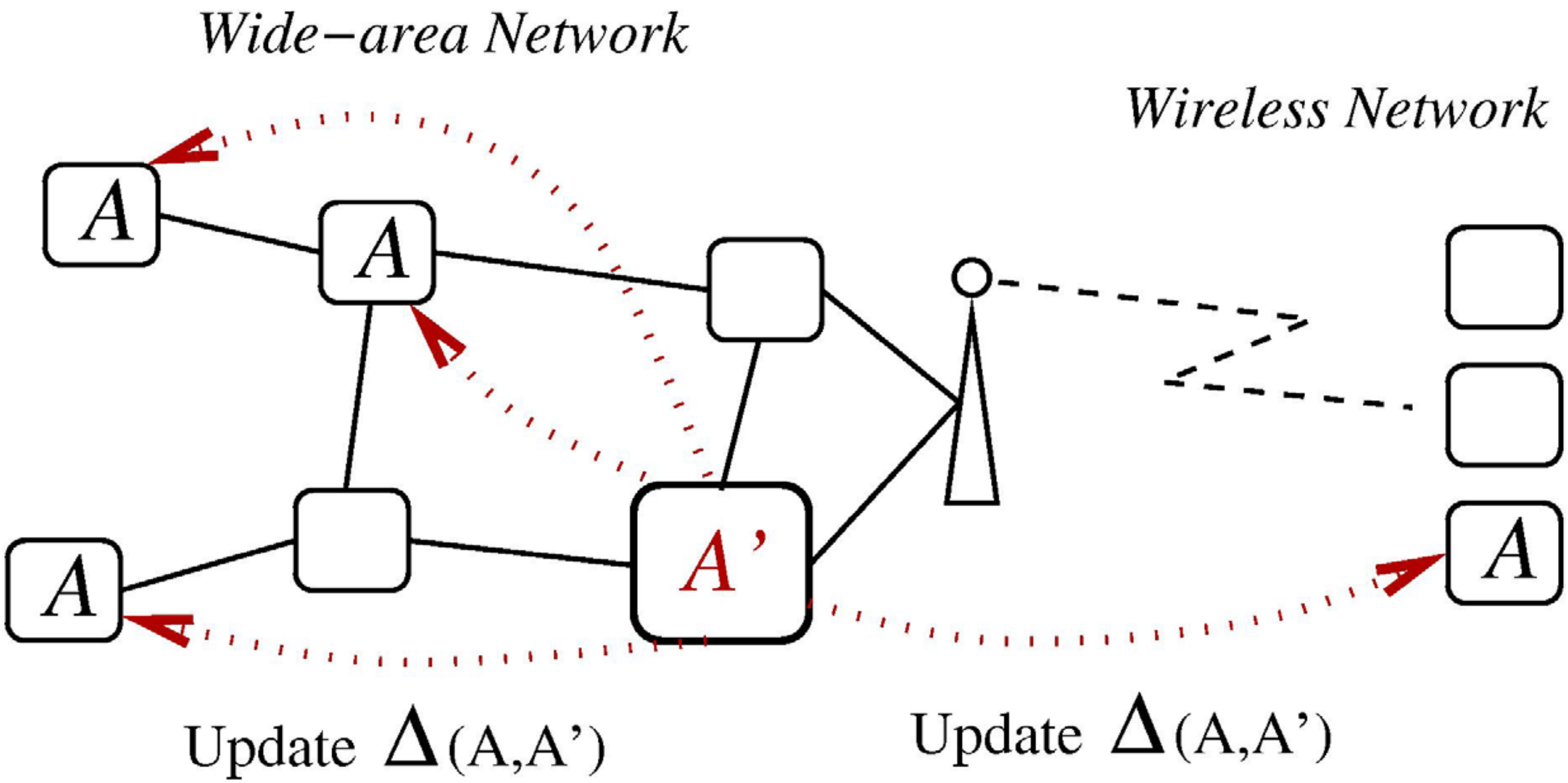
Delta Compression in Global-Scale Systems

- Represent data compactly as a small set of changes from a previous version
- Transfer deltas to update data
  - Decreased latency
    - Proven in backup/restore and Web data transfer
  - Reduced bandwidth requirements
- Gives mobile users with resource-constrained devices access to distributed data stores
  - software update
  - cache-consistency

Delta Encoding



Distributing Updates



Deltas for Mobile and Wireless Devices

- Ideal problem space
  - Big latency savings on bandwidth constrained networks
  - Good citizenship: reduce the overall network resource used to keep data consistent
- Reconstruction problems have prevented successful use
  - Applying (reconstructing) a delta to build the new version requires storage space for two versions
  - Not possible on resource-constrained devices

Mobile and Wireless Devices

- Cannot justify adding storage to resource-constrained devices for delta compression
  - Mass produced low-cost devices
  - Often use expensive non-volatile memories for persistent storage
- Devices in this class
  - Cell phones,
  - PDAs
  - Wireless handhelds

In-Place Reconstruction

- Rebuild new version in the space that the current version occupies
- No scratch space needed
  - No second version
  - Commands in delta file read and executed in order
- In-place property achieved in exchange for a small compression penalty
- In-place property brings delta compression to resource-constrained devices

