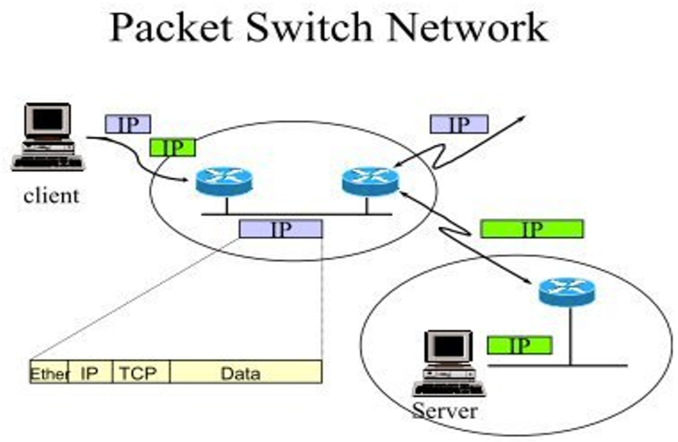
Simulating Virtual Output Queues

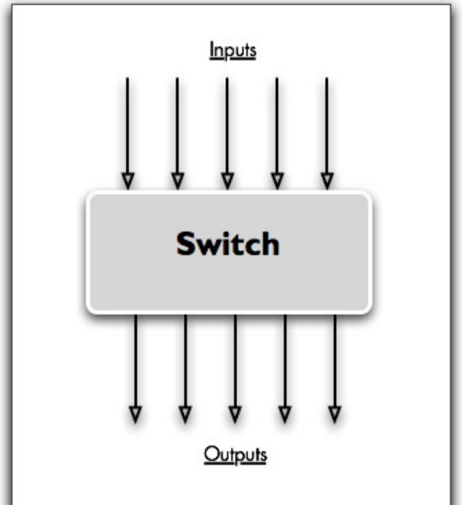
Speaker: Wenkai Dai (CT Group)

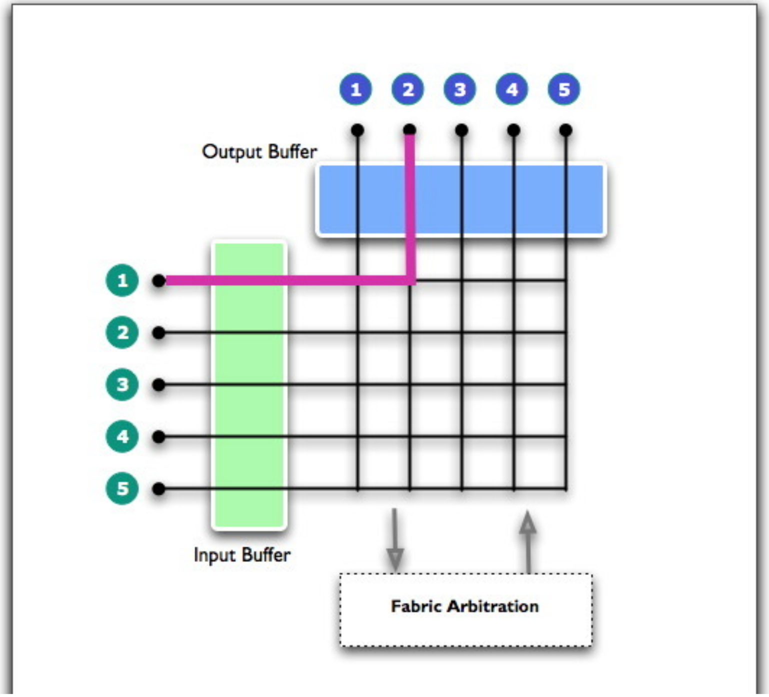
universität wien

Motivation

-







Switches Forward Packets

- **Infrastructure of Switch**:
  - input links: $(\boldsymbol{I_1}, \cdots \boldsymbol{I_n})$
  - output links: $(\boldsymbol{O_1}, \cdots \boldsymbol{O_n})$
  - Switch fabric: copying packets from input links to output links
- **Header of Packet** p: $(\boldsymbol{I}[\boldsymbol{p}], \boldsymbol{O}[\boldsymbol{p}])$ indicates input and output links respectively
- **Forwarding Steps**:
  1. a packet $p = (I[p], O[p])$ arrives at an input link $I[p]$
  2. moving p to the output link $O[p]$
  3. p leaves the output link $O[p]$
- Each input/output link can only have one packet processed at a time step

universität wien

Model 1: Virtual Output Queueing (VOQ)

