



# Packet Forwarding

Future Internet Communications Technologies

Prof. Dr. Panagiotis Papadimitriou



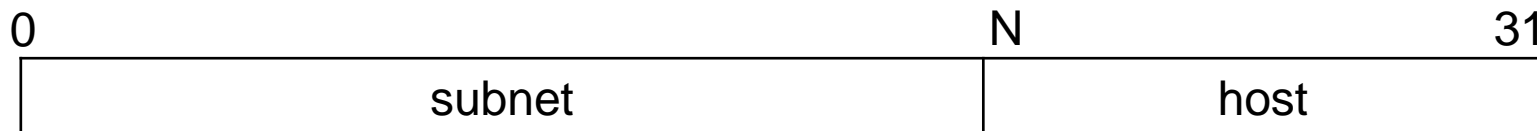
- Packet Forwarding
- IP Lookup
- Router Architectures
- Software Routers on Commodity Servers
- Scaling Software Routers



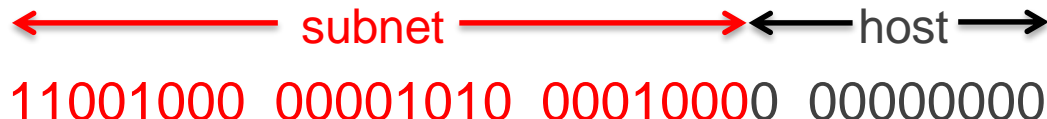
# Packet Forwarding



- CIDR:
  - IP address is subdivided into a subnet portion (of arbitrary length) and a host portion
  - Address format: A.B.C.D/N, where N represents the number of bits for the subnet portion (or prefix)
  - /N is known as the subnet mask



- Example: 200.10.16.0/23 (subnet mask: 255.255.254.0)





- A router uses separate data structures for routing (routing table) and forwarding (forwarding table)
- A routing table is optimized for handling routing updates:

Prefix	Next-hop
124.10.27	<b>192.24.30.32</b>
124.16	<b>192.24.30.78</b>

- A forwarding table is optimized for fast IP lookup and packet forwarding:

Prefix	Link Interface
124.10.27	<b>1</b>
124.16	<b>2</b>

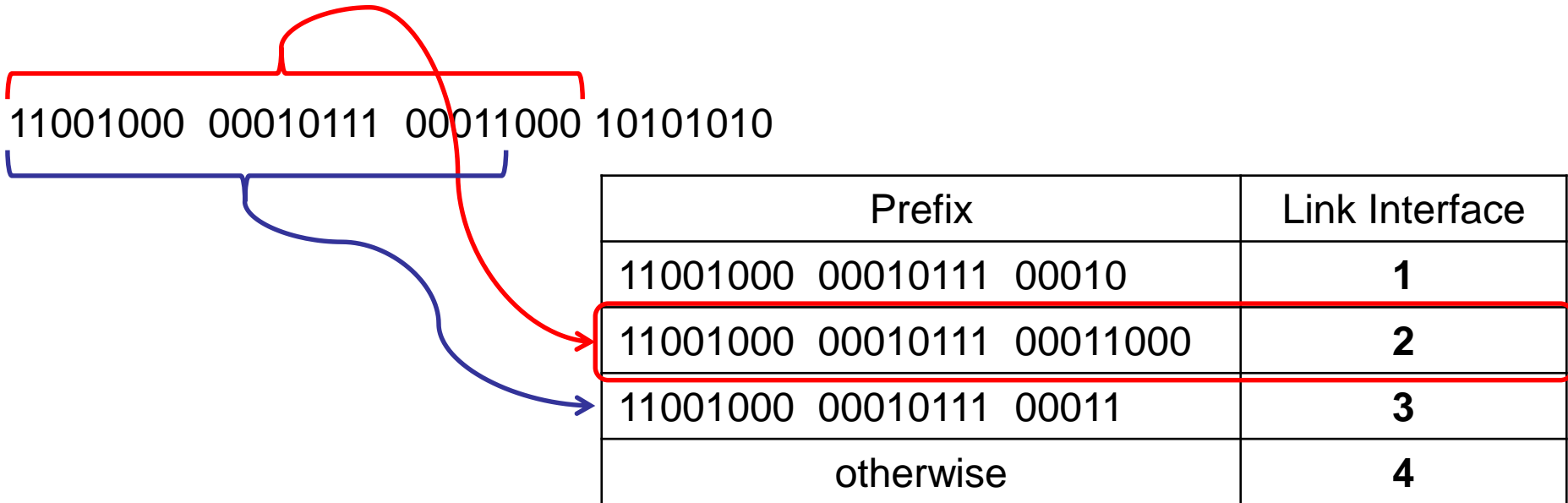


- A forwarding table:
  - indicates the router link interface that each packet is being forwarded
  - matches networks (**NOT** hosts) to router link interfaces
  - comprises a list of {**Prefix**, **Interface**} pairs

Prefix	Link Interface
11001000 00010111 00010	<b>1</b>
11001000 00010111 00011000	<b>2</b>
11001000 00010111 00011	<b>3</b>
otherwise	<b>4</b>



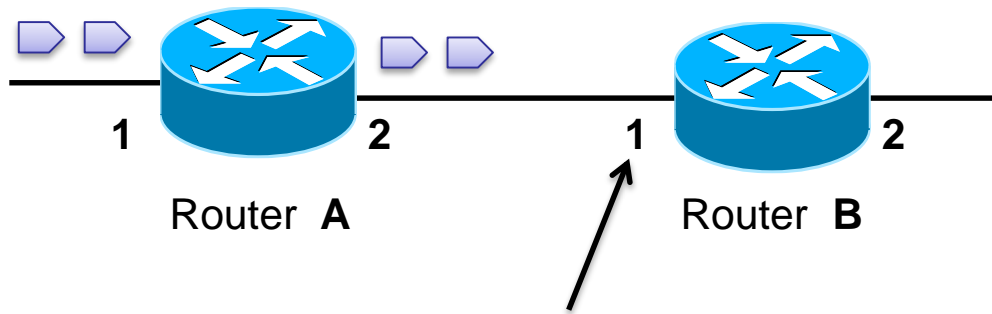
- A packet destination IP address may match more than one prefixes in a router forwarding table:



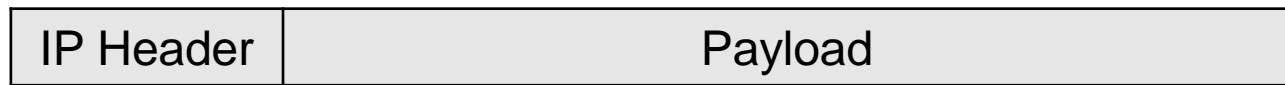
- In the case of multiple matches, the router uses the longest prefix match



- Forwarding IP packets requires:
  - the IP address of the next-hop (provided by the router)
  - the physical address of the next-hop



Next-hop interface (IP, **MAC???**)



IP packet encapsulation







- ARP is responsible for resolving physical addresses:
  - ARP allows routers/hosts in the network to build up a table of mappings between IP and physical addresses
  - each entry in the ARP table is removed after a period (TTL)

IP Address	Physical Address	TTL
172.23.180.137	84-2B-2B-A5-A5-77	13:45:00
172.23.42.42	70-F3-95-3D-14-04	11:12:00

- When a host/router wants to identify a physical address:
  - it performs a lookup in the ARP table and obtains the physical address
  - if there is no match:
    - it broadcasts an ARP query onto the network
    - the host/router that has this IP address sends a message with its physical address (which is subsequently added in the ARP table)

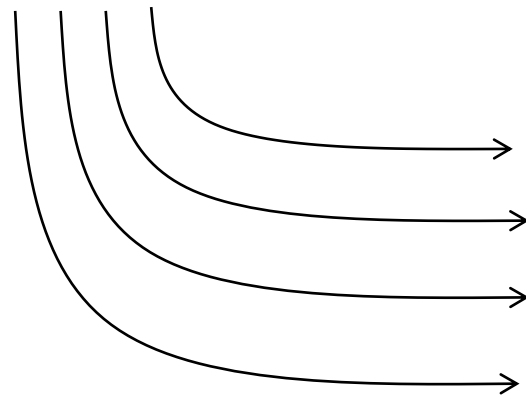


# IP Lookup



- Exact prefix match requires searching:
  - within all prefixes for the given prefix length
- Longest prefix match is complicated, as it requires searching:
  - within all prefix lengths
  - within all prefixes for each given prefix length

11001000 00010111 00011000 10101010



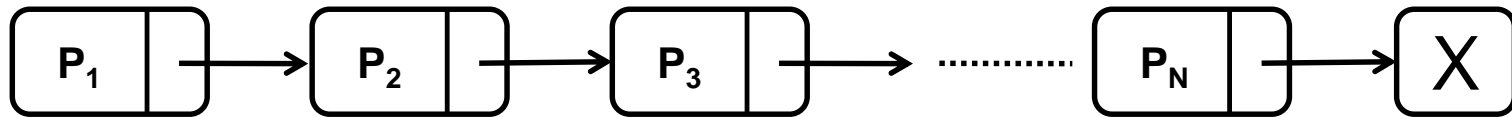
Prefix	Link Interface
11001000 00010111 00010	<b>1</b>
11001000 00010111 00011000	<b>2</b>
11001000 00010111 00011	<b>3</b>
.....	<b>4</b>



- IP lookups can be implemented using:
  - Lookup algorithms, e.g.:
    - Radix trie
    - LC trie
    - Patricia
    - Lulea
    - .....
  - Ternary Content Addressable Memory (TCAM)



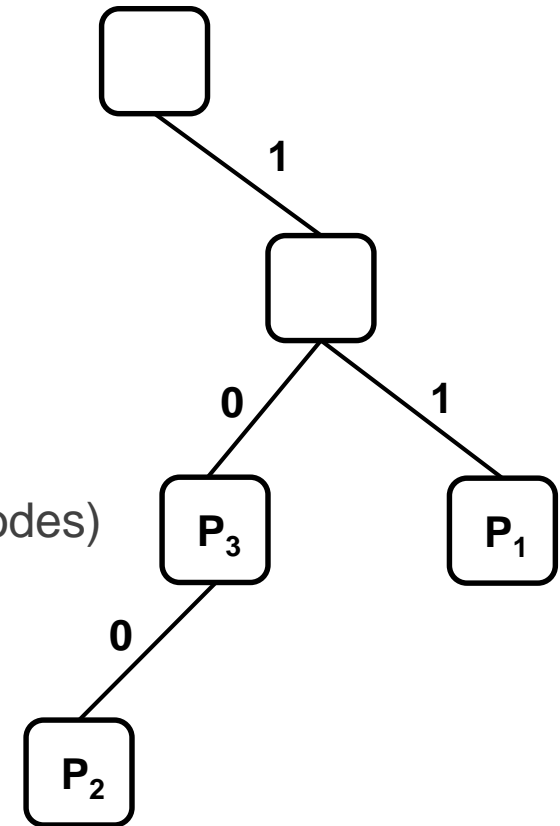
- Linked list for all (N) prefixes:
  - Lookup:  $O(N)$
  - Storage:  $O(N)$
  - Update:  $O(1)$



- Lookup time is very slow:
  - the whole list should be traversed in order to identify the longest prefix match
  - a sorted linked list (by prefix length) can result in shorter lookup time



- A Radix trie is a binary tree whose nodes are labeled with bits (0 or 1):
  - Lookup is performed bit-wise
  - For  $W$ -bit prefixes, maximum depth is  $W$  nodes
  - Lookup:  $O(W)$
  - Storage:  $O(NW)$
  - Update:  $O(W)$
- Limitations:
  - Worst-case lookup is slow
  - Waste of storage space (null pointers, 1-degree nodes)



$N$ : number of prefixes

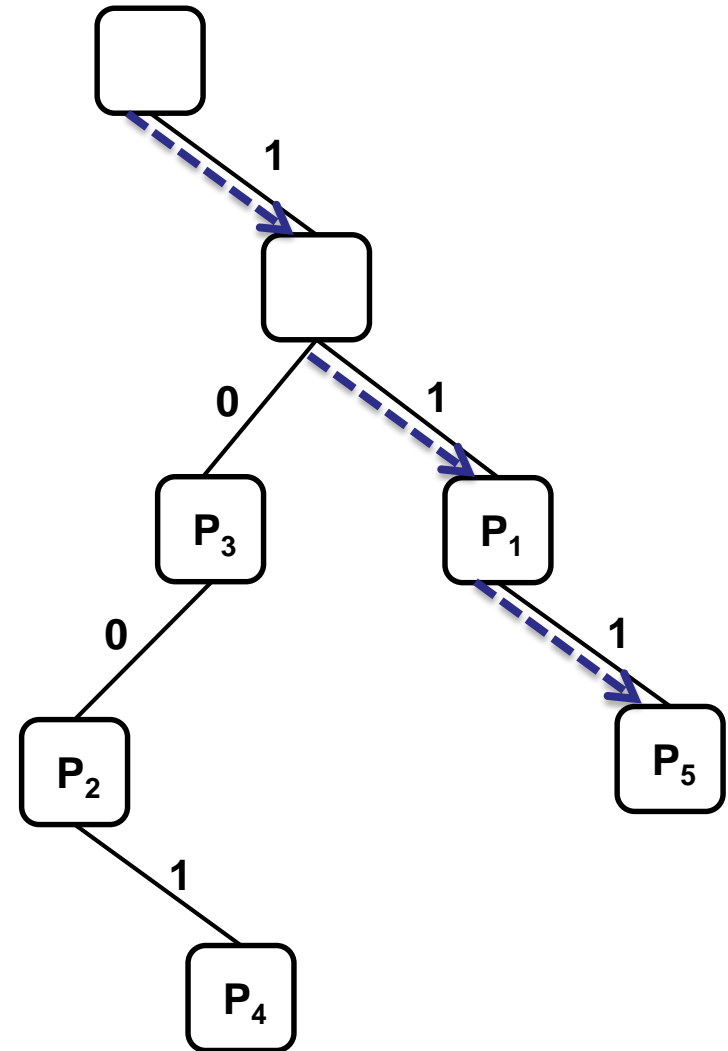
# Radix Trie Search Example



index	prefix	next-hop
P <sub>1</sub>	11*	A
P <sub>2</sub>	100*	D
P <sub>3</sub>	10*	F
P <sub>4</sub>	1001*	B
P <sub>5</sub>	111*	E



11101000 00010111 00011000 10101010





- Storing data structures in RAM and using lookup algorithms result in multiple memory accesses:
  - Linear search:  $N$  memory accesses
  - Radix trie: up to  $W$  memory accesses
- Content-Addressable Memory (CAM):
  - fully-associative memory
  - compares input string with all the entries in parallel
  - provides an exact match operation in a single clock cycle





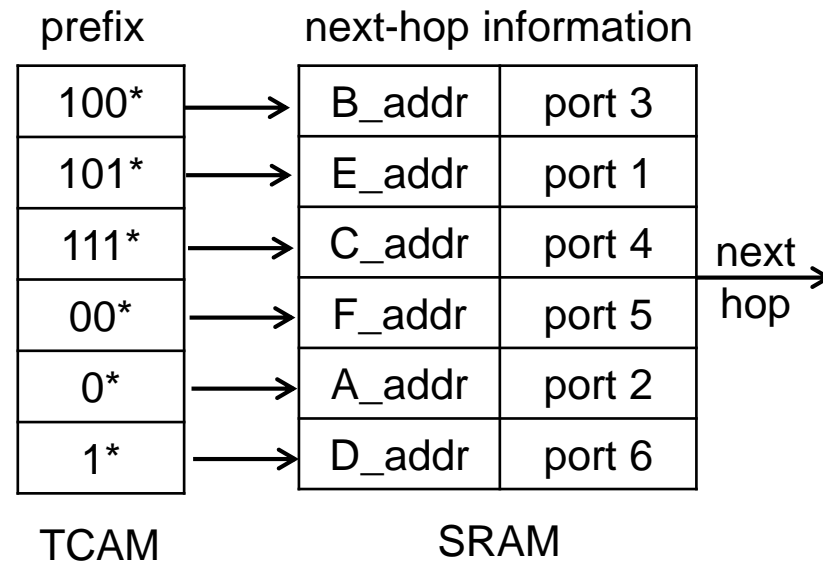
- Longest prefix match is still costly with a CAM:
  - Sequential exact matches for all possible prefix lengths using a single CAM, or
  - Parallel exact matches using 32 separate CAMs (each one containing all N prefixes)
- Solution: Ternary-CAM (TCAM):
  - instead of 0 and 1, each memory cell can take 3 logic states: 0, 1 or X (“don’t care”)
  - provides a longest prefix match in a single clock cycle