Motivation

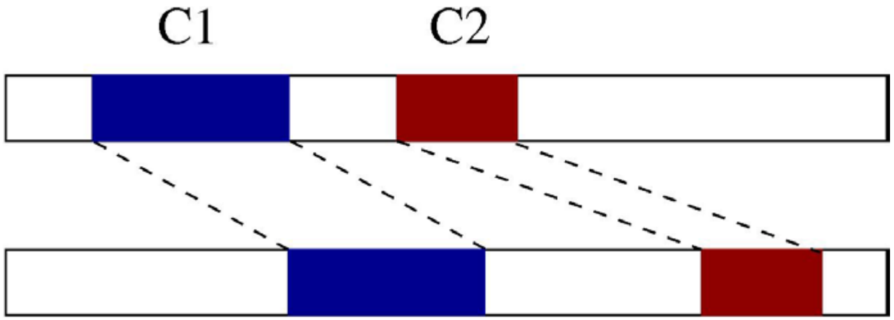
- Integrate mobile and wireless users with global-scale distributed data systems
- Distributed cache consistency
  - Workflow applications, human-computer interfaces
  - Health care, law enforcement, inventory management, military operations
- Update distribution
  - Security patches for cellular phones

Conflicts and Corruption

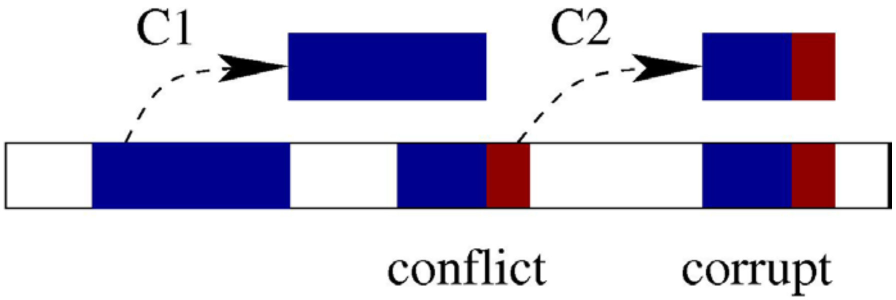
- Attempting to update a delta file in place corrupts data
  - Copy commands read from and write to file
  - Problems arise when a copy overwrites data needed for a future operation

Regular and In-Place Delta Copy

- Regular: data copied from old file version to new version



- In-place: first copy writes into the second copies read region
  - Write before read (*WR*) conflict

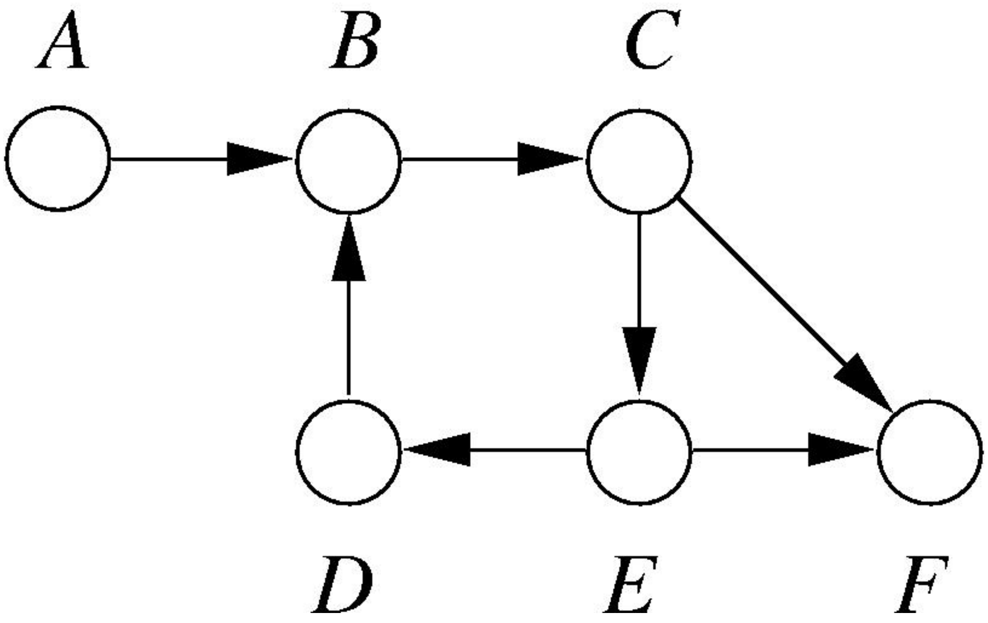


In-Place Permutation

- Detect and avoid *WR* conflicts by permuting the order in which copy commands are executed
- Avoid conflicts with add commands by conducting them after all copies are complete
  - Adds only write (do not read) data
- Encode *WR* conflicts in a directed graph
- Not all conflicts can be avoided
  - Corresponds to cycles in the graph
  - Convert copied data so that it is added explicitly

