【Internet Week 2014】
S6 パケットフォワーディング＆ルーティングの実装技術

ゼロから作る高速パケット転送用OS

東京大学大学院情報理工学系研究科
特任助教 浅井大史
<panda@hongo.wide.ad.jp>
2014年11月18日

ソフトウェアによるネットワーク機能

- SDN: Software Defined Network
  - Forwarding PlaneとControl Planeの分離
    - Control Planeをソフトウェアで実装
- NFV: Network Function Virtualization
  - ネットワーク機能の仮想化
    - ネットワーク機能を汎用CPU（仮想マシン）により実現

ネットワーク用OS

- 汎用CPUによるネットワーク機能の実現
  = ネットワーク用OS
  - ***Inexpensive***
  - ***Flexible***
  - ***Extensible***

Networking OS from Scratch

"**Networked**" Operating System manages
"**computing**" resources

"**Networking**" Operating System manages
"**computing**" *and* "**network**" resources

汎用OS（Linux/*BSD）の課題

- Not good for networking
  - プロセス・スレッド・VMによる
    ネットワーク機能の抽象化
    - Tickによる割り込みオーバーヘッド
    - コンテキストスイッチによるオーバーヘッド
      - キャッシュ汚染、TLBミス、etc.
    - パケット処理に最適でない I/Oスケジューラ
      - I/Oのボトルネックが全体の性能に影響

ゼロから作る
高速パケット転送用OS

- 「ゼロから作る」
  - 汎用ハードウェアの限界値を見極める
    - ボトルネック解析
  - 新しいアーキテクチャを検討する
    - スケジューラ
    - メモリ管理
    - 権限管理
    - プロトコルスタック
    - ルーティングテーブル管理インスタンス

ゼロから作る
高速パケット転送用OS

- 「ゼロから作る」
  - 汎用ハードウェアの限界値を見極める
    - ボトルネック解析
  - 新しいアーキテクチャを検討する
    - スケジューラ
    - メモリ管理
    - 権限管理
    - プロトコルスタック
    - ルーティングテーブル管理インスタンス

汎用ハードウェアの限界を目指す

- 汎用NICにおける高レート転送でのボトルネック解析
  - ボトルネック
    - CPU?
    - Memory?
    - PCIe bus?
    - or something else?

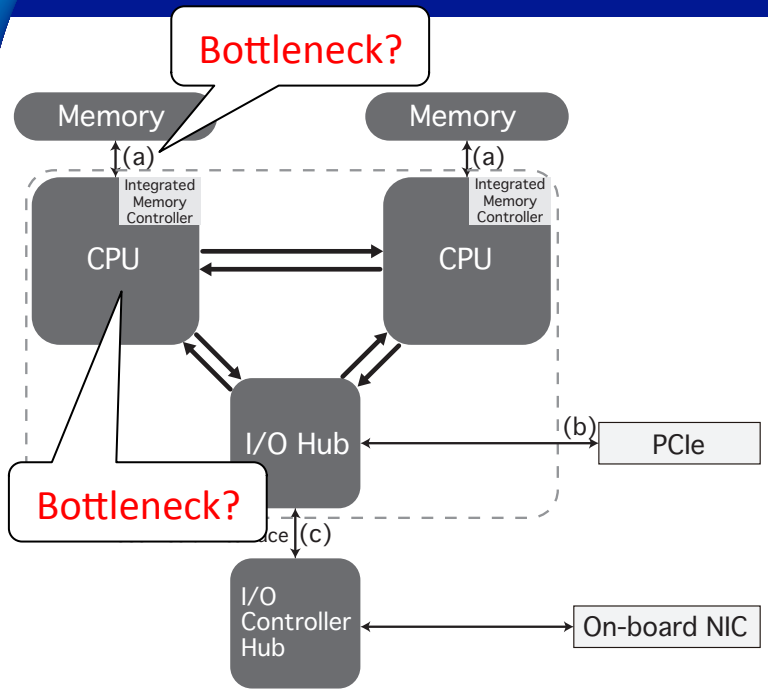
10/40/100GbE時代のパケット転送

- Ethernet
  - 最短フレーム長：64-Byte（=最高レート）
    - 1GbE: 1.488Mpps
      = 672 ns/packet
    - 10GbE: 14.88Mpps
      = 67.2 ns/packet
    - 40GbE: 59.52Mpps
      = 16.8 ns/packet
    - 100GbE: 148.8Mpps
      = 6.72 ns/packet

Myths

- 汎用機における
  パケット転送のボトルネック
  - CPU：CPUが遅い
  - メモリ：メモリコピーが遅い
    - メモリコピーの削減
      - netmap [Rizzo, USENIX ATC 2012]
      - Intel® DPDK
  - 割り込み：割り込みが重い
    - Linuxにおける解決策：NAPI
    - ポーリング
      - Intel® DPDK

汎用機におけるスループット



(a) 3.3GHz clock CPU
- 0.3ns per cycle (220 cycles / packet)
  - + スーパースカラ

(b) CPU-Memory bus (N.B., 64 bit wide access)
- DDR3-1333 Dual Channel: 21.333GB/s (170.667Gbps)
- DDR3-1600 Dual Channel: 25.600GB/s (204.800Gbps)
- DDR3-1866 Dual Channel: 29.867GB/s (238.933Gbps)

(c) PCIe bus
- Gen2: 500MB/s (x1) = 4Gbps
  - usually x8 for a two-port 10GbE NIC
  - x16 is not enough for a two-port 40GbE NIC
- Gen3: 985MB/s (x1) = 7.88Gbps

(d) DMI bus
- v1.0: 2GB/s (1GB/s per direction = 8Gbps)
- v2.0: 4GB/s (2GB/s per direction = 16Gbps)