TCAM			
11101000	00010XXX	XXXXXXXXXX	XXXXXXXXXX
11101000	00010111	00011000	XXXXXXXXXX
:			
11001001	11000010	0001XXXX	XXXXXXXXXX



prefix	next-hop
0*	A
1*	D
00*	F
100*	B
101*	E
111*	C

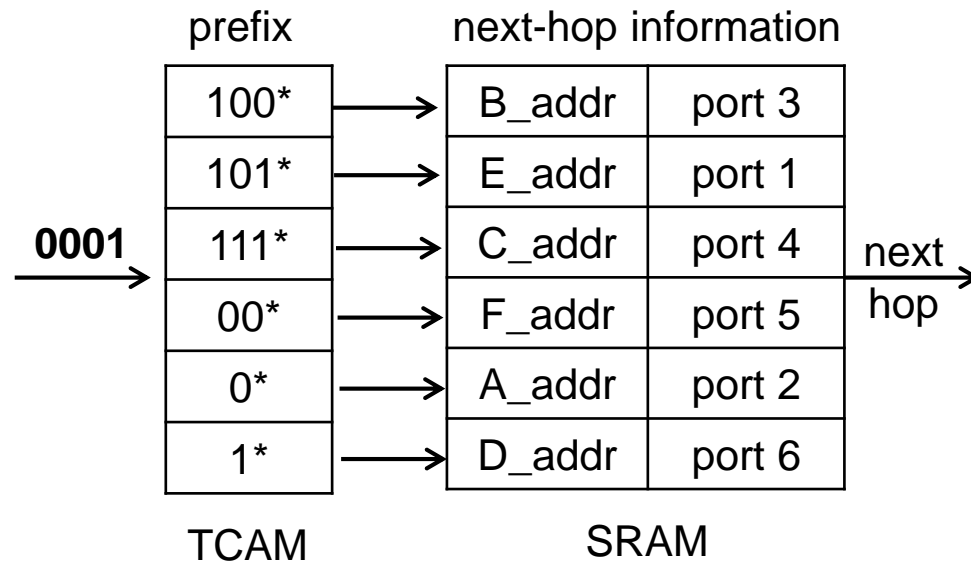
forwarding table





prefix	next-hop
0*	A
1*	D
00*	F
100*	B
101*	E
111*	C

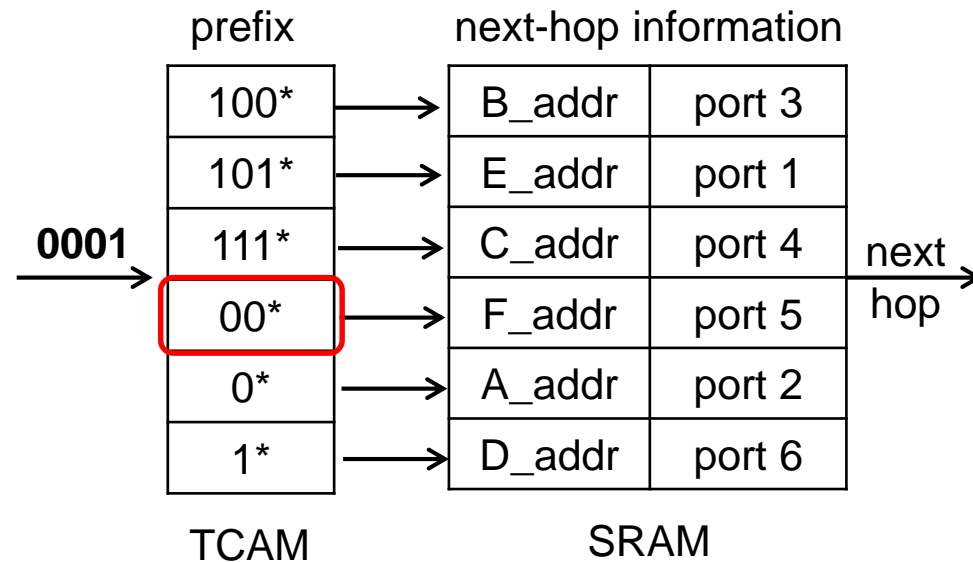
forwarding table





prefix	next-hop
0*	A
1*	D
00*	F
100*	B
101*	E
111*	C

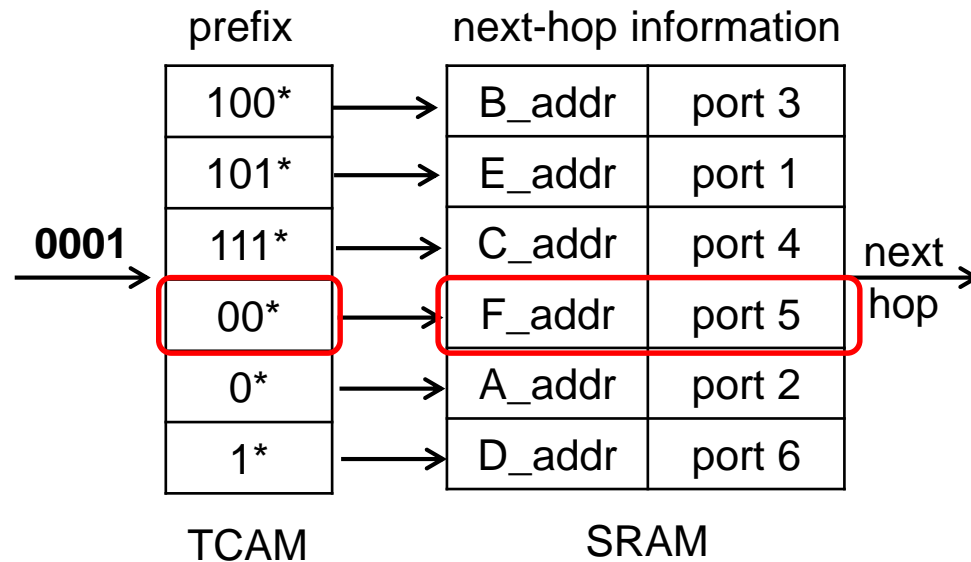
forwarding table





prefix	next-hop
0*	A
1*	D
00*	F
100*	B
101*	E
111*	C

forwarding table

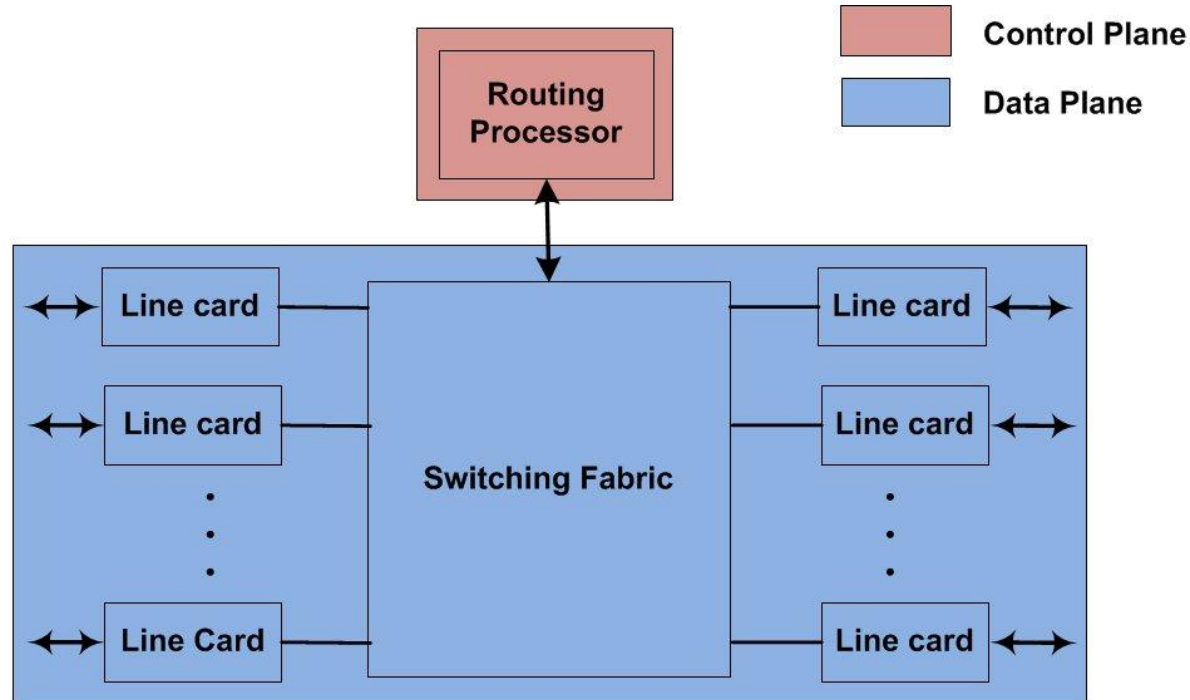




# Router Architectures

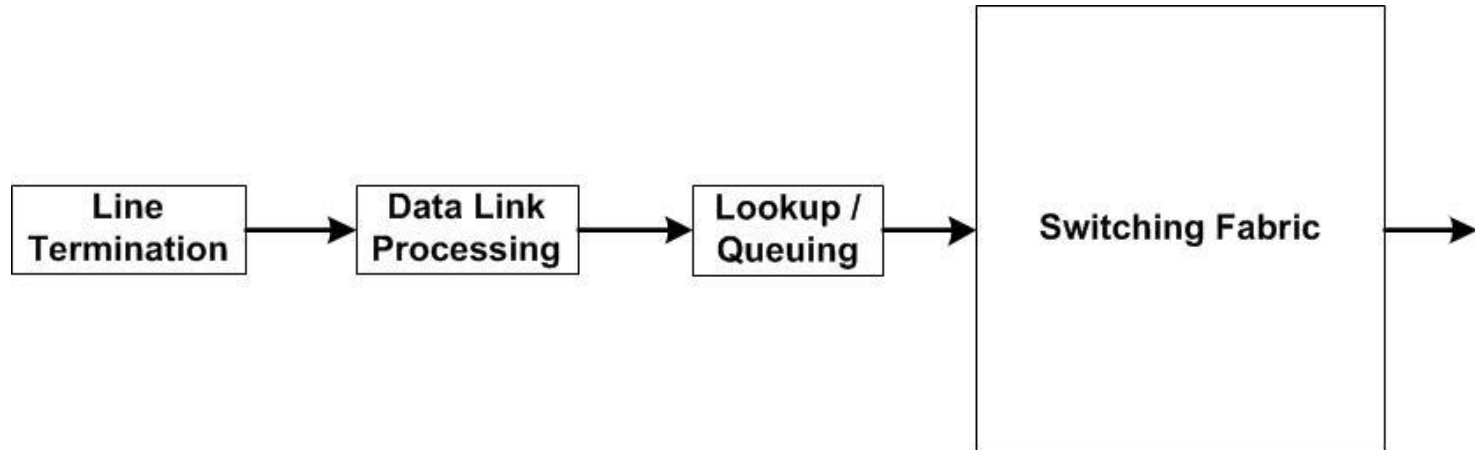


- Data plane:
  - Forwards packets from input to output
- Control plane:
  - Runs routing and management protocols (RIP, OSPF, BGP, etc.)





- Line termination (physical layer)
  - Bits reception
- Data link processing (link layer)
  - Packet decapsulation
- IP lookup, queuing (network layer)
  - Output port lookup based on forwarding table
  - A local copy of the forwarding table is kept at input port memory



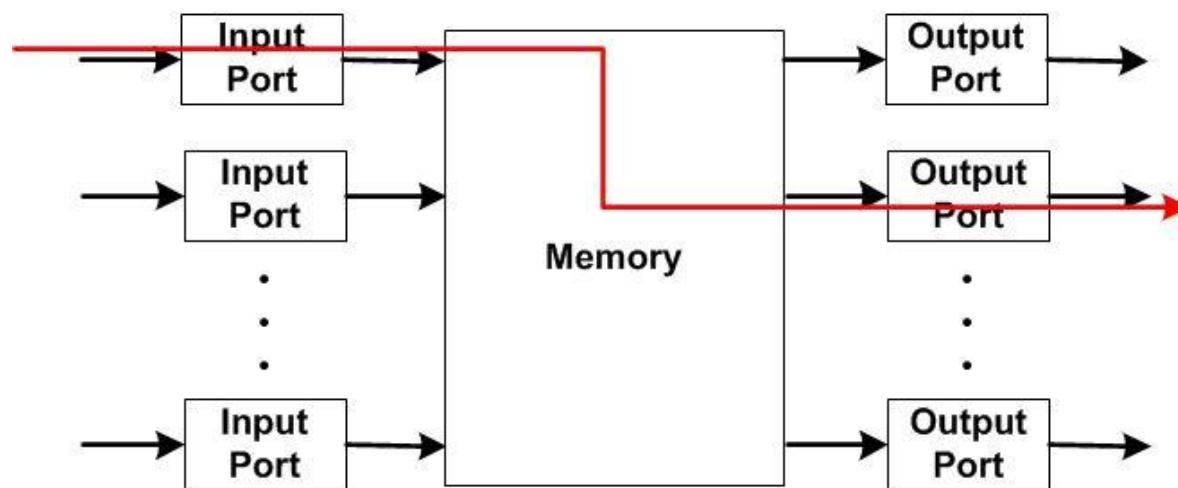




- Switching via Memory
  - Input ports write packets to a memory location
  - Output ports read packets from this memory location
- Switching via Shared Bus
  - Packets are transferred from input to output over a shared bus
- Switching via Interconnection Network (Crossbar)
  - Any input port is connected to all output ports via a matrix of buses
  - Higher bandwidth than a single bus

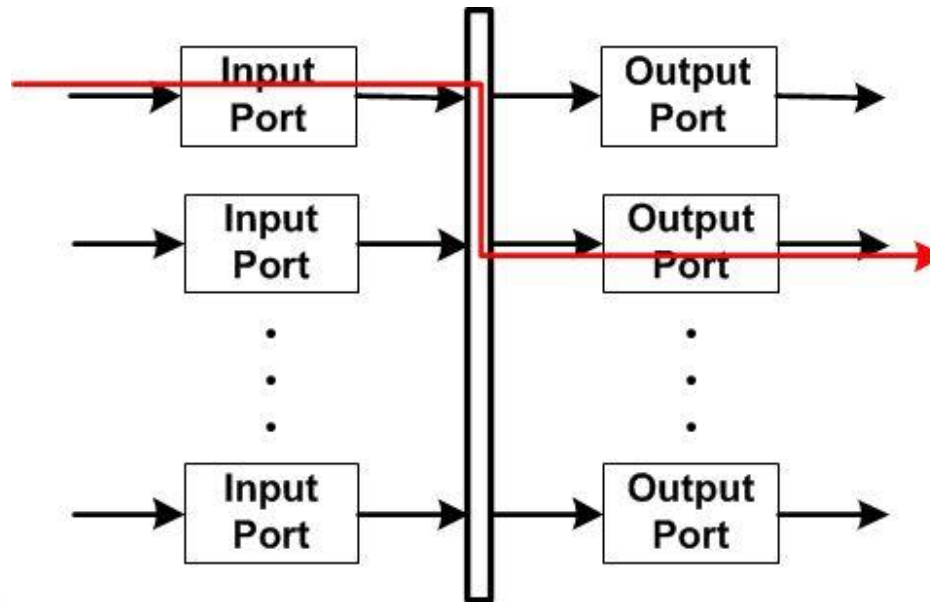


- Memory bandwidth (B) is the bottleneck
  - 2 memory bus crossings per packet transaction
  - Maximum forwarding rate is  $B/2$
- Shared-memory routers:
  - Cisco Catalyst 8500
  - Software routers on PCs



