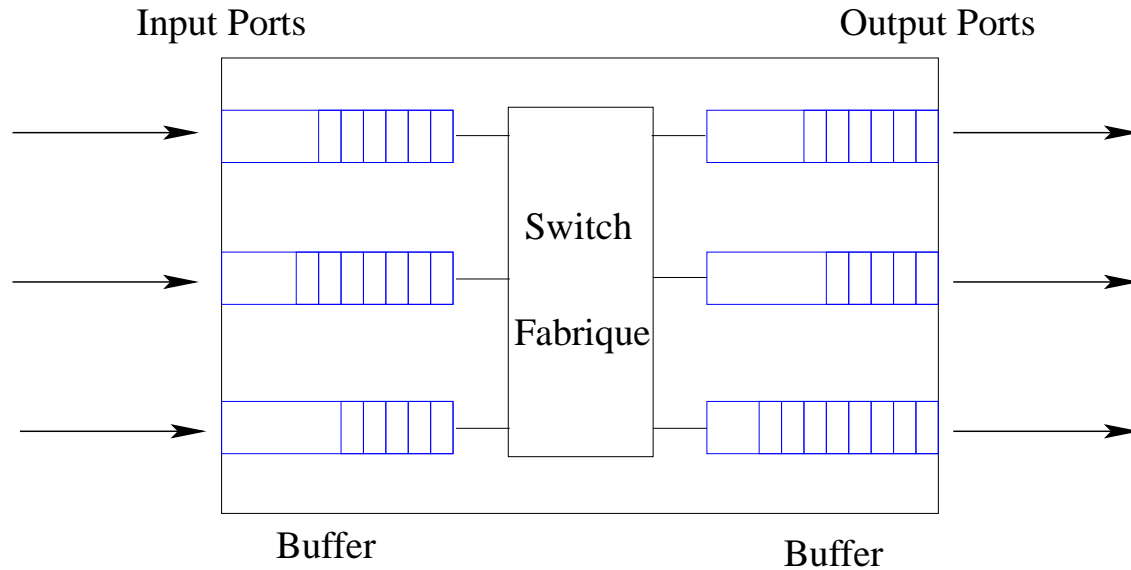


Buffer Management in Network Switches

Susanne Albers

Switches

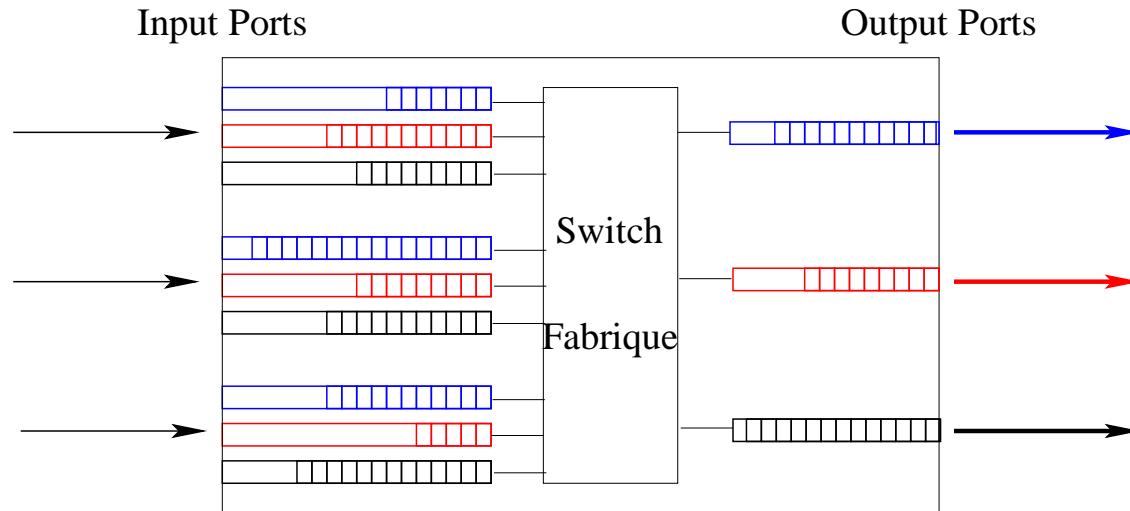


Switches forward data packets.

Buffers store packets temporarily if capacity available.

Goal: maximize **throughput**.

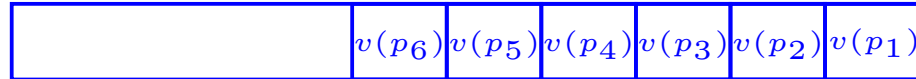
Virtual output queueing



Each input port i maintains for each output port j a queue Q_{ij} .

Each output port j maintains a queue Q'_j .

Single-buffer problem



1 buffer that can store B packets.

Packet p has value $v(p)$

In each time step

– new packets arrive online

P : set of packets in buffer

N : set of new packets

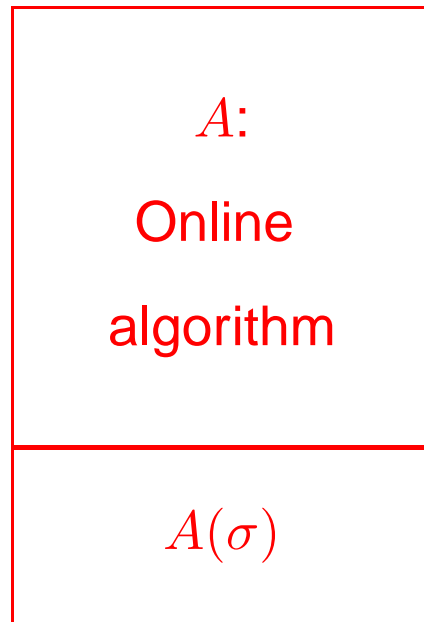
$|N| + |P| - B$ packet must be dropped if overflow

– one packet can be sent to output

Goal: maximize total value of transferred packets

Competitive analysis

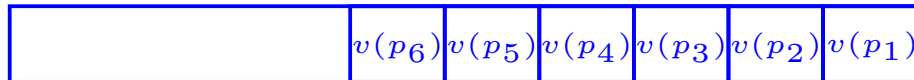
Online problem



A is c -competitive if $\exists b$ such that for all sequences σ

$$A(\sigma) \geq \frac{1}{c} \cdot OPT(\sigma) - b.$$

Settings



FIFO model: Packet from the buffer must be transmitted in the order of arrival.

If p arrives later than p' it must not be transmitted earlier.

Preemptive model: Packets from the buffer may be dropped.

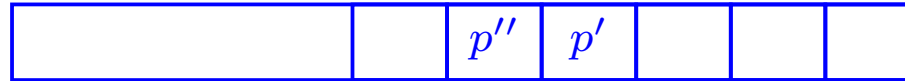
Preemptive FIFO model

Algorithm Greedy: Drop packet with smallest values in event of overflow.

Thm: Greedy is 2-competitive.

Kesselman et al. STOC 2001

Preemptive FIFO model



$$v(p'') > v(p')$$
$$v(p)/\beta > v(p')$$

Algorithm β -Preemptive Greedy: Packet p of value $v(p)$ arrives.

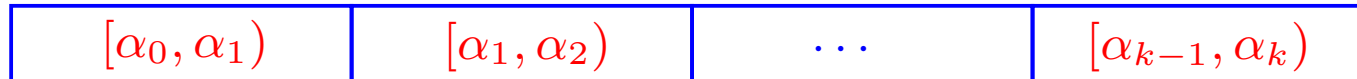
- (1) Find first packet p' in FIFO order whose value is **less than $v(p)/\beta$** and less than that of **following packet** in the buffer. If p' exists, preempt it.
- (2) Accept p if space available. Otherwise evict the available packet with smallest value.

Thm: For $\beta = 4$, β -Preemptive Greedy is **1.75-competitive**.

Thm: Any deterministic online alg. has competitive ratio of **at least 1.419**.

Bansal et al. 2004; Kesselman et al. 2003

Non-preemptive model



Algorithm Exponential Interval Round Robin: Packet values in $[1, \alpha]$

- Divide buffer into k **partitions** of size B/k , where $k = \lceil \ln \alpha \rceil$.
- Split $[1, \alpha]$ into $[\alpha_0, \alpha_1), [\alpha_1, \alpha_2), \dots, [\alpha_{k-1}, \alpha_k)$, where $\alpha_j = \alpha^{j/k}$.
- Each partition is associated with one of the subintervals, accepting packets from that subinterval if space permits.
- Partitions **take turn** in sending packets.