汎用機におけるレイテンシ

- Data access latency (*)
  - L1 cache: 4-5 cycles ~ 1.2-1.5ns
  - L2 cache: 12 cycles ~ 3.6ns
  - L3 cache: 27.85 cycles ~ 8.4ns
  - RAM: 28 cycles + 49-56 ns ~ 65ns
    - ただし、アウトオブオーダー実行などで軽減

(*) http://www.7-cpu.com/cpu/SandyBridge.html

本当のボトルネックは、どこ？

PCIe NICのボトルネック

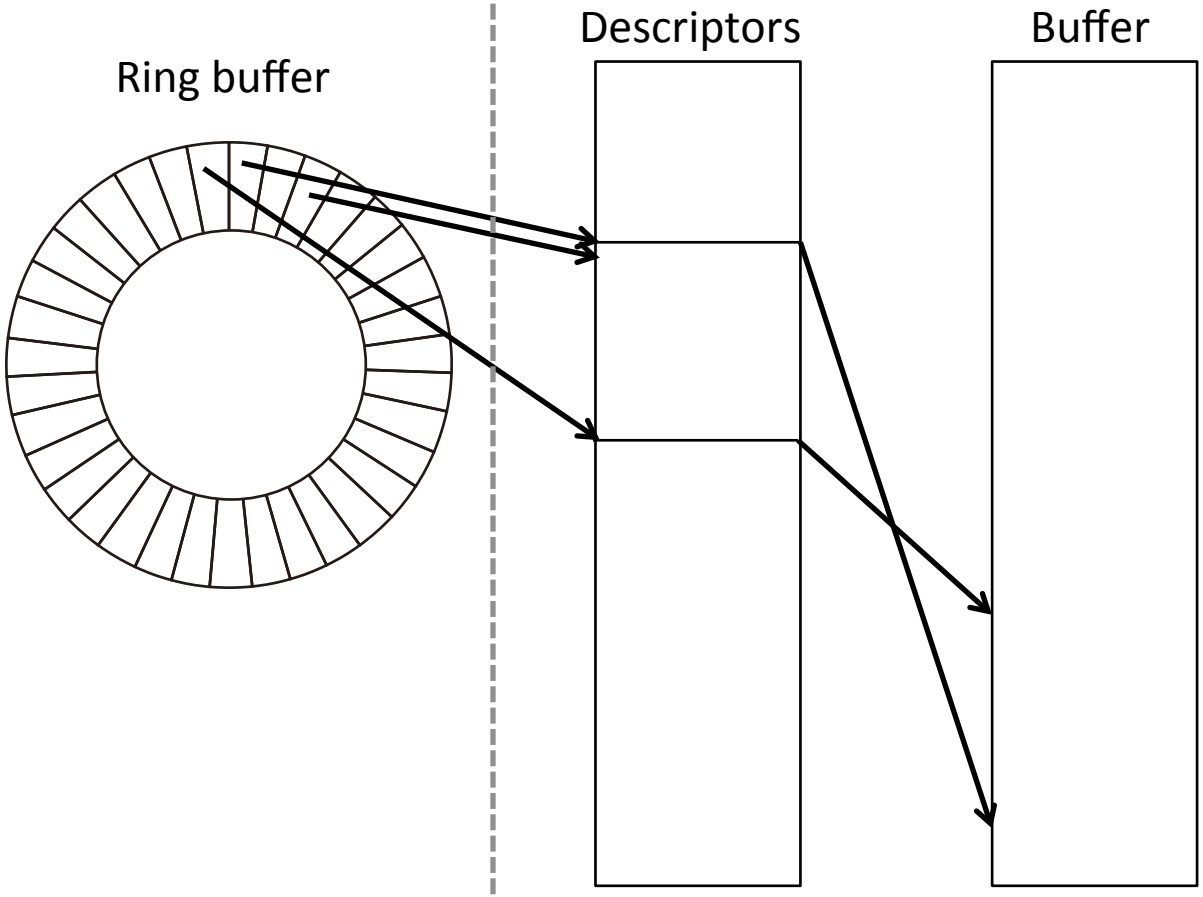
- PCIeのレジスタアクセス
  = Memory Mapped I/O (MMIO)
  - （当然）キャッシュが効かない
  - 1529.17 cycles / read
    - 392.1 ns / read
  - 282.621 cycles / write
    - 72.47 ns / write

