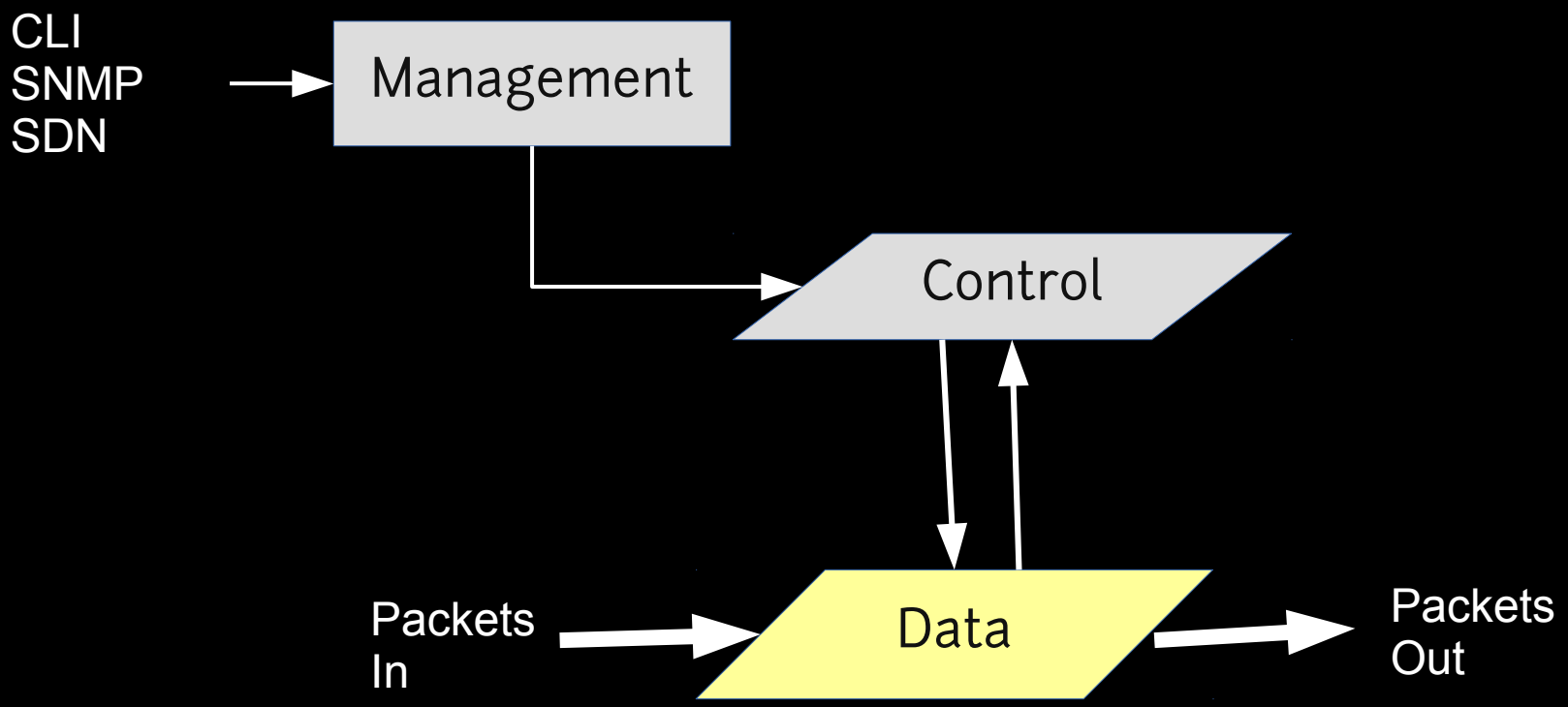
Making a Virtual Router a reality with DPDK, RCU and ØMQ

Stephen Hemminger
shemming@brocade.com

Network Equipment Architecture



Performance Requirements

	Control	Forwarding
Packet Size	1024	64
Packets/second	1.2 Million	14.88 Million
Arrival rate	835 ns	67.2 ns
Clock cycles @ 3Ghz	2504	201

Performance Tax

Operation	Cost
L2	4.3ns
L3	7.9ns
Cache Miss	32ns
LOCK instruction	16.5ns
System Call	87.7ns