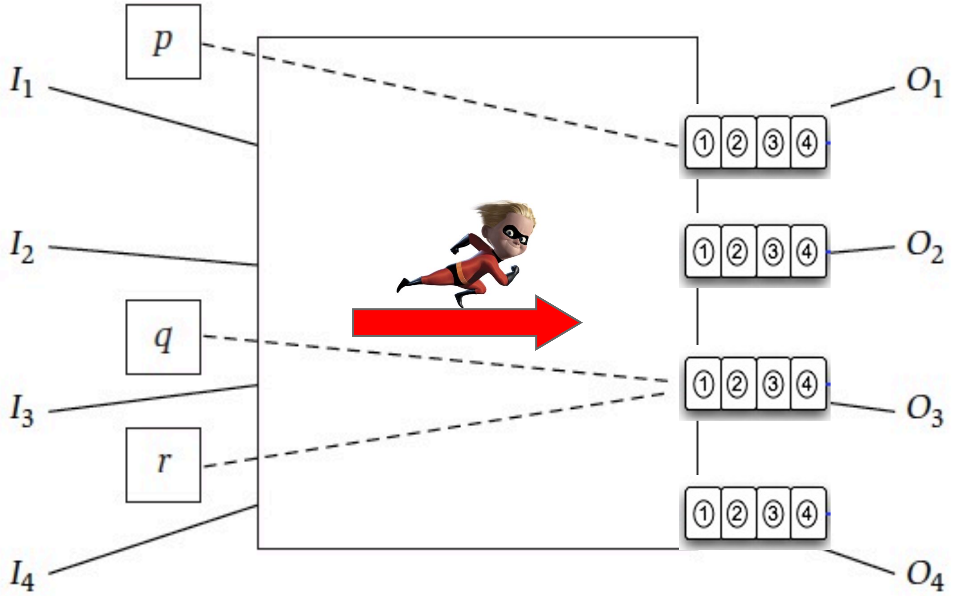
- Each output link contains a buffering queue
- No time cost for switching fabric
- Only one packet can leave at each output link at each time step

One step under pure output queueing:

Packets arrive on input links

Each packet p of type $(I[p], O[p])$ is moved to output buffer $O[p]$

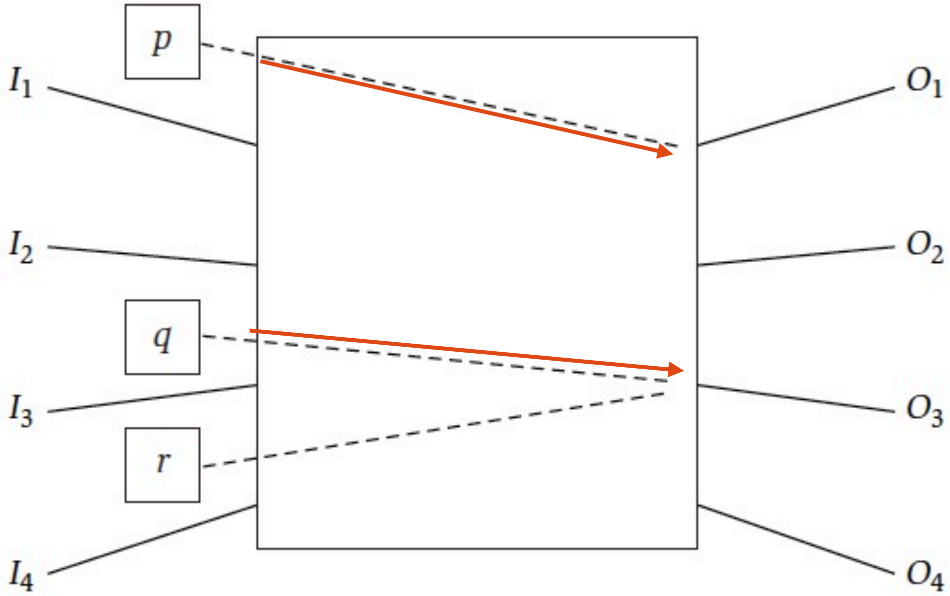
At most one packet departs from each output buffer