※同一のレジスタに1M回アクセスしたときの平均を
CPUのPerformance Monitoring Counter (PMC)により計測

CPU: Intel Core i7 4770K
Memory: Corsair DDR3-1866 8GB x4
NIC: Intel X520-DA2

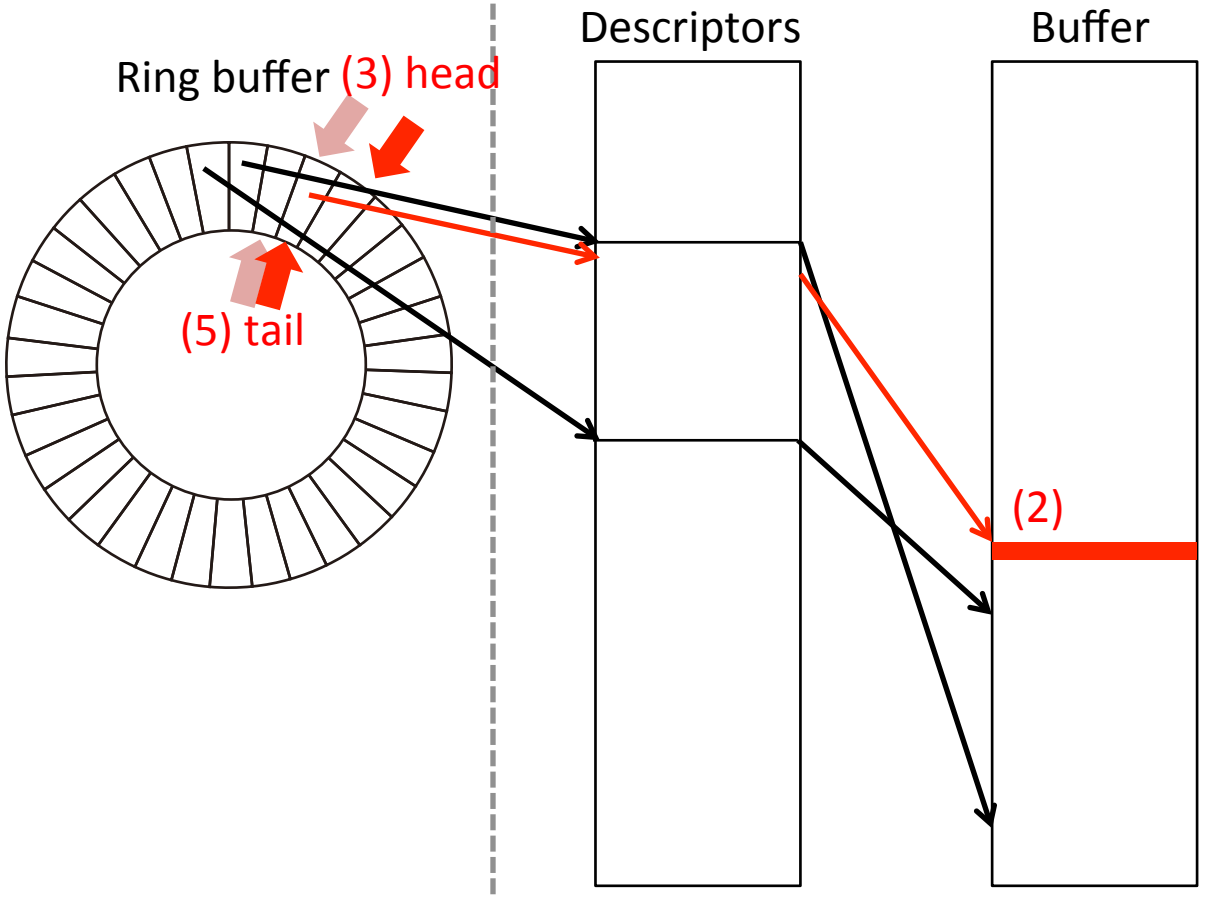
汎用NICにおけるボトルネック

Generic NIC architecture



汎用NICにおけるボトルネック

Generic NIC architecture

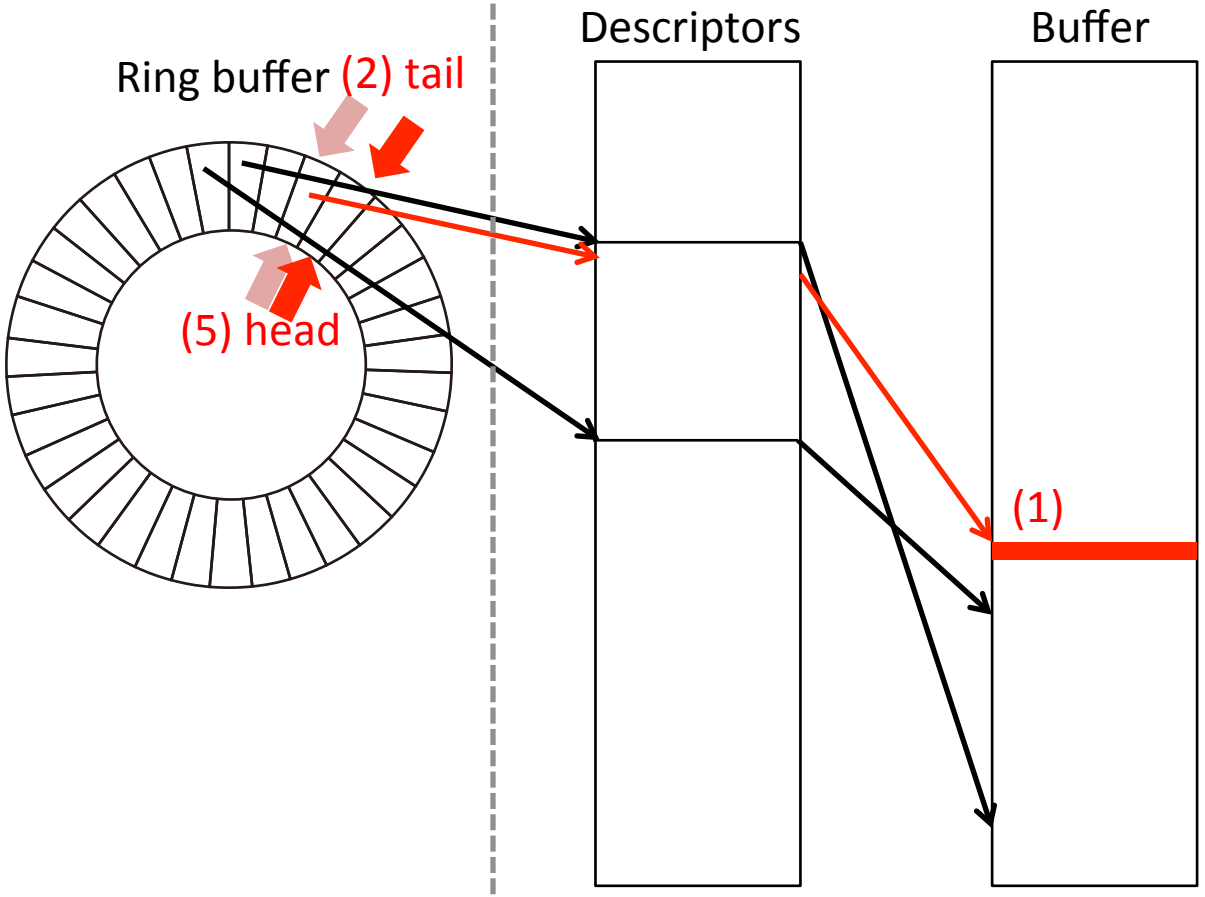


Packet reception

1. NIC receives a packet
2. NIC transfer the packet data to a buffer in RAM via DMA
3. NIC proceeds the head pointer
4. Software processes the packet
5. Software proceeds the tail pointer to release the packet