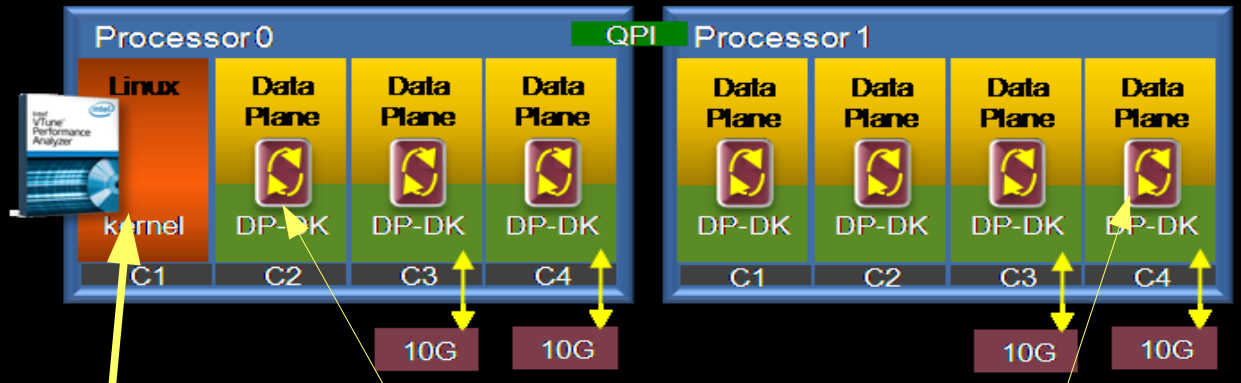
Can only afford one cache miss per packet!

What is DPDK?

DataPlane Development Kit

- Memory Manager – huge TLB
- Buffer Manager – mbuf's
- Queue Manager – lockfree ring
- Poll Mode Driver
- Longest Prefix Match
- Hash tables

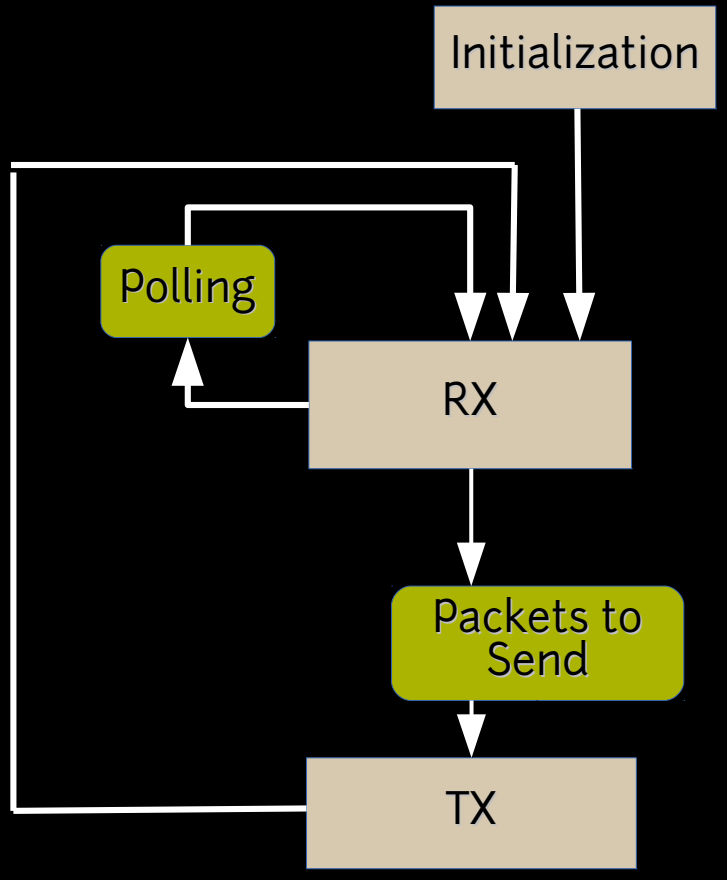
DPDK Architecture



Control Plane

Data Plane

DPDK Application



- Initialization
- Packet Reception
  - Poll Rx queues and receive bursts
- Packet Transmission
  - Transmit the received packets

