P2P Data Copy Program bbcp

http://www.slac.stanford.edu/~abh/CHEP2001/p2p_bbcp.htm

Andrew Hanushevsky, Artem Trunov, Les Cottrell
Stanford Linear Accelerator Center
September, 2001

Produced under contract DE-AC03-76SF00515 between Stanford University and the Department of Energy

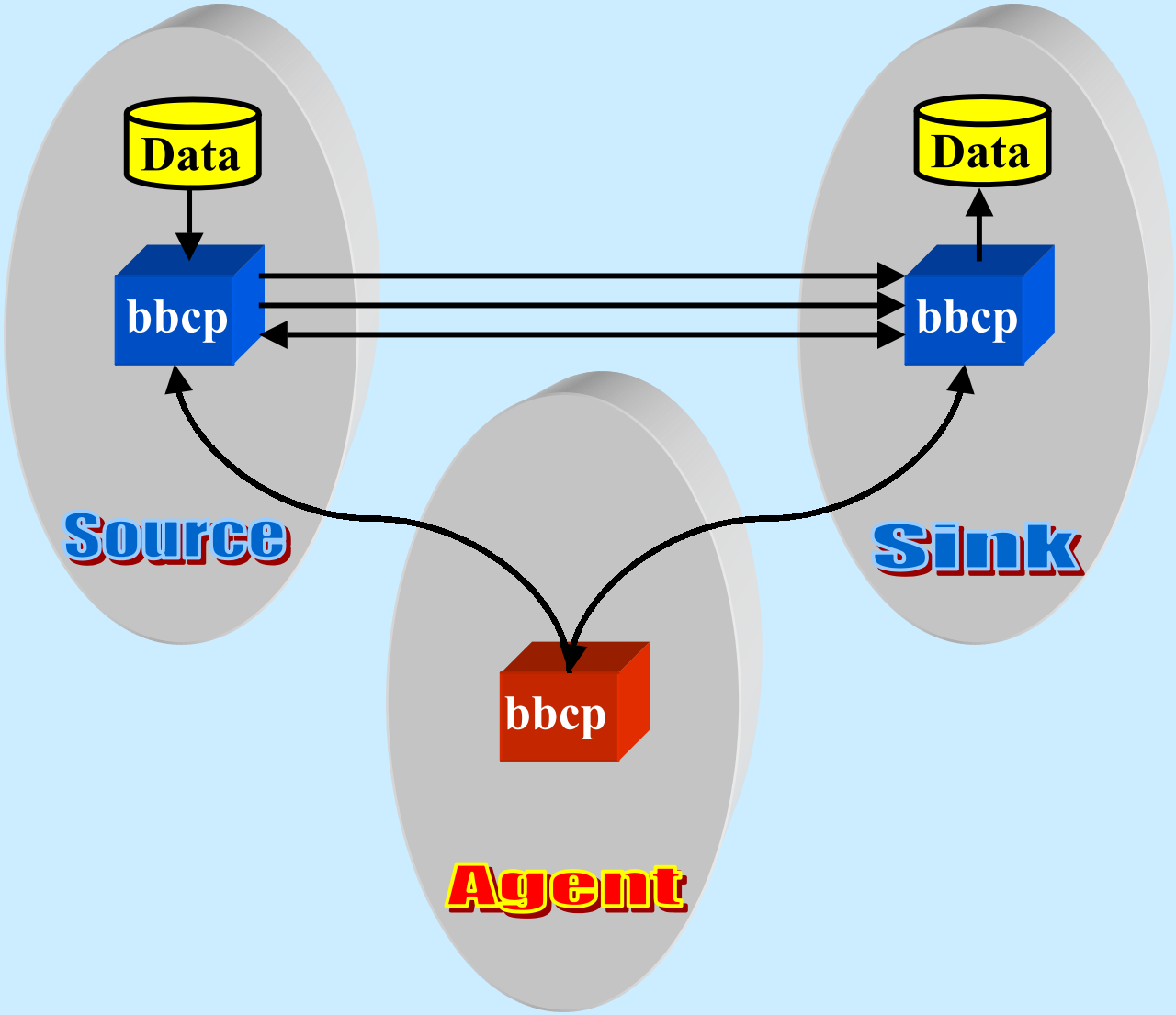
Why bbcp?