汎用NICにおけるボトルネック

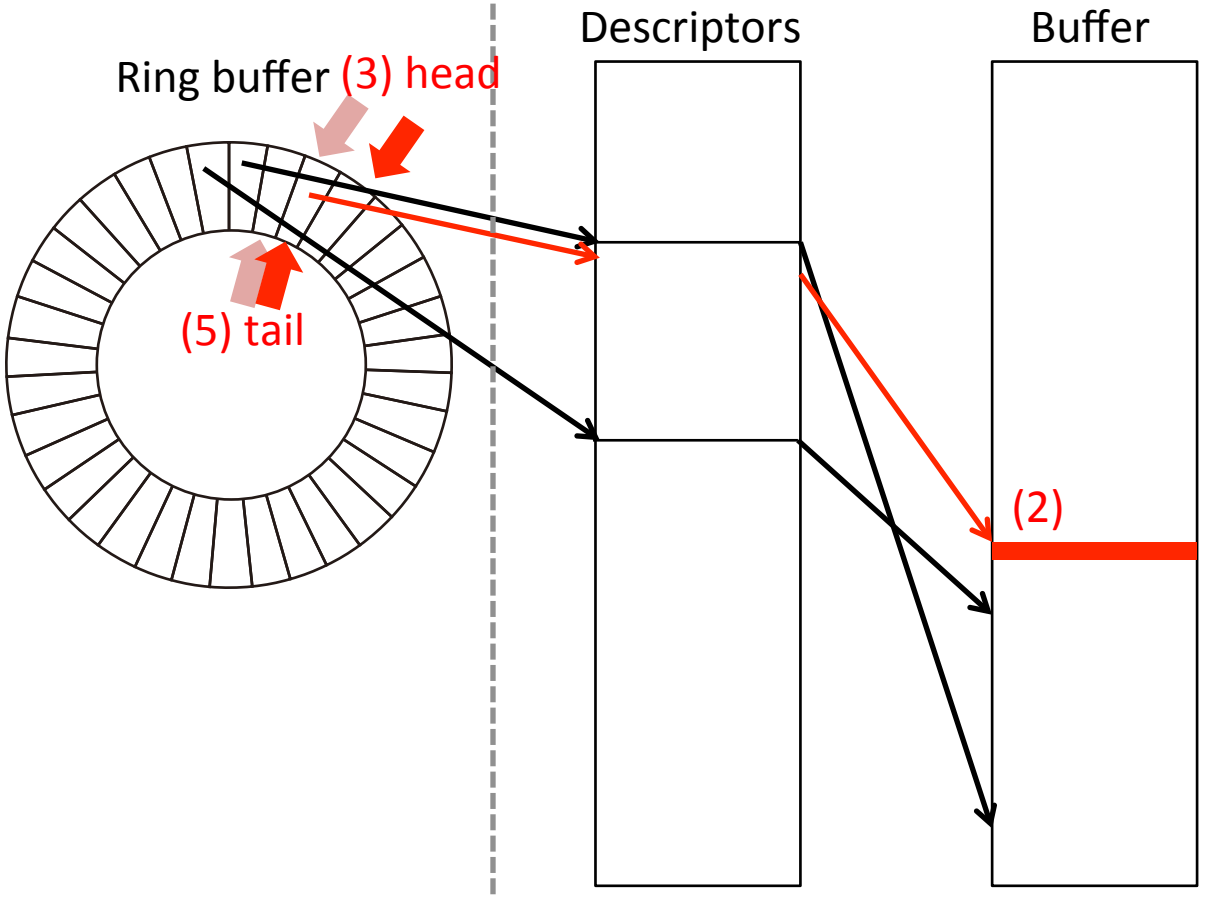
Generic NIC architecture



Packet transmission

1. Software writes a packet to a buffer in RAM
2. Software proceeds the tail pointer to commit the packet
3. NIC transfer the packet data from the buffer in RAM via DMA
4. NIC transmit the packet
5. NIC proceeds the head pointer to notify the packet is transmitted

