Enabling High Performance Bulk Data Transfers With SSH

Chris Rapier
Benjamin Bennett
Pittsburgh Supercomputing Center
TIP ‘08

Moving Data

- Still crazy after all these years
  - Multiple solutions exist
    - Protocols
      - UDT, SABUL, etc…
    - Implementations
      - GridFTP, kFTP, bbFTP, hand rolled and more…
    - Not to mention
      - Advanced congestion control, autotuning, jumbograms, etc…

Many Solutions No Answers

- All developed as a solution to the same problem
  - Moving lots of a data very fast can be very difficult
- Unfortunately, no single solution meets all needs.
  - Fast, easy to use, inexpensive to maintain, flexible, secure

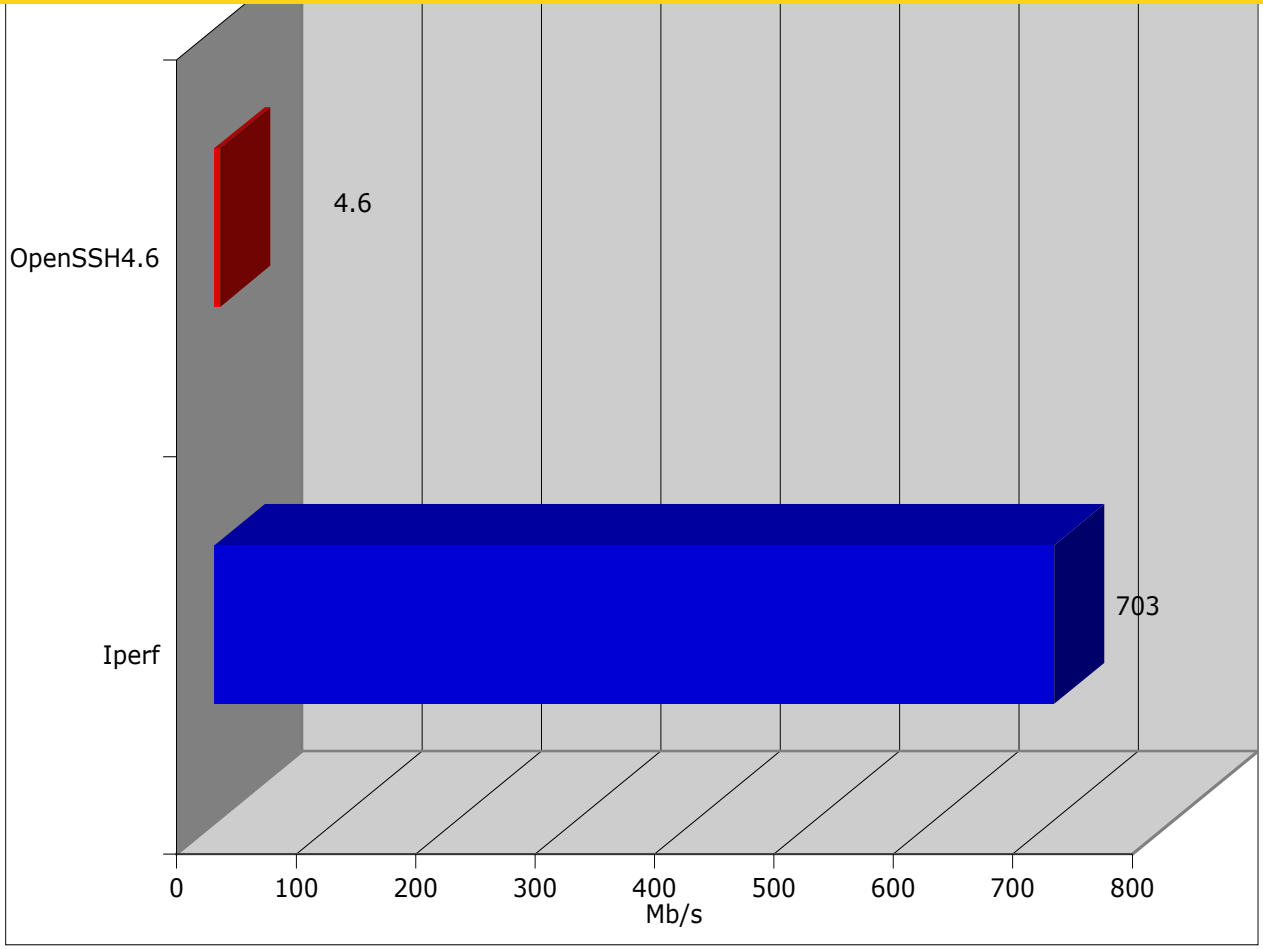
What About SSH?

- Easy to use.
- Cheap to maintain.
- Installed everywhere.
- Flexible.
- Strong cryptography.

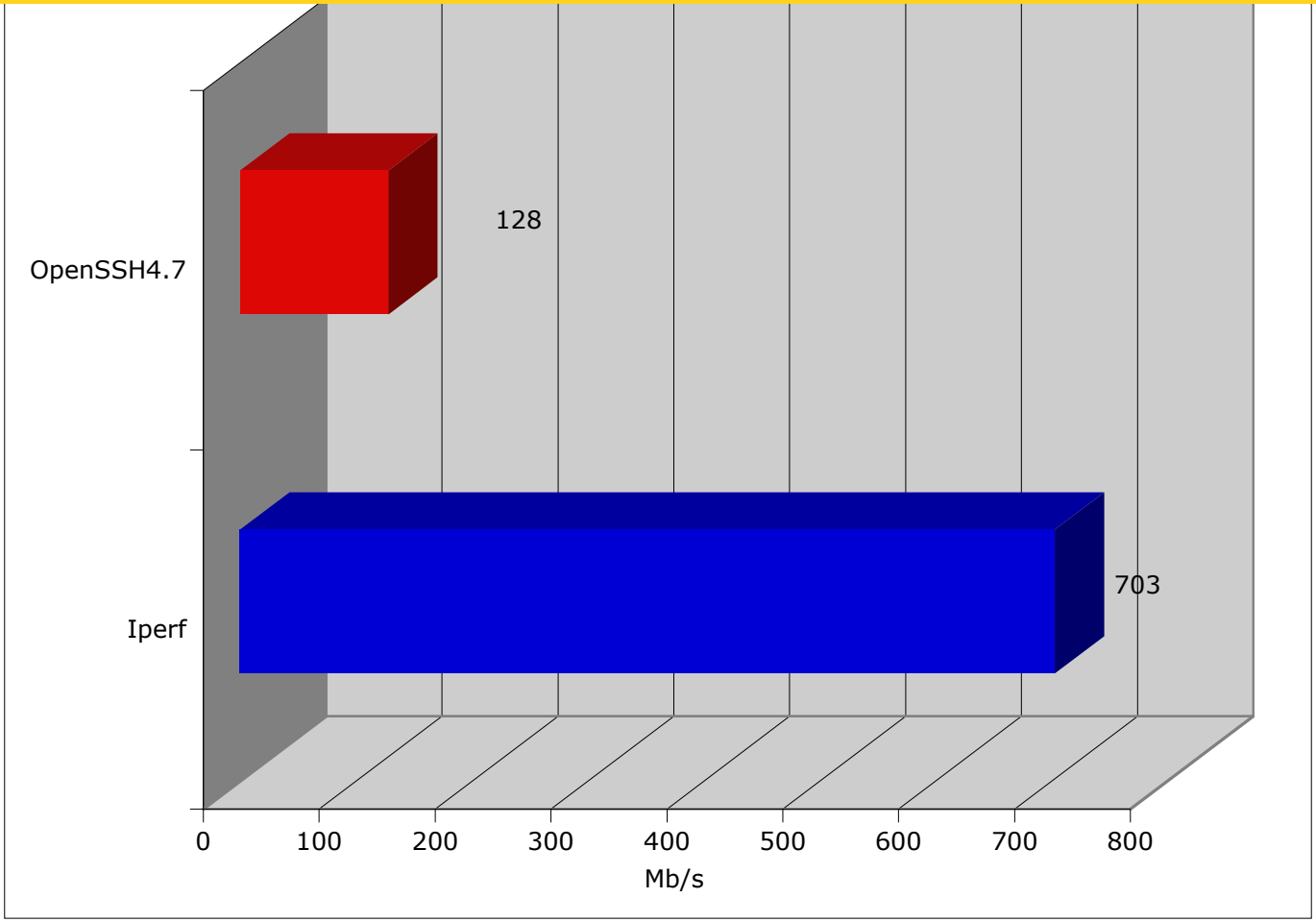
Why not SSH?

- It can be really really slow.

How slow?