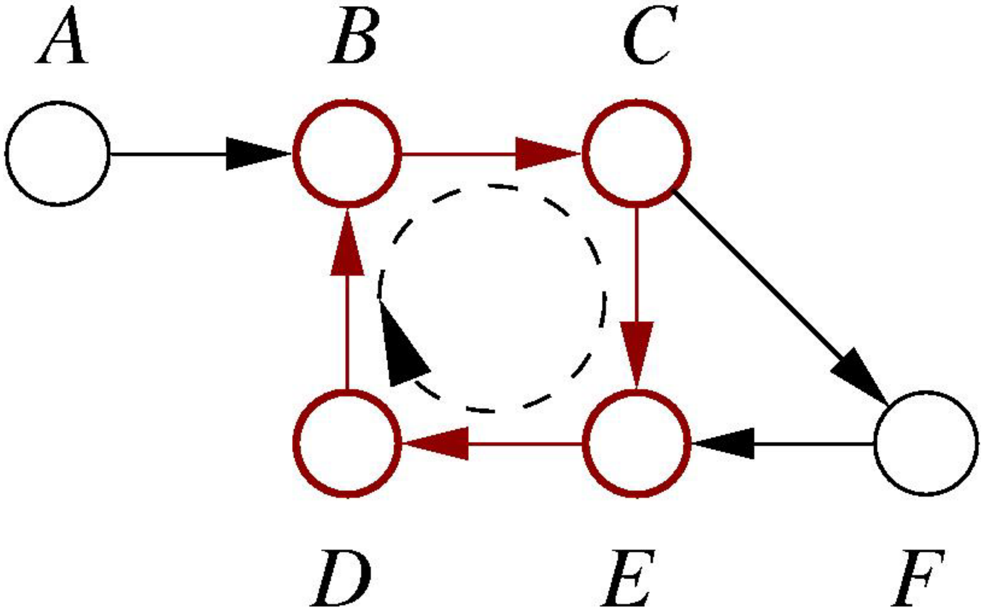
Generating an Ordering

- Topological sort (depth first search) the graph to detect conflicts and generate an ordering
- Initial graph, search starts at *B*