- Assess peer-to-peer technology
  - Fast deployment (minimal administration)
- Test novel data transfer algorithms
  - C++ component design
- Assess high level, low cost security
  - ssh-based authentication (control path)
  - Single use passwords (data path)
- Test familiar syntax to expedite use
  - Syntax same as for scp

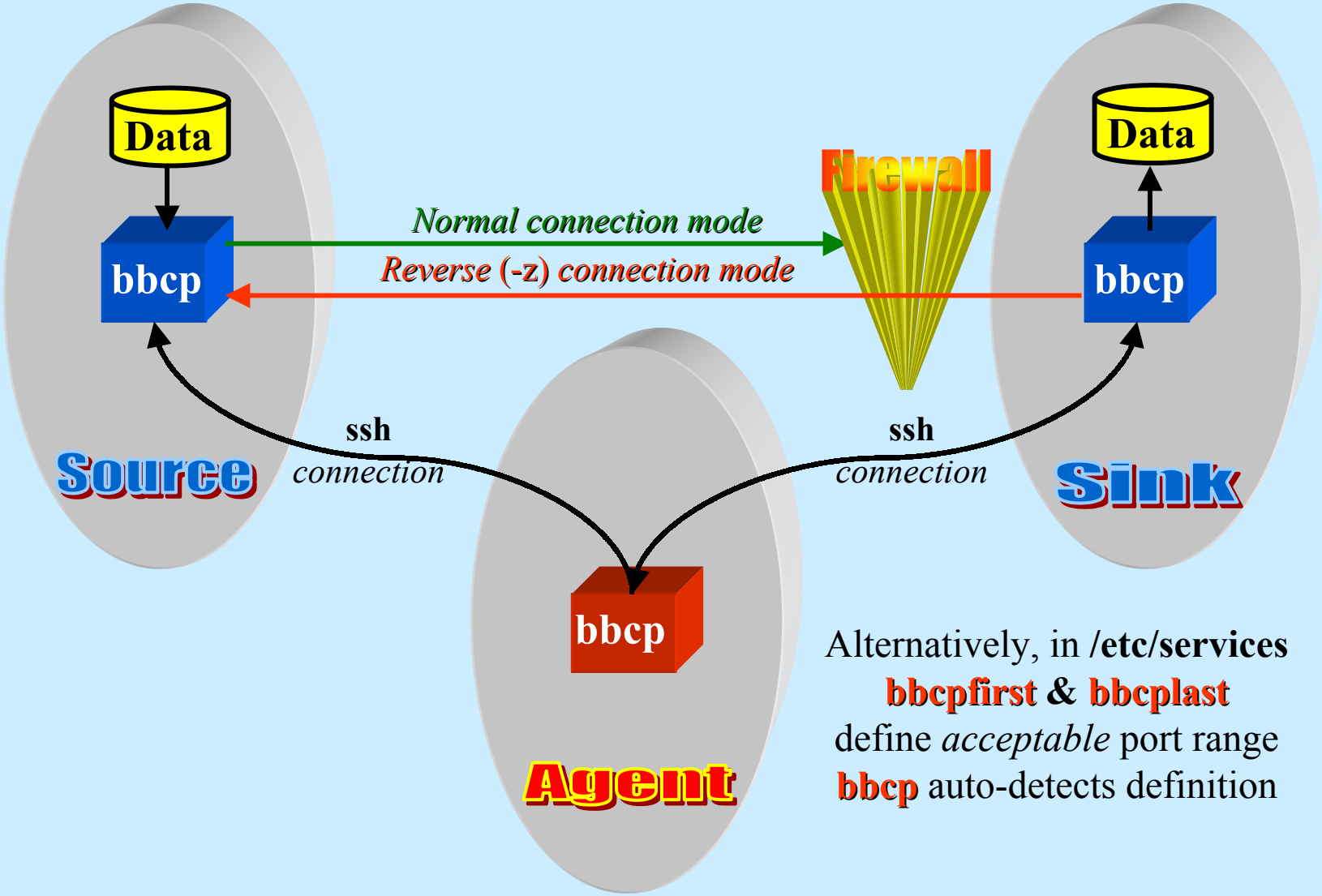
Peer-to-Peer Technology

- No identifiable client or server
  - Any node can act as data source or sink
- Well established for file sharing
  - Napster, Aimster, Gnutella, etc.
  - In many ways, rcp and scp similar to P2P
- Well suited for fast service deployment
  - If you have the program you have the service
    - Usually no need for administrators to get involved