Data sharing

- Most data updated by only one CPU
- Locking kills performance

Answer: use userspace RCU

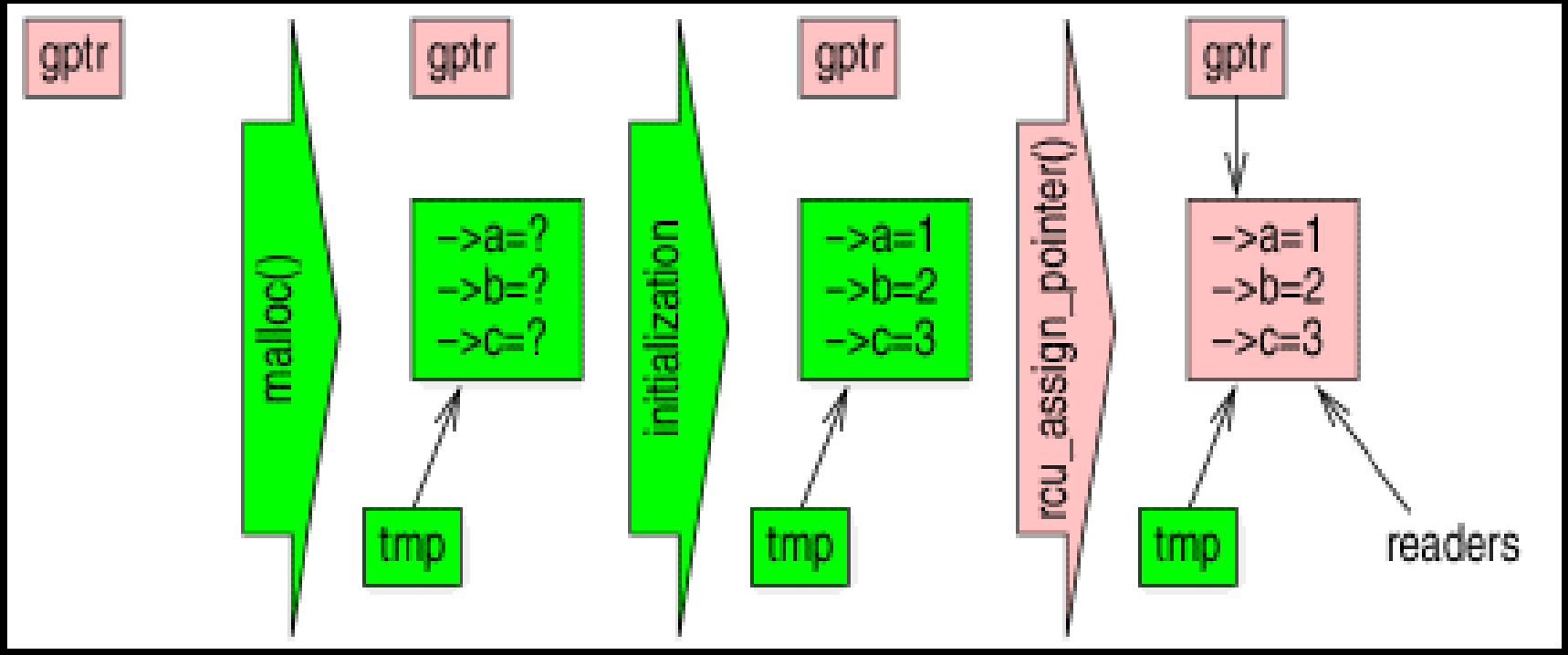
What is RCU?

- Read-copy-update
- An alternative of rwlock
- Allow low over-head wait-free read
- Update can be expensive: need to maintain old copies if in use