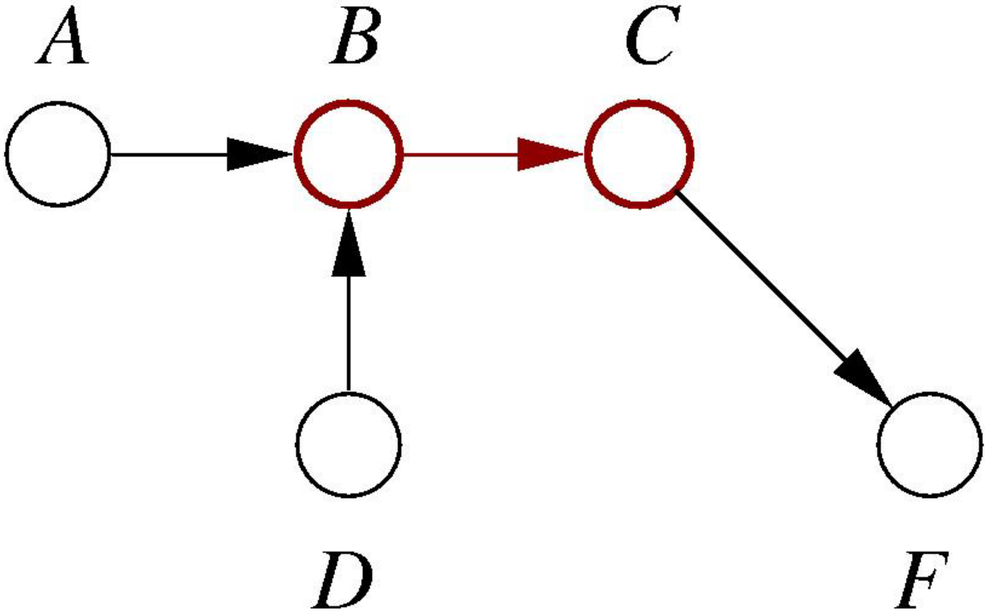
Generating an Ordering

- Search finds cycle *BCED*



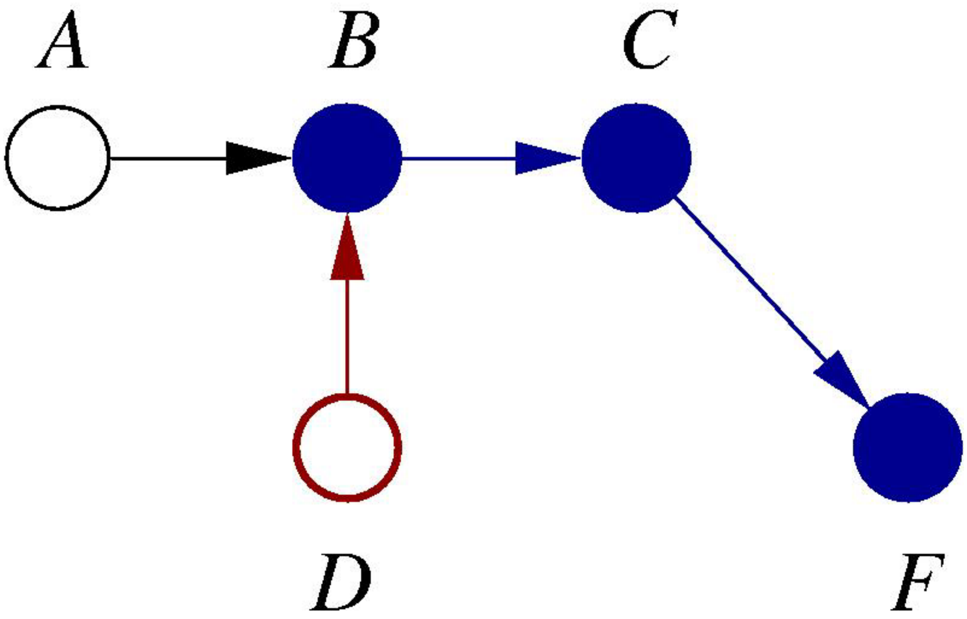
Generating an Ordering

- Cycle broken by converting *E* into an add command



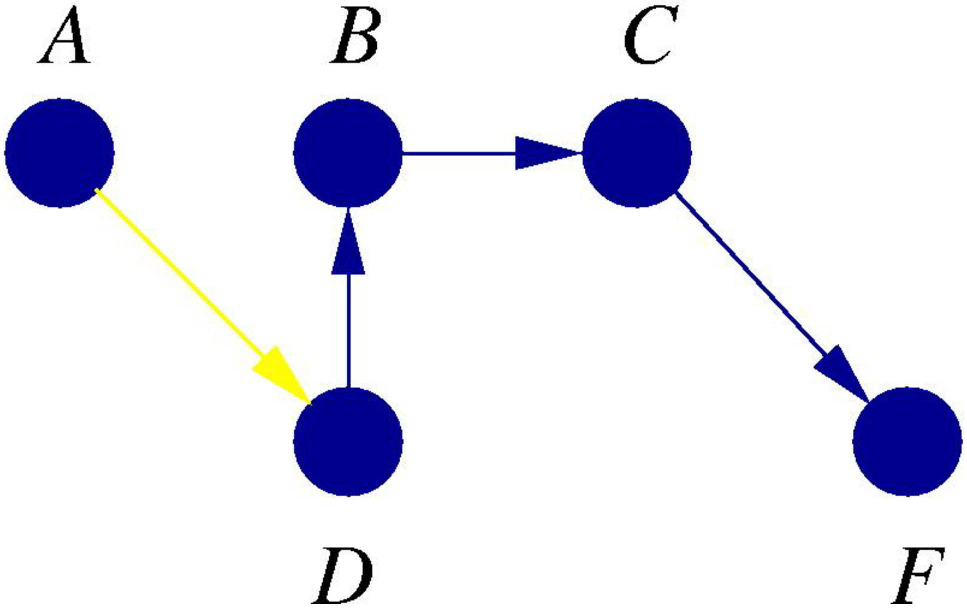
Generating an Ordering

- Finish component and start search again at remaining unordered nodes



Generating an Ordering

- Complete topological ordering
  - Involves new edges created during sort (*AD*)



Compression Loss

- Delta file grows larger when converting copies to adds
- Algorithm must select nodes to break cycles
  - Optimal solution is NP hard
- Two heuristics for breaking cycles on detecting a cycle
  - Select last node search: constant-time policy
  - Look at all nodes in cycle: local minimum
    - Quadratic time (in number of nodes)
- Experimentally compare these policies and find a (somewhat) surprising result

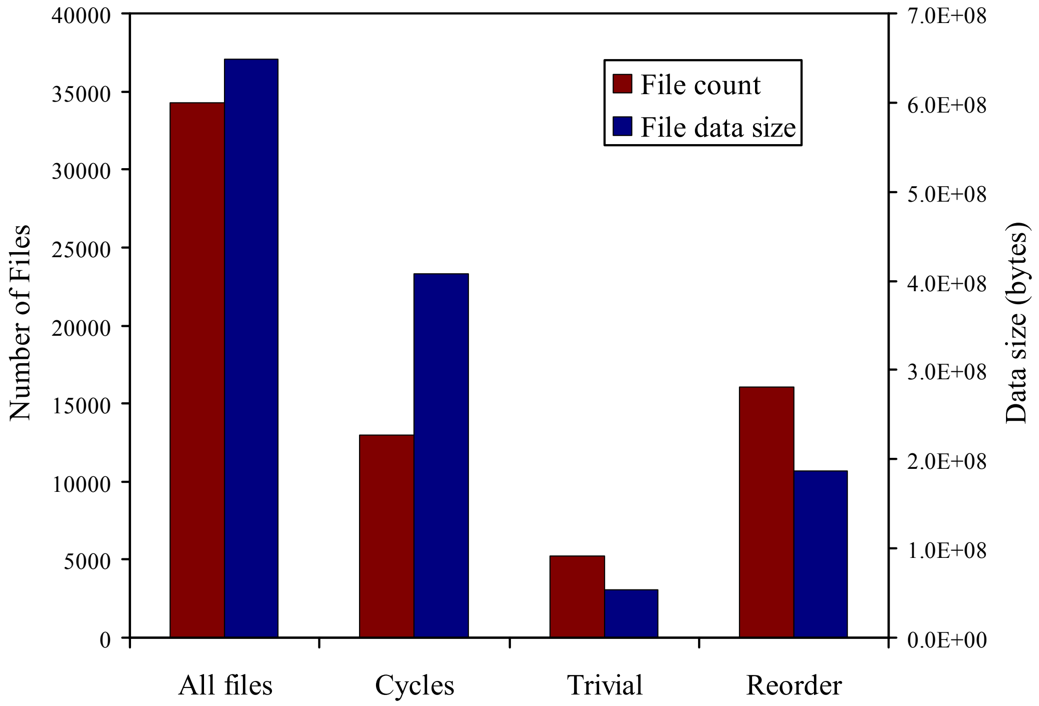
Experimental Goals

- Quantify the compression loss
- Compare cycle breaking heuristics
  - Algorithmic performance versus compression performance
- Characterize files based on their in-place properties

- Establish in-place reconstruction as a viable technique for large-scale mobile and wireless applications
  - As regular delta compression is for backup and the Internet