Peer-to-Peer Architecture



Peer-to-Peer & Firewalls

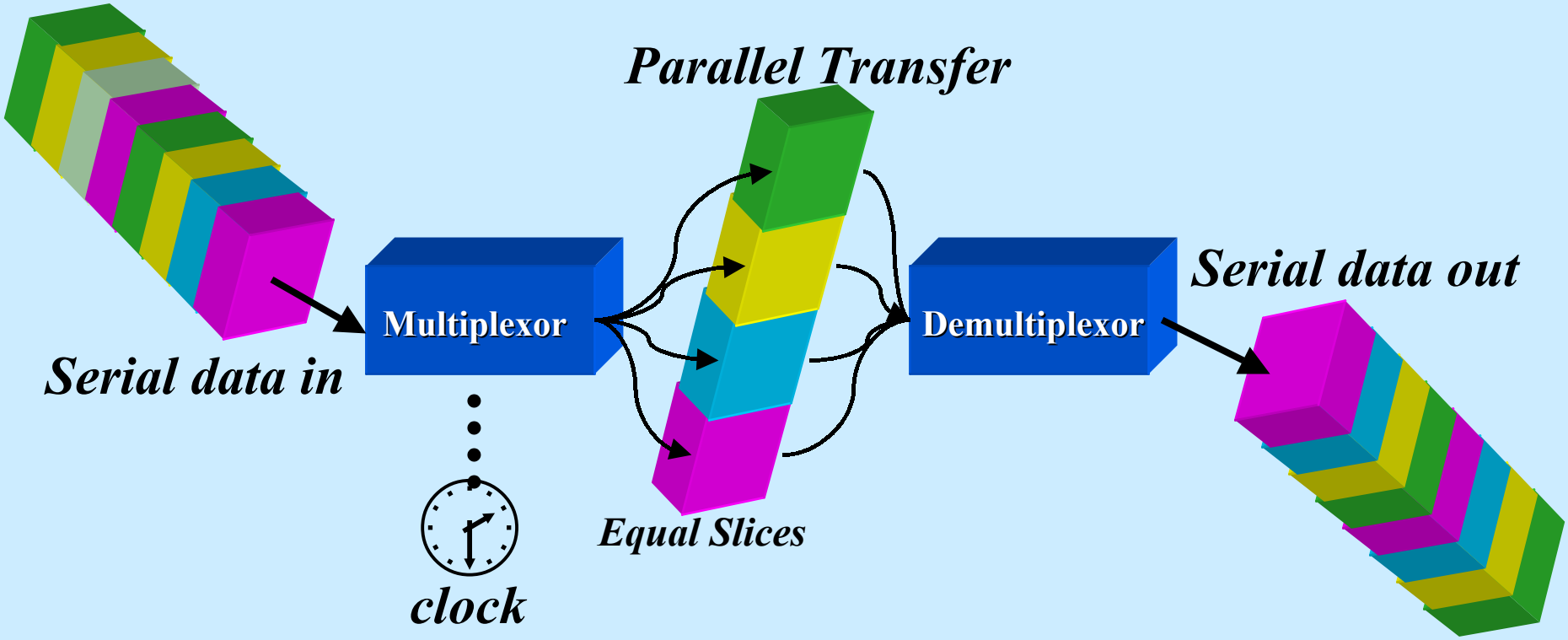


Alternatively, in **/etc/services**
bbcpfirst & **bbcplast**
define *acceptable* port range
bbcp auto-detects definition

Novel Algorithms

- Data pipelining
  - Multiple streams “simultaneously” pushed
    - Automatically adapts to router traffic shaping
    - Maximum rate controlled via data “clocking”
- Coordinated buffers
  - All buffers same-sized from end to end
    - Data queues never over- or under-filled
- Page aligned buffers
  - Allows direct I/O on many filesystems
    - Veritas