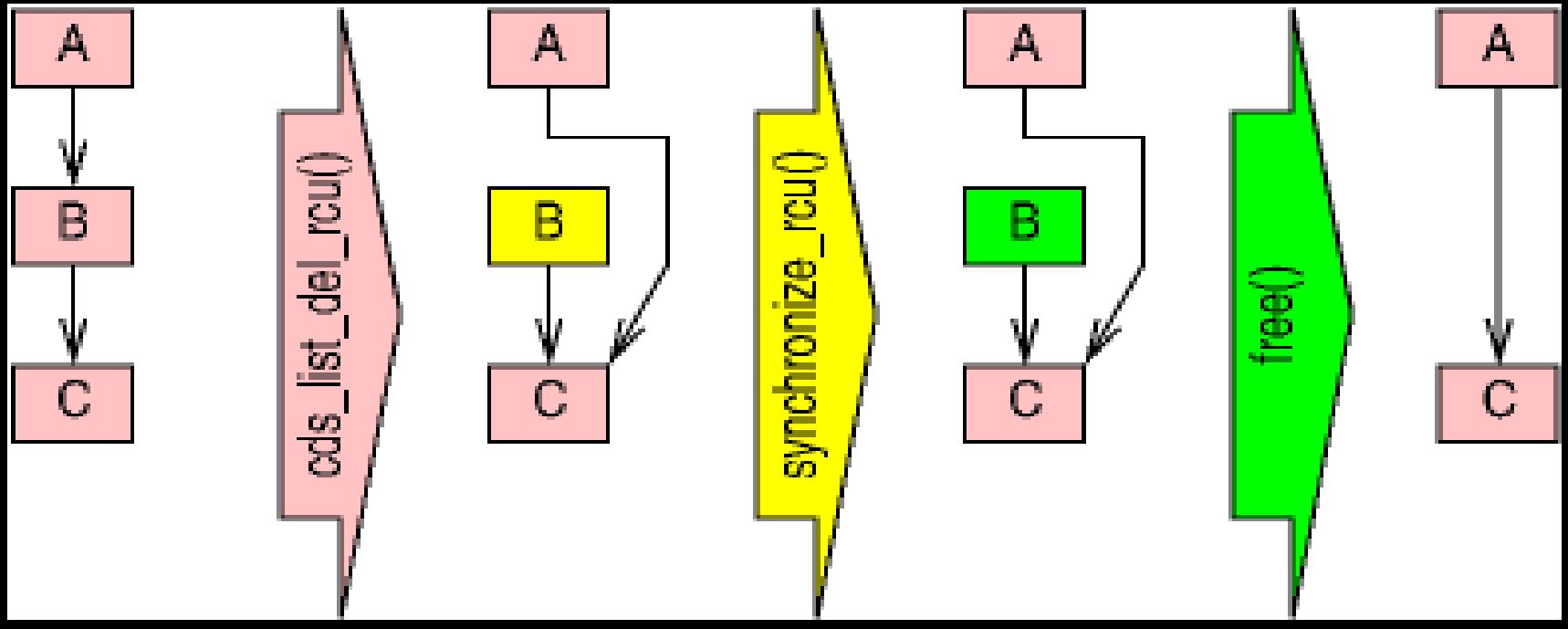
RCU Basis

- Split update into removal and reclamation phases
- Removal is performed immediately, while reclamation is deferred until all readers active during the removal phase have completed
- Takes advantage of the fact that writes to single aligned pointers are atomic on modern CPUs