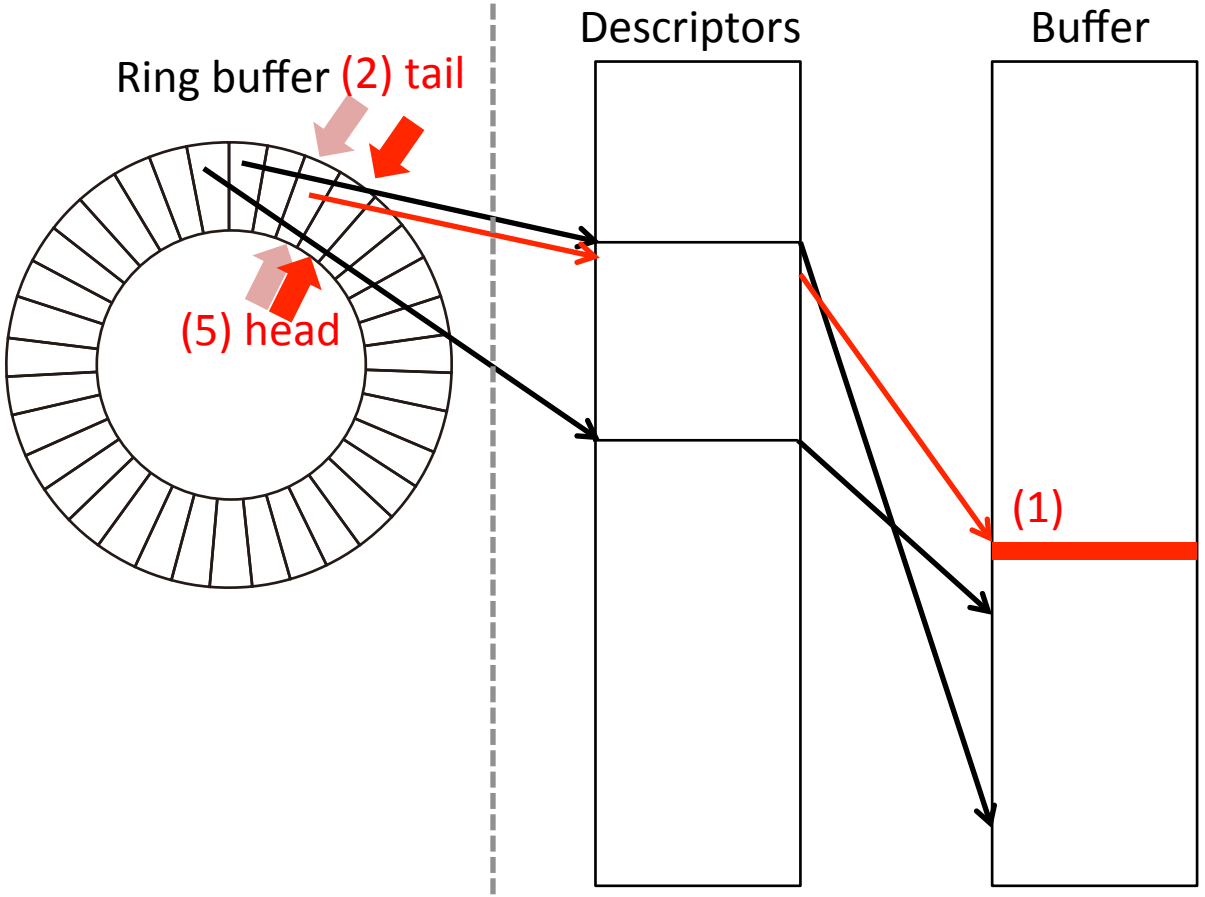
汎用NICにおけるボトルネック



Packet reception

1. NIC receives a packet
2. NIC transfer the packet data to a buffer in RAM via DMA
3. NIC proceeds the head pointer
4. Software processes the packet
5. Software proceeds the tail pointer to release the packet

汎用NICにおけるボトルネック



Packet transmission

1. Software writes a packet to a buffer in RAM
2. Software proceeds the tail pointer to commit the packet
3. NIC transfer the packet data from the buffer in RAM via DMA
4. NIC transmit the packet
5. NIC proceeds the head pointer to notify the packet is transmitted

（これを踏まえて、）Txでラインレートを出すには？

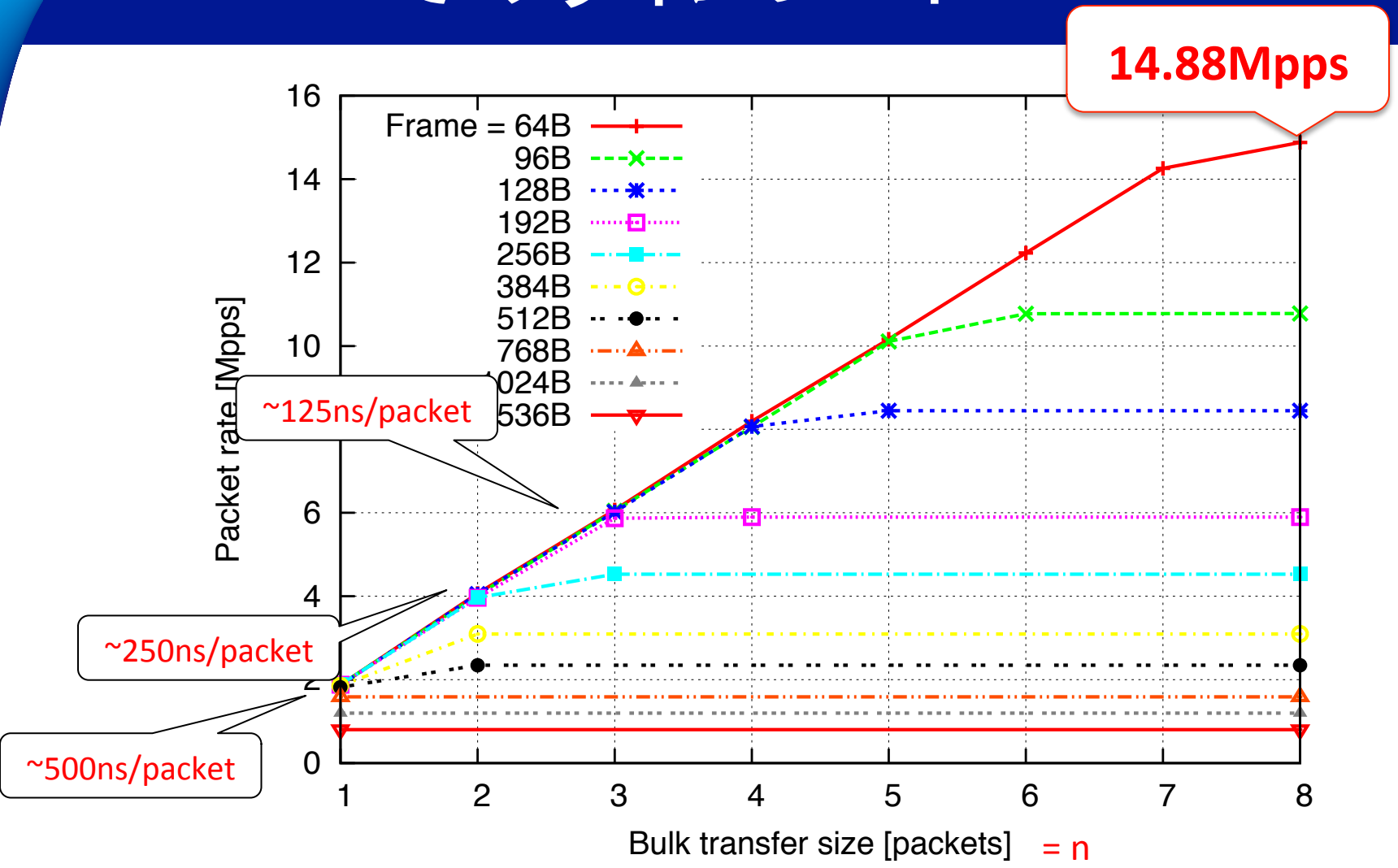
- 手順
  - 予めUDPパケットを構成（CPUサイクルを使わない）
  - Txのリングバッファが空いていれば n パケット一気に送る
    - Descriptorの構成
    - nパケットごとにTx tailを進める

```
txq_tail = 0;
for ( ;; ) {
     txq_head = read_txq_head();
     /* Available Tx queue length */
     txq_len = txq_sz
          - (txq_sz - txq_head + txq_tail) % txq_sz;
     /* Check the available Tx queue length */
     if ( txq_len < n ) continue;
     for ( i = 0; i < n; i++ ) {
          // Set packet to the ring buffer to txq_tail
          txq_ring[txq_tail].pkt = pkt_to_transmit;
          txq_tail = (txq_tail + 1)  % txq_sz
     }
     /* Commit */
     write_txq_tail(txq_tail);
}
```

