Buffer Management and Scheduling with Packet Dependencies

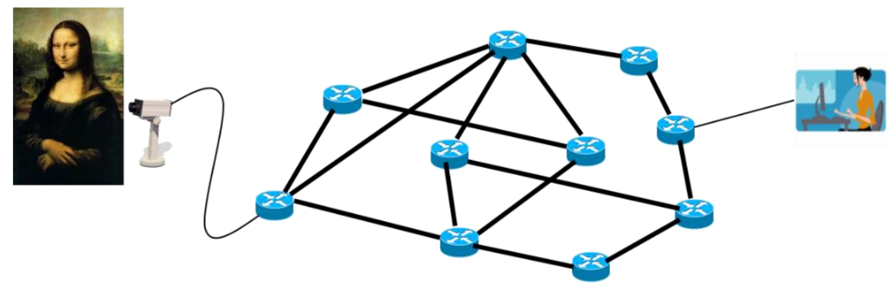
Gabriel Scalosub, Ben Gurion University

Based on joint works with:

Alex Kesselman, Boaz Patt-Shamir, Peter Marbach, and Jörg Liebeherr

Motivation: Video Streaming

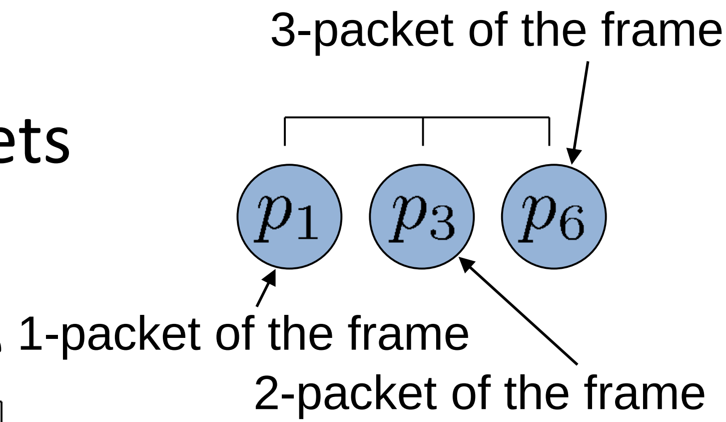
- Smart encoding:
  - Suffices to recover many

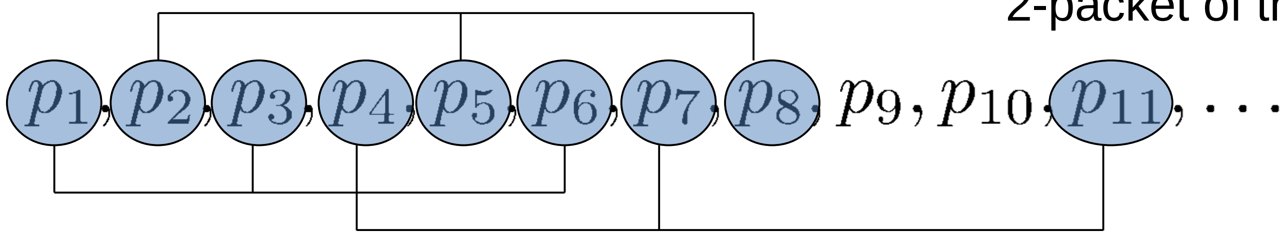


- Every video frame is fragmented into packets
- Restoration depends on recovering all packets
- If packets are lost:
  - Affects other packets as well (become redundant)
  - Streaming: retransmission is not an option
    - Popularity: Live TV over IP constantly increasing