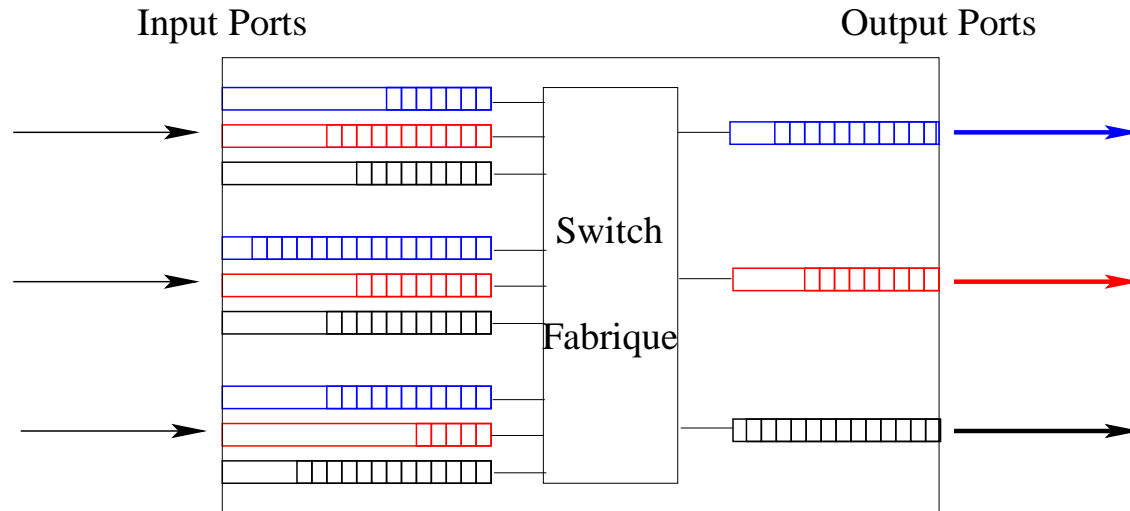
Thm: Exponential Interval Round Robin is $e^{\lceil \ln \alpha \rceil}$ -competitive.

Thm: Any deterministic online alg. has competitive ratio of **at least** $2 + \ln \alpha$

Best upper bound: $2 + \ln \alpha + O(\ln^2 \alpha / B)$

Andelman, Mansour, Zhu 2003; Andelman, Mansour 2003

Virtual output queueing



Each input port i maintains for each output port j a queue Q_{ij} .

Each output port j maintains a queue Q'_j .

Multi-buffer problem

m buffers, each of which can store
 B packets p of value $v(p)$.

In each time step

– new packets arrive online

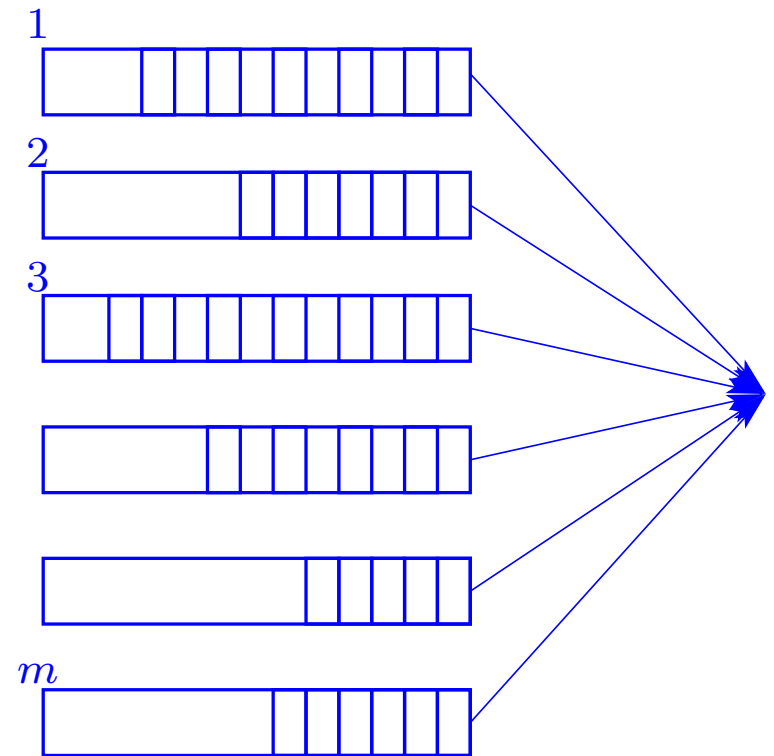
P_i : set of packets in buffer i

N_i : set of new packets at buffer i

packet loss: $\max\{|N_i| + |P_i| - B, 0\}$

– one buffer can send **one packet**
to the output

Goal: max. total value of transferred packets



Arbitrary packet values

Algorithm Transmit-Largest (TL): (preemptive non-FIFO)

1. **Admission control: Greedy-approach.** At any buffer i enqueue packets if space available. In case of overflow, drop packets with the smallest values.
2. **Transmission:** In each time step transmit packet with the **largest value** in the m queues.

Azar, Richter 2005

Arbitrary packet values

Algorithm Generic Switch:

1. **Admission control:** Apply admission control strategy ALG to any of the m buffers.
2. **Transmission:** Run simulation of TL (in the preemptive non-FIFO model) with online packet arrival sequence σ . In each time step transmit packet from head of the queue served by TL .

Thm: If ALG c -competitive, then *Generic Switch* is $2c$ -competitive.

Cor: [Preemptive FIFO] *Greedy* 4-competitive; 3.5-competitive alg.

Cor: [Non-preemptive] $2e^{\lceil \ln \alpha \rceil}$ -competitive alg.

Unit-value packets

m buffers, each of which can store B packets.

In each time step

– new packets arrive online

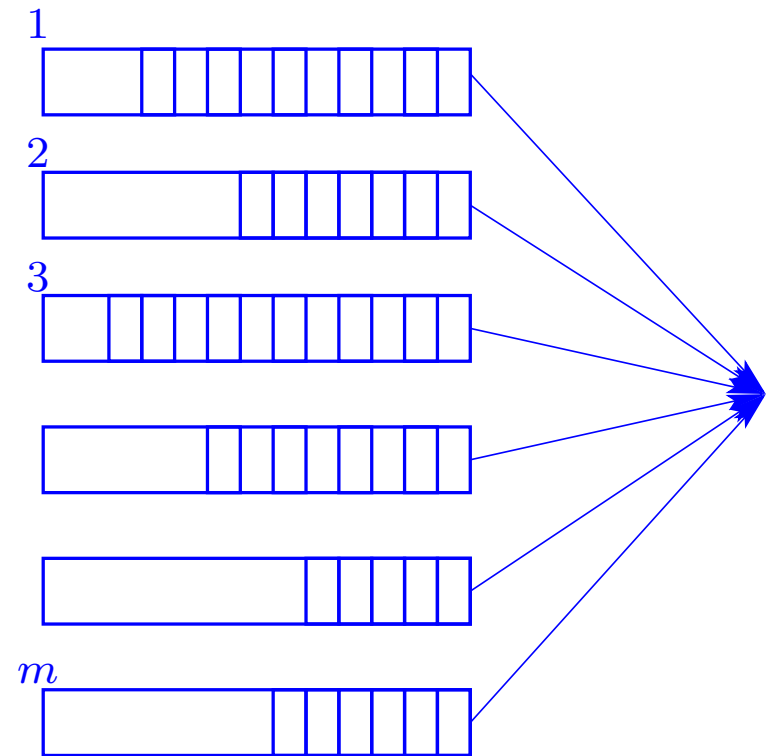
P_i : #packets in buffer i

N_i : #new packets at buffer i

packet loss: $\max\{N_i + P_i - B, 0\}$

– one buffer can send **one packet**
to the output

Goal: maximize #transferred packets



Known algorithmic results

Work conserving strategies are 2-competitive.

Greedy strategies are exactly 2-competitive.

Albers, Schmidt 04

Deterministic algorithms

HOPT: 2

Albers, Jacobs 07

Semi Greedy: $17/9 \approx 1.89$

Albers, Schmidt 04

Waterlevel: $\frac{e}{e-1} \left(1 + \frac{\lfloor H_{m+1} \rfloor}{B}\right)$

Azar, Litichevsky 04

Lower bound: $e/(e-1) \approx 1.58$

Albers, Schmid 04

Randomized algorithms

Random Schedule: $e/(e-1)$

Azar, Richter 03

Random Permutation: 1.5

Schmidt 05

Lower bound: 1.46

Albers, Schmidt 04

Offline problem polynomially solvable.

Work conserving alg. A

Partition σ into phases

P_1 starts at beginning of σ

P_i starts at the end of P_{i-1}

P_i ends when, for the first time, A 's buffers are empty and new packets arrive in next time step.