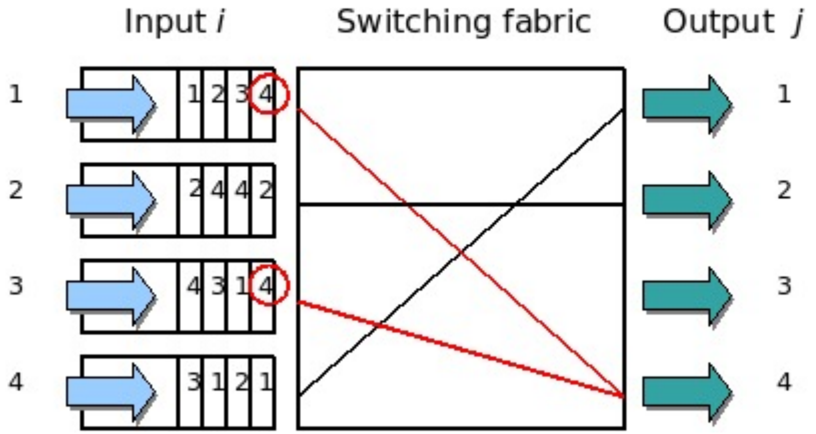
Model 2: Input/Output Queueing

- Each input/output link contains a queue
- Switching fabric need a time step to move packets
- Only one packet can leave at each output link at each time step

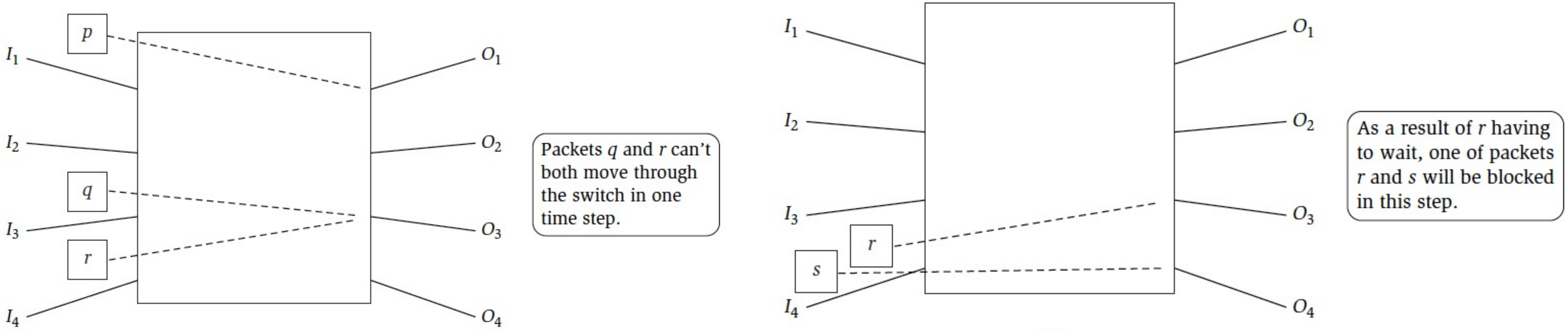
```
One step under input/output queueing:
  Packets arrive on input links and are placed in input buffers
  A set of packets whose types form a matching are moved to their
    associated output buffers
  At most one packet departs from each output buffer
```



Head-of-Line Blocking



Epilogue: Algorithms That Run Forever



(a) (b)

Simulate Virtual Queueing By Input/Output Queueing

- Speed-up switching fabric by factor of two is enough

```
One step under sped-up input/output queueing:
  Packets arrive on input links and are placed in input buffers
  A set of packets whose types form a matching are moved to their
    associated output buffers
  At most one packet departs from each output buffer
  A set of packets whose types form a matching are moved to their
    associated output buffers
```