A little better

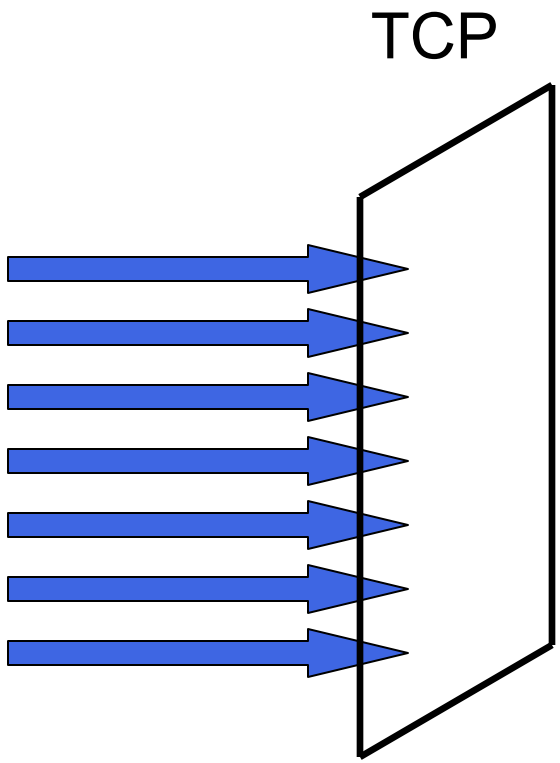


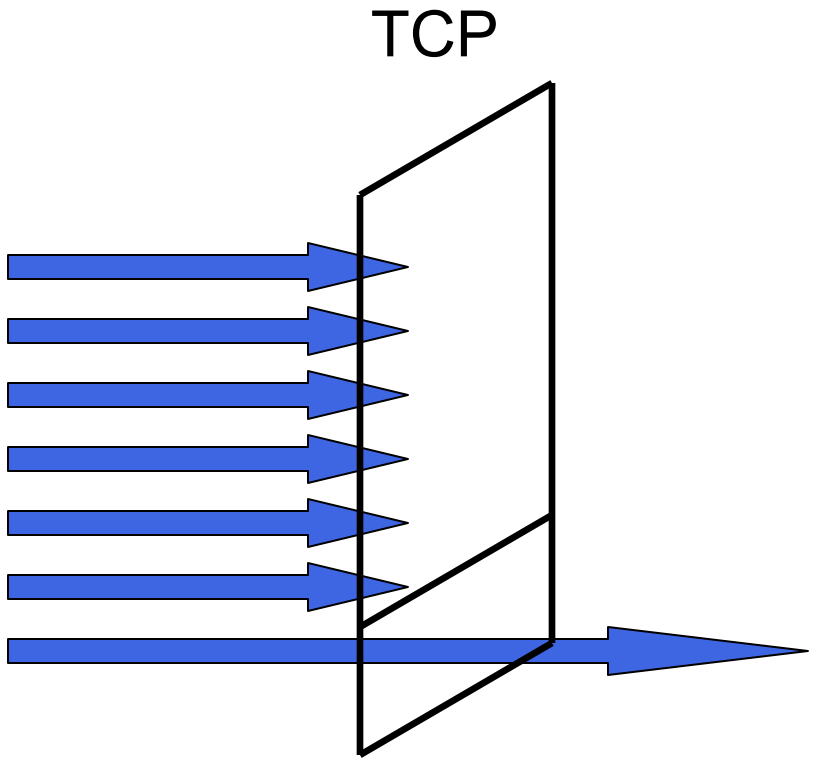
What changed?

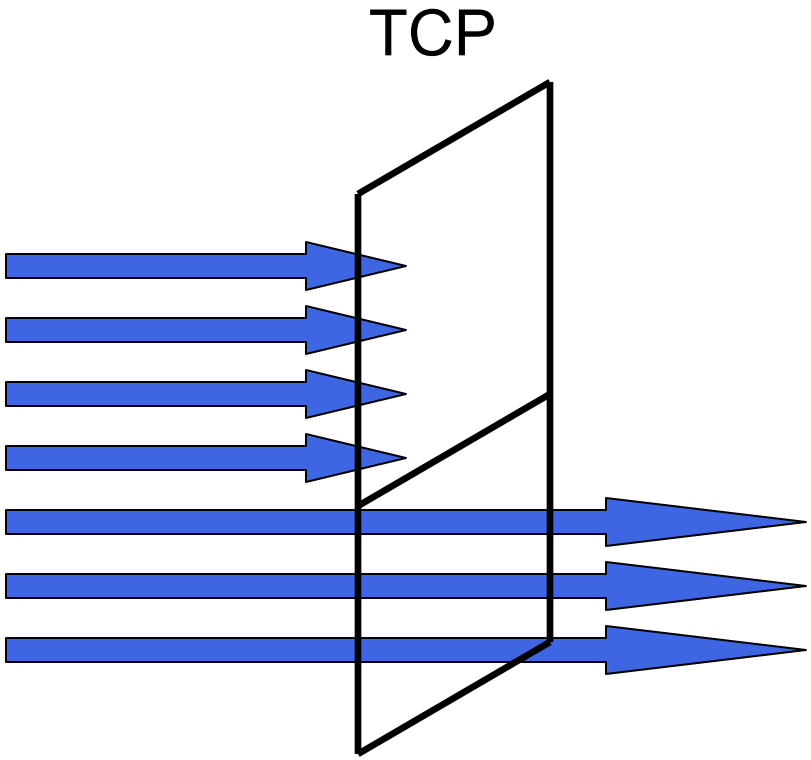
- Why the improvement in OpenSSH4.7?
  - SSH is a multiplexed application
    - Each channel requires its own flow control which is implemented as a receive window
  - In 4.7 the maximum window size was increased to ~1MiB up from 64KiB

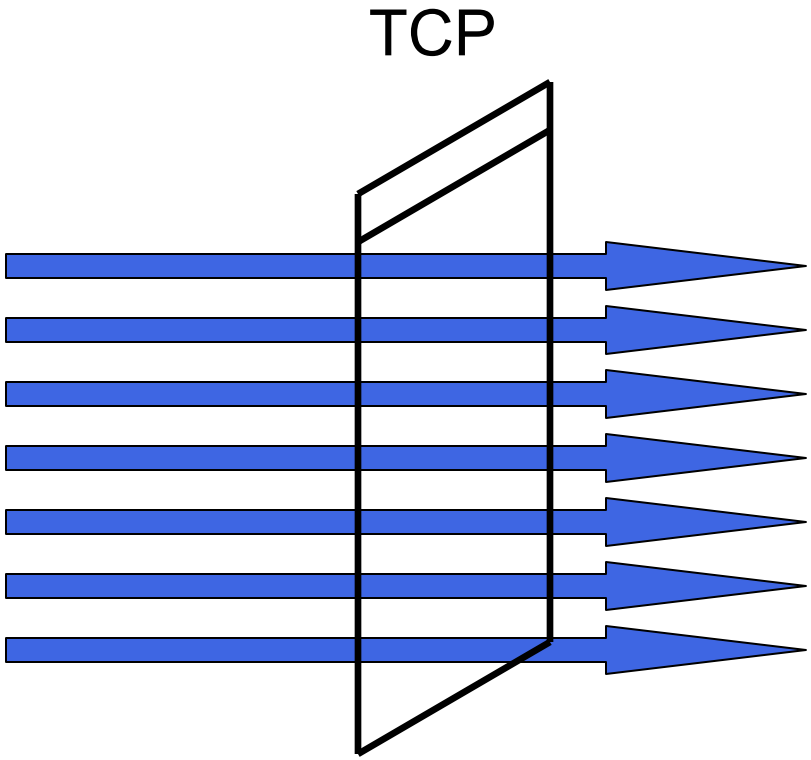
Windows

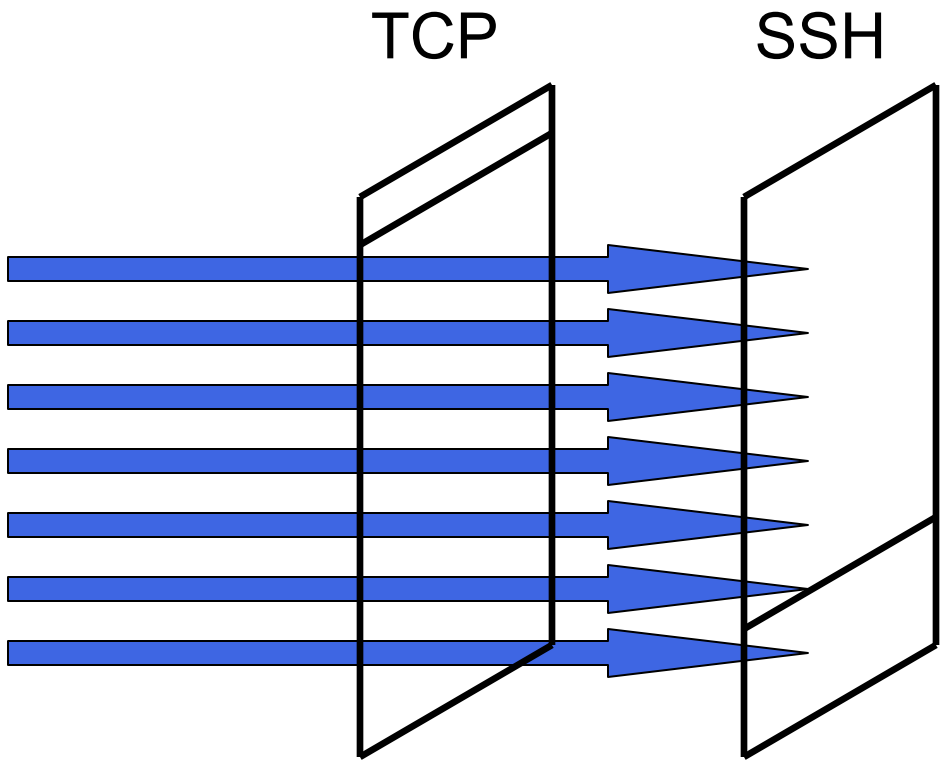
- Receive windows advertise the amount of data a system or application is willing to accept per round trip time.
- Effective window size is the minimum of all windows; protocol and application.
- Each window must be tuned and in sync to maximize throughput.
  - If any one is out of tune the entire connection will suffer.