Phase P_i

– $T_i = \#$ time steps during which A transmit packets

– $N_i = \#$ packets in OPT 's config. at the end of time step T_i

$$N_i \leq T_i$$

A transmits T_i packets

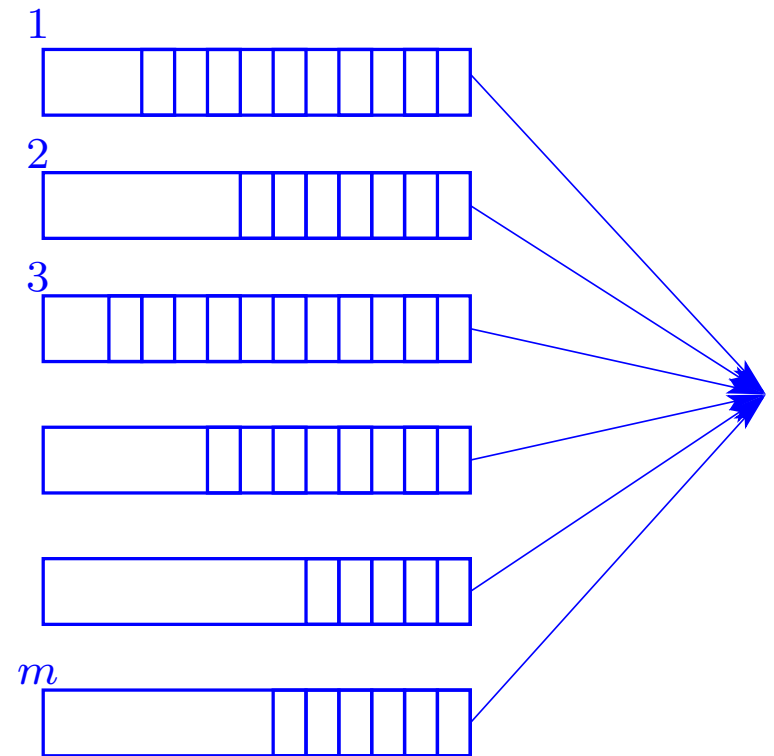
OPT transmits $T_i + N_i \leq 2T_i$ packets

Greedy algorithms

Greedy: Always serve a buffer currently storing a maximum number of packets.

Advantages:

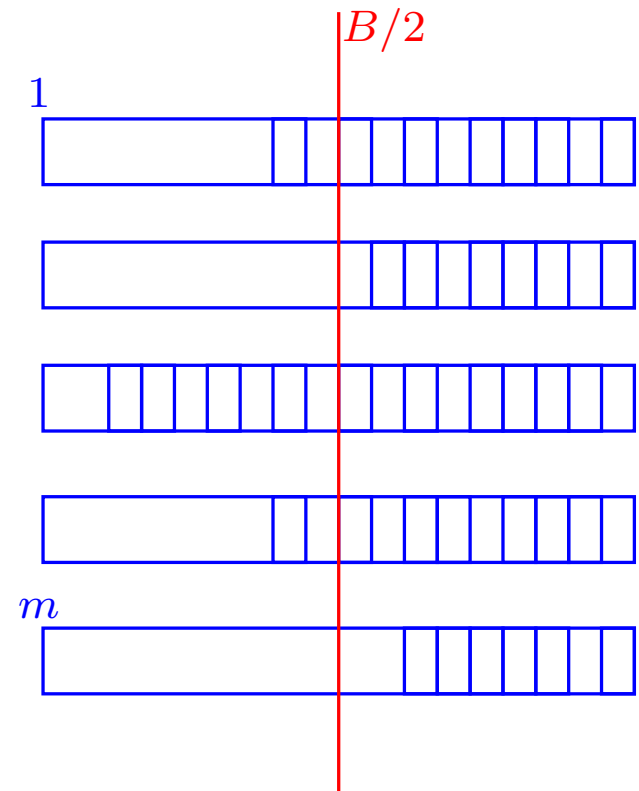
- fast
- little extra memory
- natural strategy to avoid packet loss



Semi-Greedy

In each time step execute the first applicable rule.

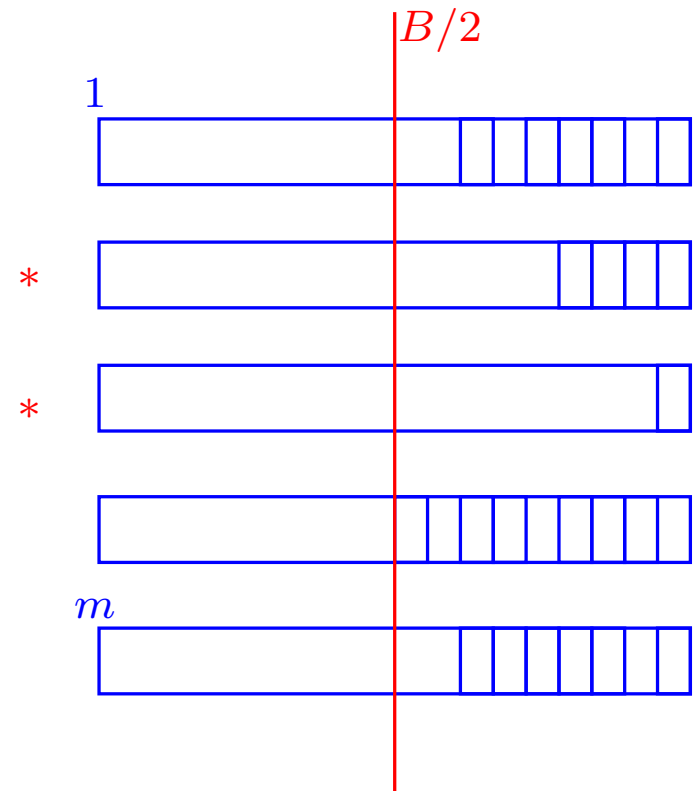
1. \exists buffer with $> B/2$ packets
→ serve a buffer with max. number of packets
2. \exists non-empty buffer that has never been full
→ amongst these, serve one with max. number of packets
3. Serve a buffer with max. number of packets



Semi-Greedy

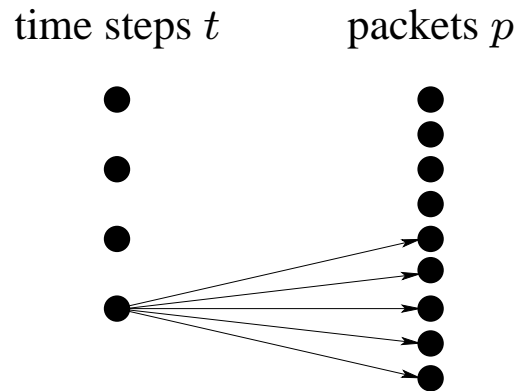
In each time step execute the first applicable rule.

1. \exists buffer with $> B/2$ packets
→ serve a buffer with max. number of packets
2. \exists non-empty buffer that has never been full
→ amongst these, serve one with max. number of packets
3. Serve a buffer with max. number of packets



Waterlevel

Fractional bipartite matching



Connect t to $p \in P_t$ $P_t = \bigcup_{i=1}^m \{\text{last } B \text{ packets at port } i\}$

s_t^p = total extent to which p has been served so far.

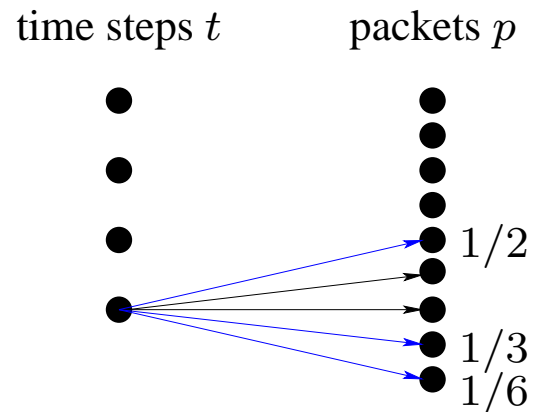
Fractional Waterlevel: $\forall t \forall p \in P_t$ match extent of

$$x_t^p = \max\{h - s_t^p, 0\}$$

h is maximum value s.t. $\sum_{p \in P_t} x_t^p \leq 1$.

Waterlevel

Fractional bipartite matching



Connect t to $p \in P_t$ $P_t = \bigcup_{i=1}^m \{\text{last } B \text{ packets at buffer } i\}$

s_t^p = total extent to which p has been served so far.

Fractional Waterlevel: $\forall t \forall p \in P_t$ match extent of

$$x_t^p = \max\{h - s_t^p, 0\}$$

h is maximum value s.t. $\sum_{p \in P_t} x_t^p \leq 1$.

Discretization

D(FWL): Buffers of size $B + 1 + \lfloor H_m \rfloor$

- S_i = total volume of packets transmitted from buffer i by Fractional Waterlevel up to (and including) time t
 S'_i = # packets transmitted from buffer i
Serve buffer i for which $S_i - S'_i$ is largest.