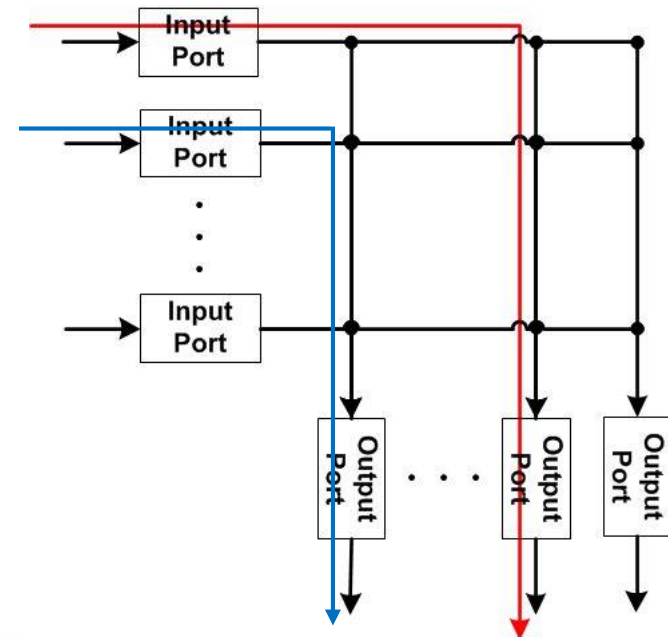


- Shared bus bandwidth (B) is the bottleneck
  - Maximum forwarding rate is B
- Shared-bus routers:
  - Cisco 5600
  - High-speed shared bus (32 Gbps)



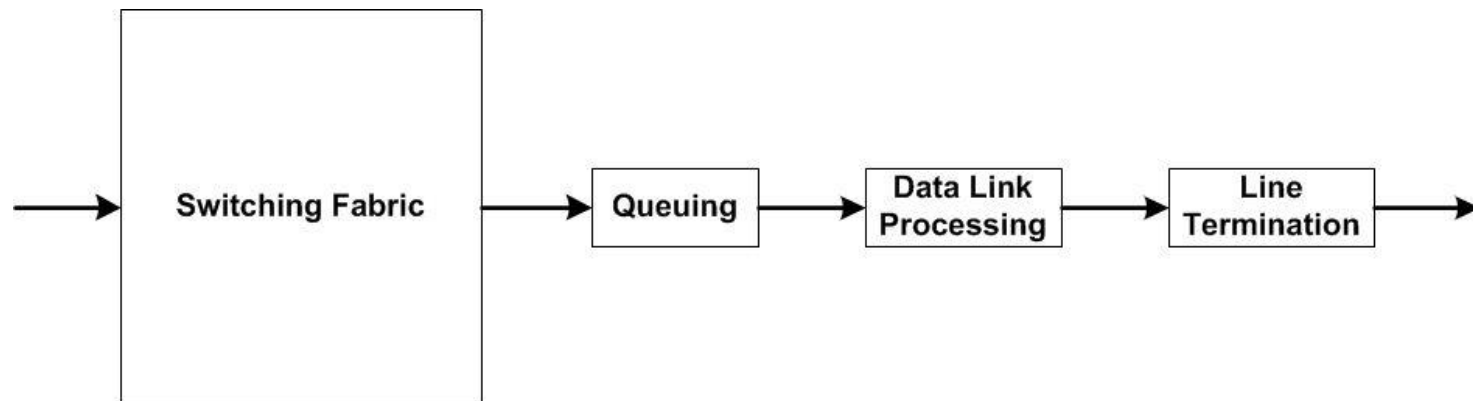


- Matrix of buses to provide more bandwidth than a shared bus
  - $2n$  buses for  $n \times n$  router
- Limitations:
  - Only one packet at a time can be transferred over a bus
  - Packet transfers over vertical buses can be blocked by concurrent packet transfers
- Crossbar routers:
  - Cisco 12000 (60 Gbps)



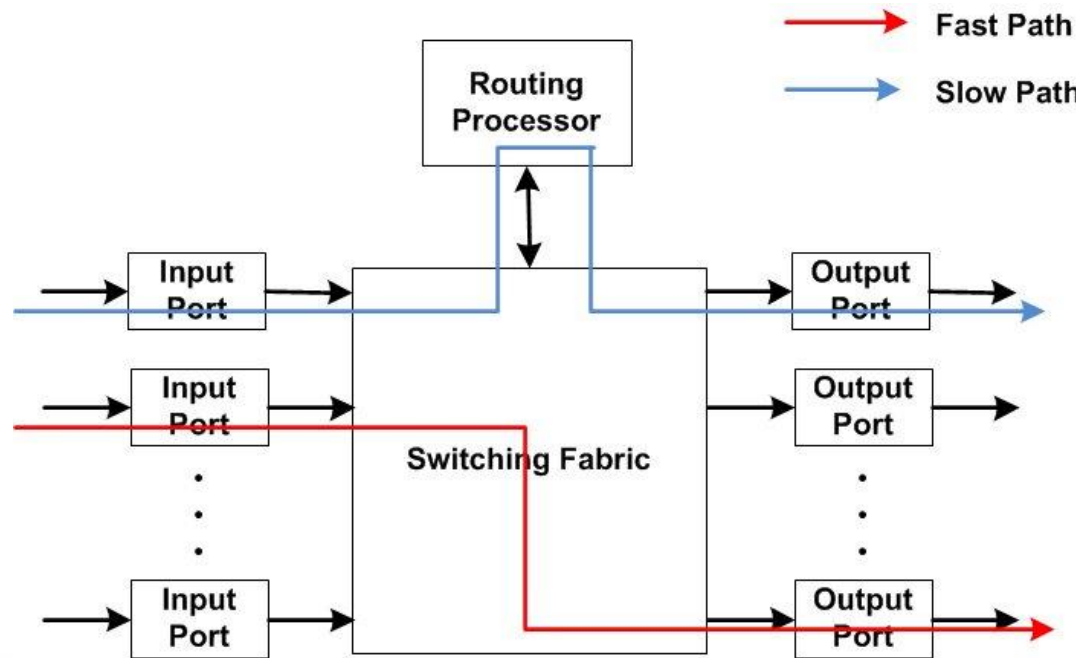


- Queuing (network layer)
  - Required when packet arrival and forwarding rate exceeds transmission rate in the output interface
- Data link processing (link layer)
  - Packet encapsulation
- Line termination (physical layer)
  - Bits transmission





- Fast Path:
  - Direct packet switching from input to output interface (used for typical packet forwarding, such as IPv4)
- Slow Path:
  - Packets are processed and forwarded via the routing processor (used by control packets when access to the full routing table is needed or in the case of packet processing, such as encryption, timestamping)

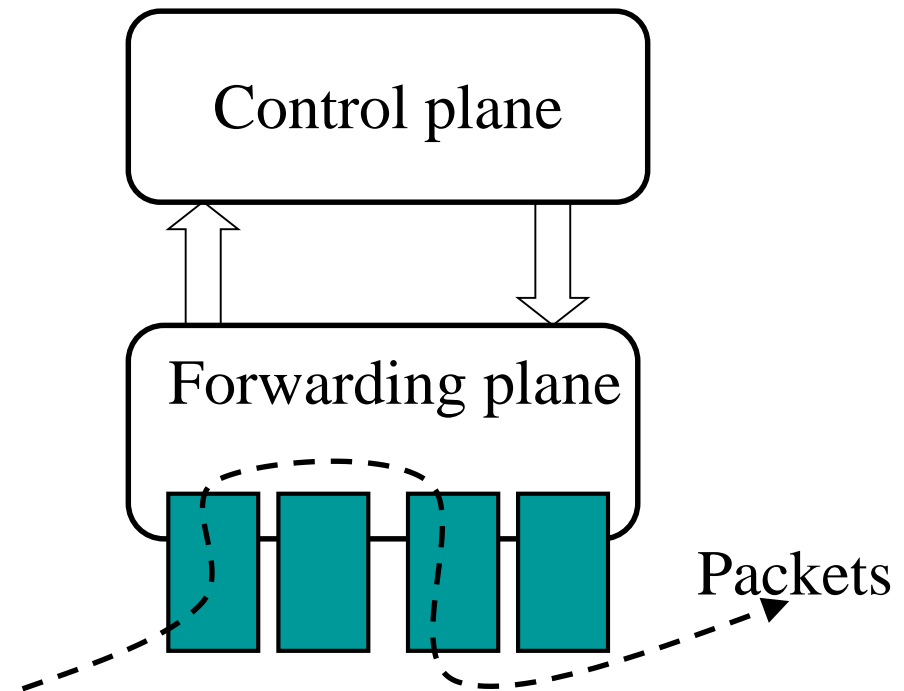




# Software Routers on Commodity Servers



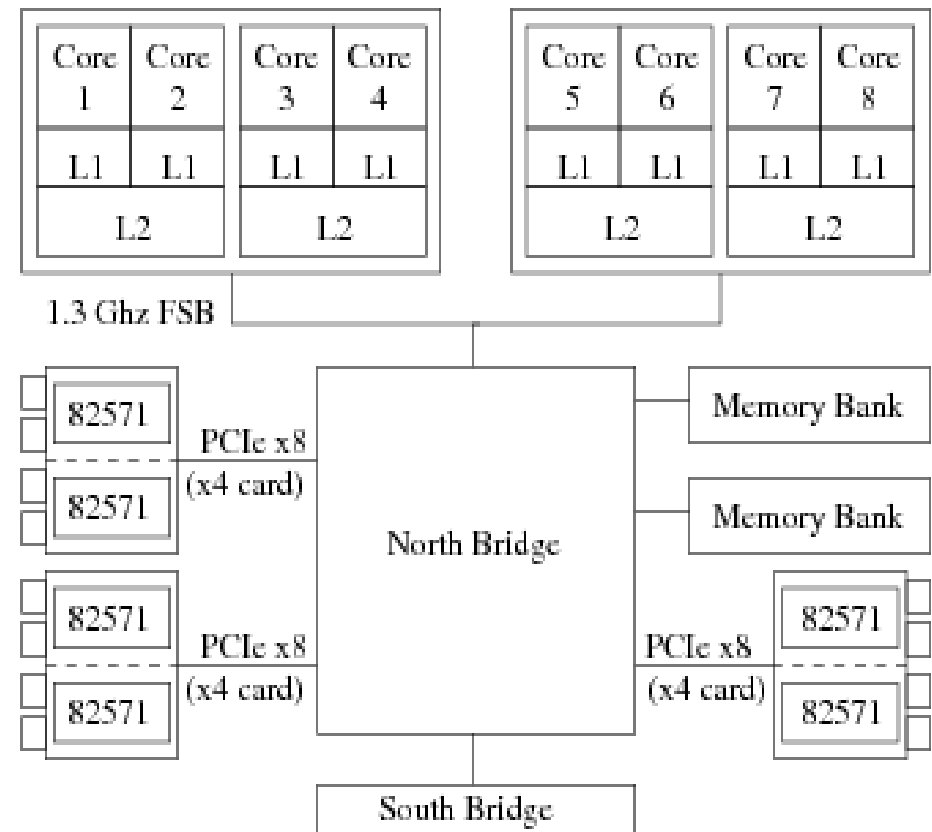
- Programmability:
  - Fully programmable forwarding plane (e.g., Click Modular Router)
  - Extensible control plane (e.g., XORP)
- Performance:
  - Can a software router achieve line rates?

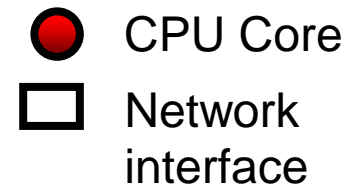
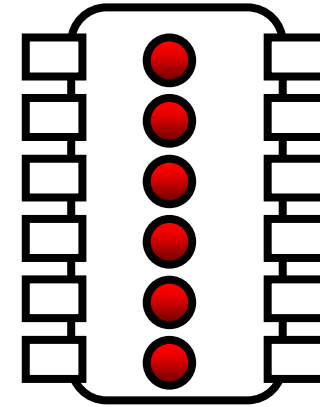
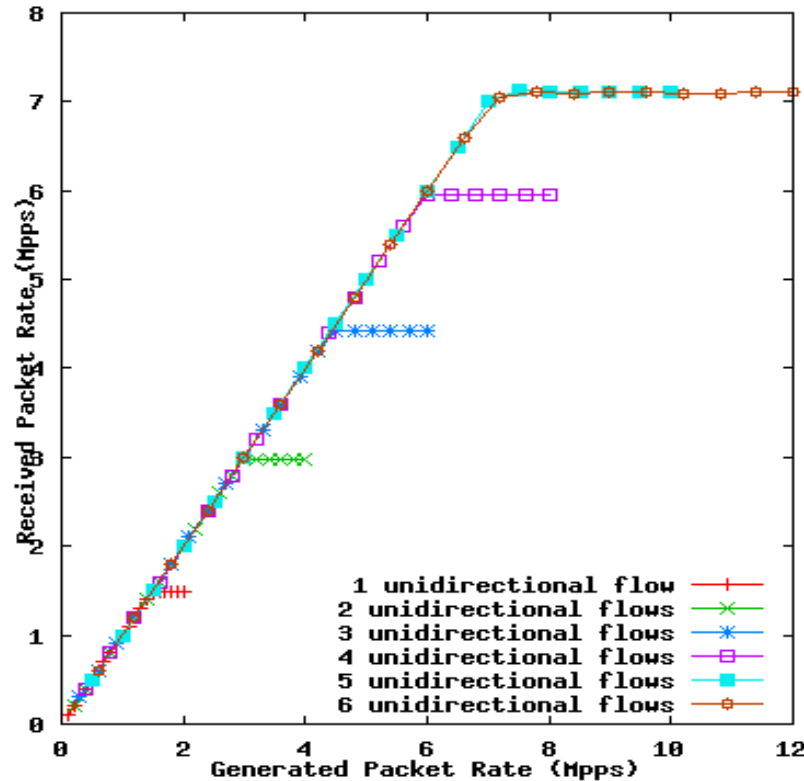






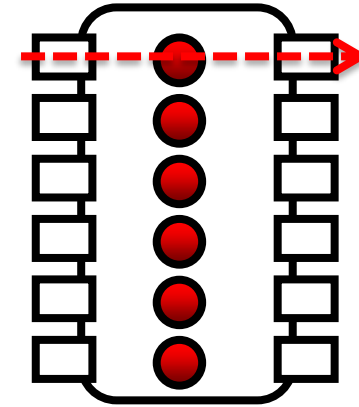
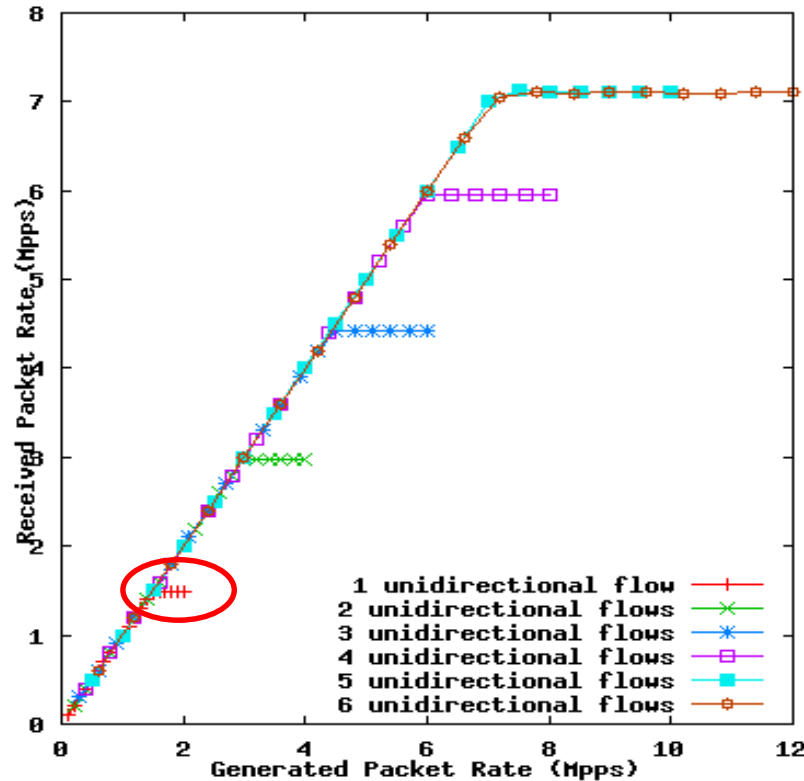
- Dell PowerEdge 2950 (SMP)
  - 2 Intel Xeon X5355 CPUs (quad-core at 2.66GHz)
    - 32K L1d, 32K L1i, 4M L2 shared cache (shared by 2 cores)
  - 8 x1GB system memory
    - PC-5300 (667 MHz)
  - 12 Gigabit Ethernet ports
    - On 3 NICs
    - Each NIC had 4 PCIe lanes
    - Intel 82571EB controllers