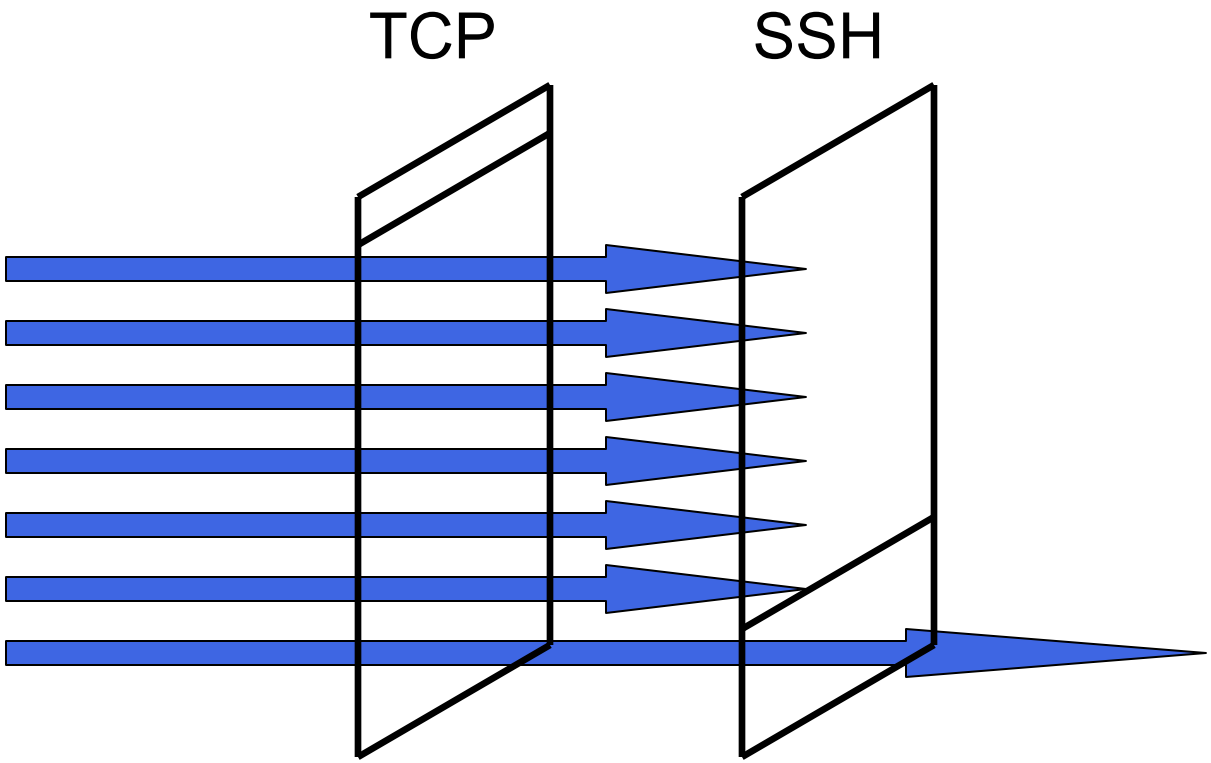

TCP


TCP


TCP

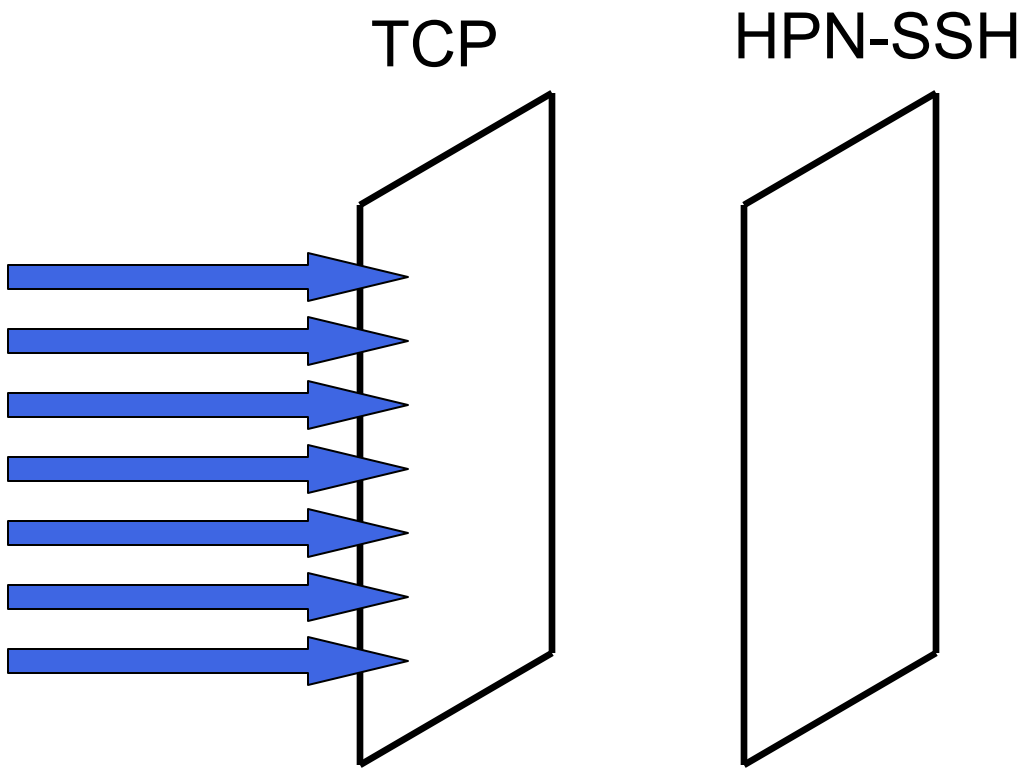

TCP

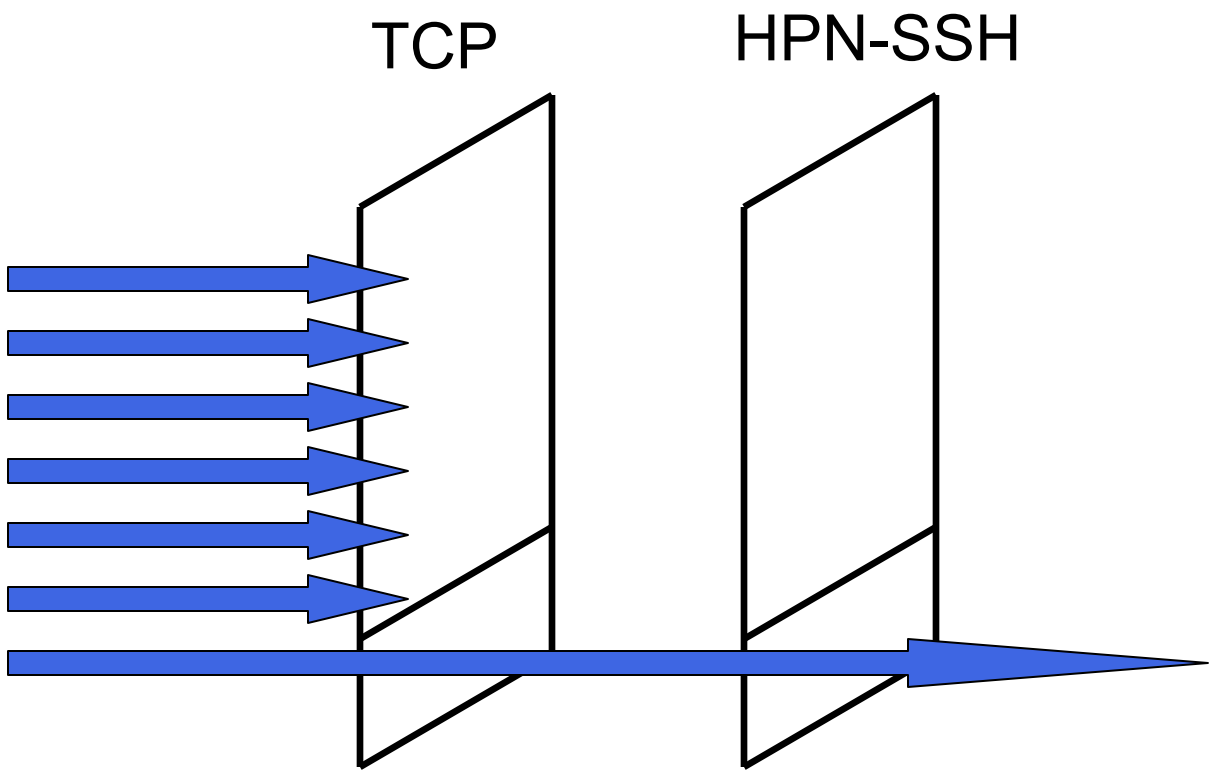

TCP
SSH

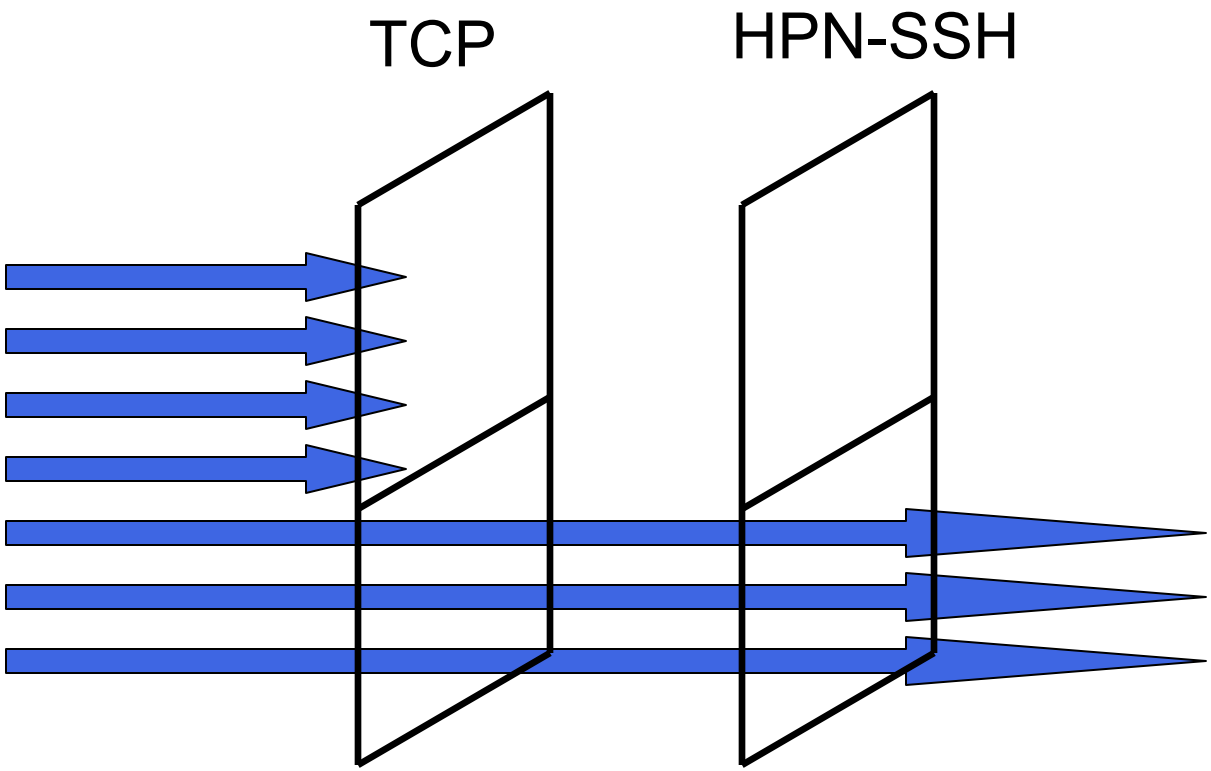

TCP
SSH

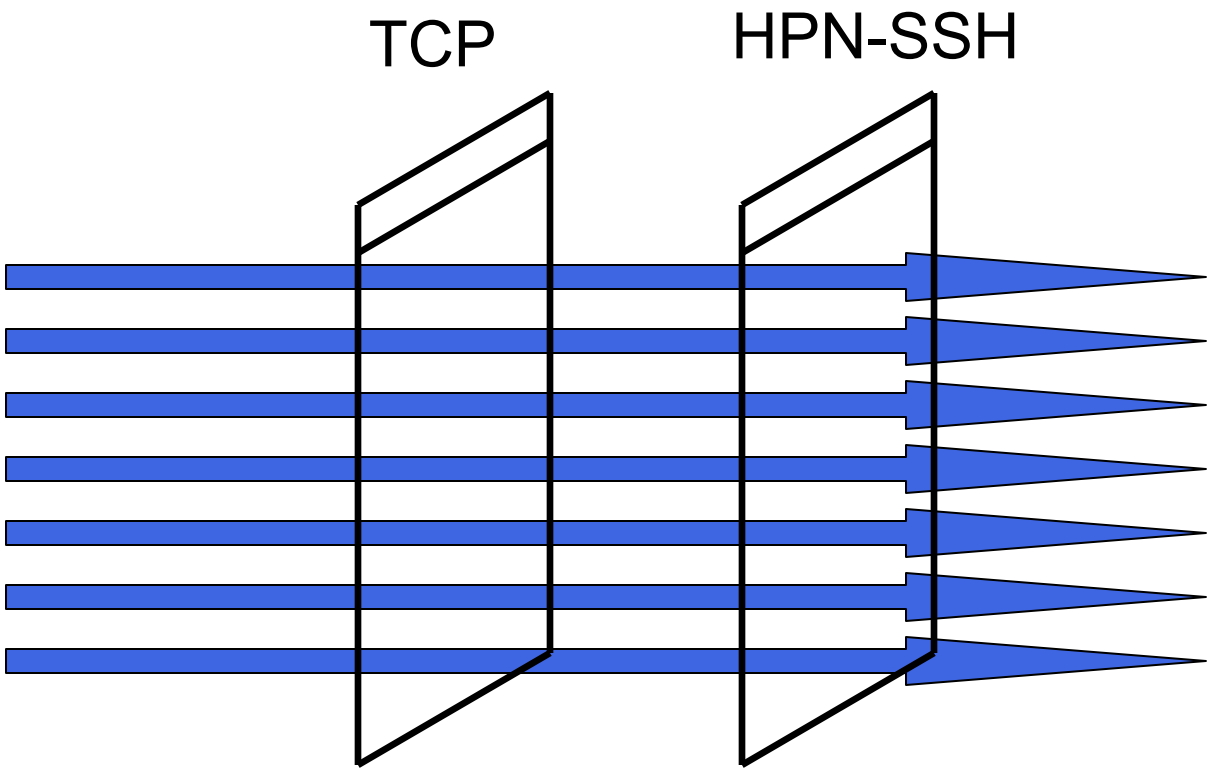
Windows in HPN-SSH

- Dynamically defined receive window size grows to match the TCP window.
  - Set to TCP RWIN on start.
  - Grows with RWIN if autotuning system.
  - Dynamic sizing reduces issues of over-buffering problems.