Waterlevel: At each time step:

- Serve same buffer as D(FWL) if it is non-empty.

HOPT

Algorithm HOPT: Simulate OPT by estimating future packet arrival rates.

$\alpha \in [0, 1]$

$$R_i(t) = \alpha R_i(t-1) + (1-\alpha)N_i(t)$$

$$T_i^{\text{ov}}(t) = (B - P_i(t))/R_i(t)$$

Serve buffer with smallest T_i^{ov} .

Experiments: $\alpha \approx 0.9$ is best.

Random Schedule

Assign each packet a **random priority from $[0,1]$** . Always serve the queue containing the largest priority.

Random Permutation

Reduce problem to one of maintaining **mB unit-size queues**. Fix a random permutation Π on these. Serve non-empty queue occurring first in Π .

Goals experimental study

- Evaluate **experimentally observed competitiveness**
- Establish **relative performance ranking** among algorithms
- Determine **running time**: average time to determine queue to be served (total time / # time steps)
- Determine **extra memory requirements**

Test environment

Traces from Internet Traffic Archive (ACM SIGCOMM)

Name	Date	# Packets	Place
DEC-PKT-1	08.03.1995 22.00–23.00	2.1 mio	DEC
DEC-PKT-2	09.03.1995 02.00–03.00	2.6 mio	DEC
DEC-PKT-3	09.03.1995 10.00–11.00	2.8 mio	DEC
DEC-PKT-4	08.03.1995 14.00–15.00	3.8 mio	DEC
LBL-PKT-4	21.01.1994 14.00–15.00	1.3 mio	LBL
LBL-PKT-5	28.01.1994 14.00–15.00	1.3 mio	LBL
LBL-TCP-3	20.01.1994 14.10–16.10	1.8 mio	LBL

Test environment

- Varying m

- Varying B

- Varying switch speeds s

$$f_T = (\# \text{ packets in } T) / (\text{length time horizon of } T)$$

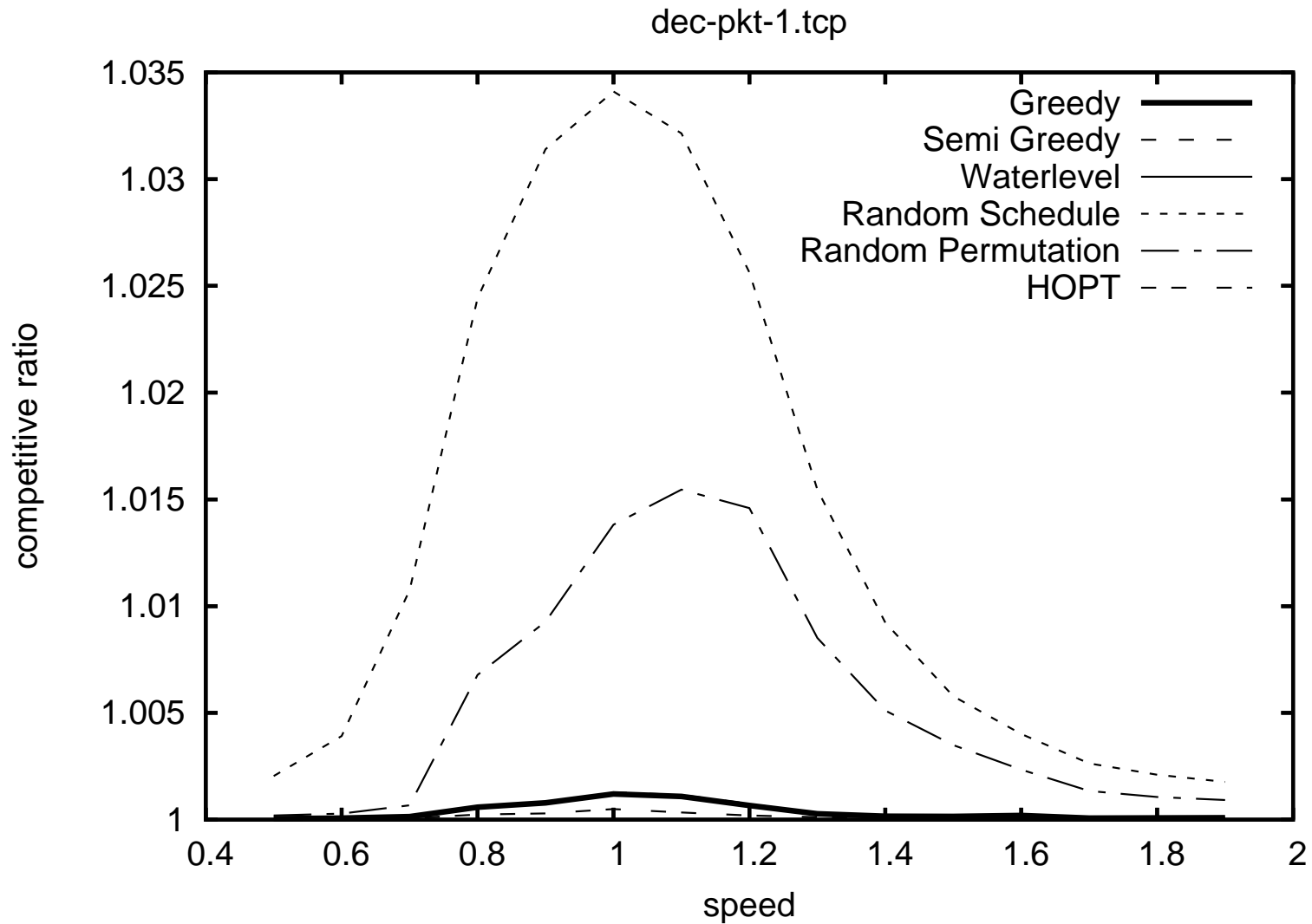
Speed s : Switch can forward $s f_T$ packets per time unit

Results

- **Results consistent** for all the traces
- This talk: DEC-PKT-1 (with 2.1 million packets)
- Basic setting: $m = 30$ $B = 100$

Competitiveness

$m = 30$ $B = 100$



Competitiveness

Greedy, Semi Greedy, Waterlevel

have “identical performance”