1-6 unidirectional flows, 64-byte packets, 1 flow per core

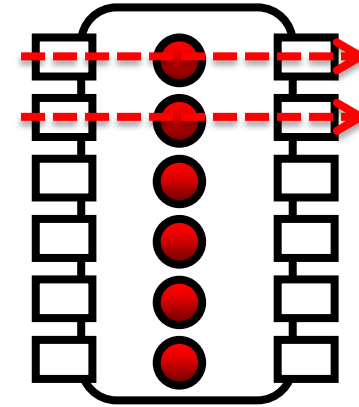
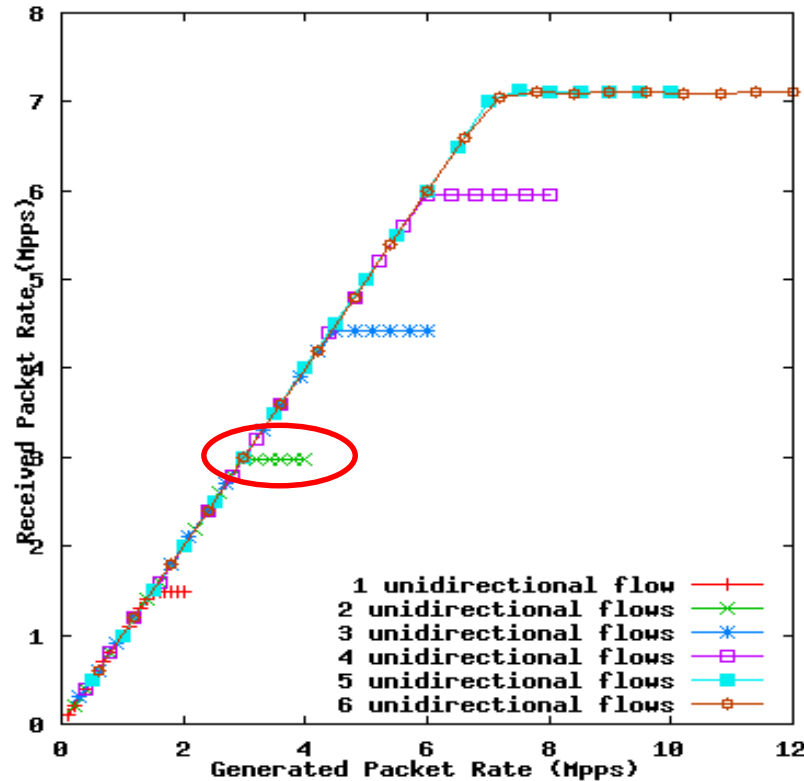
- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets



● CPU Core  
□ Network interface

1-6 unidirectional flows, 64-byte packets, 1 flow per core

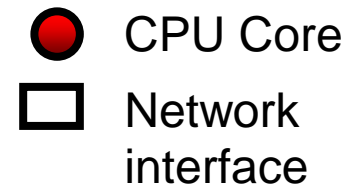
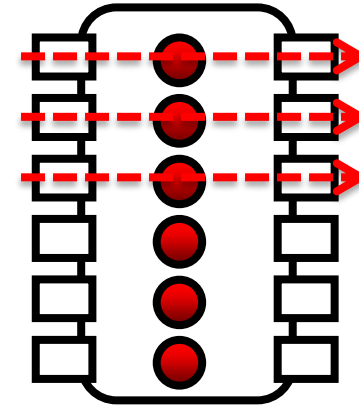
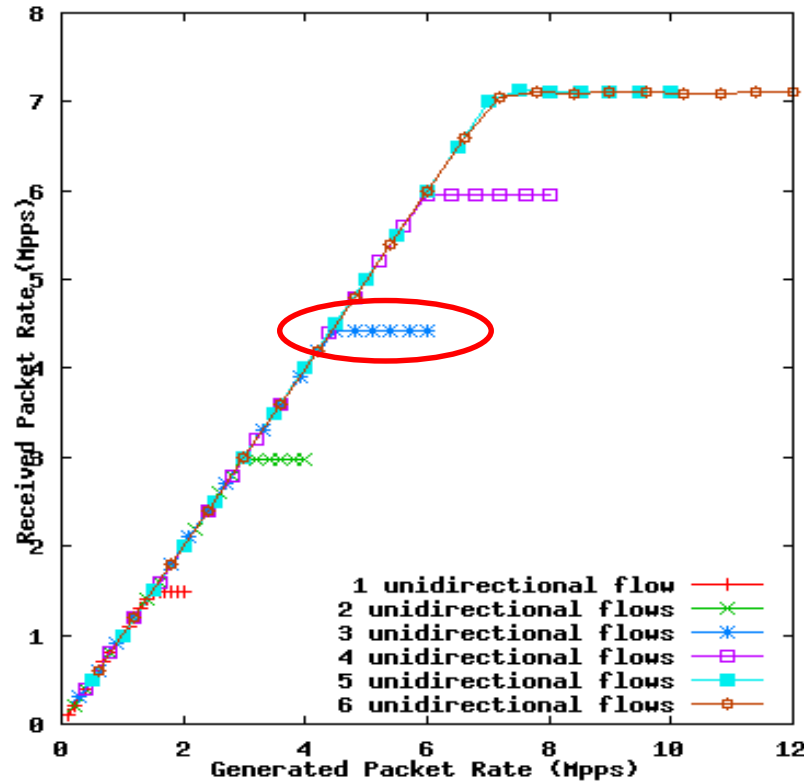
- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets



● CPU Core  
□ Network interface

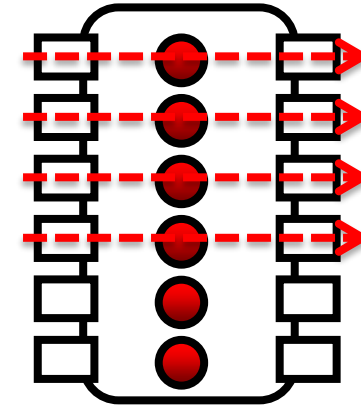
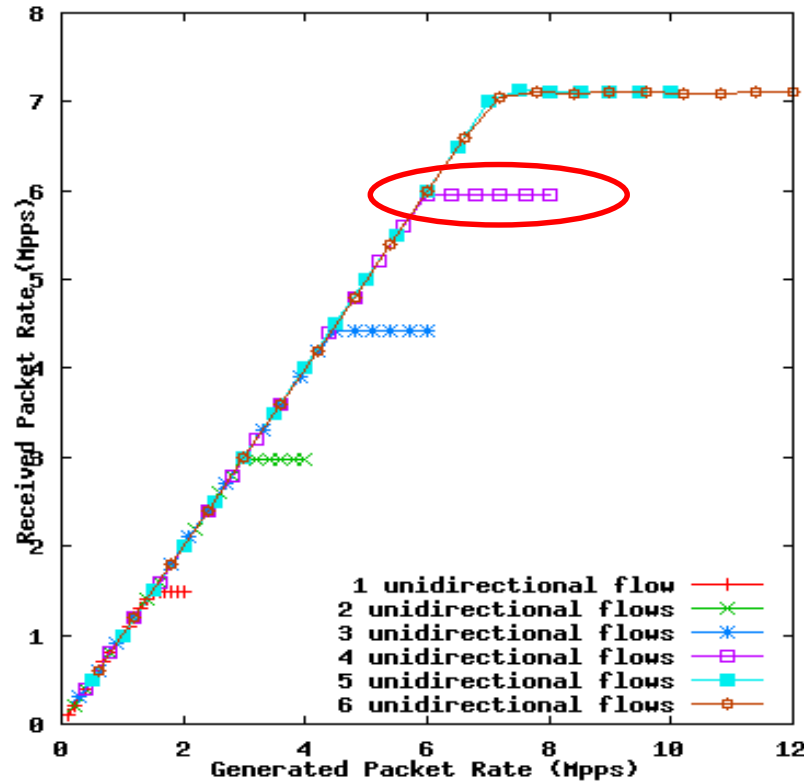
1-6 unidirectional flows, 64-byte packets, 1 flow per core

- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets



1-6 unidirectional flows, 64-byte packets, 1 flow per core

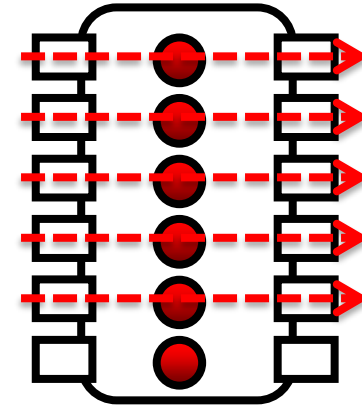
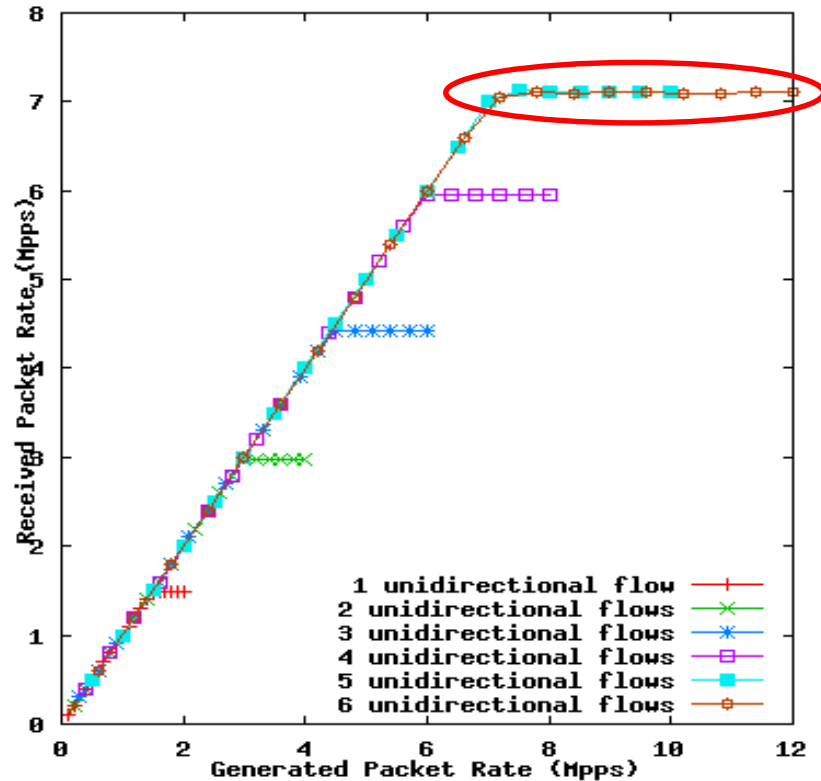
- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets



● CPU Core  
□ Network interface

1-6 unidirectional flows, 64-byte packets, 1 flow per core

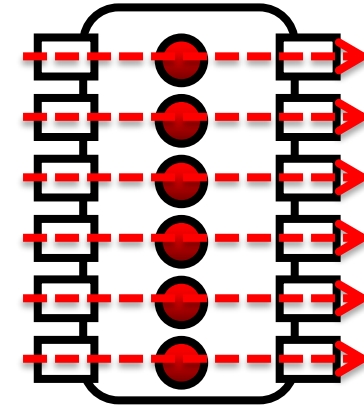
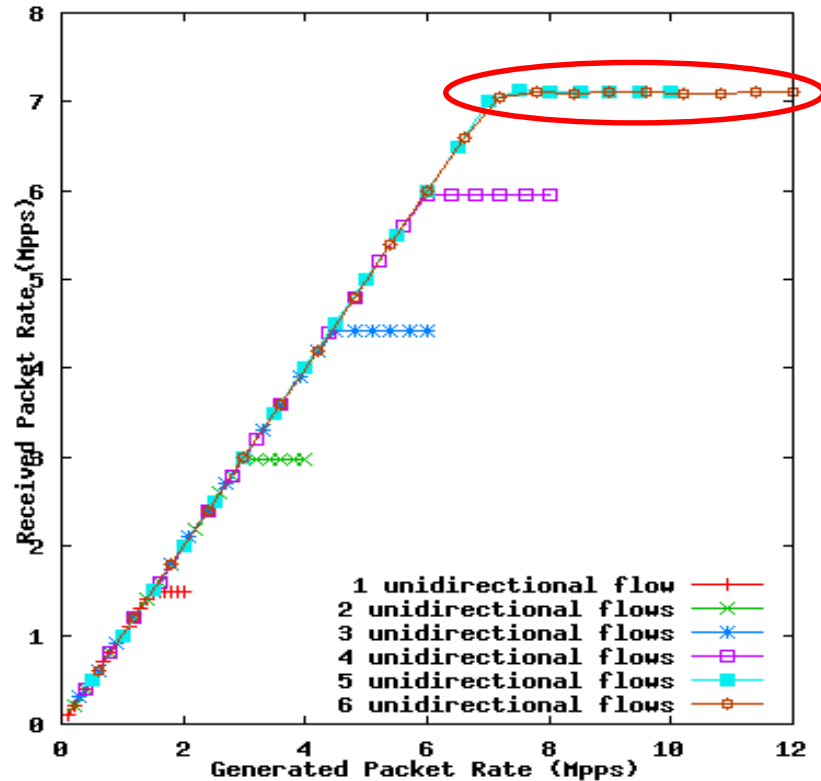
- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets