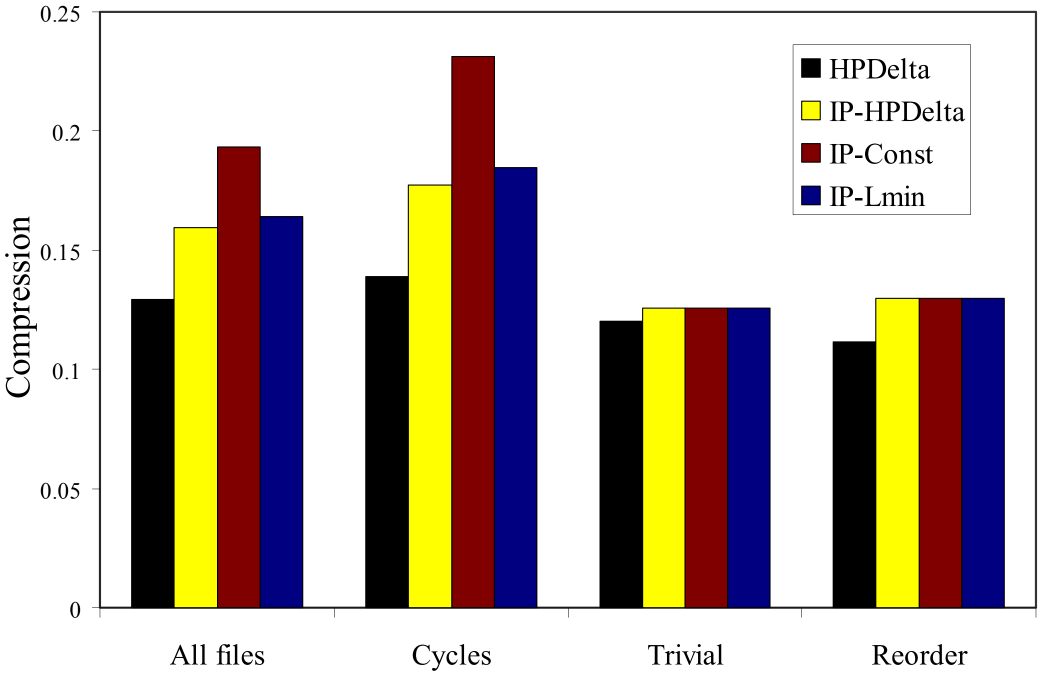
Characterizing Files and Data

- 34,000 files distributed on the Internet
  - GNU tools
  - BSD kernels
  - Source code, executables, and binary data
- The majority of the data (but not files) require algorithms to break cycles
- In-place is non-trivial
  - Cycles of >15,000 nodes



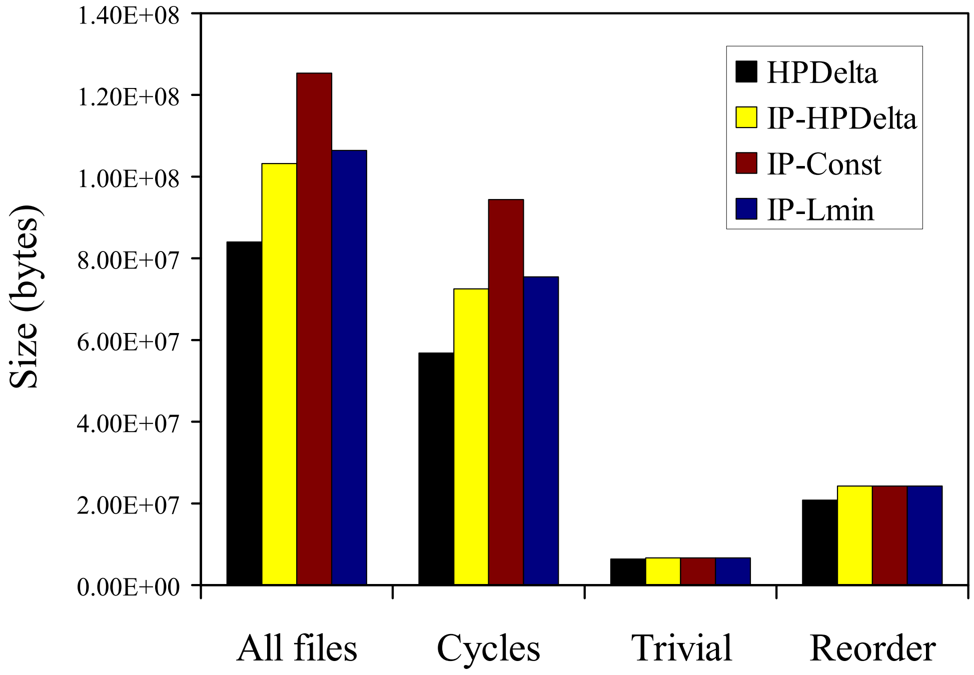
Compressibility of Files

- No compression loss on trivial or reorder classes
- Encoding loss
  - Different codewords needed for in-place reconstruction
- Cycle breaking loss avoidable through heuristics
  - IP-Lmin < 0.5%
  - IP-Const > 3%