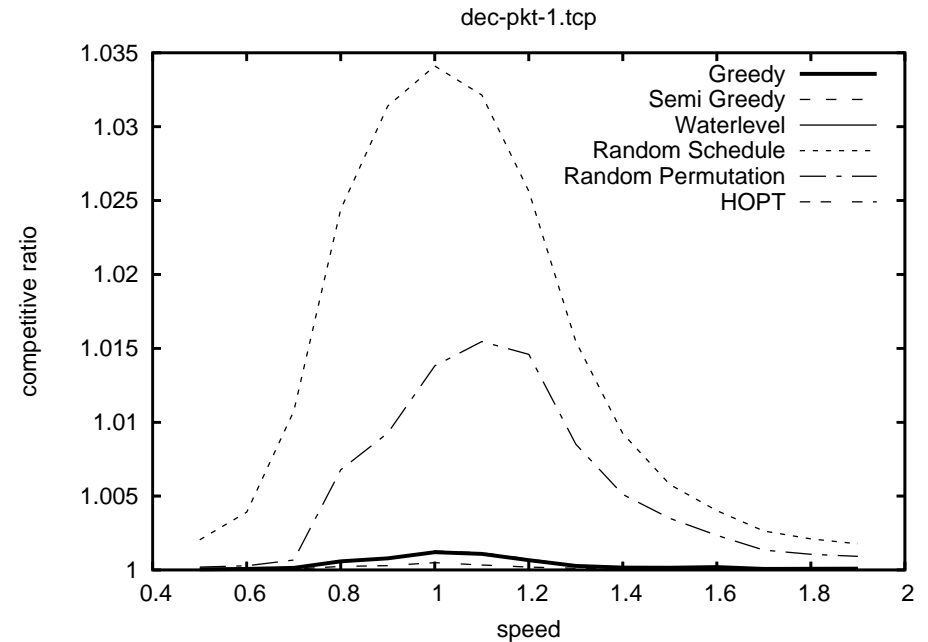
Competitiveness below 1.002

HOPT has best performance

Competitiveness below 1.001

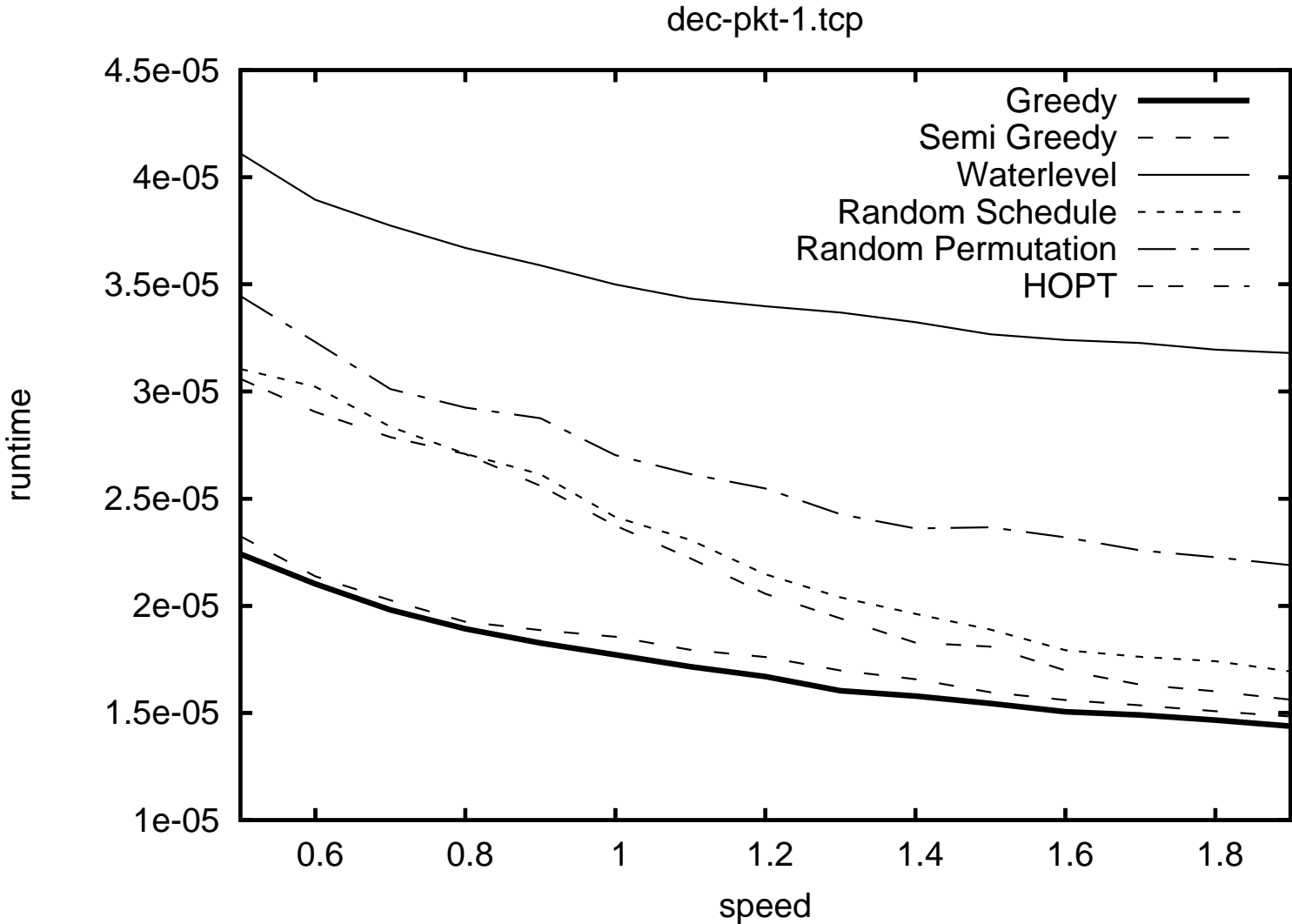
Random Schedule is worst.

Worst ratios around $s = 1.0$



Running times

$m = 30$ $B = 100$

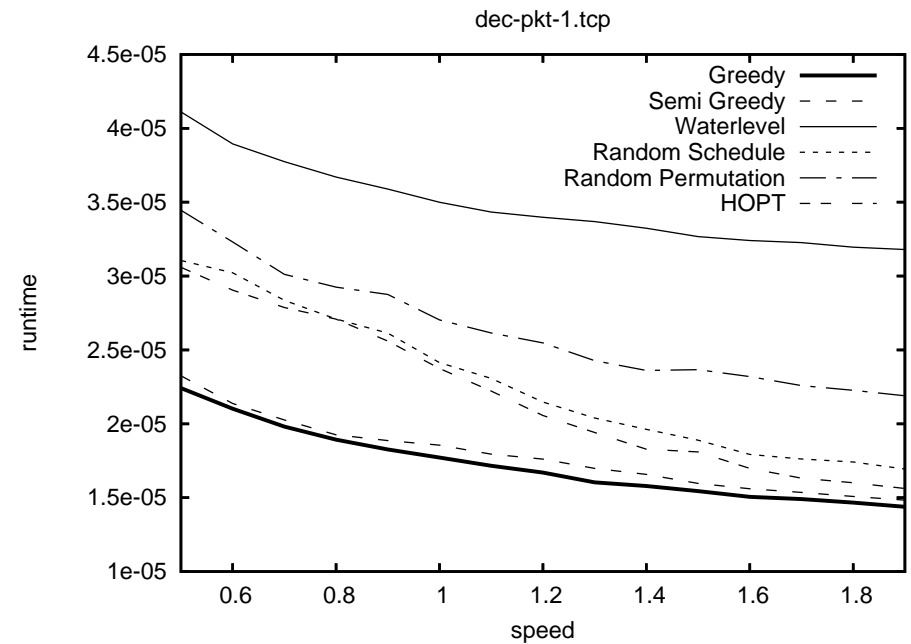


Running times

20–40 milliseconds for one time step

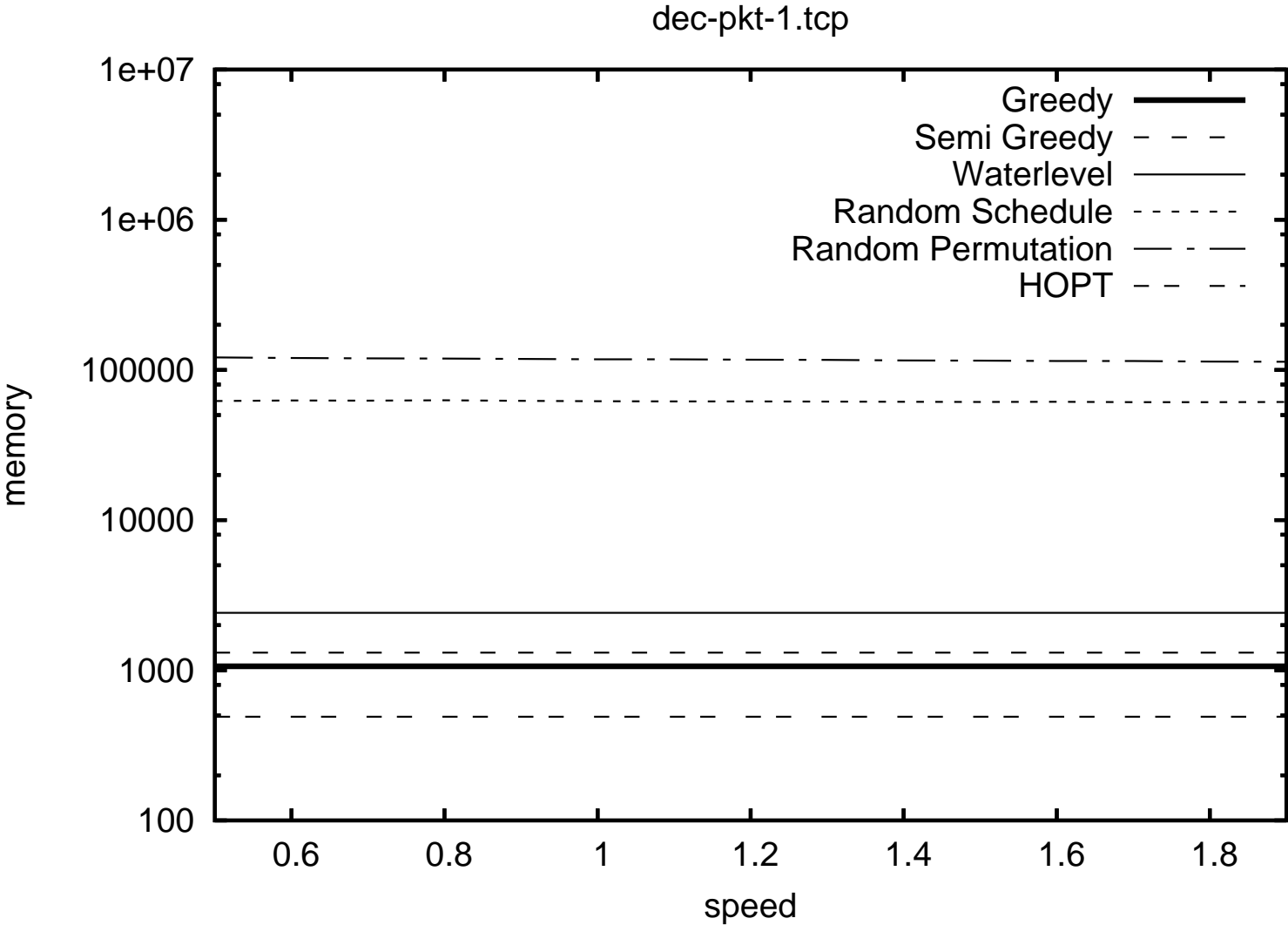
Times decreasing for increasing s
as buffers tend to be empty