● CPU Core  
□ Network interface

1-6 unidirectional flows, 64-byte packets, 1 flow per core

- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets

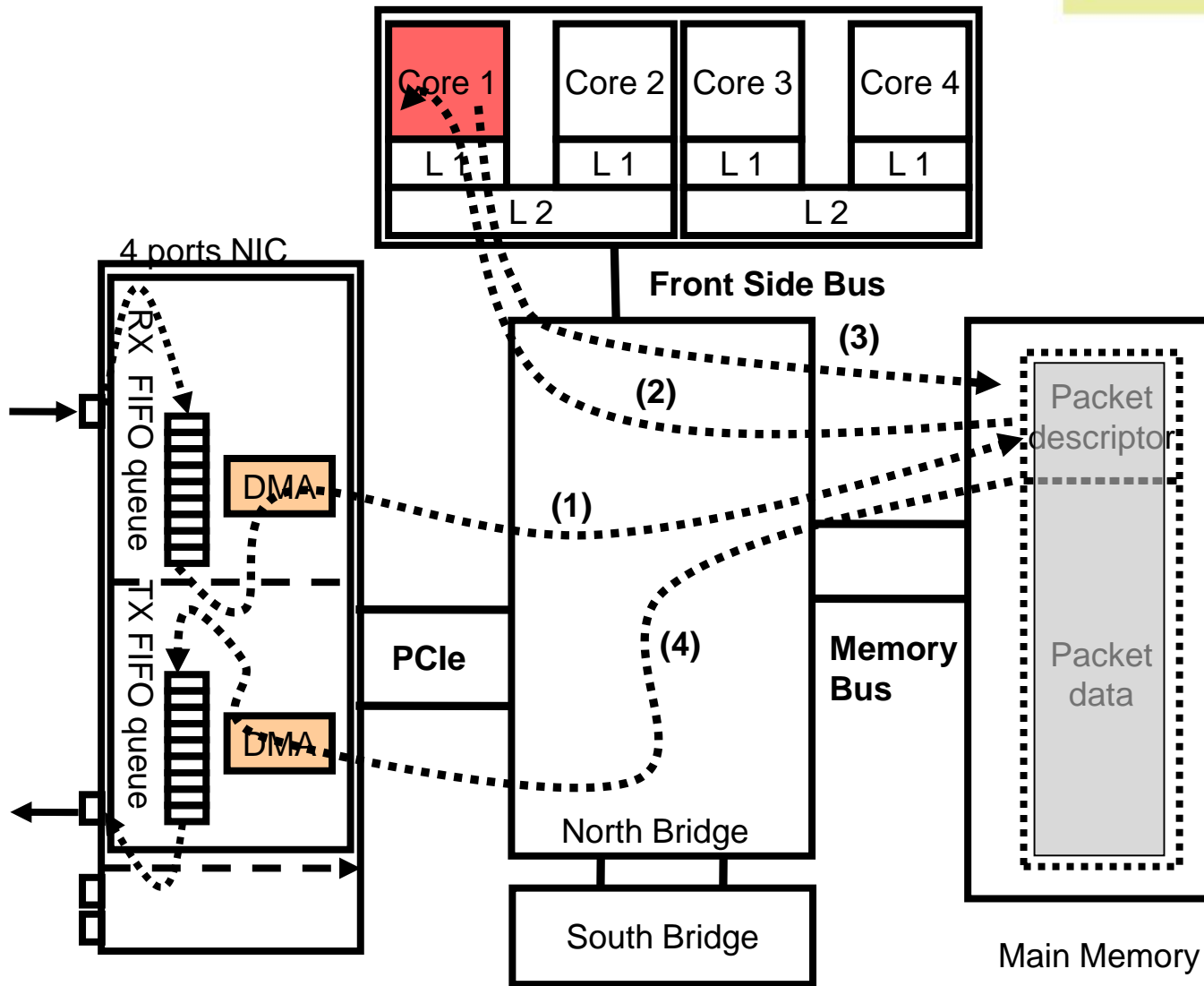


● CPU Core  
□ Network interface

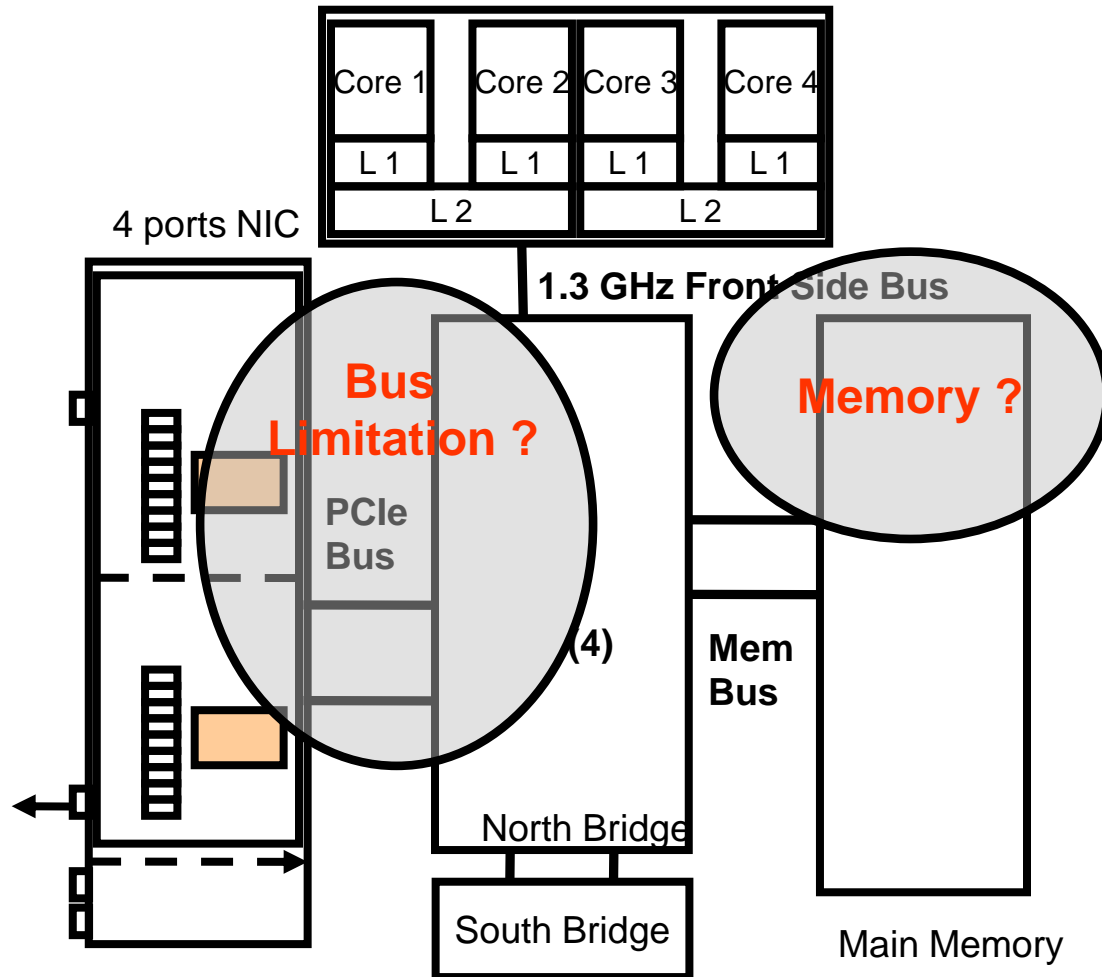
1-6 unidirectional flows, 64-byte packets, 1 flow per core

- Forwarding rates exceed 7 Mpps with small packets (faster than cheap Cisco routers)
- Line rate with large packets





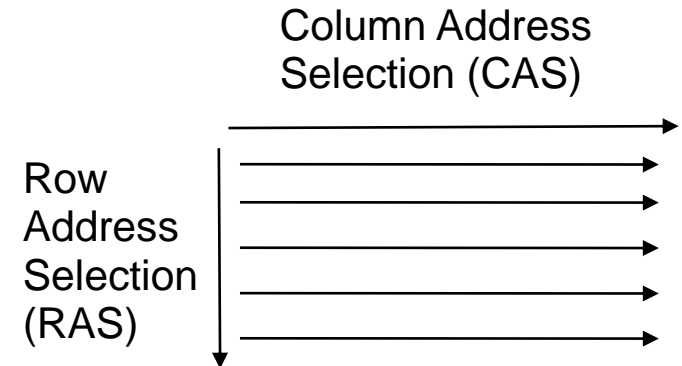
# Where is the Bottleneck?





- For PC-5300 (DDR2 667 MHz):

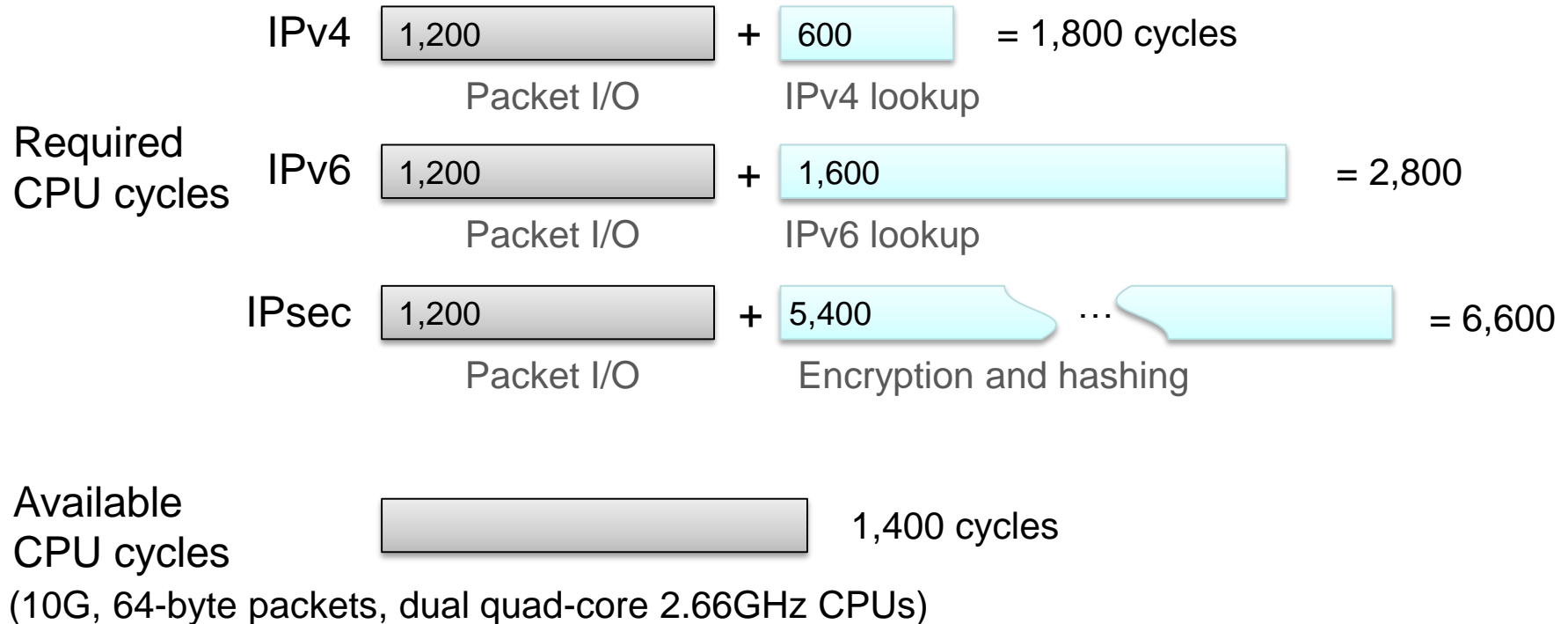
- Frequency: 333 MHz
- Latency: 3 ns/cycle
- CAS: 5 cycles
- RAS: 5 cycles



- Each memory access has to be CAS asserted (best case)
- Cost per memory access (due to CAS):
  - 5 cycles
  - 5 cycles x 3 ns/cycle = 15 ns



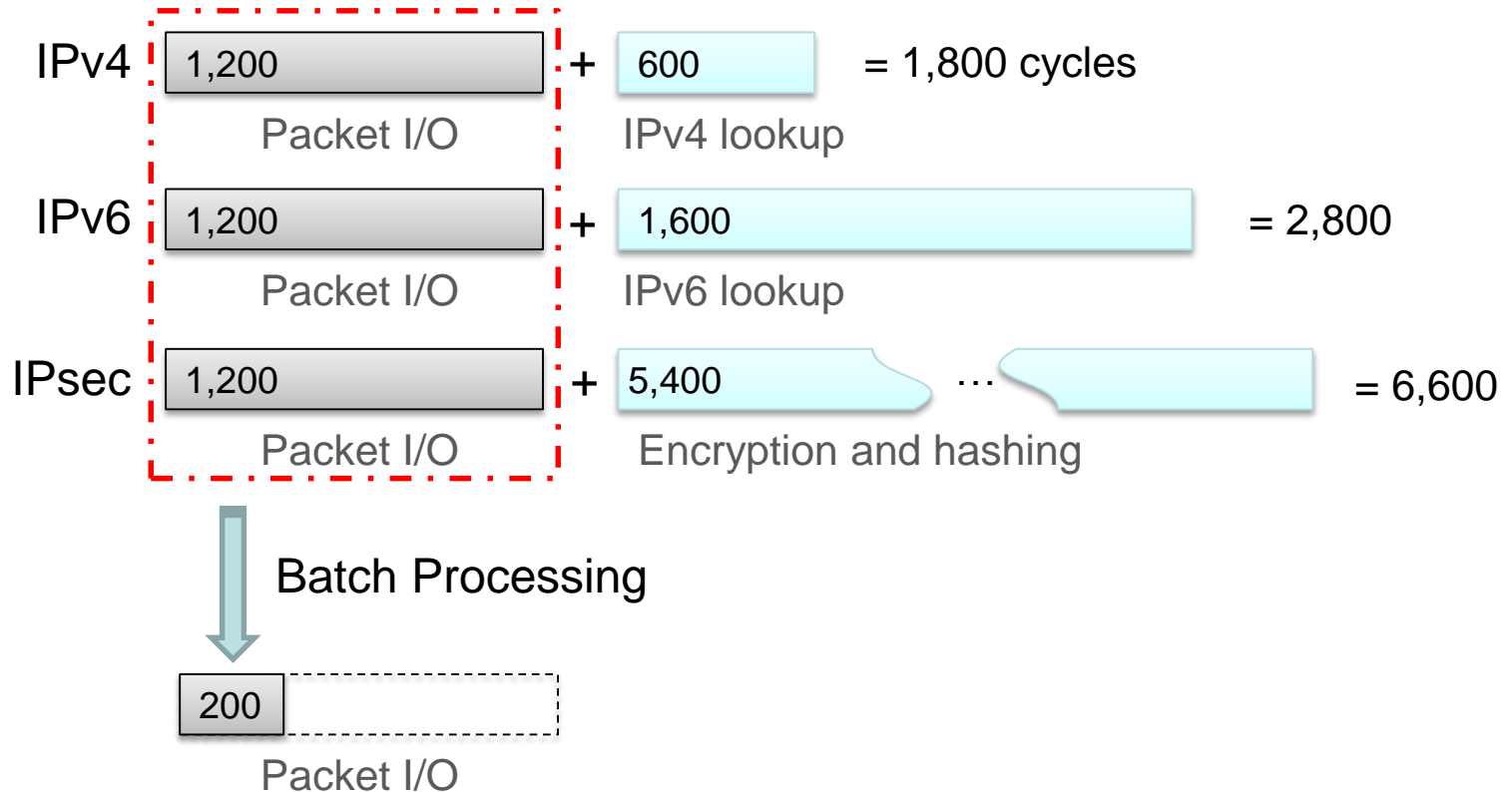
- Memory accesses per packet transaction with raw forwarding (without IP lookup):
  - 6 accesses from NIC (3 for transmit and 3 for receive path)
  - 4 accesses from CPU
- Total memory access delay:
  - $10 \text{ accesses} \times 15 \text{ ns} = 150 \text{ ns}$
- Hence, maximum forwarding rate is 6.7 Mpps:
  - for 1500-byte packets: 78 Gbps
  - for 64-byte packets: 3.3 Gbps (performance bottleneck)



