Resequencing in Networks with Dynamic Routing

Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst
Motivation

- Dynamic routing can lead to out-of-order delivery
  - routers generally expected to maintain order under normal conditions
  - requires resequencing on output side
Resequencing with Sequence Numbers

- **Drawbacks**
  - each output needs $N$ resequencing arrays
  - initialization when line card comes on-line
  - timeouts needed to cope with lost cells
  - multicast requires per flow resequencing arrays
**Time-Based Resequencing**

- Single resequencing buffer per output
- Cells held until “age” exceeds *threshold* \( (T) \)
- Options for late cells
  - discard (strict resequencing) or buffer (loose)
Henrion’s Strict Resequencer Design

- Implemented using linked lists in common memory
- Constant time per cell
Implementing Loose Resequencing

- Cannot just insert late cells into output list

- Only approximates loose resequencing
  - must still discard “really late” cells
  - low cost of large timing wheel allows good approximation
Performance of Strict Resequencing

- Simple random traffic.
- 3 stage network, 8 port SEs, 512 (shared) cell buffers.
- 1st stage SEs use round robin load balancing for each input.

Late cells rare with small speedup

For systems with 10G links, delay for 256 cells is 10μs.
Performance on Adversarial Traffic

- 2:1 overload at "target" output
- Delay drops as SC buffers drain
- fixed, strict, $T-128$
- Growing network delay
- Cells discarded when delay exceeds $T$
- Resequencer recovers when delay drops below $T$
Performance on Bursty Traffic

- 100% input load
- Input picks an output at random – overload at 1 in 4 outputs
- Stays with target output for geometrically distributed time
Loose Reseq. with Adversarial Traffic

Arriving cells are younger than oldest waiting cells, so no resequencing errors.

- Overload
- Fixed loose, $T=128$
- Age of oldest cell
- Age threshold
- Network delay
- Resequencer occupancy

Time:
- 250
- 500
- 750
- 1000
- 1250
- 1500
- 1750
- 2000
Loose Reseq. with Bursty Traffic

- mean dwell time
- late probability
- speedup = 1.1
- bursty traffic
- loose, fixed, \( T = 128 \)

Tolerates about 3x longer bursts than strict reseq.
Adaptive Resequencing

- Adjust age threshold to match observed delay.
  - window size (W), short term delay diff. bound (Δ)
  - variables: max delay in current measurement window
    (d₀) and previous measurement window (d₋₁)
  - age threshold = Δ + max{d₀, d₋₁}
  - implement by extending loose, fixed threshold design

- Theorem (simplified). If cell c₁ enters network just
  before c₂ and exits no later than Δ after c₂, then an
  adaptive resequencer with W ≥ Δ forwards c₁
  before c₂

- Resequencing errors caused by excessive delay
  variability, rather than large delays
Performance on Adversarial Traffic

- Arriving cells are younger than oldest waiting cells, so no resequencing errors
- Age threshold tracks network delay
- Resequencer adds small increment to network delay
- Resequencer occupancy stays bounded

Age of oldest cell
Network delay
Resequencer occupancy
Receptive, $W=\Delta=32$
Performance on Bursty Traffic

- Performance degrades when switch buffers fill. Caused by delay variation in first stage.
- Less sensitive to extremely large bursts.
- Tolerates about 2x longer bursts than loose reseq.
Boosting Performance for Long Bursts

- Bursty traffic (dwell=100) speedup=1.2
- Adaptive resequencer
- First stage buffer capacity
- Smallest buffer reduces switch throughput by 2%
- Small first stage buffer gives good performance with modest "extra" delay
- Limiting first stage buffering cuts variability
Avoiding Out-of-Order Packets

- Can design dynamic routing network to maintain packet order
- Each SE holds cells with timestamp $t$ until it knows it cannot receive any new cells with timestamp $< t$
  - assume each upstream neighbor always sends cells in timestamp order
  - then, once SE $A$ gets a cell from SE $B$ with timestamp $t$, it will never get another cell from $B$ with timestamp $< t$
  - once $A$ has received cells from all its neighbors with timestamp $t$, it can forward all cells with timestamp $< t$
- SEs send timestamp floor cells on otherwise idle links
  - promise no future cell will have smaller timestamp than floor
  - floor timestamp is min({latest timestamp from input $i$})
Fast-Forwarding the Lag Pointer

- Lag pointer must advance to next non-empty list on every cell time for constant time operation
  - no time to check successive pointers in timing wheel
  - use fast-forward bits to speed-up process

- Three memory reads suffice to find next slot
  - 32 bit words allows range of 1024
  - 128 bit words allows range of 16,384
Synchronization

- Time-based resequencing requires synchronization of all line cards
  - in small routers, requires just a common backplane signal
  - in large routers, line cards connected to network only by optical data cables
- Requires low-level clock synchronization protocol
  - "master" line card issues periodic broadcast synchronization messages
  - network forwards sync messages with constant delay
    - only approximate synchronization is necessary
  - new clock master selected on failure
- Independent line card clocks require adjustments
  - suspend transmission when delaying clock