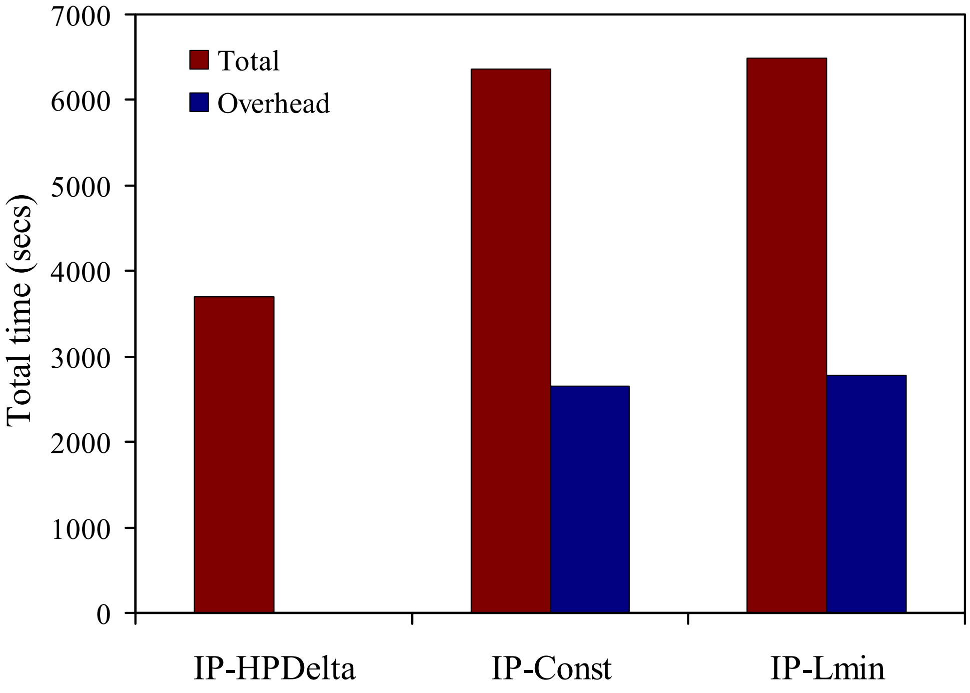
Compressibility of Data

- Files with cycles dominate all results
- Compression loss can have a large impact on overall performance
  - Encoding overhead
  - Loss from cycles



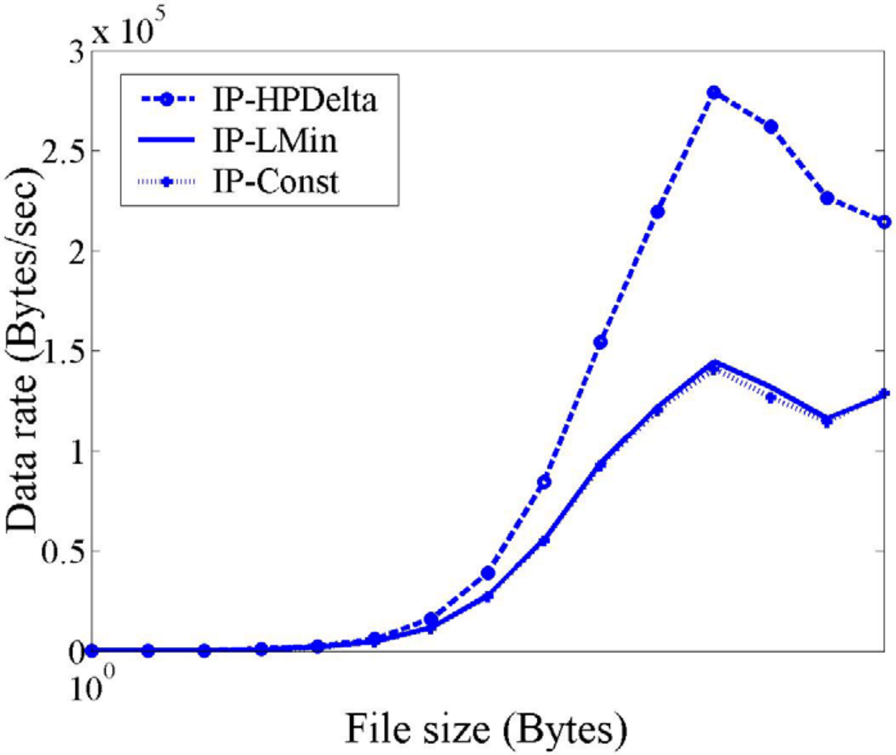
Execution Time

- Almost all overhead from I/O
  - Any in-place algorithm does nearly twice the I/O
- Computation not an important factor
- Constant-time heuristic less efficient than “slower” local-min
  - Larger delta files means more I/O
- In-place algorithms I/O bound