CPU cycles from PacketShader and RouteBricks

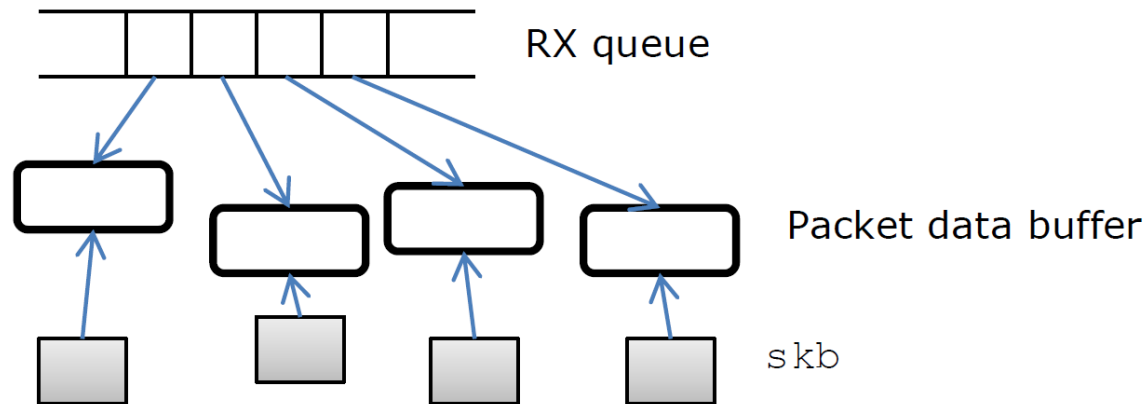


Required  
CPU cycles

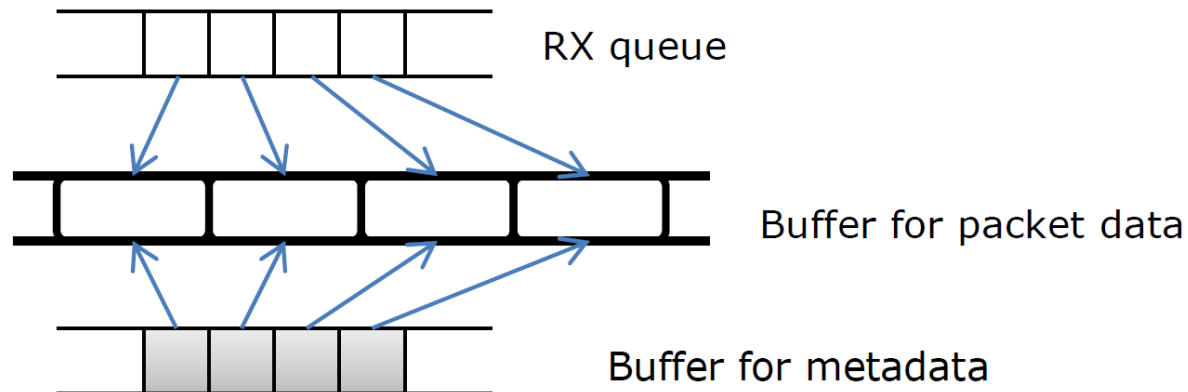


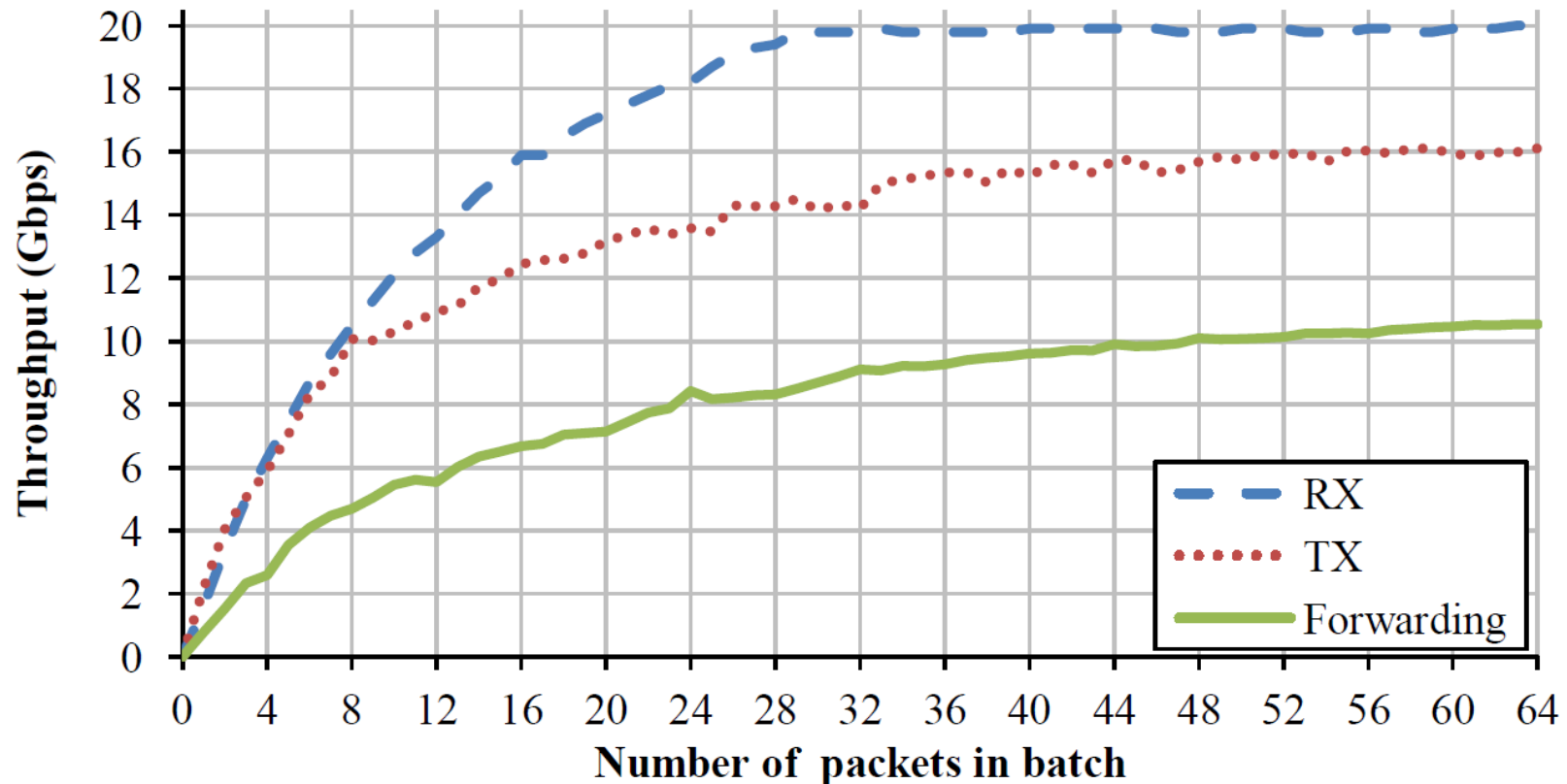
CPU cycles from PacketShader and RouteBricks

- Per-packet buffer allocation (Linux):



- Huge packet buffer for batch processing (PacketShader):





- Without batch processing:
  - 1.6 Gbps for Rx
  - 2.1 Gbps for Tx
  - 0.8 Gbps for forwarding

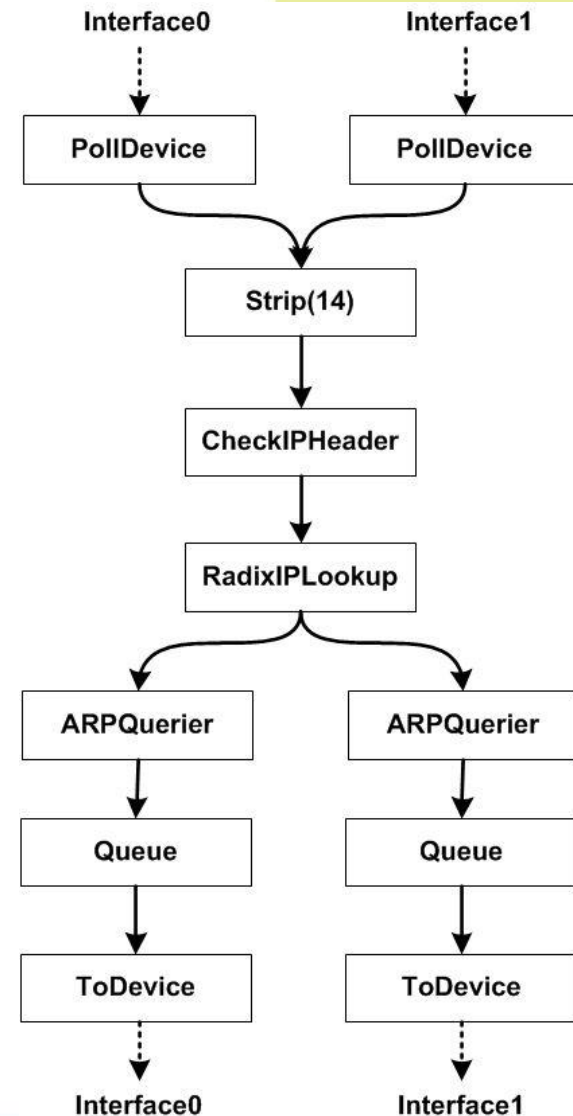




- Data Plane:
  - Linux Forwarding
  - Click Modular Router
- Control Plane:
  - XORP
  - Quagga (Cisco)

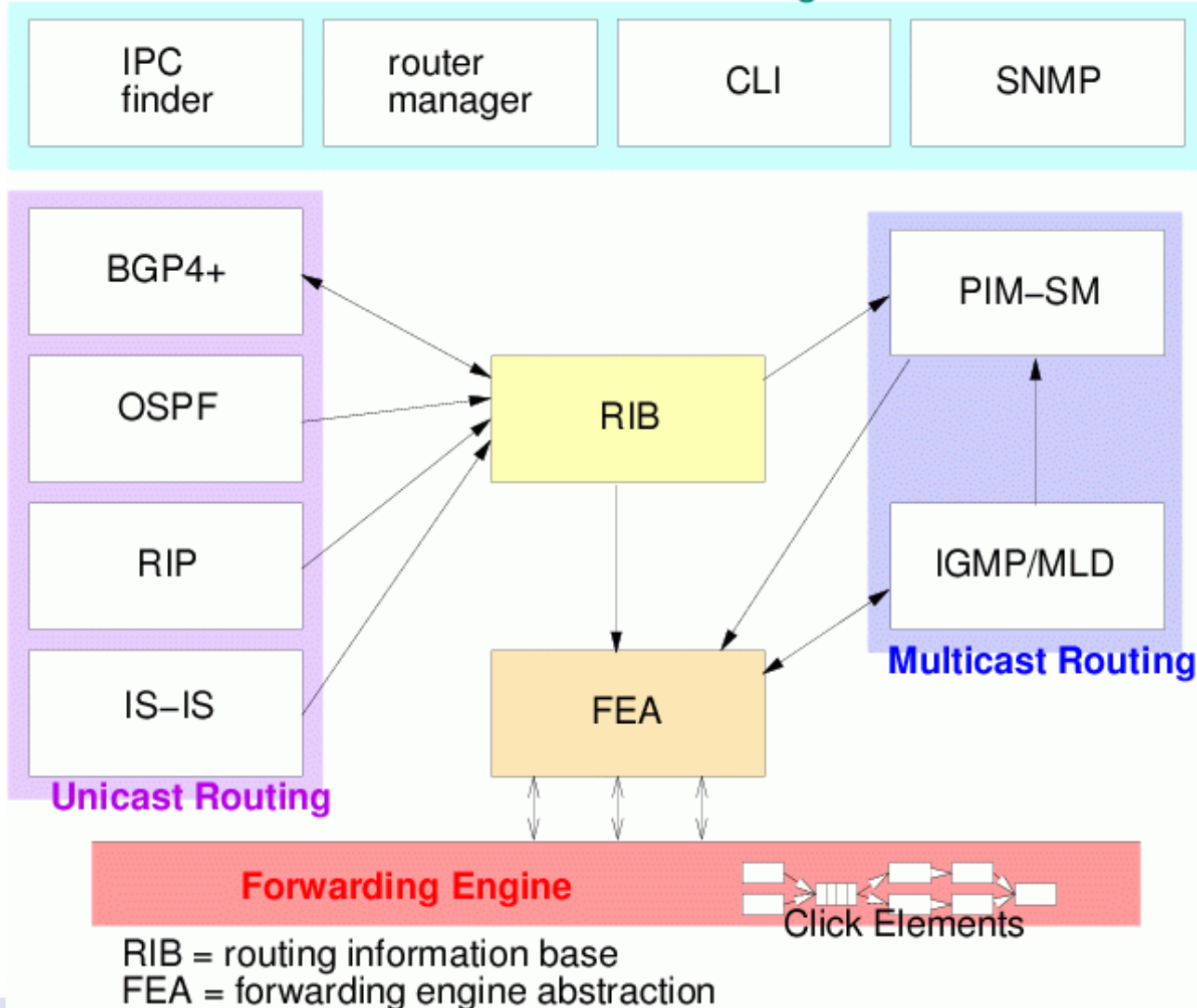


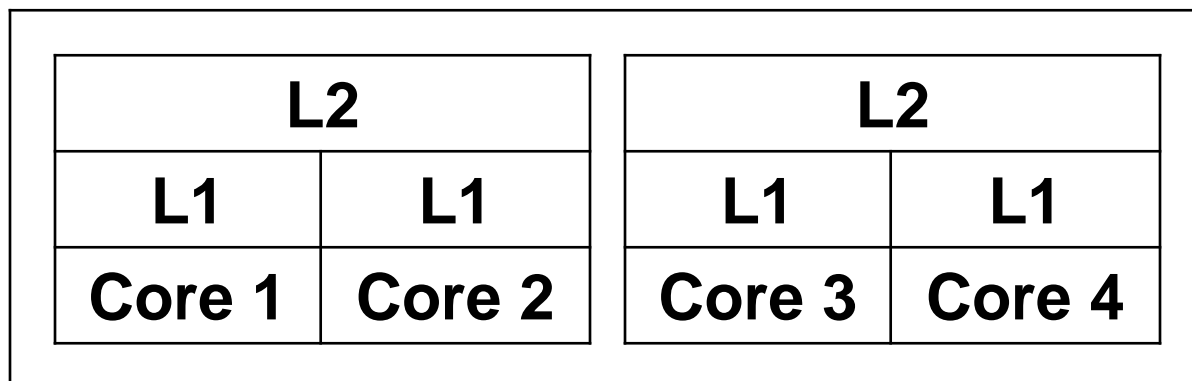
- Click Modular Router
  - Running in the kernel
  - Plenty of available elements
  - Easy implementation of new elements
  - High performance
- Click Task Scheduling
  - Click assigns tasks to threads (using Stride scheduler)
  - Linux schedules threads