TCP
HPN-SSH

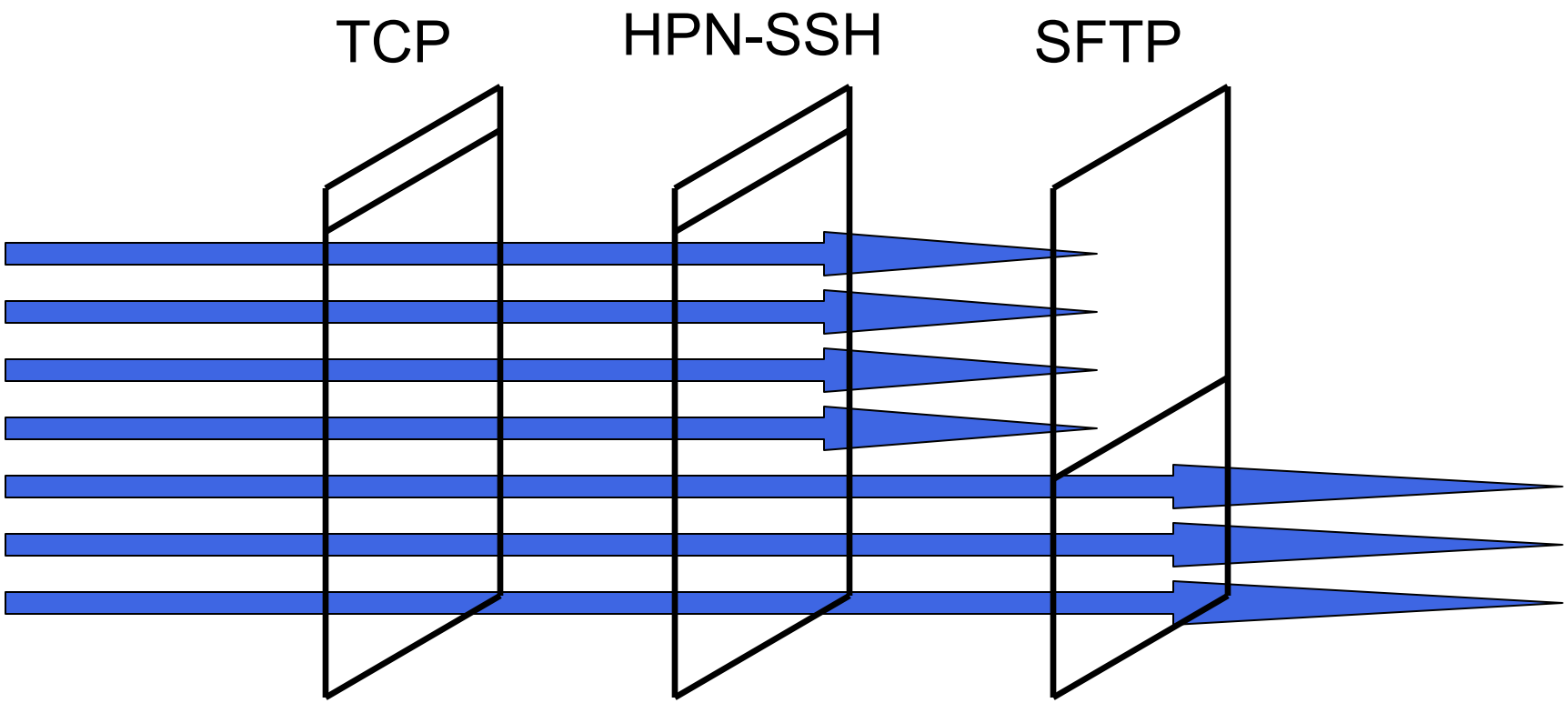

TCP
HPN-SSH


TCP
HPN-SSH

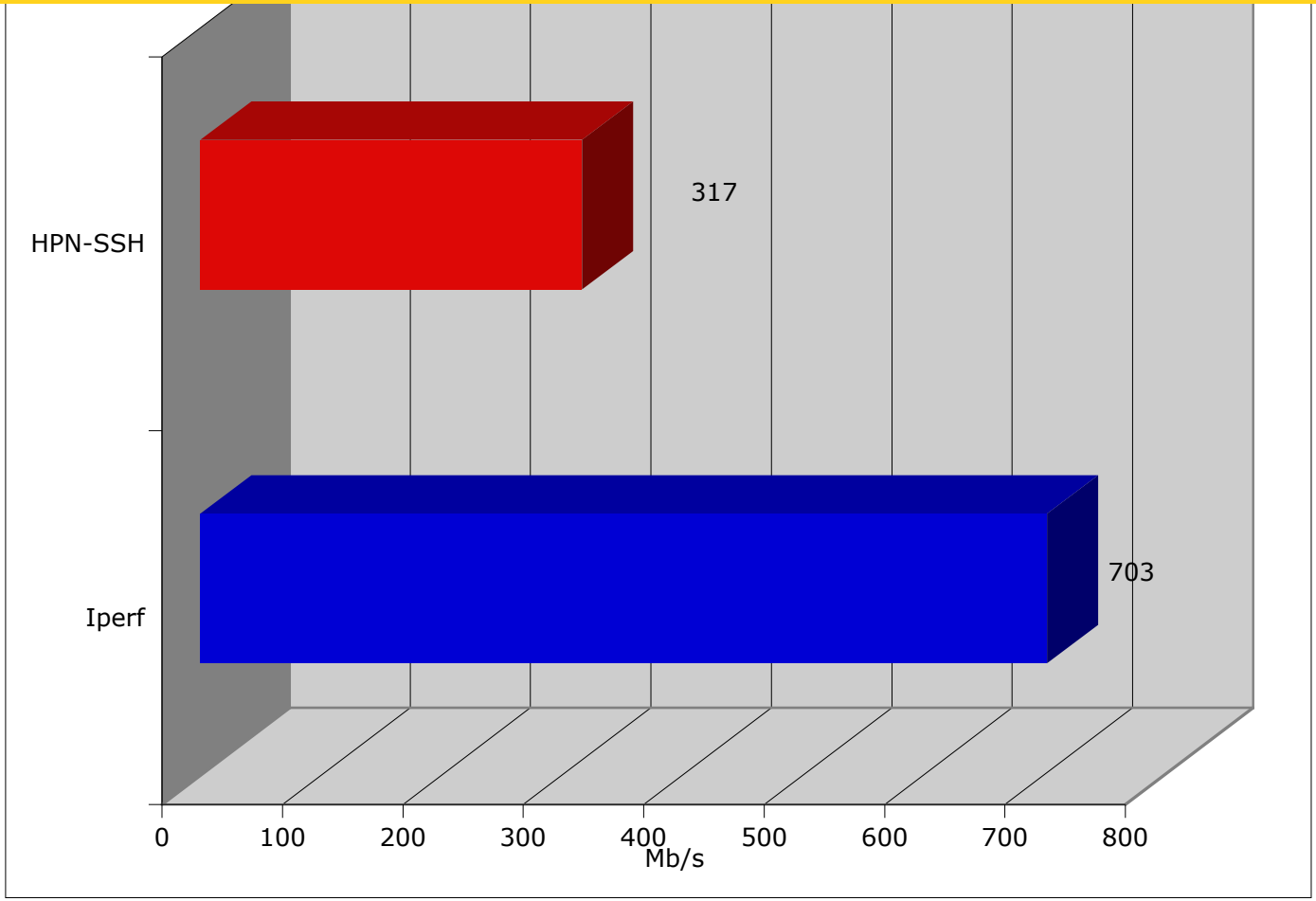

TCP
HPN-SSH

SFTP is Special

- SFTP adds *another* layer of flow control.
  - All SFTP packets are treated as requests
  - By default no more than 16 outstanding requests.
  - Results in a 512KiB window
  - Increase using -R on command line


TCP
HPN-SSH
SFTP

A lot better



But…

- As the throughput increases crypto demands more of the processor.