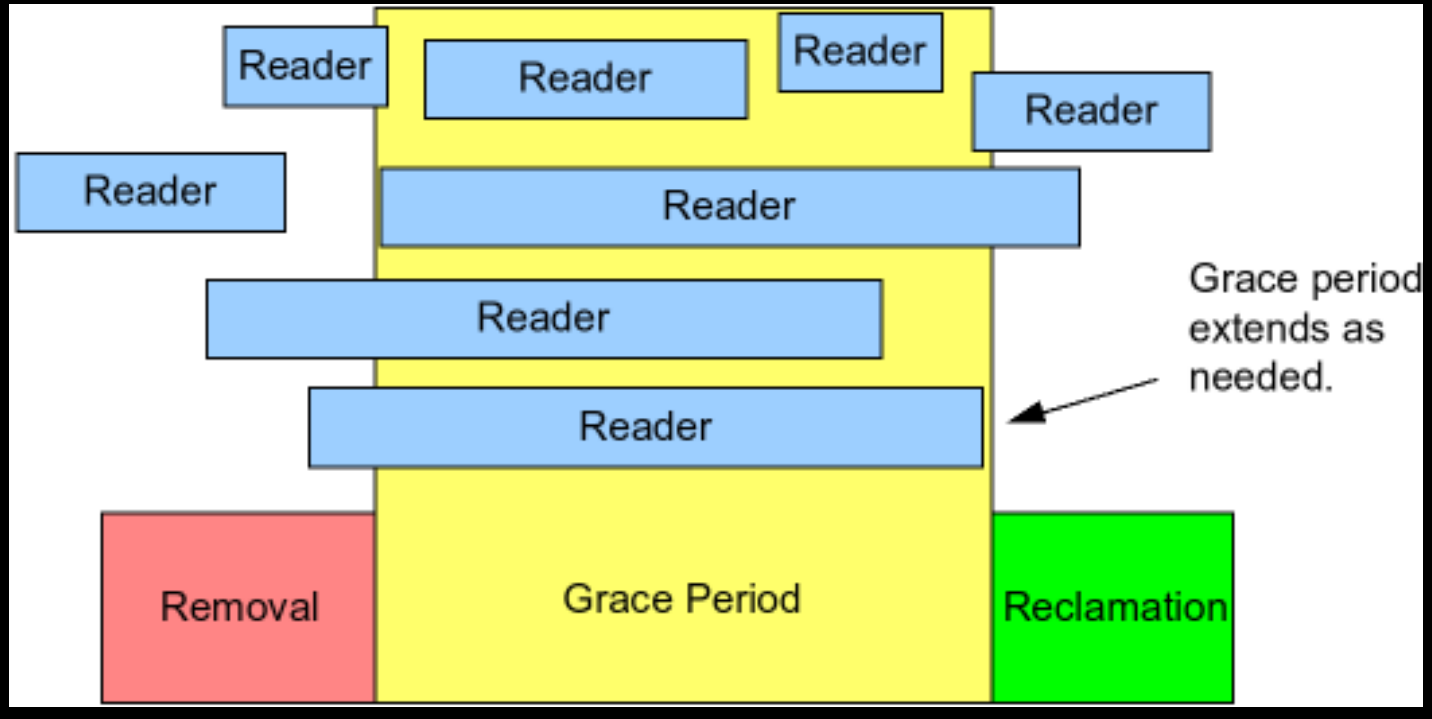
RCU Insertion



RCU Deletion



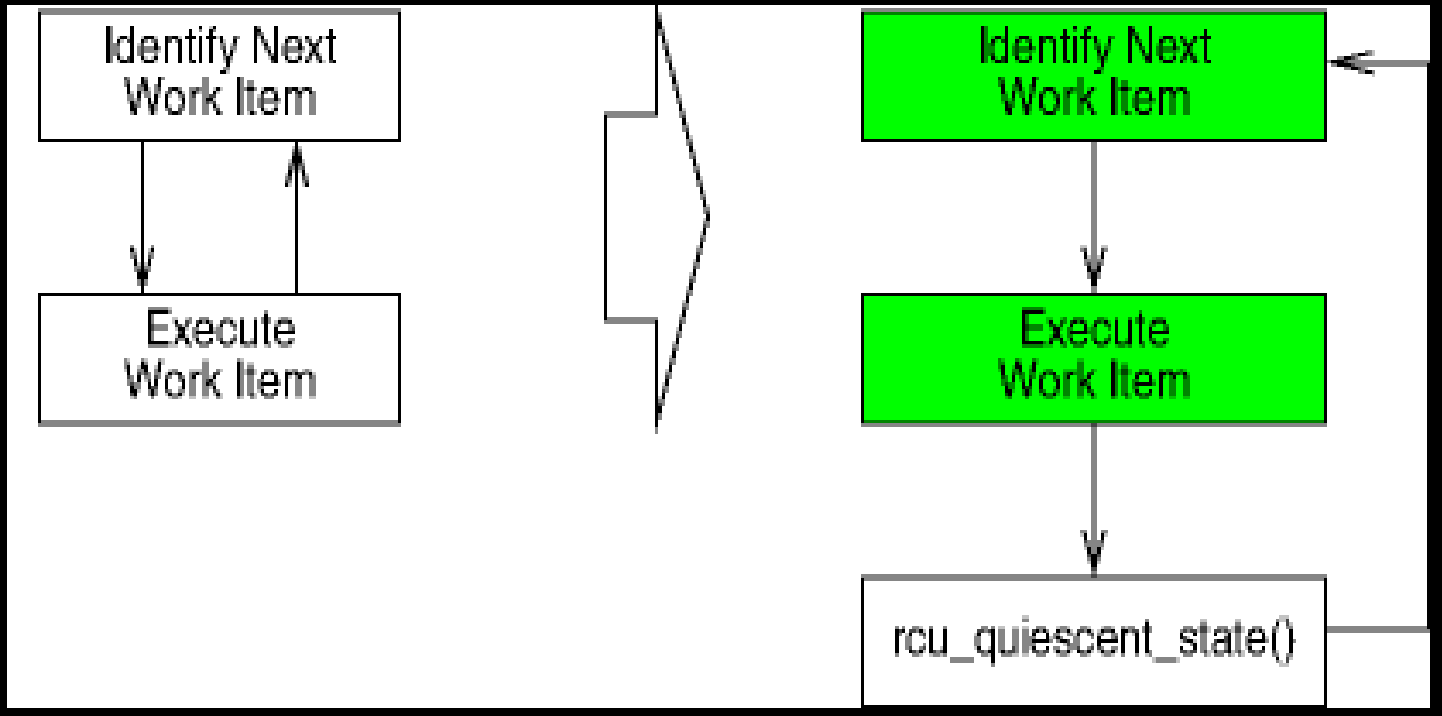
RCU Grace Period



RCU Lock Free Hash

- Lock free
  - Lookup
  - Insertion
  - Delete
- Cache friendly
- Auto resizing

Work Loop


Identify Next Work Item
Execute Work Item
Identify Next Work Item
Execute Work Item
rcu_quiescent_state()

What is ØMQ?