~392.1ns

~72.47ns

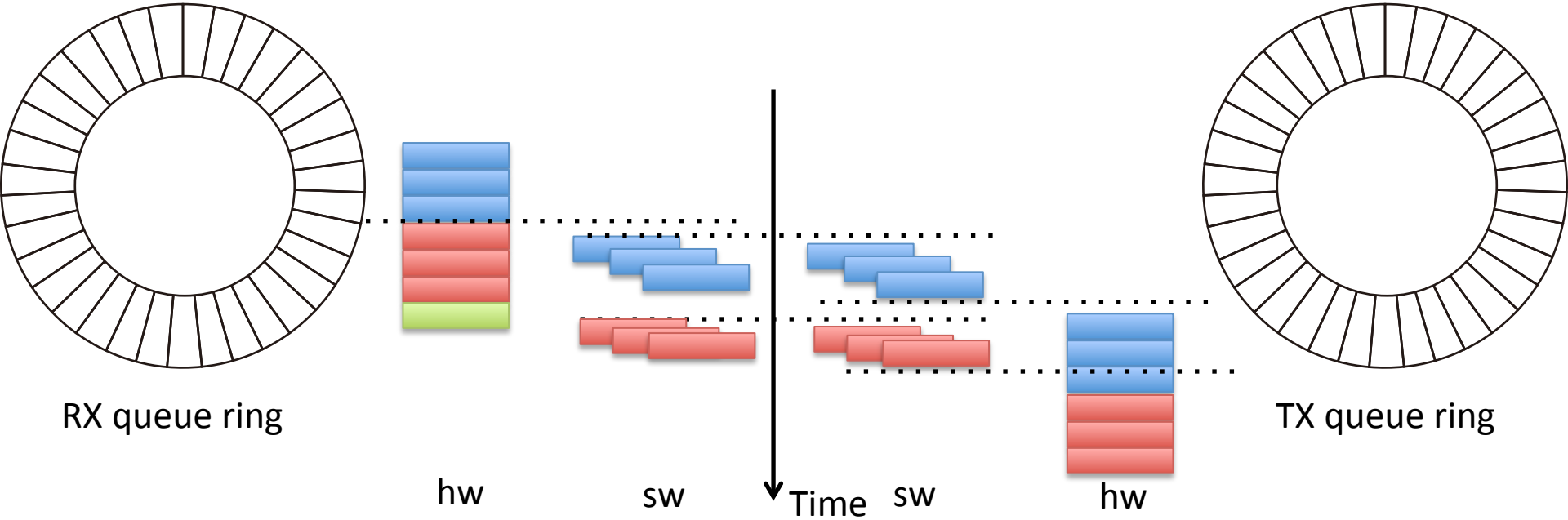
Txでのラインレート


14.88Mpps
Frame = 64B
96B
128B
192B
256B
384B
512B
768B
024B
536B
~125ns/packet
~250ns/packet
~500ns/packet
Packet rate [Mpps]
Bulk transfer size [packets]
= n
0
2
4
6
10
12
14
16
1
2
3
4
5
6
7
8