Greedy, Semi Greedy are fastest



Extra memory

$m = 30$ $B = 100$



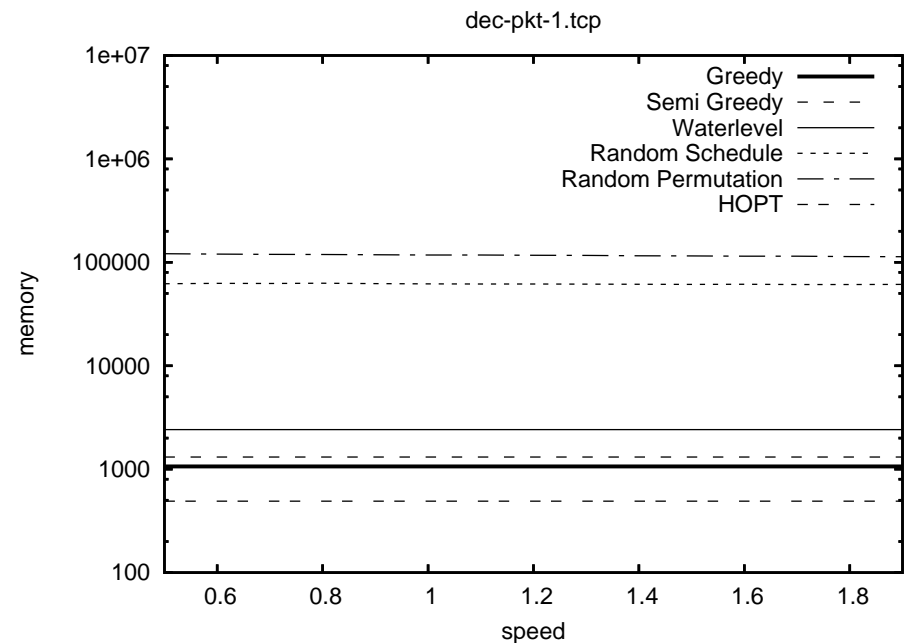
Extra memory

Logarithmic scale

HOPT, Greedy, Semi Greedy use
little space

Waterlevel double amount

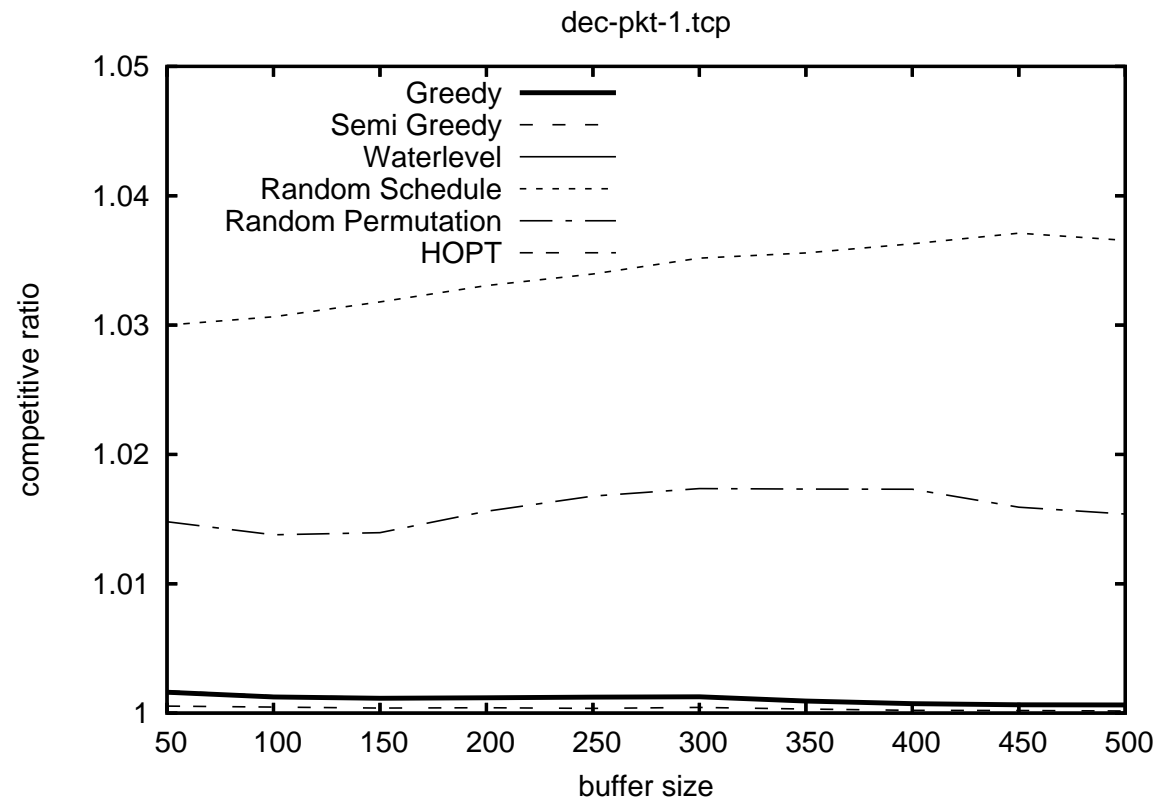
Randomized algs. huge requirements



Varying buffer size

$s = 1.0$

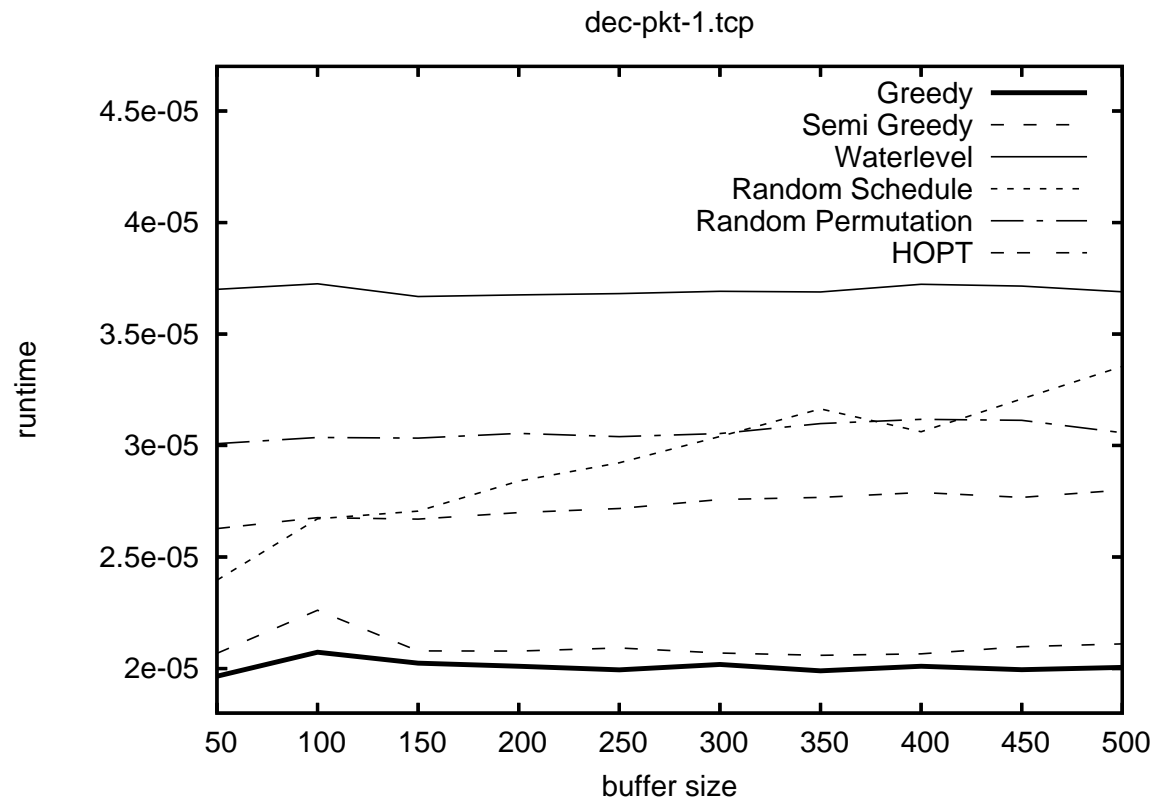
Competitiveness is stable



Varying buffer size

Running times