Traffic, Dependencies, and Goodput

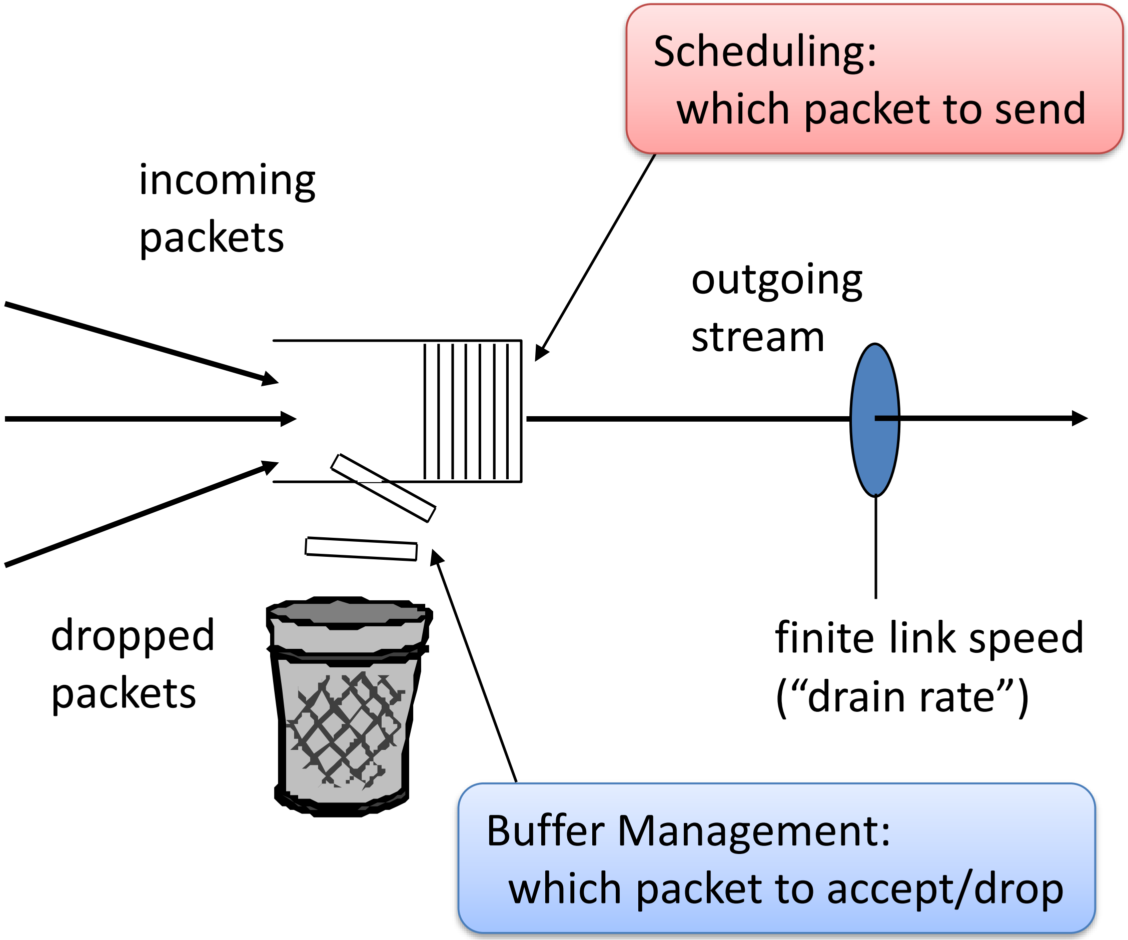
- Each frame consists of k packets
- Traffic is a sequence of packets
  - Underlying dependency structure





- Goodput: number of whole frames delivered
- Throughput vs. Goodput
  - Throughput (packet-level) ≠ Goodput (frame-level)
  - Also verified by experimental studies (e.g., MPEG)

Buffering Schematics



Buffer Model

- Single FIFO queue of size B

"fix" scheduling,
design buffer management

- Discrete time:
  - Delivery substep
    - One packet delivered from head of queue
  - Arrival substep
    - Packets arrive
    - Some packets may be dropped
    - Packets accommodated in the buffer

Methodology

- Adversarial traffic
  - An adversary generates the traffic dynamically
  - "Provides" an optimal solution for this traffic
  - Analysis globally applicable
    - Independent of the process generating the traffic
- Assumption:
  - An online algorithm has no knowledge of future traffic
  - OPT is the optimal clairvoyant (offline) algorithm
- Algorithm A is c-competitive if $\forall$ finite traffic σ

$$A(\sigma) \geq \frac{1}{c} \cdot \mathrm{OPT}(\sigma)$$

The Problem

- Given:
  - Finite size buffer
  - FIFO scheduler
  - Incoming traffic with packet dependencies
- Goal:
  - Maximize the goodput
- Main difficulty:
  - What to do upon overflow?? Which packet(s) to drop?

Our Contibutions

- Design guidelines for algorithms
  - ***No-regret***:

    Once a packet is admitted to the buffer (not necessarily sent), make every attempt possible to deliver the frame.
  - ***Ensure progress***:

    Deliver a complete frame as soon as possible
- Competitive algorithms (following these guidelines)
  - Analytic guarantees
  - Good performance in simulations
- Lower bounds