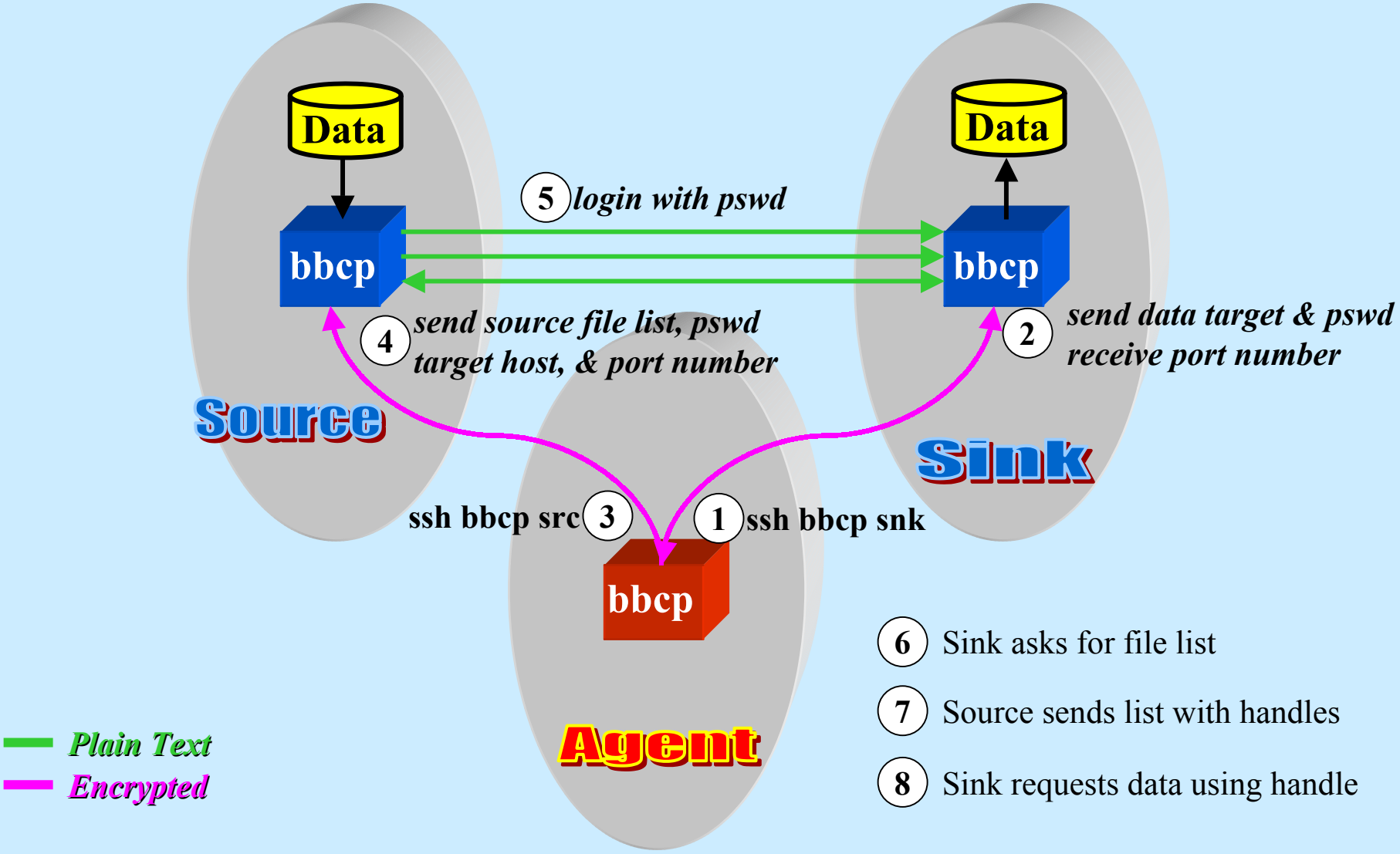
Algorithm Design



Security

- Low cost, simple and effective security
  - Leveraging widely deployed infrastructure
    - If you can ssh there you can copy data
- Sensitive data is encrypted
  - One time passwords and control information
- Bulk data is not encrypted
  - Privacy sacrificed for speed
- Minimal sharing of information
  - Source and Sink do not reveal environment

Protocol Interactions



6 Sink asks for file list

7 Source sends list with handles

8 Sink requests data using handle

Invocation

- Familiar syntax
  - **bbcp** [*options*] *source* [*source* […]] *target*
- Sources and target can be anything
  - [[*username*@]*hostname*:]]*path*
  - /dev/zero *or* /dev/null
- Easy but powerful
  - Can gather data from multiple hosts
  - Many usability and performance options
- http://www.slac.stanford.edu/~abh/bbcp