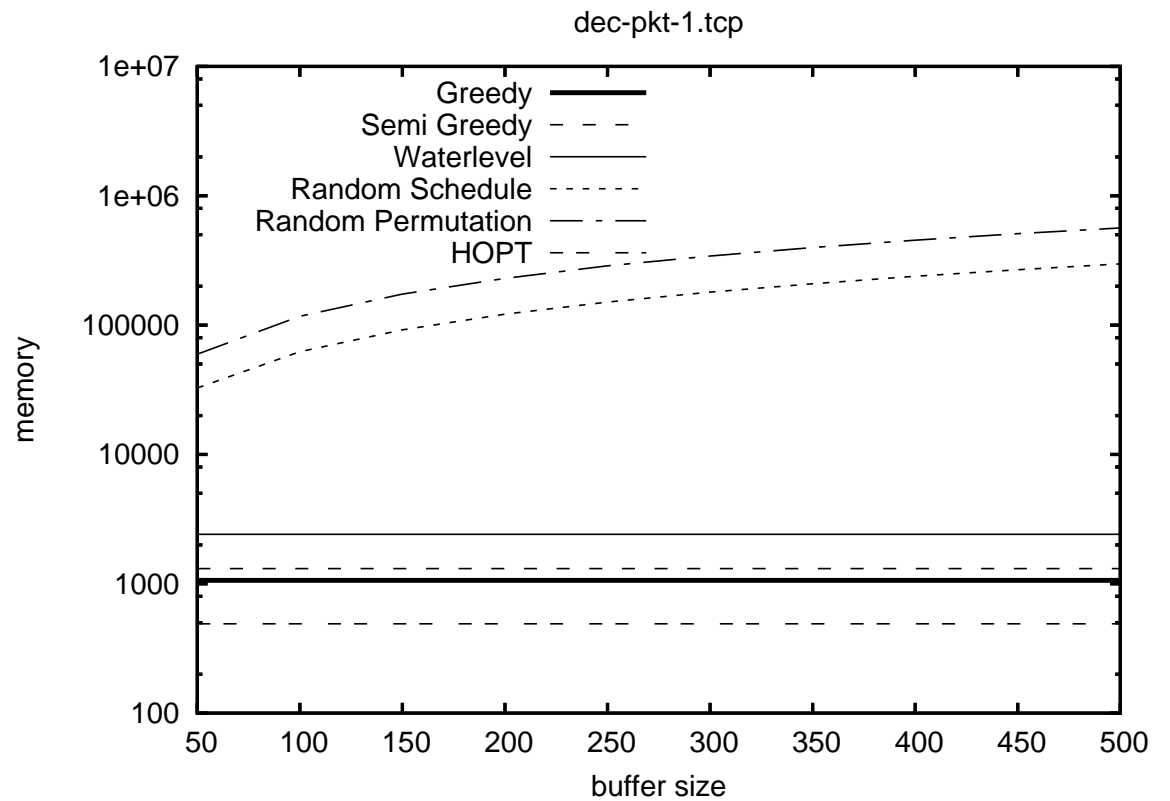
Only **Random Schedule** has increasing times



Varying buffer size

Memory requirements

Randomized algorithms experience linear increase (demand depends linearly on mB).

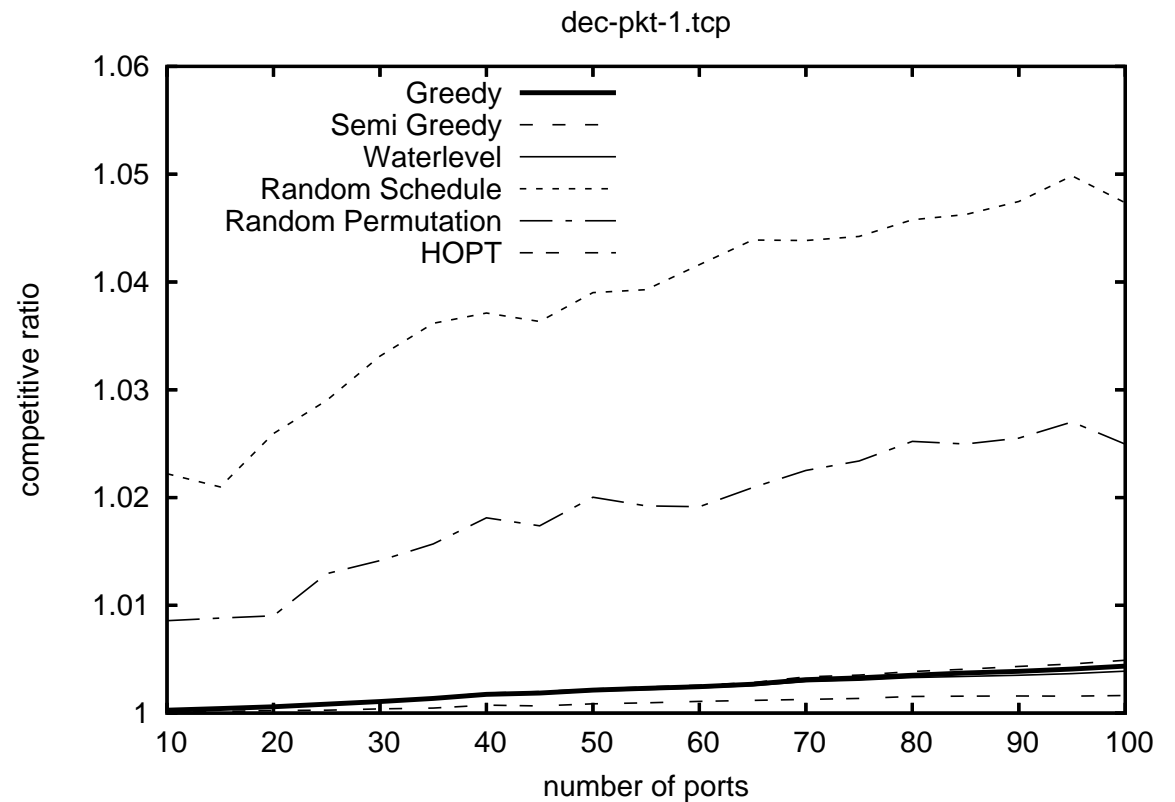


Varying the number of ports

$s = 1.0$

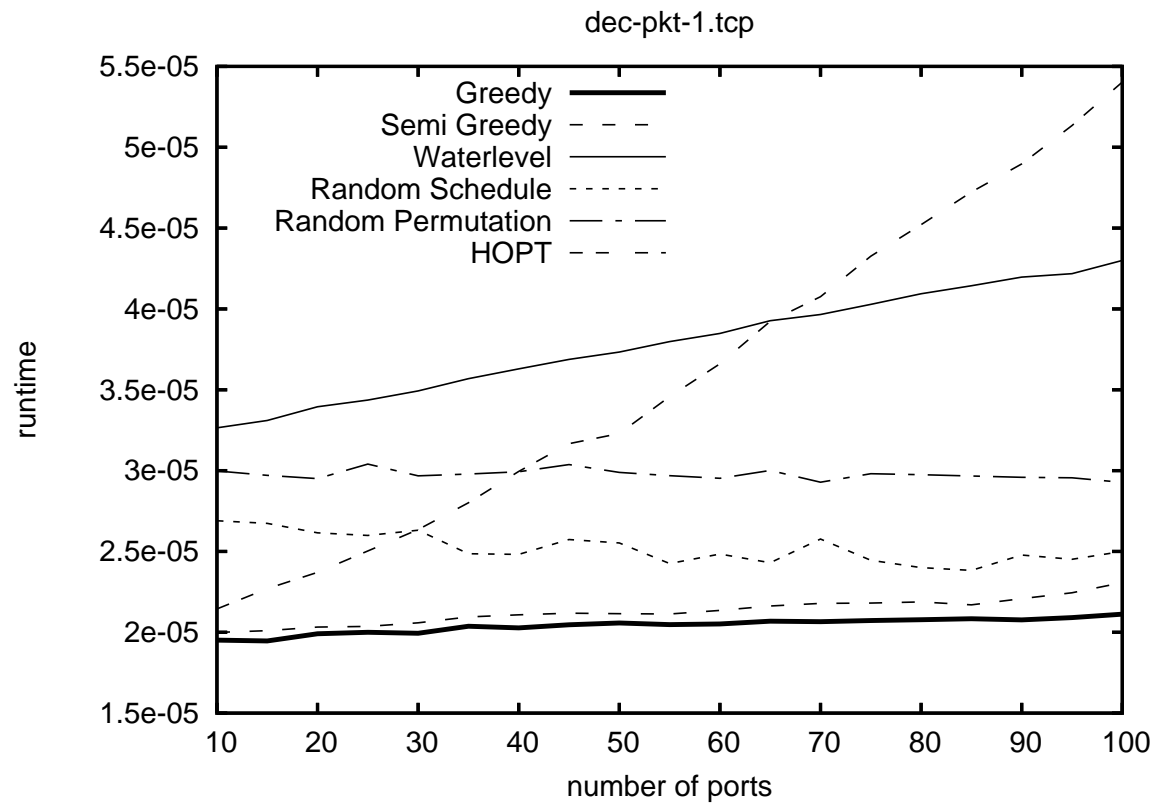
Competitiveness slightly
increasing

Increase more pronounced for
randomized algs.