- Intelligent socket library for messaging
- Many kinds of connection patterns
- Multiplatform, multi-language (30+)
- Fast (8M msg/sec, 30usec latency)
- Small (20K lines of C++ code)
- Open source LGPL (large community)

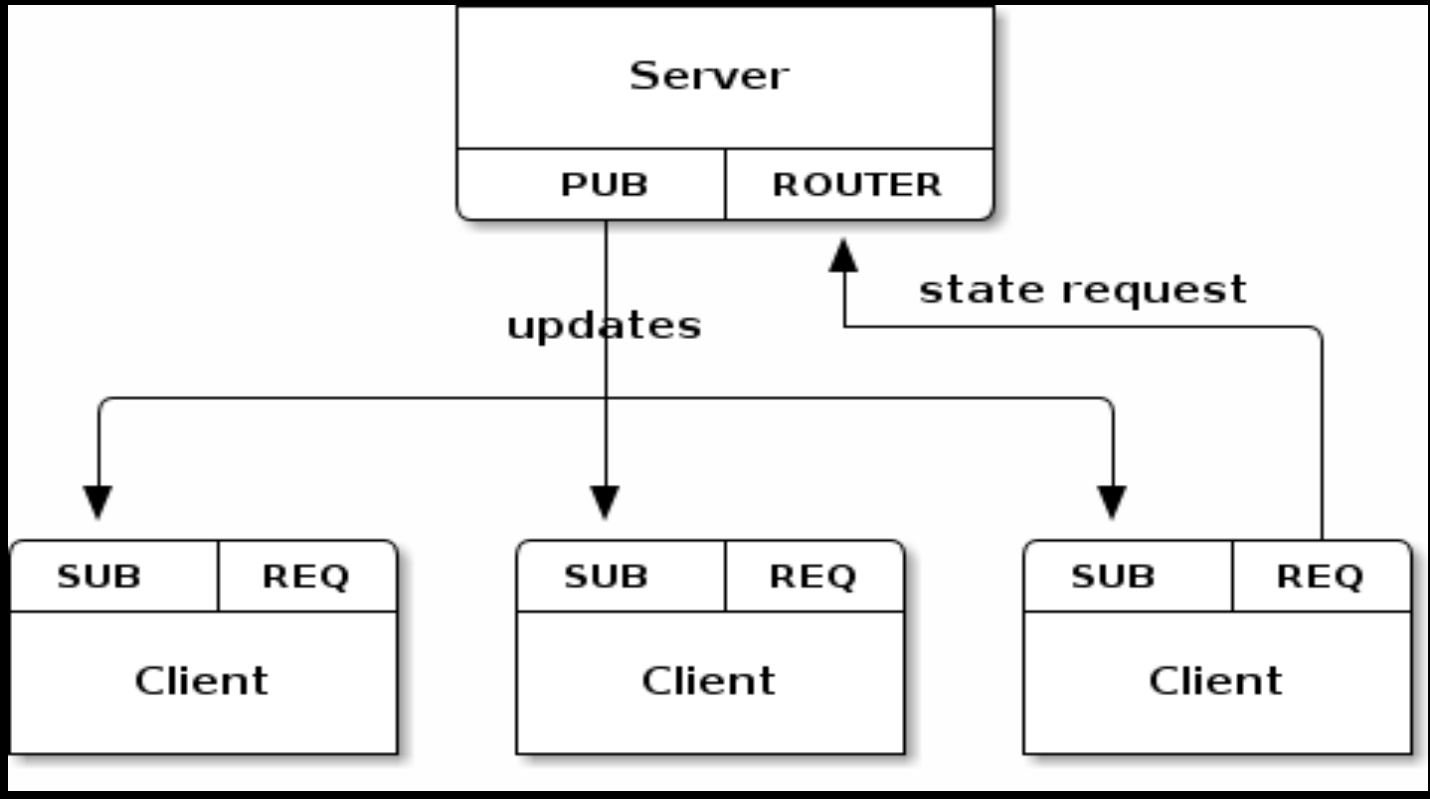
Why use ØMQ?