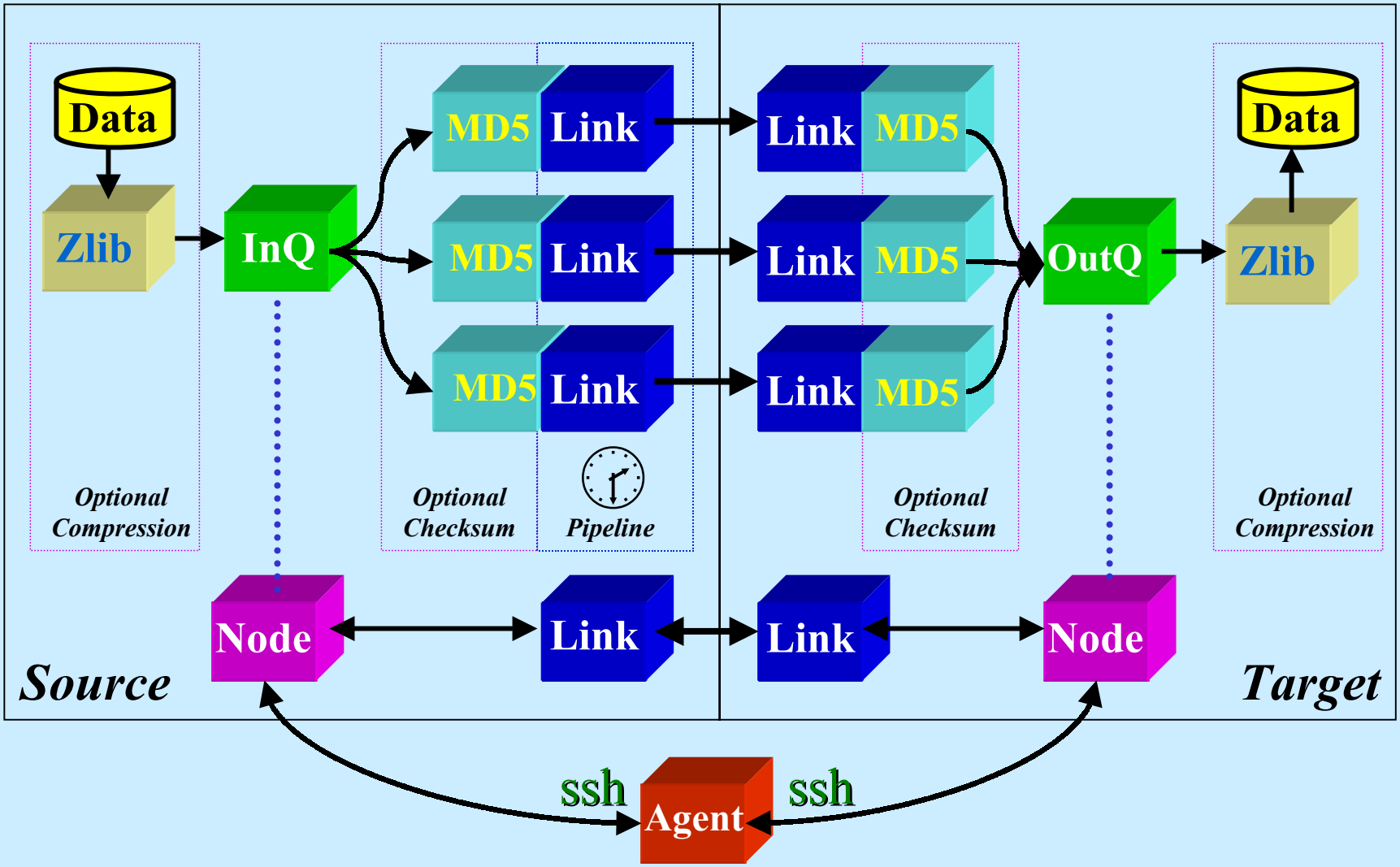
Usability Features

- Serial Input & Output
  - Source & target can even be a pipe or tape drive
- Auto-resume failed copies
  - Only un-copied data transferred after failure
- Preserve group ownership and file times
- Auto-create directories
- Command line include files that list files to be copied
- History log file
- Periodic progress messages
- Transfer rate limiting
- MD5 Checksums
- Compression
- Multiple performance tuning options