  - The transfer is now processor bound

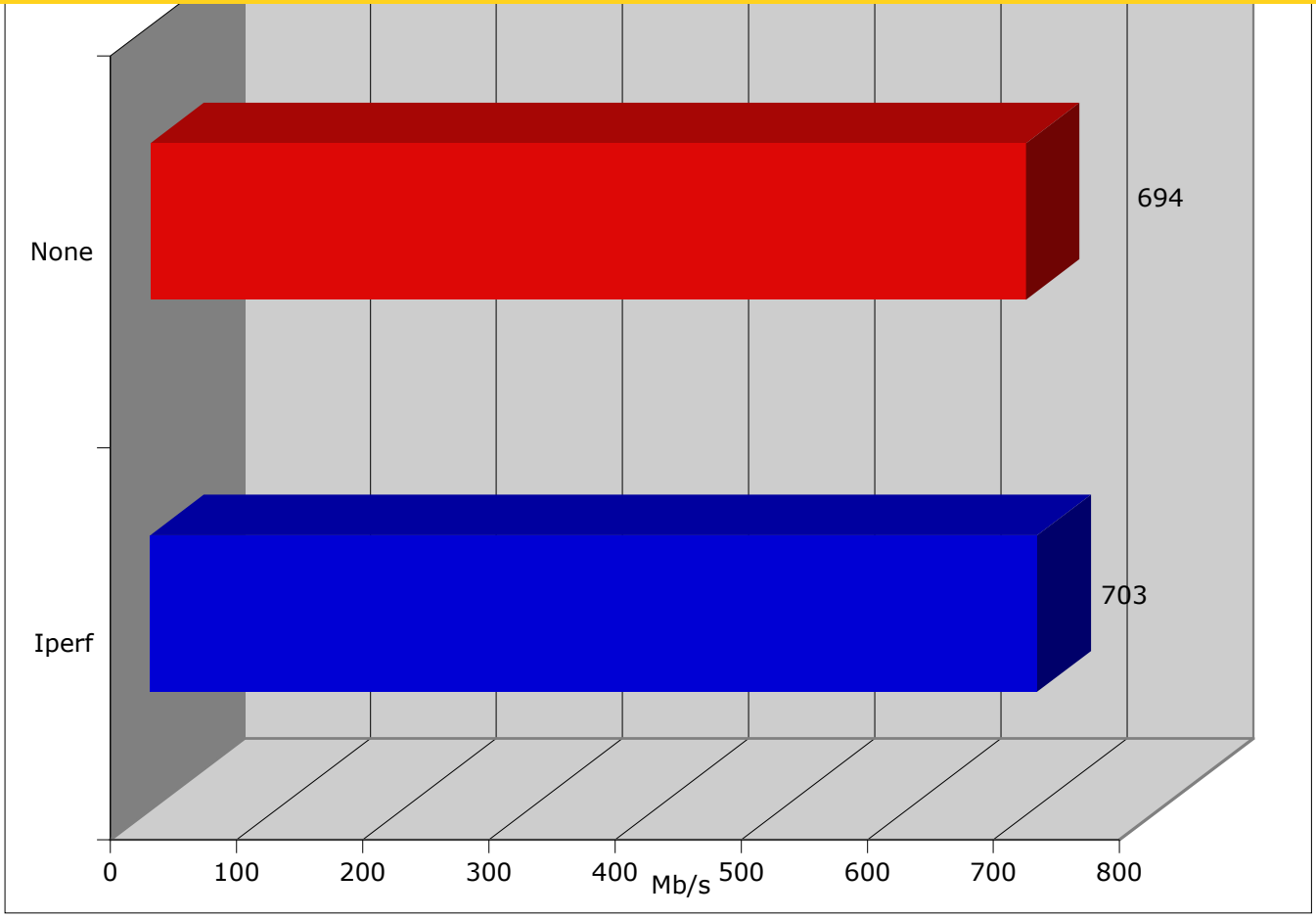
We Need More Power?

- Two solutions to processor bound transfers
  - Throw more processing power at the problem
  - Do the work more efficiently
    - Define ‘work’

The None Switch

- Many people only need secure authentication. The data can pass in the clear.
  - HPN-SSH allows users to switch to a ‘None’ cipher after authentication.

Done!



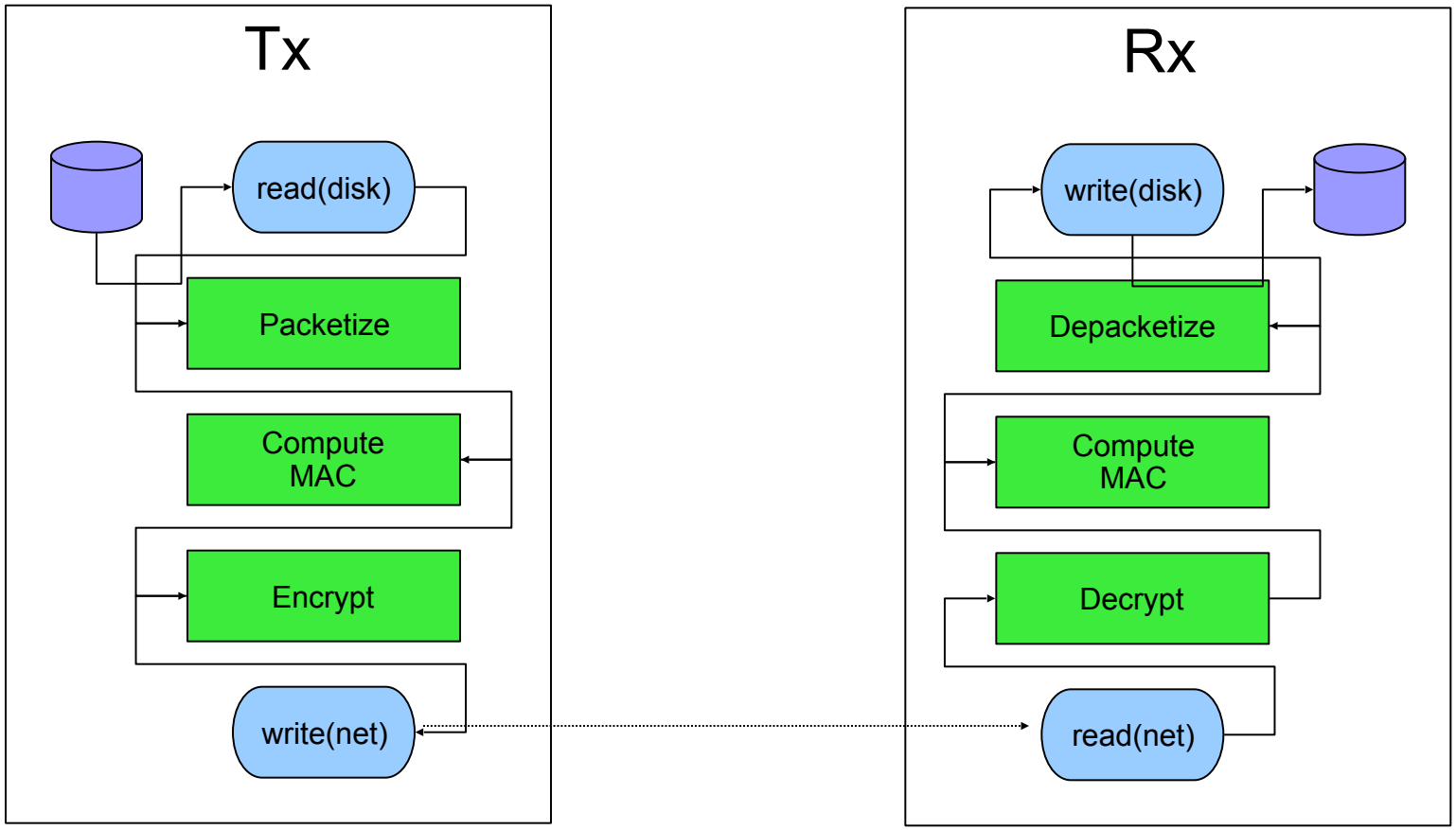
As far as we can go?