- Language neutral
- Active friendly community
- Great documentation
- Design Patterns

ØMQ Transports

- Threads in one process (inproc://)
- Processes on one box (ipc://)
- Processes on one network (tcp://)
- Multicast group (pgm://)

Typical ØMQ Design



ØMQ Routing

- Round-robin (REQ, PUSH, DEALER)
- Multicast (PUB)
- Fair-queuing (REP, SUB, PULL, DEALER)
- Explicit addressing (ROUTER)
- Unicast (PAIR)

ØMQ Guide

- Design Patterns
  - Client/server
  - Reliable client
  - Reliable server
  - State resynchronization
  - Broker
  - Distribution

CZMQ Frontend

- Version independent
- Object-like API
- Higher level services
  - Discovery
  - Certs
  - Containers
  - Message management
  - Event model

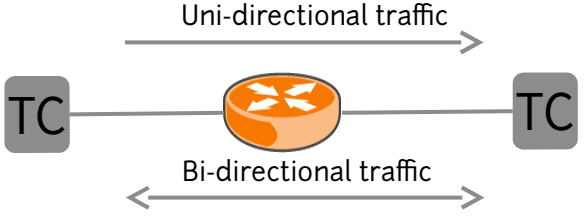
Brocade 5600 virtual router

- Control Plane
  - Linux based
  - Routing protocols
- Dataplane
  - DPDK application
  - Bare metal or guest

Test Environment

Test bed

- Server with two 10GE NICs
- 2 Spirent TestCenter ports connected to port 1 of two NICs
- IP packets; 1M streams



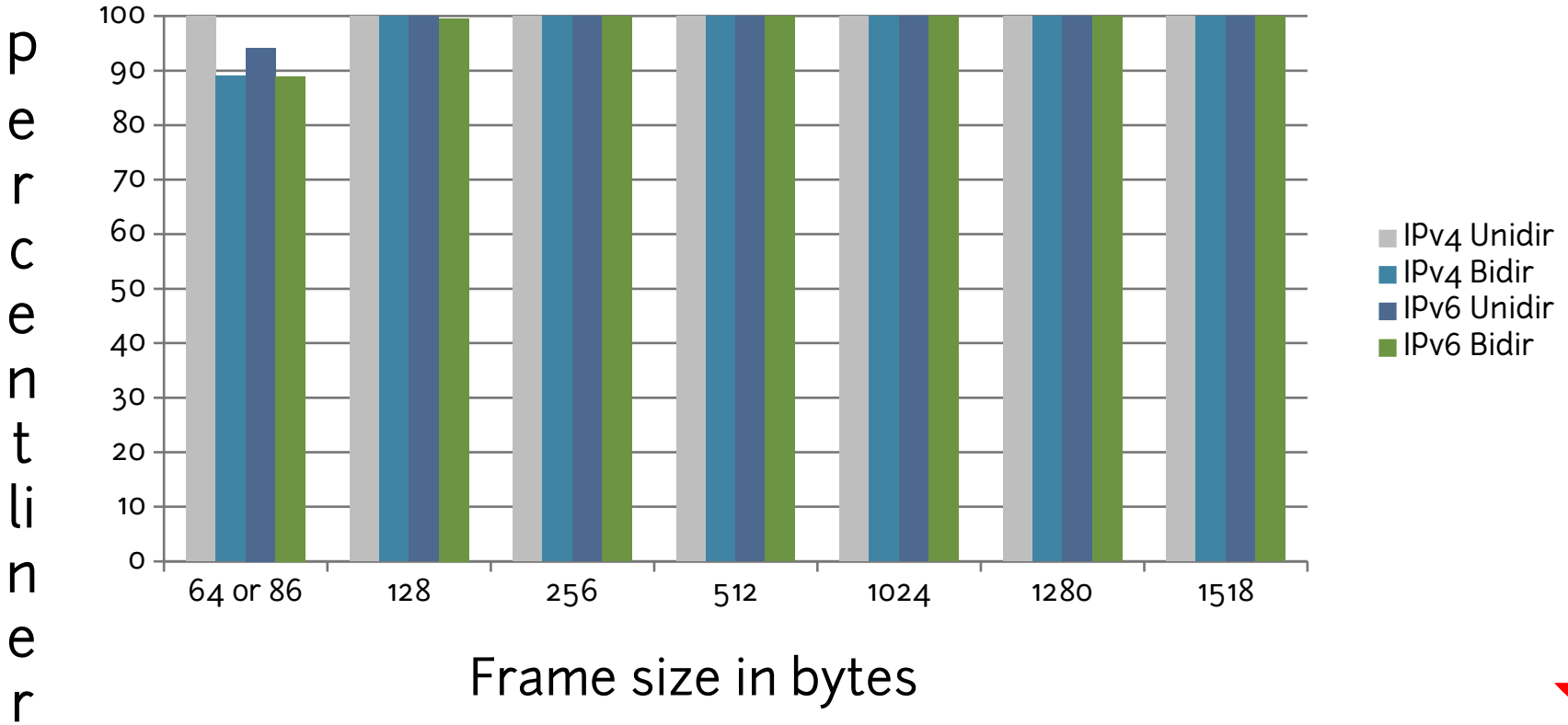
Server Spec

Dell PowerEdge R720 Server with

- 2 X Intel E5-2667v2 3.30GHz
  - 16 cores/32 threads
  - 3.3GHz clock speed
- 64 GB RAM
- Two Intel X540 NICs with 10G Base-T