Varying the number of ports

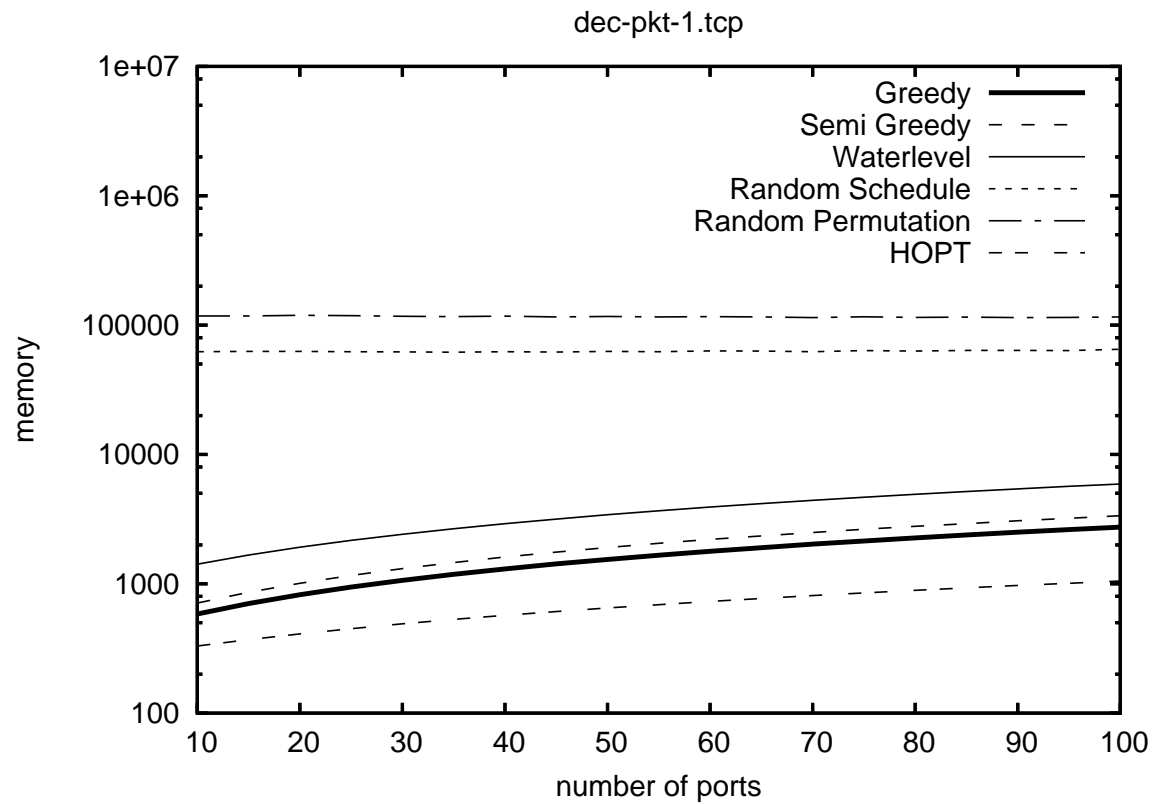
HOPT, Waterlevel have increasing running times



Varying the number of ports

Memory requirements

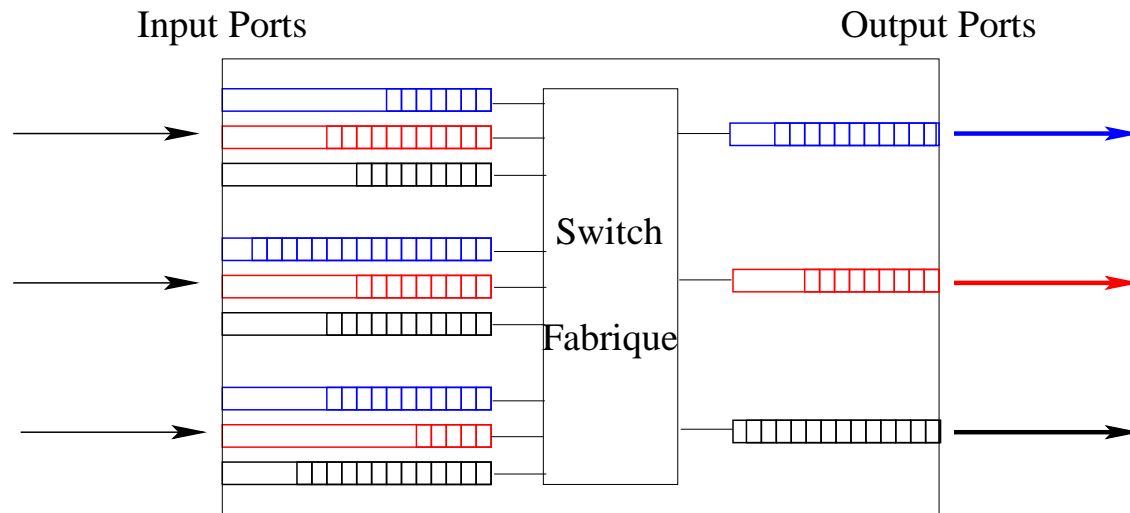
Deterministic algs. have
slightly increasing demands



Summary

- **Greedy:** Excellent competitiveness. Low, stable running times and memory requirements.
- **Semi Greedy:** Excellent competitiveness. Running times and memory requirements are slightly higher.
- **HOPT:** Best competitiveness. High running times for large m . Very low memory requirements.
- **Waterlevel:** Excellent competitiveness. Running times and memory requirements are substantially higher (factor 2).
- **Random Schedule, Random Permutation:** Very high competitiveness. High running. Huge memory requirements.

CIOQ switches



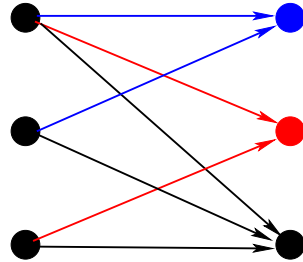
Input queues Q_{ij} , $1 \leq i, j \leq m$.

Output queues Q'_j , $1 \leq j \leq m$

Time step t

- (1) Packet arrivals at Q_{ij}
- (2) Scheduling in S rounds (speed). In any round s
At most one packet from heads of Q_{i1}, \dots, Q_{im} can be dequeued.
At most one packet can be enqueued at Q'_j .
- (3) Transmission from head of Q'_j

Matching



For any time t and round t_s , $1 \leq s \leq S$

$Q(t_s)$ = packets in queue Q

$h(Q(t_s))$ = packet at head of Q at t_s

$\min(Q(t_s))$ = minimum packet value in Q at t_s

$H(t_s) \subseteq \{h(Q_{ij}(t_s)) \mid 1 \leq i, j \leq m\}$ induces $G(t_s) = (U, V, E(t_s))$.

$U = V = \{1, \dots, m\}$

$(i, j) \in E(t_s)$ iff $H(t_s)$ contains packet p from Q_{ij} . Edge has weight $v(p)$.

Goal: Find maximum (weighted) matching in $G(t_s)$.

Unit-value packets

Algorithm Greedy: Each time step t .

- **Input buffer management:** Admit a new packet arriving at a buffer if space permits; otherwise drop it.
- **Scheduling:** In each round t_s , compute maximum matching in $G(t_s)$ induced by $H(t_s) = \{h(Q_{ij}(t_s)) \mid 1 \leq i, j \leq m \text{ and } Q'_j \text{ is not full}\}$.

Thm: For any speed S , Greedy is 3-competitive.

Kesselman, Rosen 2003

Arbitrary packet values

Algorithm Admin(β): At any time t , admit packet p arriving at buffer Q if the buffer not full or if it is full and $v(p) > \beta \min(Q(t))$.
In latter case discard packet of value $\min(Q(t))$.

Algorithm Greedy(β): Each time step t .

- **Buffer management:** At input queues use $Admin(1)$. At output queues use $Admin(\beta)$.
- **Scheduling:** In each round t_s compute max. weighted matching in $G(t_s)$ induced by $H(t_s) = \{h(Q_{ij}(t_s)) \mid 1 \leq i, j \leq m \text{ and } Q'_j \text{ is not full or } v(h(Q_{ij}(t_s))) > \beta \min(Q'_j(t_s))\}$.

Thm: For any S , best choice of β yields **8-competitive alg.**

Azar, Richter 2005