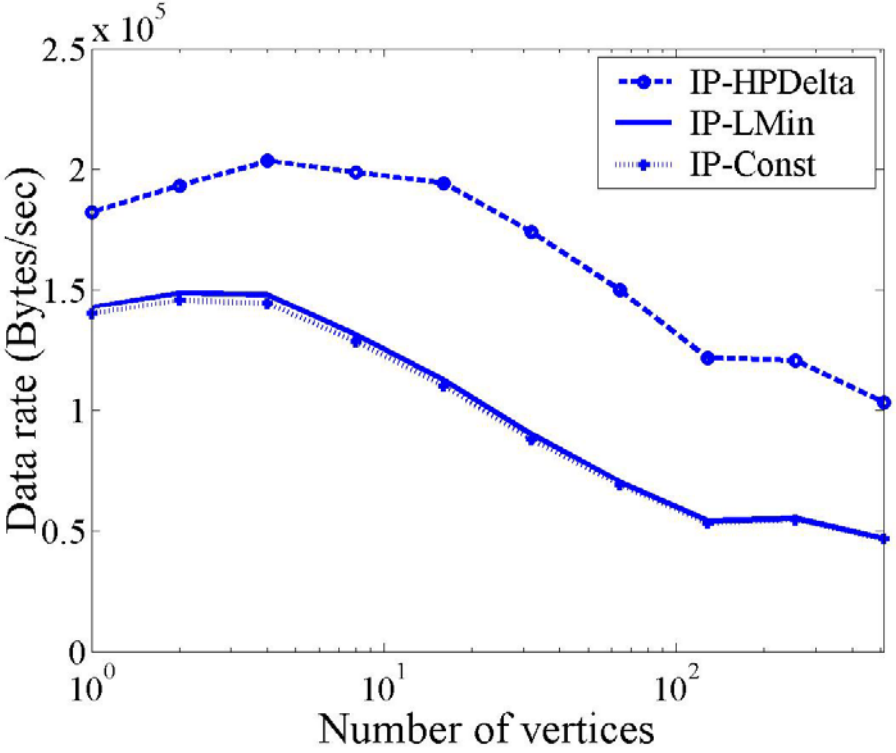
Performance Scaling

- Effects of asymptotic bounds cannot be seen
  - Not with increasing file size



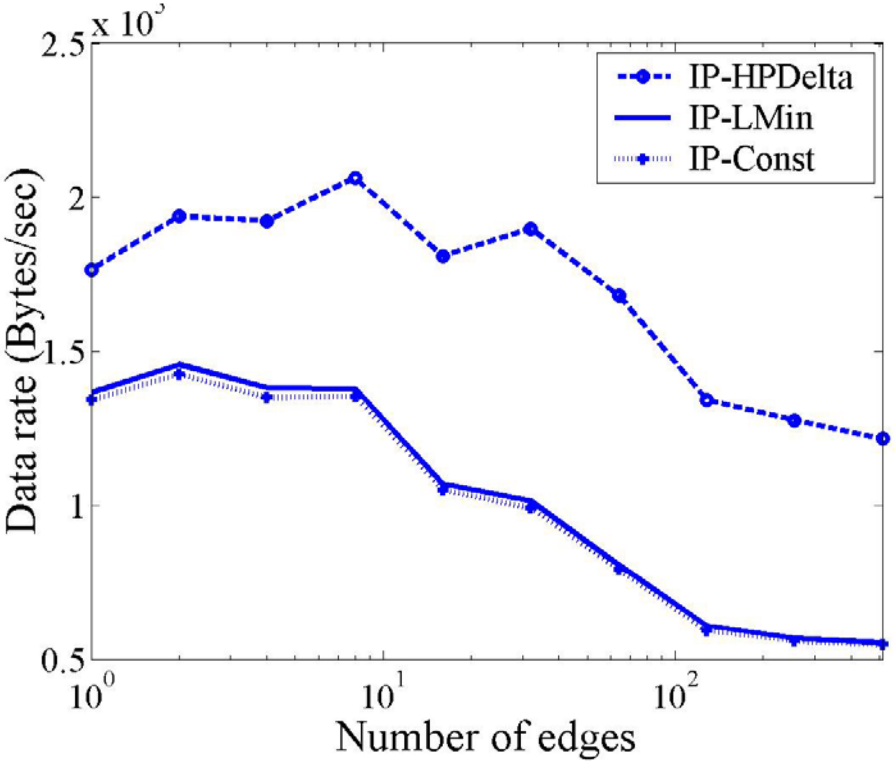
Performance Scaling

- Effects of asymptotic bounds cannot be seen
  - Not with increasing file size
  - Not with more vertices



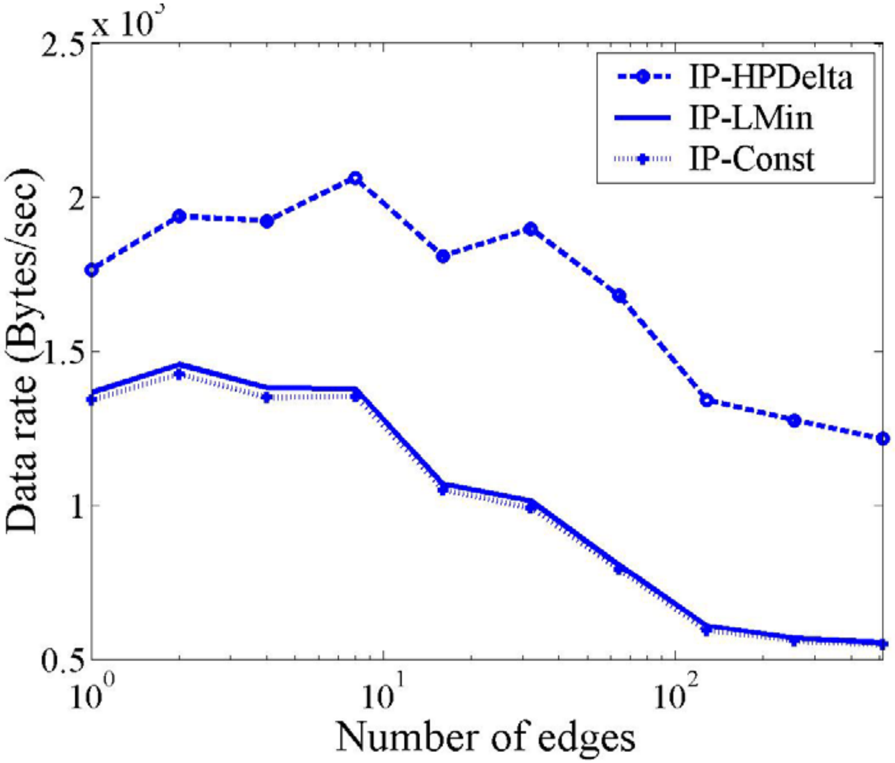
Performance Scaling

- Effects of asymptotic bounds cannot be seen
  - Not with increasing file size
  - Not with more vertices
  - Not with more edges