Related Work

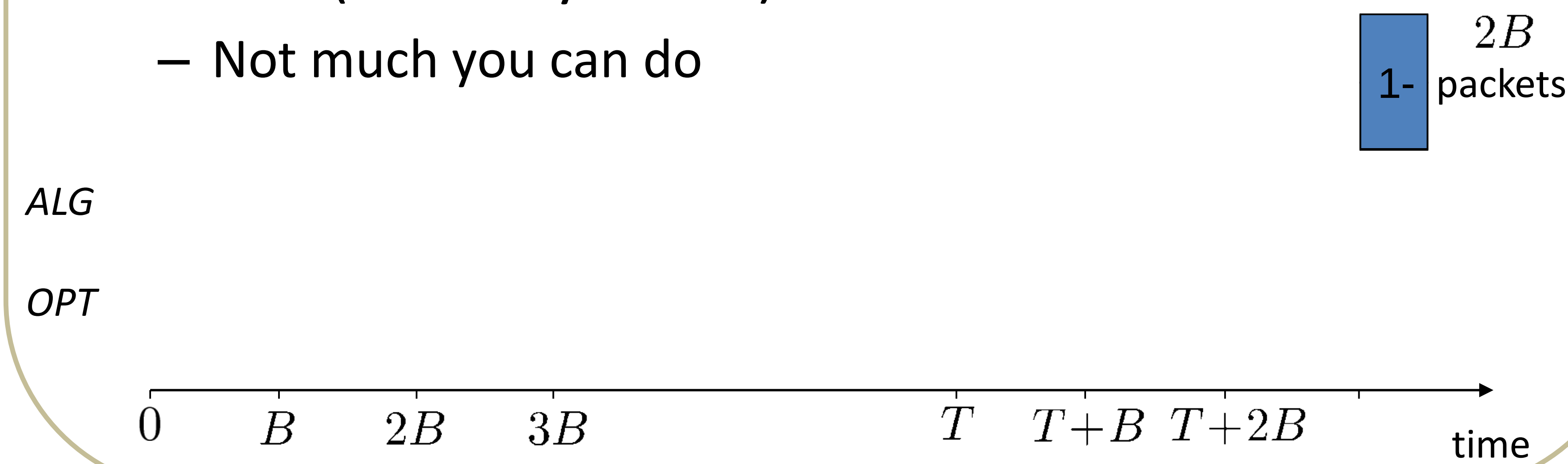
- MPEG inter-frame dependencies
  - Consider frames as “basic” entities (although routers decide on packets)

  [Zhang et al., 2001], [Awad et al., 2002]
- Inter-packet dependencies
  - Implementation [Ramanathan et al., 1995]
  - ATM guaranteed frame rate [Bonaventure & Nelissen, 2001]
- Buffer management for QoS
  - Multi-valued packets, competitive analysis

  [Lapid et al., 2000], [Kesselman et al., 2004], [Englert & Westerman, 2006]
- Proactive coding and FEC [Albanese et al., 1996]

Preliminaries

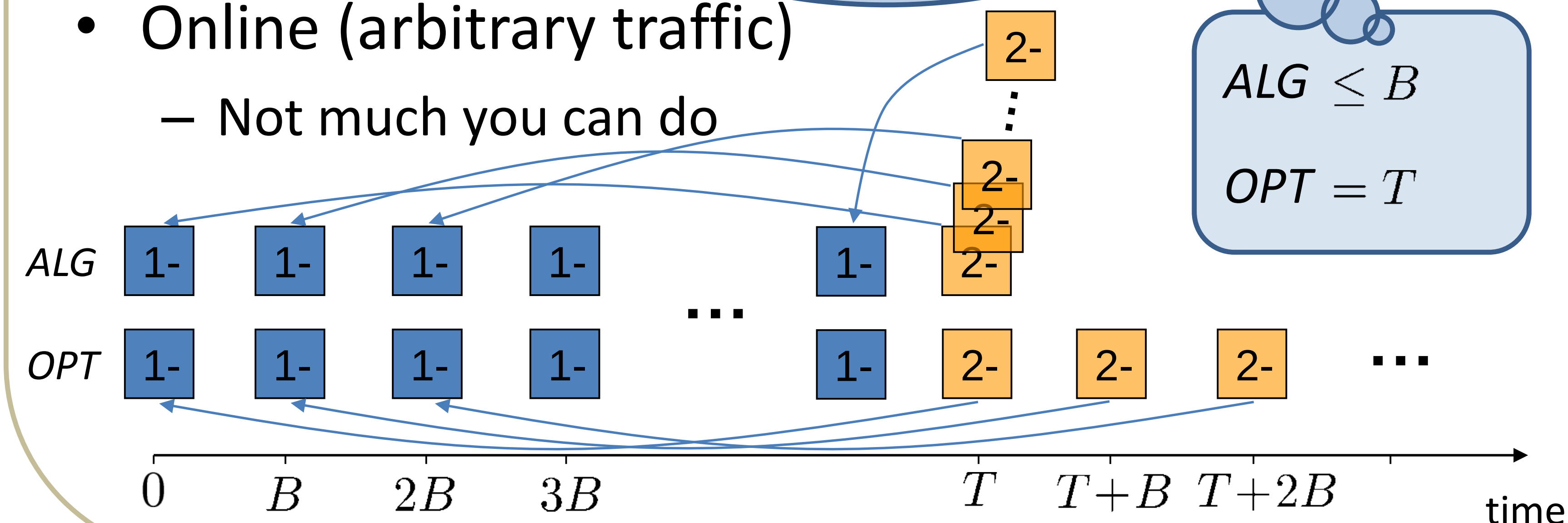
- Offline
  - Closely related to *k*-DM (Ω(k/log k)-hard to approx.)
  - Simple greedy algorithm is a (*k*+1)-approximation
- Online (arbitrary traffic)
  - Not much you can do



Preliminaries

Some thoughts:
(1) Is this reasonable traffic?
(2) Maybe the adversary is too strong??