ルーティングは？

Strategy

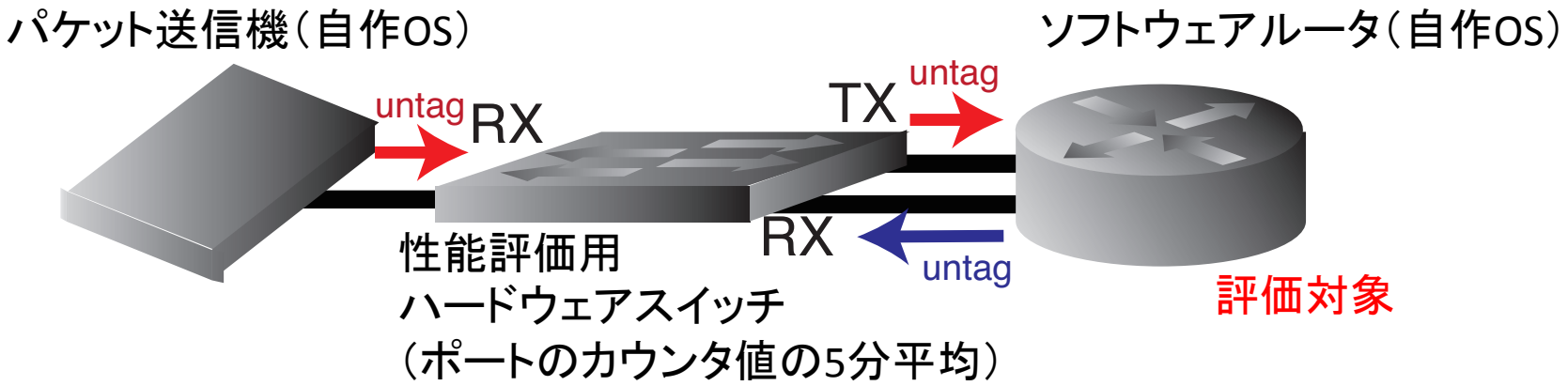
- バルクで転送する！
  - PCIeのレジスタアクセス中に溜まったパケットはバルク処理



ルーティングは？

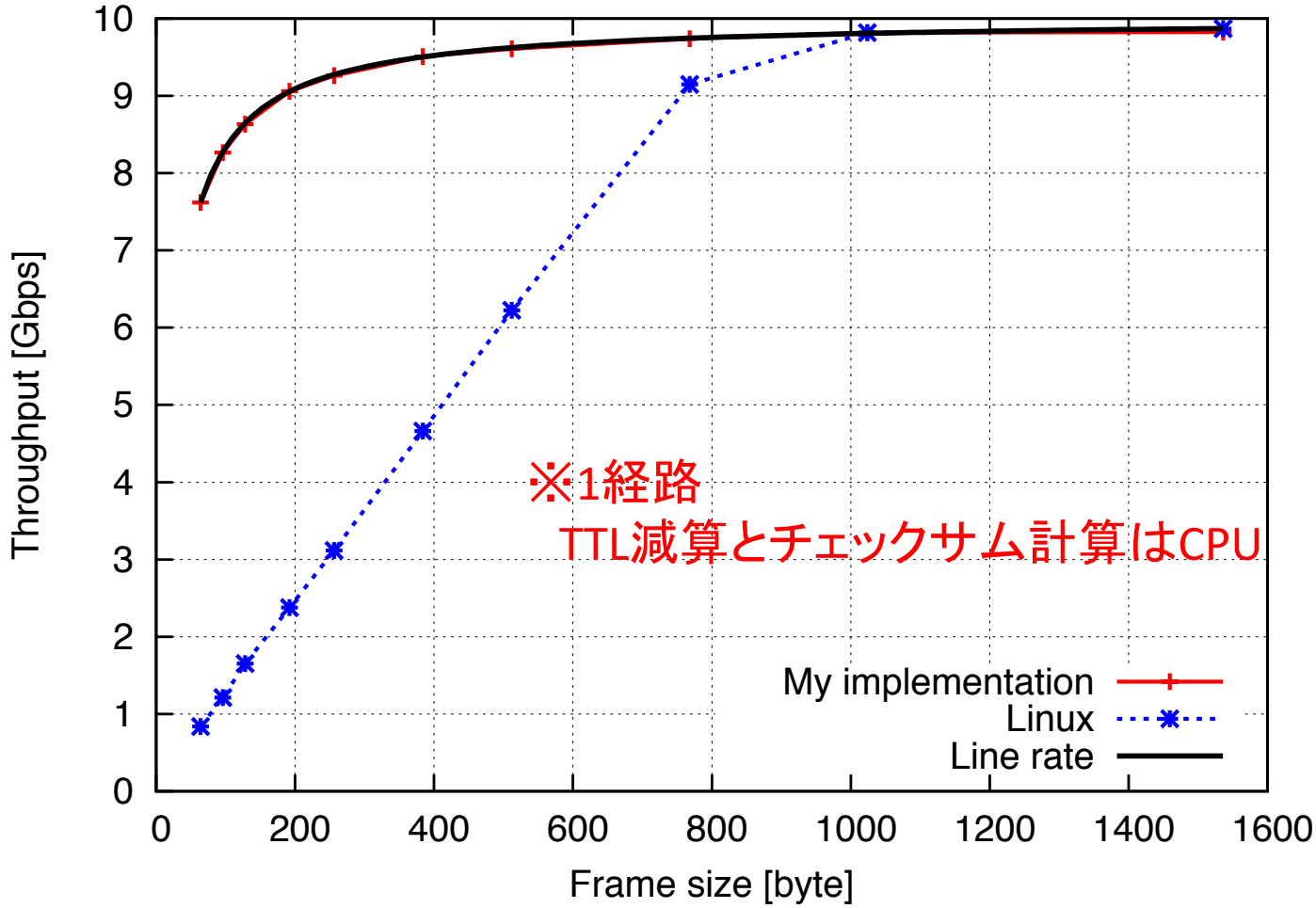
```
rxq_tail = txq_tail = 0;
blkcnt = 0;
/* # of packets to be routed in bulk transfer */
nr_blk = 256 /* can be another value */;
for ( ;; ) {
     /* Rx queue head */
     rx_desc = GET_RX_DESC_HEAD(netdev);

      if ( DMA_COMPLETED(rx_desc) ) {
          // Lookup routing table and copy from Rx to Tx
          // Rewrite destination MAC address, TTL--,
          // and calculate checksum
          blkcnt++;
          if ( blkcnt >= nr_blk ) {
               blkcnt = 0;
               write_rxq_tail(rxq_tail);
               write_txq_tail(txq_tail);
          }
     } else {
          blkcnt = 0;
          write_rxq_tail(rxq_tail);
          write_txq_tail(txq_tail);
     }
}
```