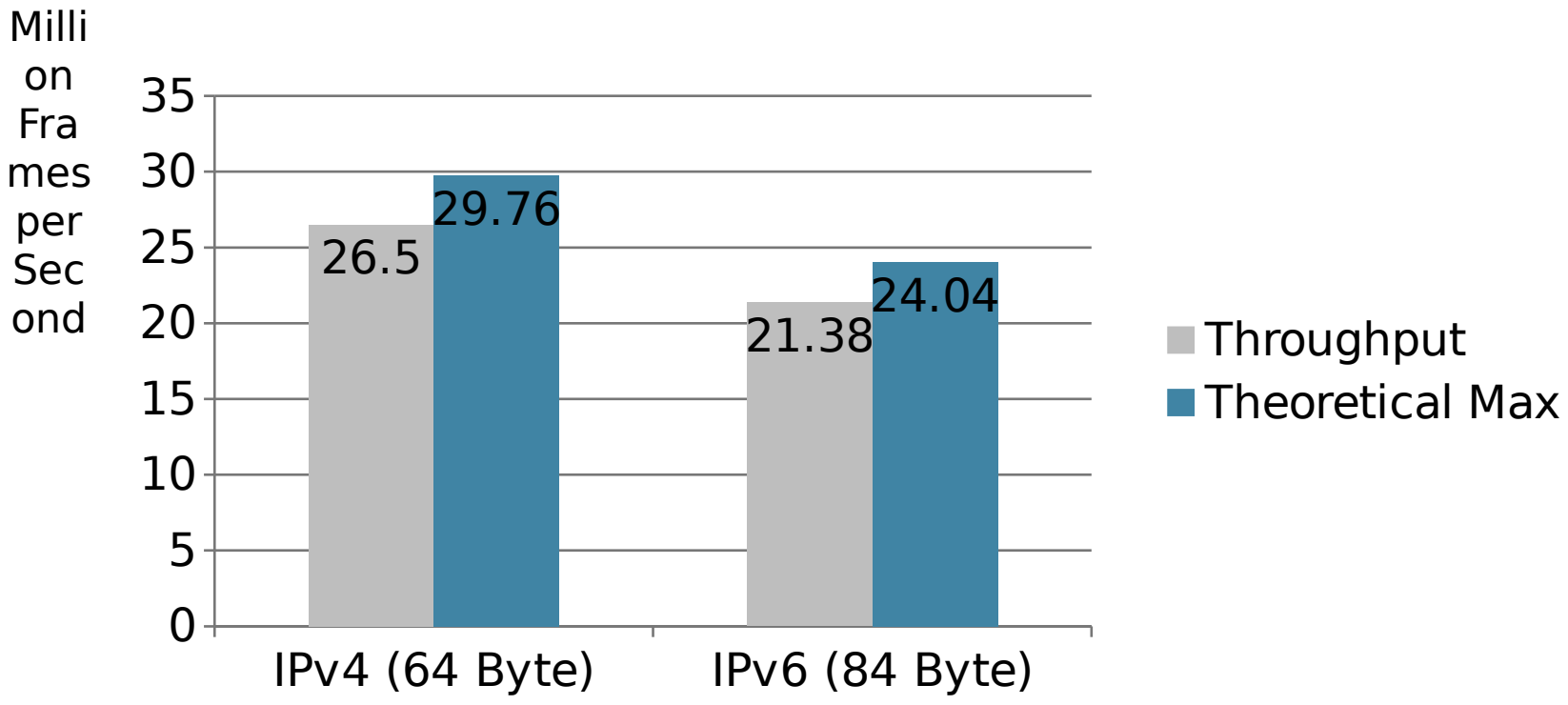
Throughput Results of Brocade 5600 on KVM

Dual-socket Intel E5-2667v2 server, RFC2544 Test, 3.2R1



Aggregate Throughput in Frame Rates

Dual-socket Intel E5-2667v2 server, RFC2544 Test, 3.2R1



Resources

	License	Release	Website
DPDK	BSD	0.6.0	http://dpdk.org
RCU	LGPL	0.8.3	
ØMQ	LGPL	4.0.3	http://www.zeromq.org

Thank you

Stephen Hemminger
@networkplumber

shemming@brocade.com