Preliminaries of Proof

- $TL(p)$: the leave time of p under VOQ model (deadline)
- Output queues: unordered and a packet can leave at any place
- Input queues: ordered by deadlines, but arbitrarily insert and move to the head
- Input cushion $IC(p)$: the number of packets before p in its input queue
- Output cushion $OC(p)$: packets with earlier deadline in the output queue of p
- Define $Slack(p) = OC(p) - IC(p)$ (indicate the slackness, and larger is better)

Invariants To Be Guaranteed

(i) $Slack(p) \geq 0$ for all unprocessed packets p.

(ii) In any step that begins with $IC(p) = OC(p) = 0$, packet p will be moved to its output buffer in the first matching.

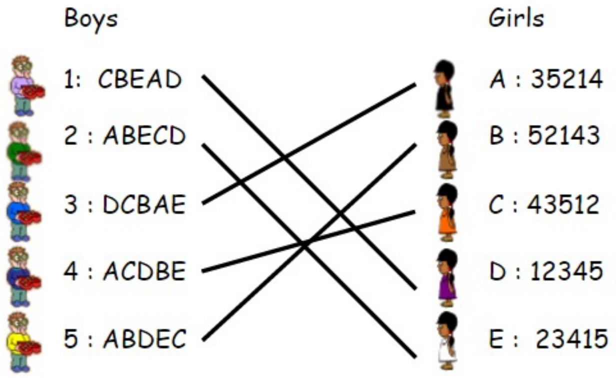
(E.1) *If properties (i) and (ii) are maintained for all unprocessed packets at all times, then every packet p will depart at its time to leave $TL(p)$.*

Proof. If p is in its output buffer at the start of step $TL(p)$, then it can clearly depart. Otherwise it must be in its input buffer. In this case, we have $OC(p) = 0$ at the start of the step. By property (i), we have $Slack(p) = OC(p) - IC(p) \geq 0$, and hence $IC(p) = 0$. It then follows from property (ii) that p will be moved to the output buffer in the first matching of this step, and hence will depart in this step as well. ■

Initialization of Invariant

Moving a Matching through a Switch When a packet p first arrives on an input link, we insert it as far back in the input buffer as possible (potentially somewhere in the middle) consistent with the requirement $Slack(p) \geq 0$. This makes sure property (i) is satisfied initially for p.

Next to Show $Slack(p) = OC(p) - IC(p)$ Increasing



- Note only when $OC(p)$ (decrease) or $IC(p)$ (increase) then $Slack(p)$ decrease
- A packet arrive/leave might decrease $Slack(p)$ by one (in total two)
- Try to increase $Slack(p)$ by one for moving each matching
- Select stable matchings
  - Preference of output O: the input I where the packet has the earliest deadline
  - Preference of input I: the packet to O is the forwardmost than other output O’
  - Stability: no other matching can satisfy the preference better

Moving A Matchings Ensures $Slack(p)$ Increasing by 1

(E.2) *Suppose the switch always moves a stable matching M with respect to the preference lists defined above. (And for each type (I, O) contained in M, we select the packet of this type with the earliest time to leave). Then, for all unprocessed packets p, the value $Slack(p)$ increases by at least 1 when the matching M is moved.*

Proof. Consider any unprocessed packet p. Following the discussion above, suppose that no packet ahead of p in $I[p]$ is moved as part of the matching M, and no packet destined for $O[p]$ with an earlier time to leave is moved as part of M. So, in particular, the pair $(I[p], O[p])$ is not in M; suppose that pairs $(I', O[p])$ and $(I[p], O')$ belong to M.

Now p has an earlier time to leave than any packet of type $(I', O[p])$, and it comes ahead of every packet of type $(I[p], O')$ in the ordering of $I[p]$. It follows that $I[p]$ prefers $O[p]$ to O', and $O[p]$ prefers $I[p]$ to I'. Hence the pair $(I[p], O[p])$ forms an instability, which contradicts our assumption that M is stable. ∎