性能評価



CPU:	Intel(R) Core(TM) i7 4770K (3.90GHz, quad core)
Memory:	32GiB, DDR3-1866
NIC:	Intel(R) X520-DA2 (2 ports)

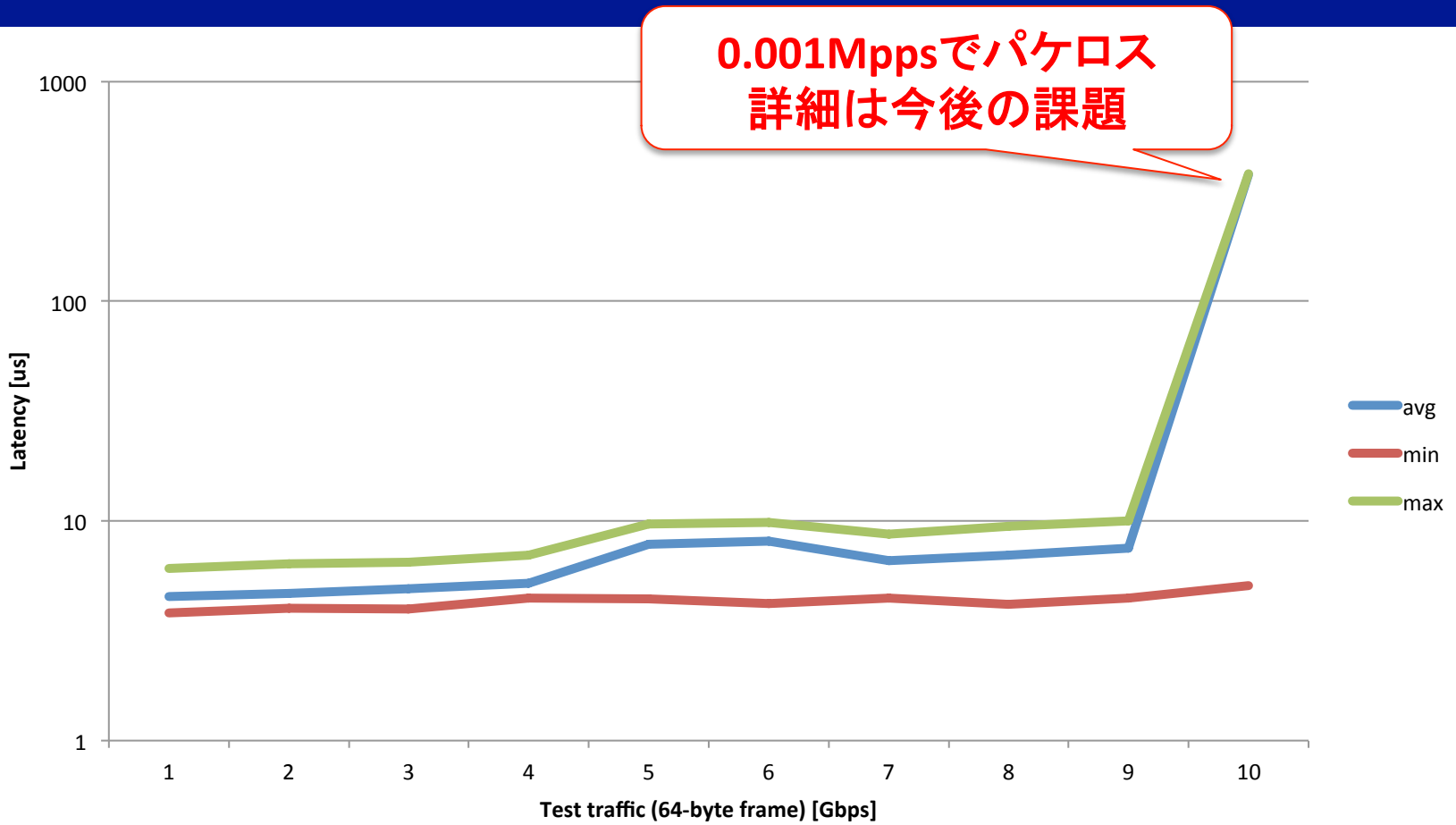
ルーティングの性能評価



遅延計測

- 実験機材
  - テスター
    - Spirent Communications社 Spirent TestCenter
      - 本計測はInterop Tokyo 2014中に
        株式会社東陽テクニカ様のご協力頂き計測しました。
    - 使用型番
      - シャーシ SPT-N4U-110
      - モジュール CV-10G-S8
  - PCルータ
    - CPU: Intel® Core i7 4770K
    - Memory: DDR-3-1866 (8GB x4)
    - NIC: Intel® X520-DA2 (1ポートのみ使用)

Low Latency



ラインレートの90%までは低遅延（~10us）を実現

今後の予定

- Networking Operating System
  - 新しいアーキテクチャの設計
    - I/Oスケジューラ etc.
  - ルーティングプロトコルなど
    ネットワークOSとして必要な機能を搭載
- 40GbE NICへの対応

まとめ

- パケット転送におけるボトルネック
  - Not CPU
    - （少数経路のルーティング程度なら）
  - Not memory
  - PCIe MMIO
- 高速パケット転送用OS
  - 検証済み
    - 10GbEのラインレートルーティング
    - 10GbE x4 の Tx 転送

- Routing
  - 4.897Mpps (59.52Mpps input)
- Tx
  - 17.83Mpps (Write Back)
  - 24.24Mpps (Polling)