Crossbar Scheduling

Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst
Performance Objectives

- Random, admissible traffic
  - uncorrelated traffic flows, no persistent overloads
  - possible to forward all traffic with speedup=1
    - that is, length of VOQs remains bounded
- Worst-case traffic
  - allow arbitrary, adversarial input traffic
  - possible to achieve ideal performance with speedup=2
  - work-conservation – no wasted output capacity
    - if there is a packet for output j in system, output j is not idle
  - exact emulation of output-queued switches
    - crossbar forwards packets at same time as ideal output-queued switch with a specified output queueing method
    - can handle any Push-in, First-out queueing method
Newest VOQ First (NVF)

- Each input maintains ordered list of active VOQs
  - VOQ goes to front of list when it becomes active
  - VOQs drop out when they become inactive

Inputs make single request for first VOQ in their list

Outputs issue grants

- NVF is work-conserving with speedup of 2
Work-Conservation Result

- Preliminaries
  - a VOQ $V_{ij}$ precedes another VOQ $V_{ih}$ at input $i$, if $V_{ij}$ comes before $V_{ih}$ in the VOQ list at input $i$
  - let $p_{ij}$ be number of cells in $V_{ij}$ plus number of cells in all VOQs that precede $V_{ij}$
  - let $q_{j}$ be number of cells in queue at output $j$
  - $slack_{ij} = q_{j} - p_{ij}$

- Sketch of proof
  - $slack_{ij}$ never decreases while $V_{ij}$ remains active
  - $slack_{ij} \geq 0$ while $V_{ij}$ is active
  - if output $j$ fails to send a cell when it could then $slack_{ij}$ must be negative
Slack$_{ij}$ Never Decreases

- Idealized switch operation for speedup 2
  - input phase – cells arrive and are placed in VOQs
  - transfer phase – cells move from inputs to outputs
  - output phase – cells sent from outputs
  - second transfer phase

- During each transfer phase
  - if $V_{ij}$ is active, but does not transfer a cell
    - either $q_i$ increases by one or $p_{ij}$ decreases by 1
    - in either case, slack$_{ij}$ increases by >1

- Input phase causes $p_{ij}$ to increase by at most 1
- Output phase causes $q_i$ to decrease by at most 1
- So, over full cycle, slack$_{ij}$ cannot decrease
NVF is Work-Conserving

- $\text{slack}_{ij} \geq 0$ before each output phase
  - right after $V_{ij}$ becomes active, $p_{ij}=1$, so $\text{slack}_{ij} \geq -1$
  - next transfer phase increases $\text{slack}_{ij}$ by 1, so $\text{slack}_{ij} \geq 0$
  - subsequent cycles can lead to no net decrease in $\text{slack}_{ij}$ so long as $V_{ij}$ remains active

- No wasted output phases
  - suppose at some output phase, output $j$ fails to send a cell, even though $V_{ij}$ contains a cell
  - since output $j$ does not send a cell, $q_j=0$
  - since $V_{ij}$ contains a cell, $p_{ij}>0$
  - hence, $\text{slack}_{ij}=q_j-p_{ij}<0$, yielding a contradiction

- So, NVF is work-conserving for speedup of 2
Least Occupied Output First (LOOFA)

- Each input maintains ordered list of active VOQs
  - VOQs ordered by output-side queue lengths
  - $V_i$ can pass $V_j$ if $q_i$ becomes strictly smaller than $q_j$

- LOOFA is work-conserving with speedup of 2
- Trickier to prove, due to changing VOQ order

Unmatched inputs and outputs repeat until no new matches can be added.
Work-Conservation for LOOFA

- Define \( \text{minSlack}_i = \min_j \text{slack}_{ij} \)

- Outline of proof
  - if \( V_{ij} \) not passed during xfer phase \( \text{slack}_{ij} \) increases by \( \geq 1 \)
  - if \( V_{ij} \) is passed by VOQs containing \( k \) cells,
    - \( \text{slack}_{ij} \) decreases by at most \( (k - 1) \)
    - and before xfer, \( \text{slack}_{ij} \geq \text{minSlack}_i + k \),
      since last VOQ \( V_{ih} \) to pass \( V_{ij} \) has \( q_h = q_i \) and \( p_n = p_i + k \)
      so, \( \text{minSlack}_i \leq \text{slack}_{ijn} = q_n - p_n \leq q_i - (p_i + k) = \text{slack}_{ij} - k \)
  - so, \( \text{minSlack}_i \) increases by at least 1 during each transfer
  - no VOQ can pass another during an input or output phase
  - \( \text{minSlack}_i \), never decreases during a busy period at input \( i \)
  - \( \text{minSlack}_i \geq 0 \) before each output phase (so, \( \text{slack}_{ij} \geq 0 \) also)
  - so, no wasted output phases
LOOFA vs. NVF

- For speedup of 2, both are work-conserving
- Any reason to prefer one over the other?
- For smaller speedup, LOOFA has better fairness
  - Example for speedup 1.2

  100 cells arrive at $A$, $B$ for $Y$
  40 cell backlog, $Y$ has 20 cells

- In NVF, $A$ favors $X$, $B$ favors $Z$, so after $Y$’s backlog clears, its output rate drops to 40%; $X$, $Z$ at 100%
- LOOFA allows all to forward at 80%
Buffered Crossbars

- Crosspoint buffers
de-couple inputs, outputs
  » so, no centralized scheduler is needed
  » simple, distributed control
  » buffers add cost to datapath
- Simple schedulers that are work-conserving
  » variants of NVF, LOOFA and others
  » also leads to simpler proofs
- Can switch variable length packets directly
  » no segmentation & reassembly, no fragmentation losses
  » requires more complex proofs to establish guarantees