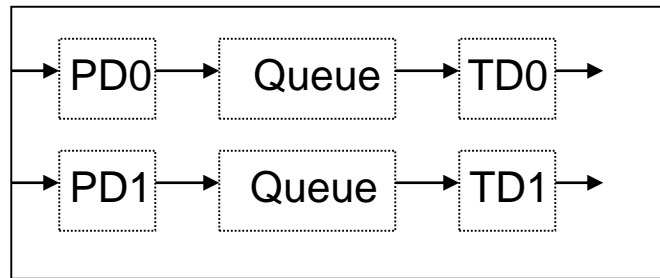
## Management Processes



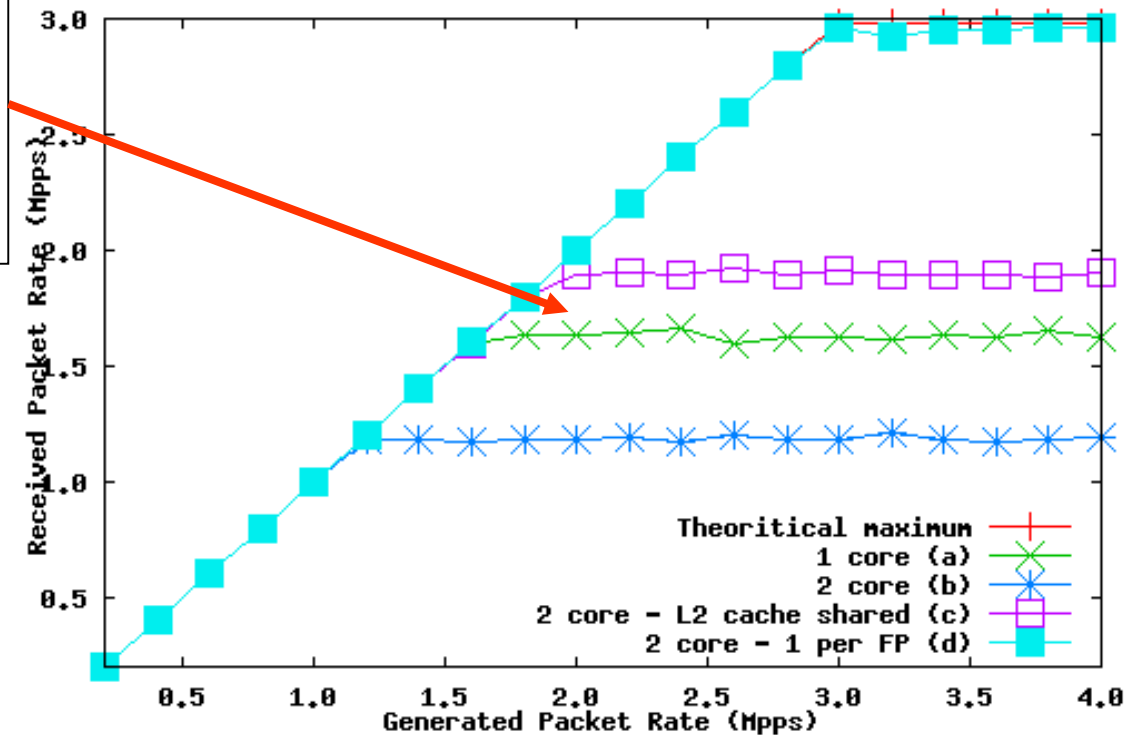


2-level Cache Hierarchy

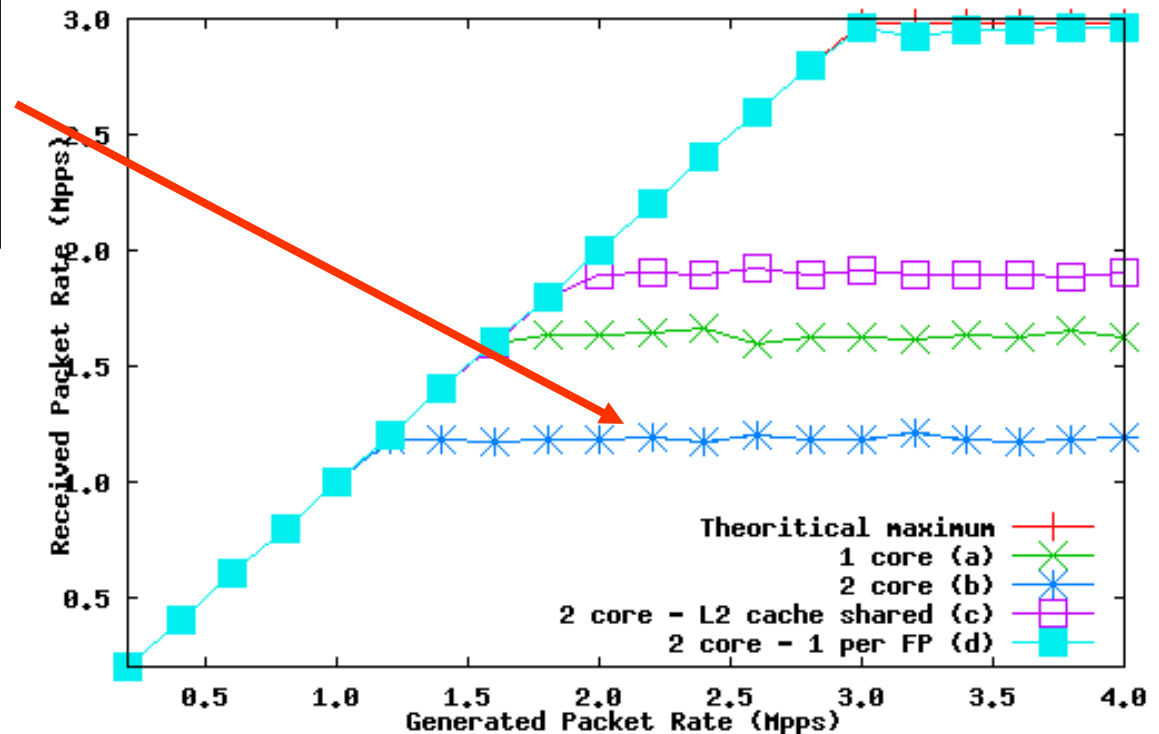
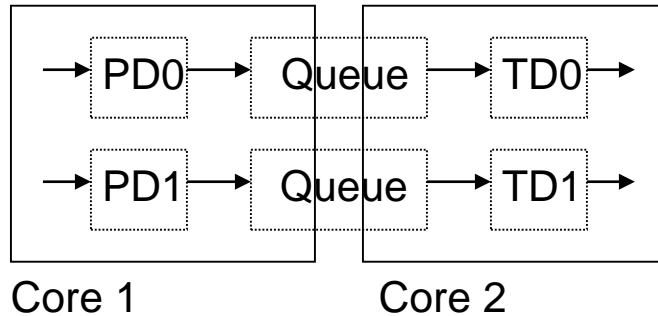
- L1 : small, very fast, non-shared
- L2: larger and slower than L1, shared



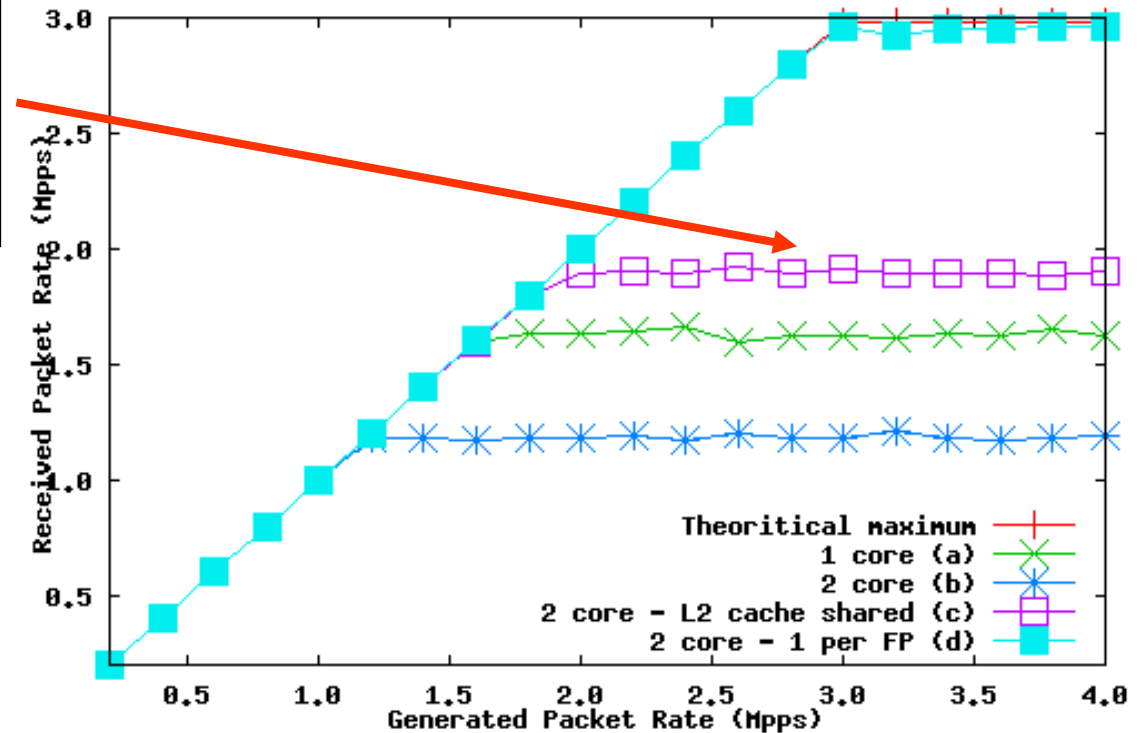
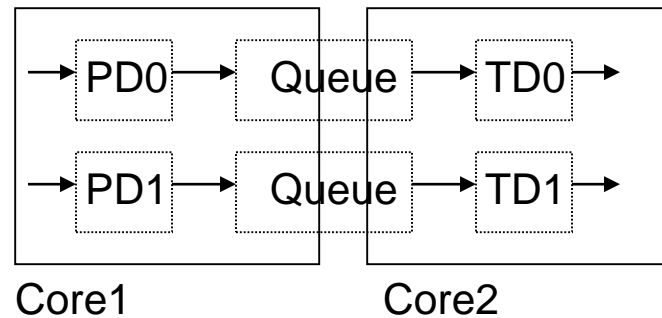
Core 1



- Everything allocated to a single core
- CPU-limited



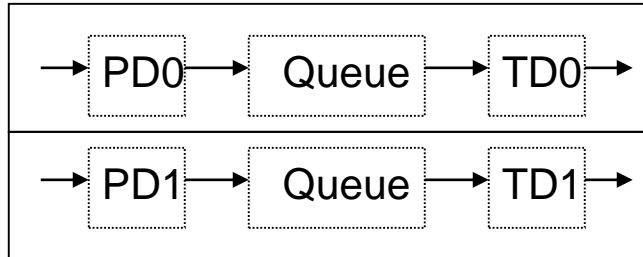
- Packets switch cores with L2 cache not shared
- Performance degradation due to increased memory accesses



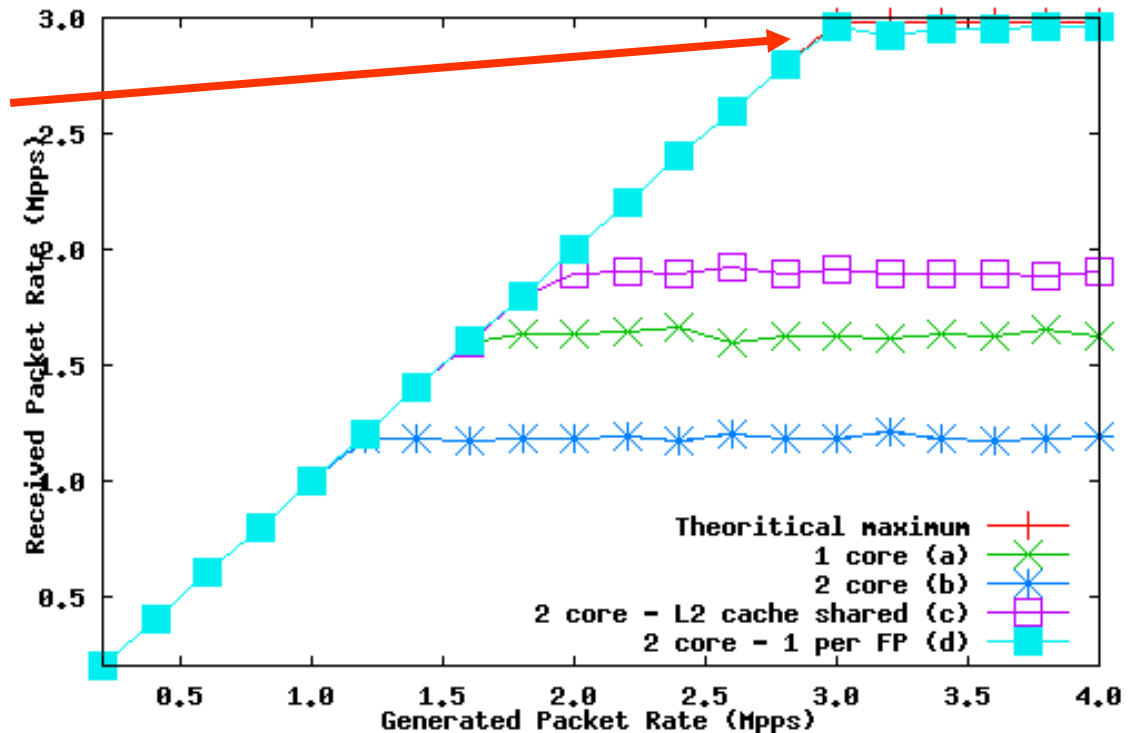
- Packets switch cores with shared L2 cache
- Improved performance



Core 1



Core 2



- One core per forwarding path
- Maximum performance

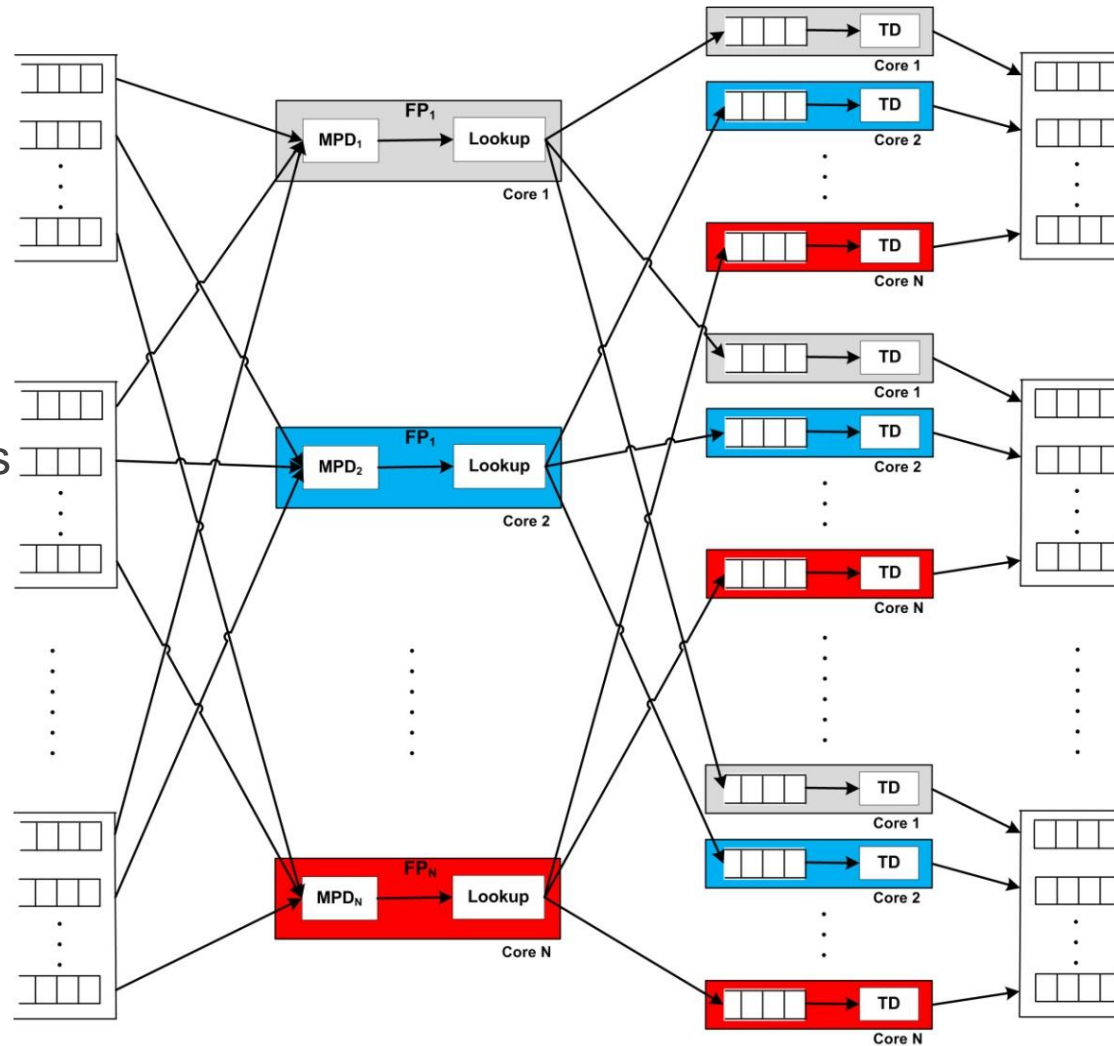




- Confine accesses to main memory
  - Packets should not traverse cache hierarchies during forwarding
- Distribute packet processing among CPU cores
  - Available processing power should be utilized efficiently
  - Less likely that packet processing will be CPU-limited
- Enable high degree of parallelization
  - Hardware multi-queuing allows multiple CPU cores to concurrently access a physical network interface
  - Multiple cores can process packets in parallel



- Forwarding path replicated across all cores:
  - Multiple cores are assigned to each physical interface
  - High level of parallelization
  - Packets do not switch cores