- Windows are already optimized.
  - No more real improvements available there
- NONE cipher is limited to a subset of transfers.
  - Sometimes you absolutely need full encryption.
- So what now?

More Power

- Common assumption that current hardware is incapable of meeting crypto demand
  - Is it true?

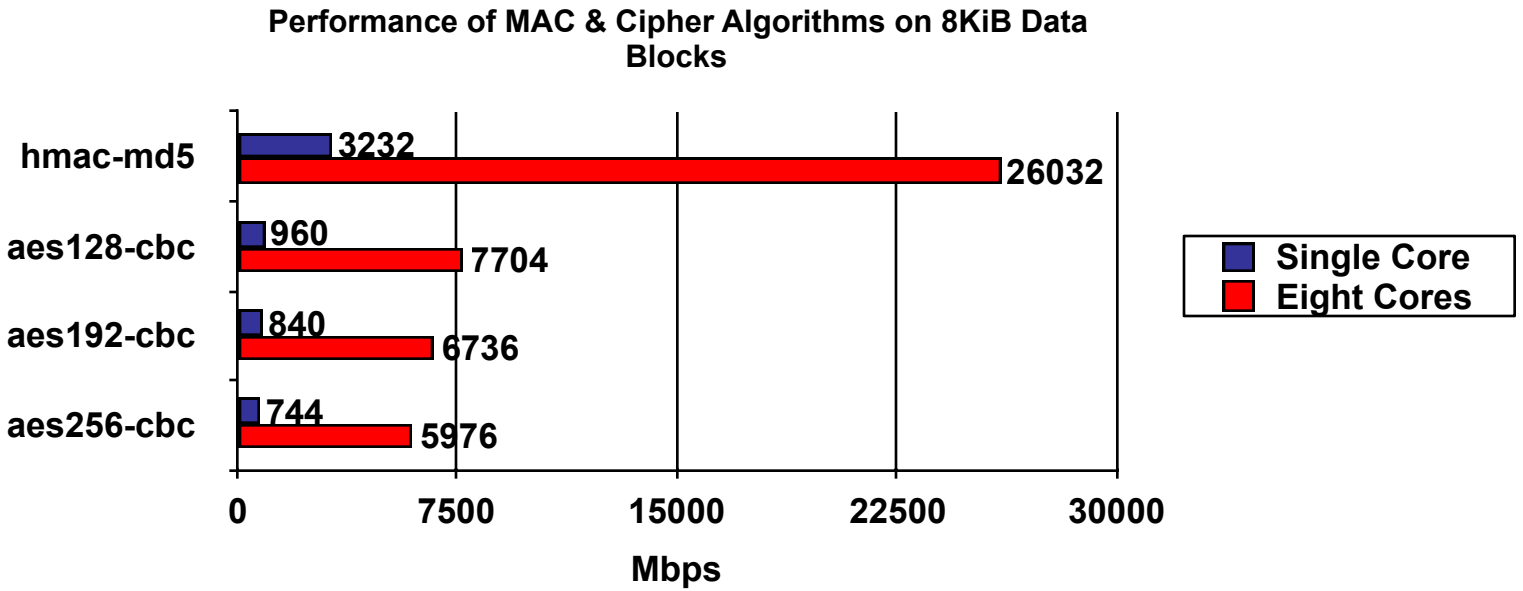
What does SSH need to do?



Today's Hardware

- Laptop
  - Two 64bit general purpose cores
  - 1GiB to 4GiB RAM
  - 1Gbps ethernet
- Desktop/Workstation
  - Two to eight 64bit general purpose cores
  - 1GiB to 8GiB RAM
  - 1Gbps ethernet

OpenSSL Benchmarks



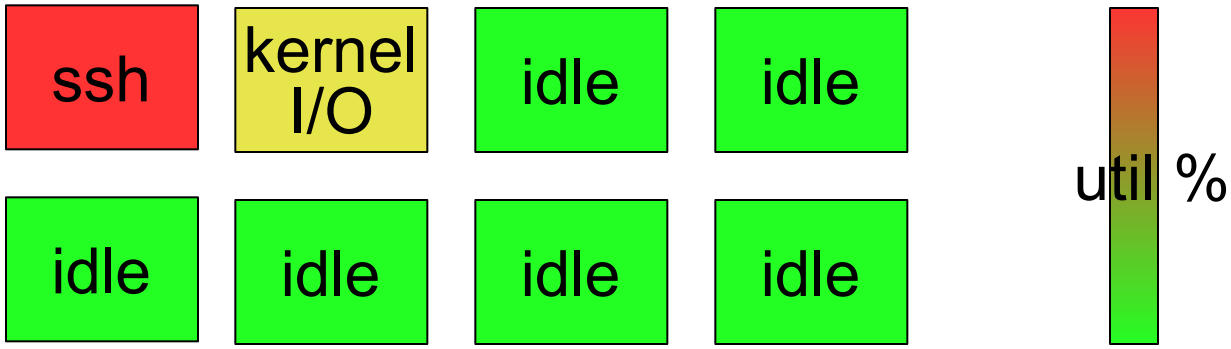
- Dual Intel Xeon 5345 Workstation
  - 4 cores per socket, 8 cores total @ 2.33Ghz
  - Fedora 7 stock OpenSSL build

We have the CPU power

- hmac-md5 @ 1Gbps, ~0.3 cores
- aes256-cbc @ 1Gbps, ~1.34 cores
- Crypto total @ 1Gbps, ~1.64 cores
- We have 8!

So what's the problem?

- MAC requires fraction of one core
- Cipher requires more than one core
- MAC, cipher, and more all within a single execution thread



How can we fix it?

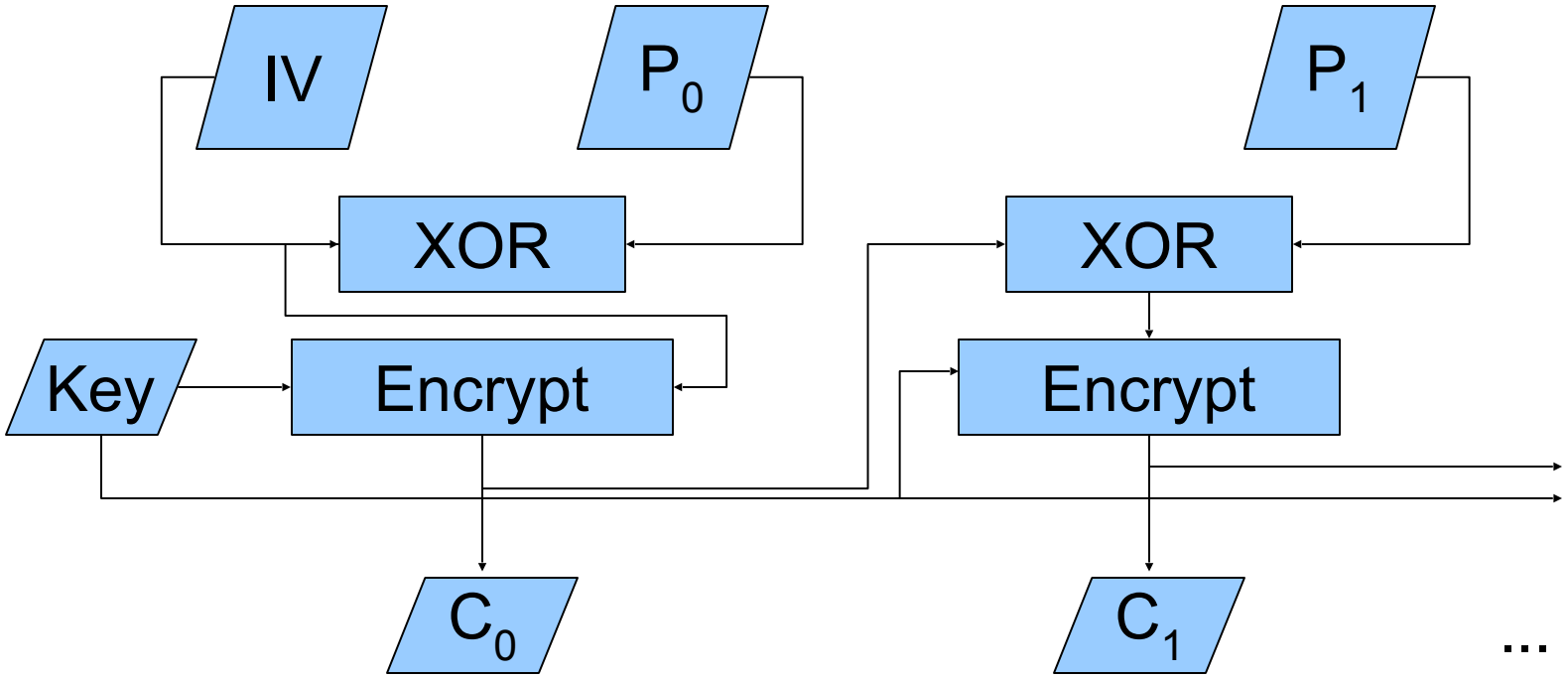
- Multi-threading on functional boundaries
  - Perform MAC and cipher on a packet concurrently
    - Possible on sender, not on receiver
  - Process multiple packets concurrently (pipeline)
  - Cipher still needs more than one core
- Multi-threading within cipher
  - Can it be parallelized?