Performance Scaling

- Effects of asymptotic bounds cannot be seen
  - Not with increasing file size
  - Not with more vertices
  - Not with more edges
- Heuristics are “supposed to” differ as edges and vertices increase



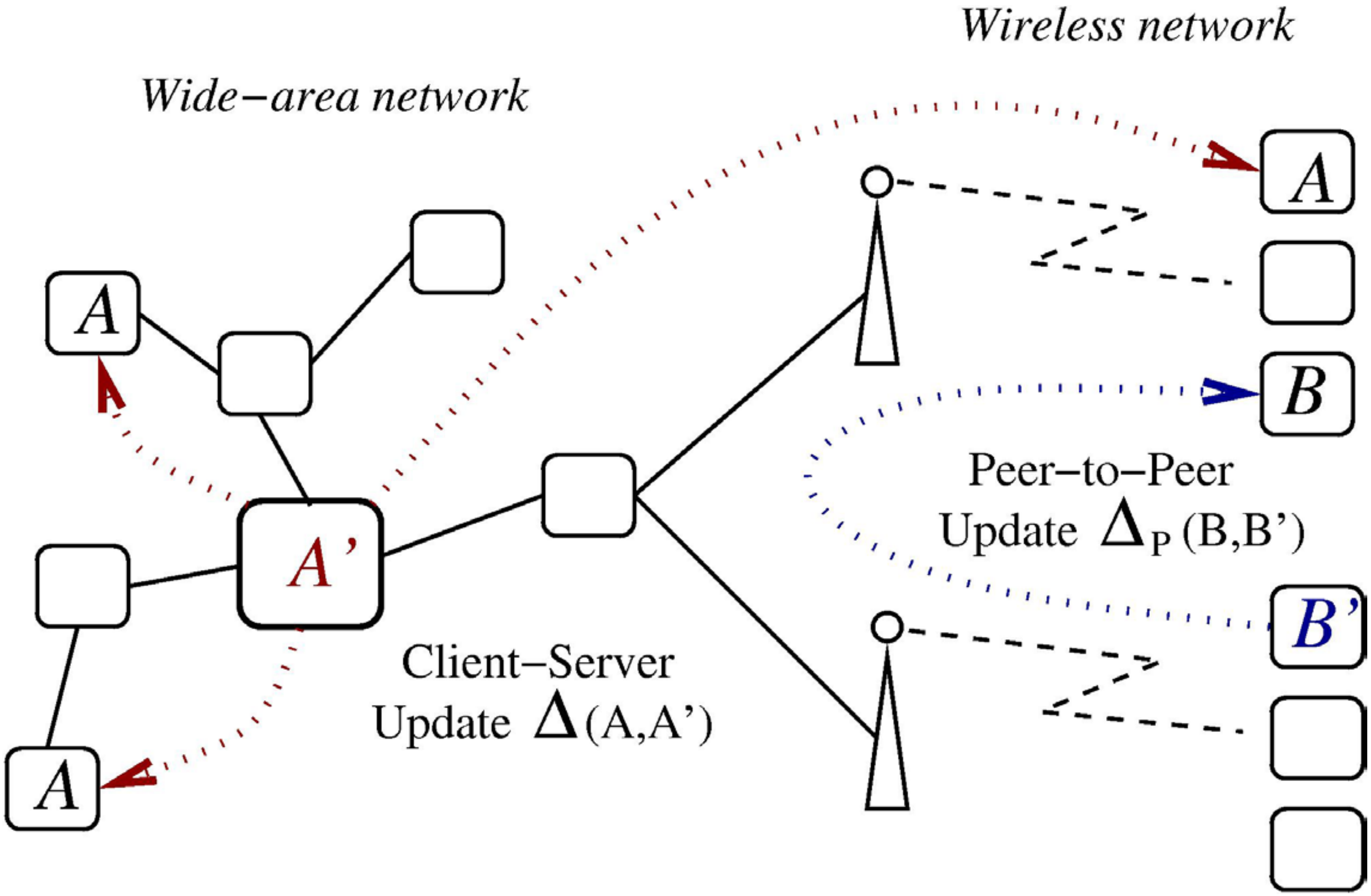
What did we learn?

- In-place delta files preserve the compression of regular delta files
  - Suitable for data distribution in wireless and mobile distributed systems
  - Provide same benefits that they do for Internet data transfer and backup/restore
- Compression loss influences algorithmic performance
  - Algorithms are I/O bound
  - Reducing compression, even if seemingly more expensive is desirable

Future Directions

- Reducing compression loss
  - Exploiting a bounded amount of scratch space
  - “Harvesting” scratch space from within the file
- Alternative data models
  - Structured data (XML, HTML), different differencing algorithms, but similar formats
  - Peer-to-peer in-place delta algorithms, analagous to Rsync
- Distribute in-place toolkit for building mobile and wireless distributed systems

An In-Place System


Wireless network
Wide–area network
A
A
A
A'
B
B'
Peer–to–Peer
Update Δ_P (B,B')
Client–Server
Update Δ(A,A')