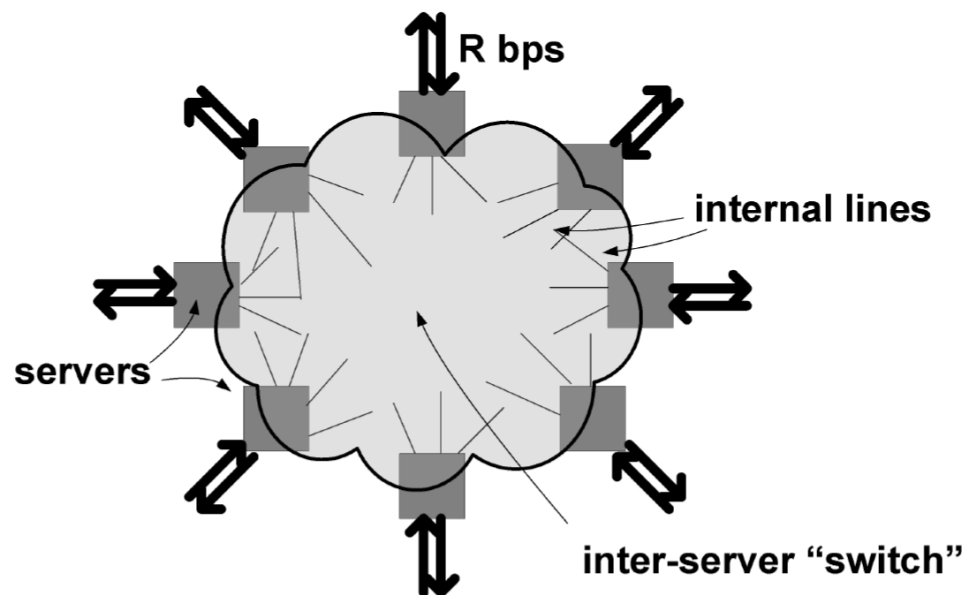


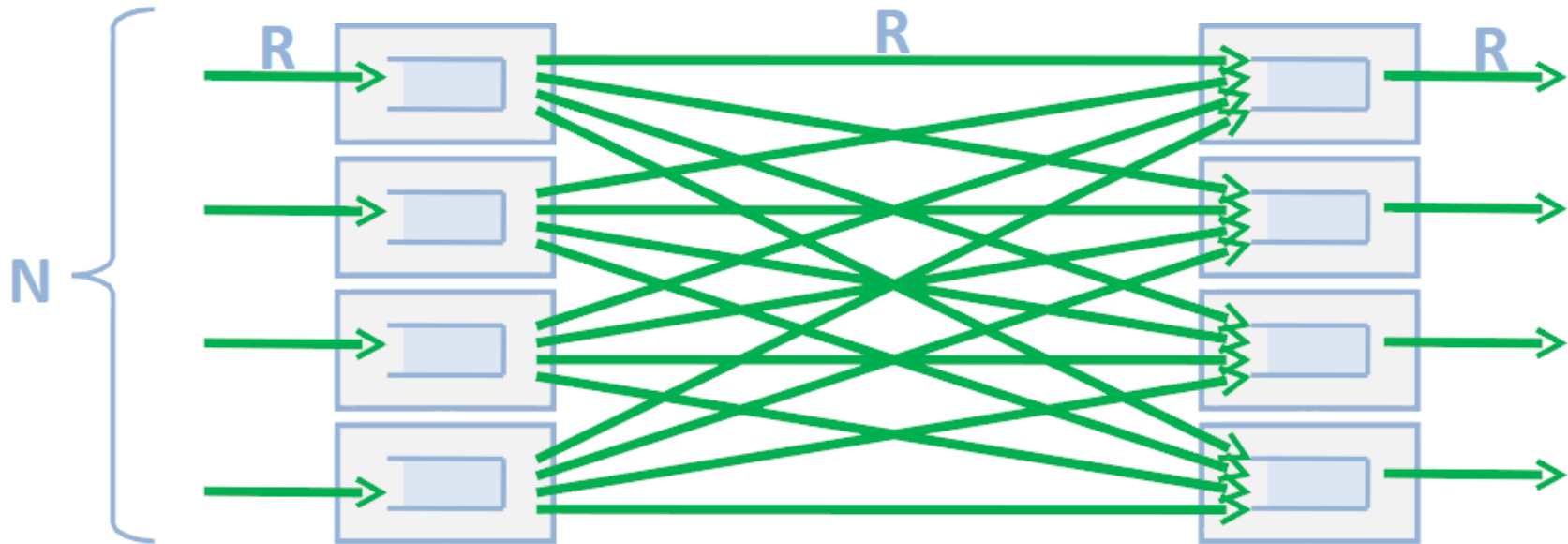
# Scaling Software Routers

Diagrams from RouteBricks paper and presentation

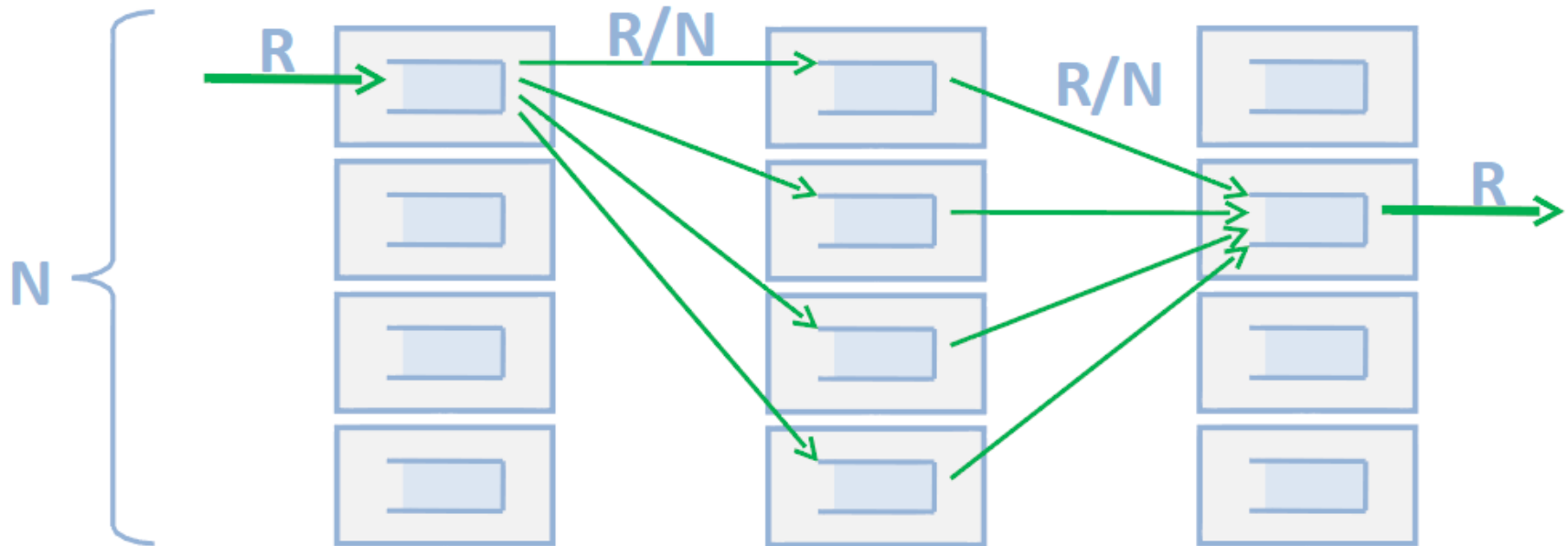


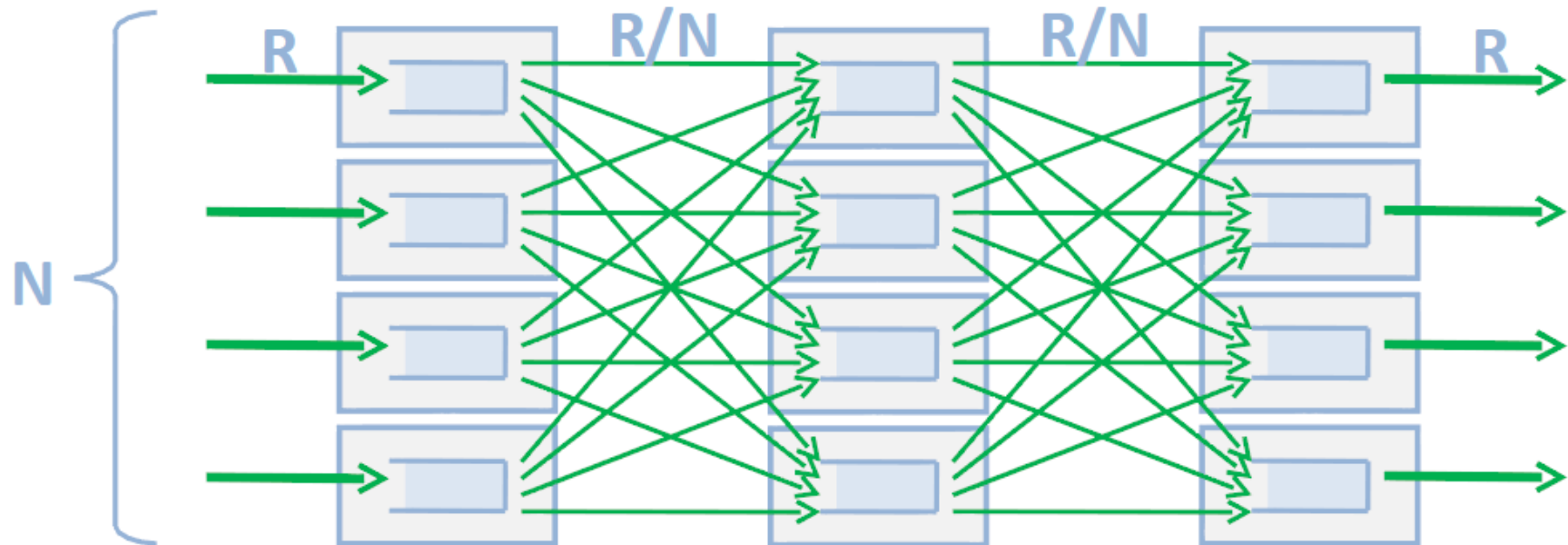
- Parallelization across multiple cores and I/O optimization result in forwarding rates of several Gbps for software routers
  - Such performance may be sufficient for edge routers but not for routers deployed in the Internet core (required throughput of 40 Gbps or higher)
- RouteBricks distributed software router:
  - Use a cluster of commodity servers to scale performance and port density



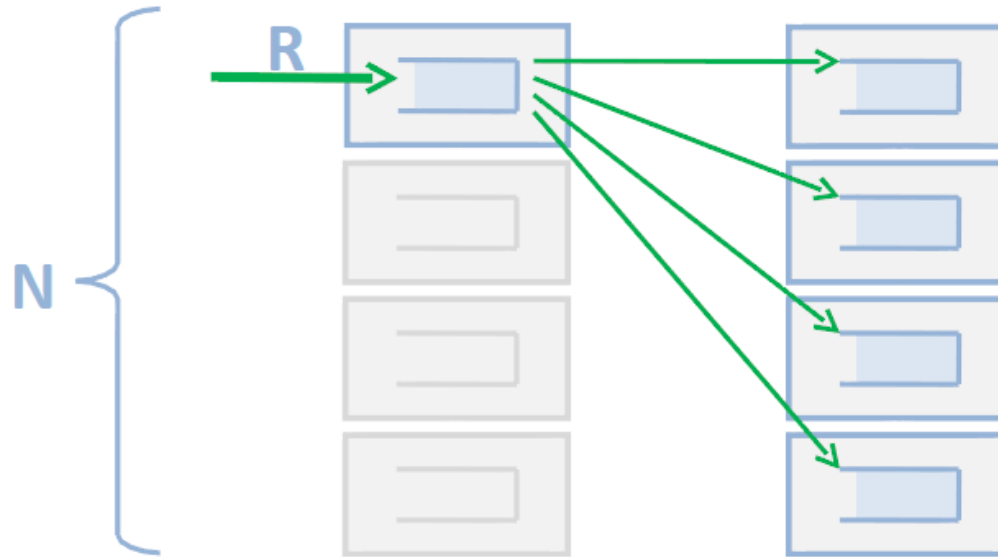


- Requirements:
  - $N$  external links of capacity  $R$
  - $N^2$  internal links of capacity  $R$
  - Per-server processing rate of  $2R$



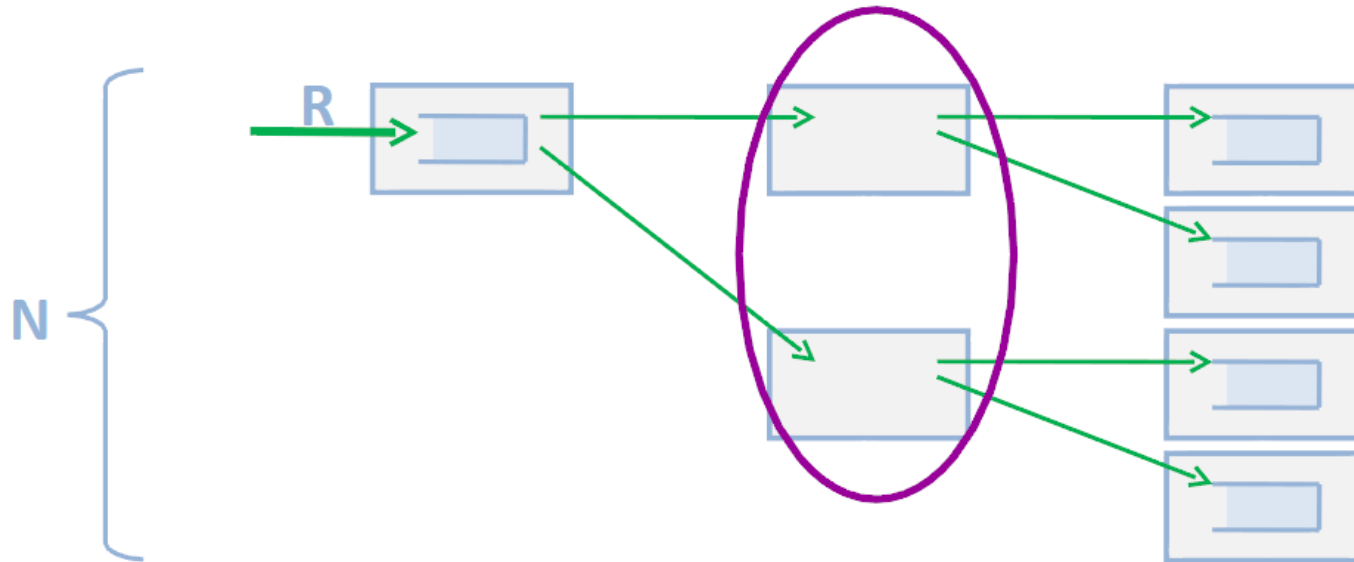


- Requirements:
  - $N$  external links of capacity  $R$
  - $N^2$  internal links of capacity  $2R/N$
  - Per-server processing rate of  $3R$



- Server port density is limited (12-16 ports)
  - When  $N$  exceeds the number of server ports, full mesh becomes infeasible





- Intermediate servers to overcome per-server port density limitation
  - Use intermediate servers only when full mesh is not possible

- P. Gupta, **Algorithms for Routing Lookups and Packet Classification**, Ph.D. Thesis, Stanford, USA, 2000
- N. Egi, et al., **Towards High Performance Virtual Routers on Commodity Hardware**, ACM CoNEXT 2008
- N. Egi, et al., **Forwarding Path Architectures for Multicore Software Routers**, ACM CoNEXT PRESTO 2010
- M. Dobrescu, et al., **RouteBricks: Exploiting Parallelism To Scale Software Routers**, ACM SOSP 2009
- S. Han, et al., **PacketShader: A GPU-Accelerated Software Router**, ACM SIGCOMM 2010
- E. Kohler, **The Click Modular Router**, Ph.D. Thesis, MIT, USA, 2001
- M. Handley, et al., **Designing Extensible IP Router Software**, USENIX NSDI 2005