SSH Cipher Modes

- CBC
  - Most common
  - RFC 4253 “The Secure Shell (SSH) Transport Layer Protocol” specifies only CBC mode ciphers, arcfour, and none.
- CTR
  - Specified in RFC 4344 “SSH Transport Layer Encryption Modes”
  - More desirable security properties than CBC

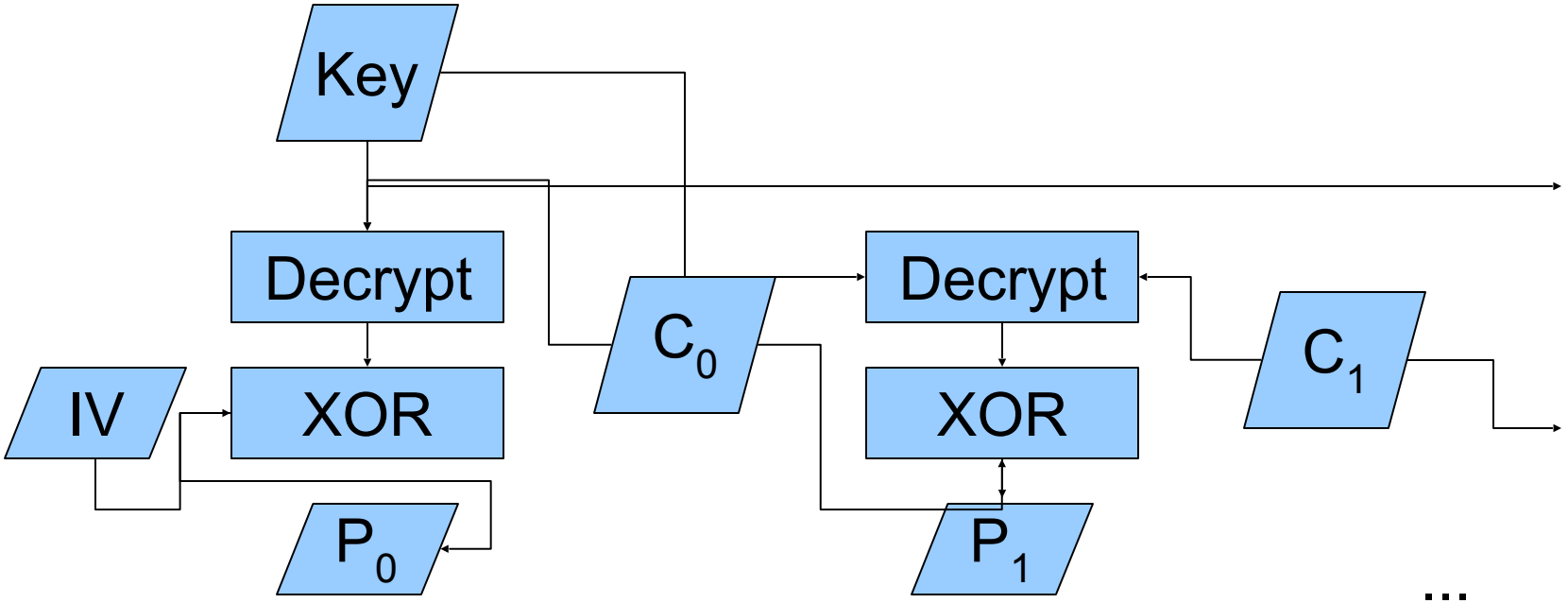
Hello, my name is CBC

- Cipher Block Chaining Mode Encryption



Hello, my name is CBC (cont)

- Cipher Block Chaining Mode Decryption

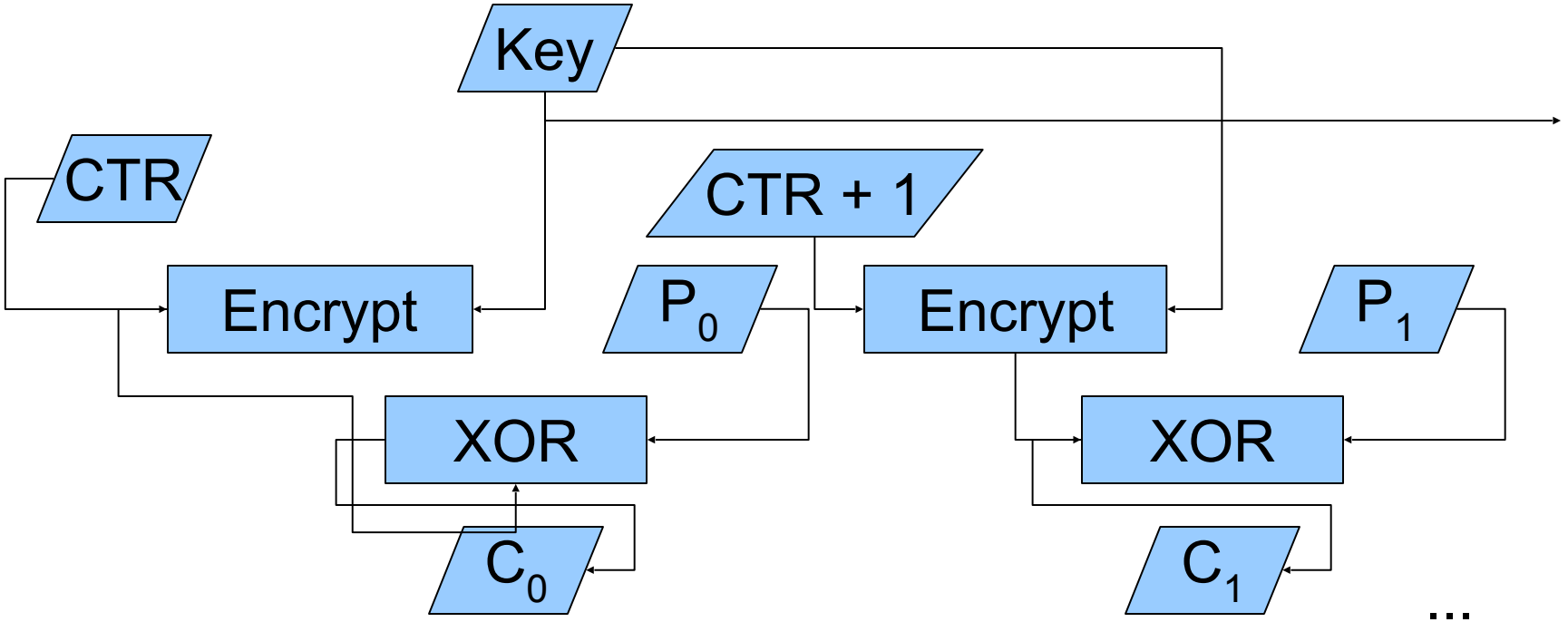


CBC Summary

- Encrypt must be serial
- Decrypt may be parallel
- That doesn't help so much :-(