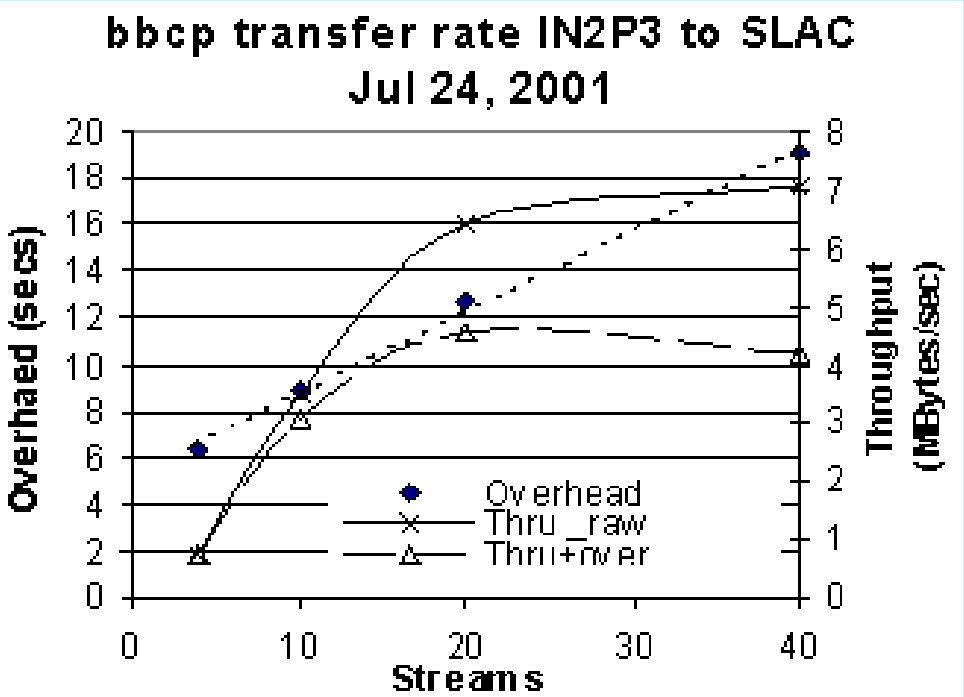
Optional

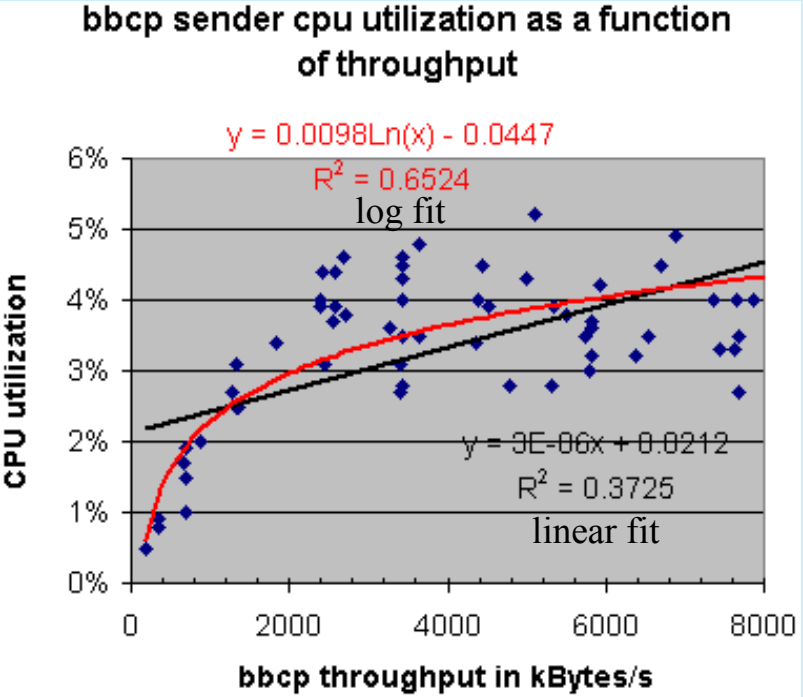
- And more …
  - http://www.slac.stanford.edu/~abh/bbcp

The Inside Details



The Results





bbcp can perform within 5% of iperf reported maximum performance

For all results visit: http://www-iepm.slac.stanford.edu/monitoring/bulk/bbcp.html

Future Work

- Reduce start-up overhead
  - Make overhead largely independent of streams
- Real-time Adaptation
  - Transfer unit size and number of streams
    - Investigating network feedback monitors
- Real-time logging
  - Incorporate netlogger

Conclusion

- bbcp algorithms are effective
  - Excellent performance
  - Allow data serialization
- A different approach
  - East vs West
    - **East:** Transfer data in balance and harmony
    - **West:** Blast the data as fast as you can