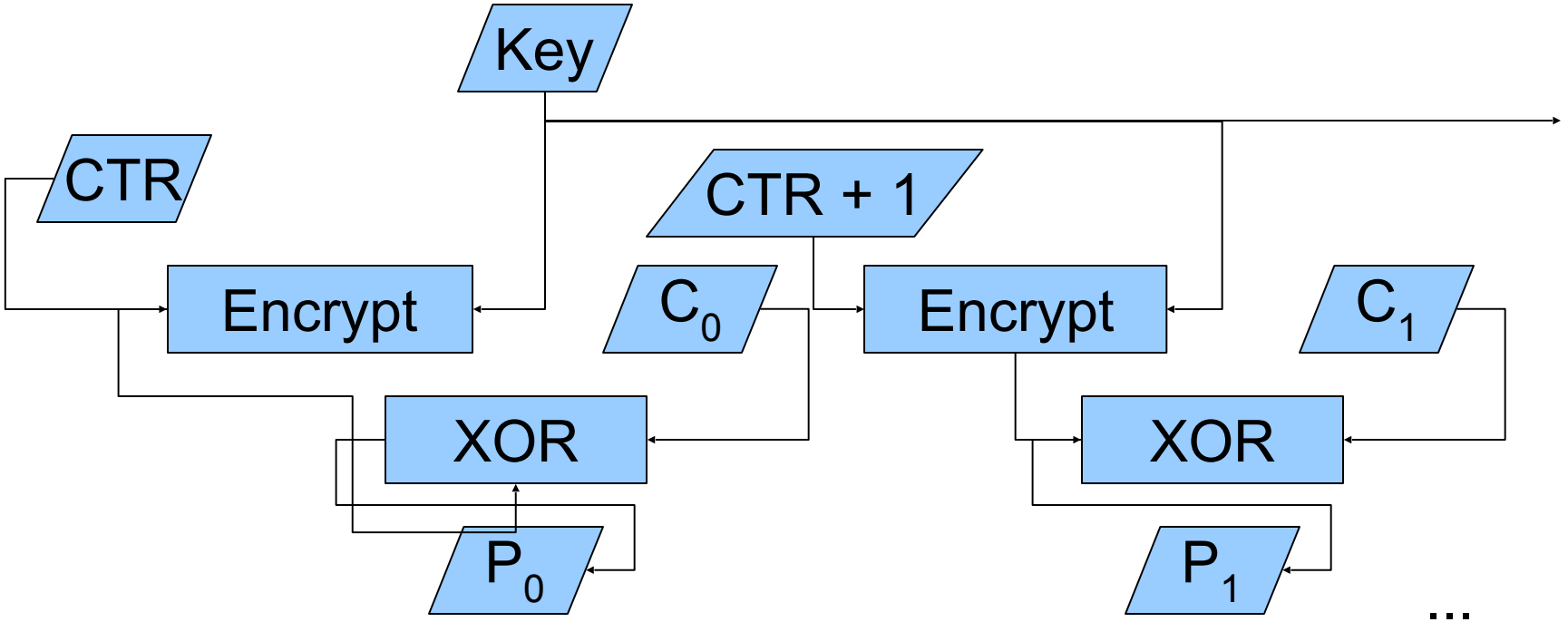
Hello, my name is CTR

- Counter Mode Encryption



Hello, my name is CTR (cont)

- Counter Mode Decryption



CTR Summary

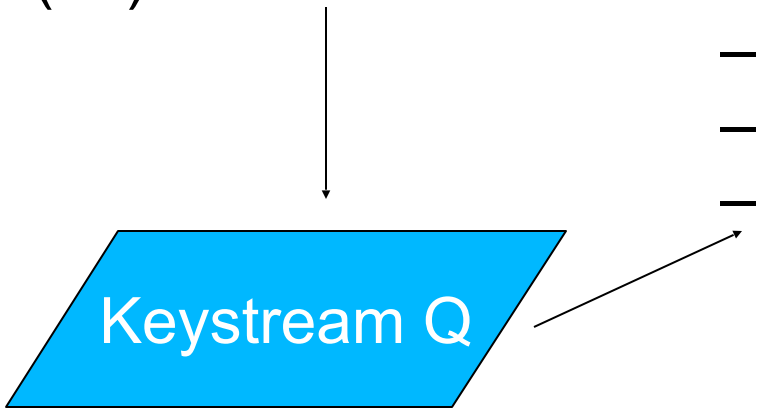
- Encrypt may be parallel
- Decrypt may be parallel
- Keystream can be pregenerated
- Let’s get to work…

Multi-threaded AES-CTR

- Uses arbitrary number of cipher threads (and cores) to generate a single keystream.
- Cipher threads pre-generate keystream, starting once a cipher context key and IV are known.
- Leaves only keystream dequeue & XOR for encrypt/decrypt operations in main SSH thread.

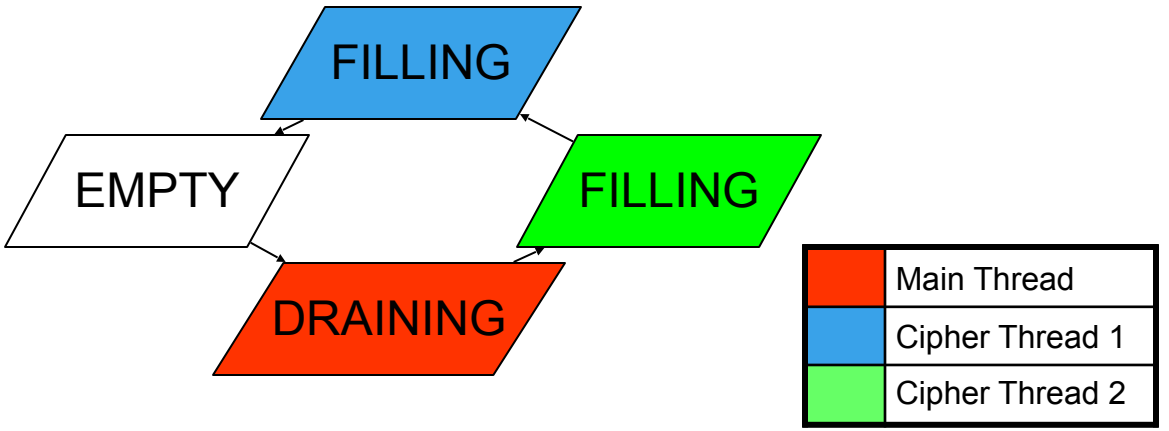
Single Cipher Thread

- Cipher Thread
  - AES_Encrypt(ctr)
  - Inc(ctr)
- Main Thread
  - read(disk)
  - Packetize
  - Compute MAC
  - XOR
  - write(net)

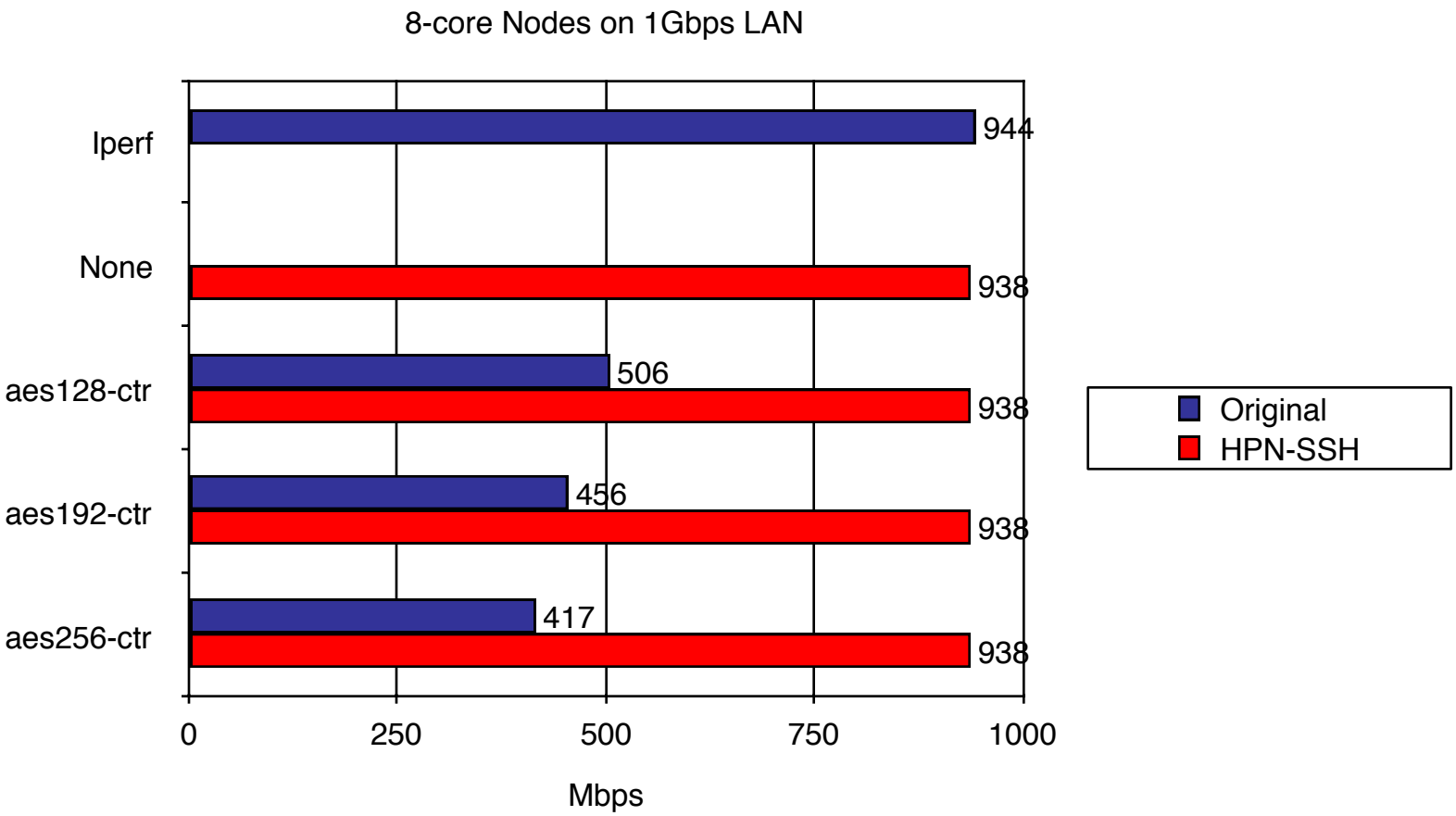


Multiple Cipher Threads

- Ring of bounded queues
  - Each queue holds a portion of keystream
  - Each queue exclusively accessed
- Queue counters offset initially and each fill



M-T AES-CTR Results



Conclusion

- SSH designed for security
  - HPN-SSH is performance enhancements to the most common SSH implementation, OpenSSH
- High throughput with high latency
  - Kernel auto-tuning adjusts TCP flow contol
  - HPN-SSH RecvBufferPolling adjusts SSH flow control
- High throughput with any latency
  - HPN-SSH None cipher for non-private data
  - HPN-SSH Multi-threaded AES-CTR cipher

Future Work

- Approaching 10Gbps
- Continued multi-threading
  - Concurrent packet processing/pipelining
- Efficiency
- Striped data transfers
- Exotic architectures

Where to get it

http://www.psc.edu/networking/projects/hpn-ssh

Email: hpnssh@psc.edu