- Online (arbitrary traffic)
  - Not much you can do



ALG $\leq B$

OPT $= T$

Order Respecting Traffic

- "Problem":
  - Selective unbounded delay/burstiness
- Model requirement ("solution"):
  - Both ALG and OPT have to deal with same delay/burstiness
- Order-respecting traffic:
  - Frame order induced by *j*-packets is the same for every *j*
    - OK: 2.1 (frame 2, part 1), 3.1, 2.2, 3.2
    - Not OK: 3.1, 2.1, 2.2, 3.2
- Lower bound for order respecting traffic: $\Omega(k)$

A Greedy Algorithm (*GA*)

- Consider k=2

At every time t where overflow occurs, keep packets according to the following priority:

1. 2-packets whose 1-packets were delivered
2. Complete frames (both 1-packets and 2-packets)
3. Remaining 1-packets

- Theorem: *GA* is constant-competitive for k=2
- What if k>3?

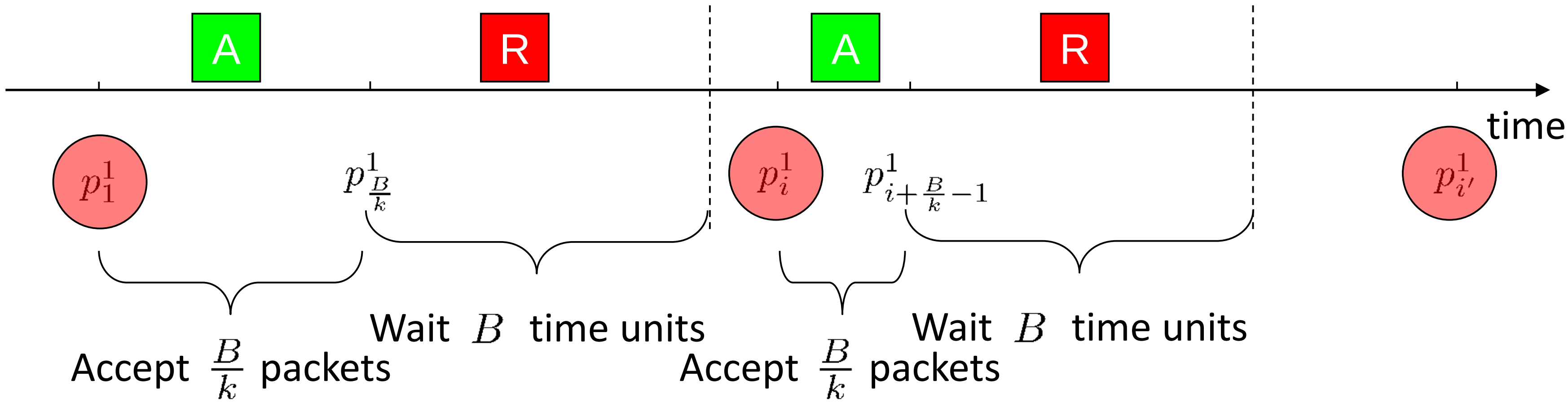
Static-Partitioning Algorithm (*SPA*)

- Intuition
  - Think ahead: focus on packet admission
  - Virtually partition the buffer into *k* levels of size $\frac{B}{k}$
    - Buffer is still FIFO!!
  - Level *j* only holds *j*-packets
  - Level *j* accepts *j*-packets that are "evenly" spaced in time
    - Alternating accept/reject periods
  - Levels synchronize on frame index
    - Ensures delivered packets correspond to the same frame
- Extra perk: non-preemptive

Example: *SPA* for k=2

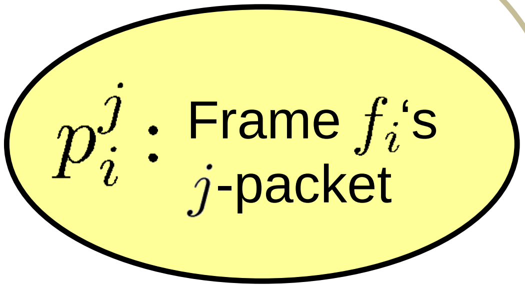
p_i^j : Frame f_i's j-packet

- Consider level 1, i.e., 1-packets

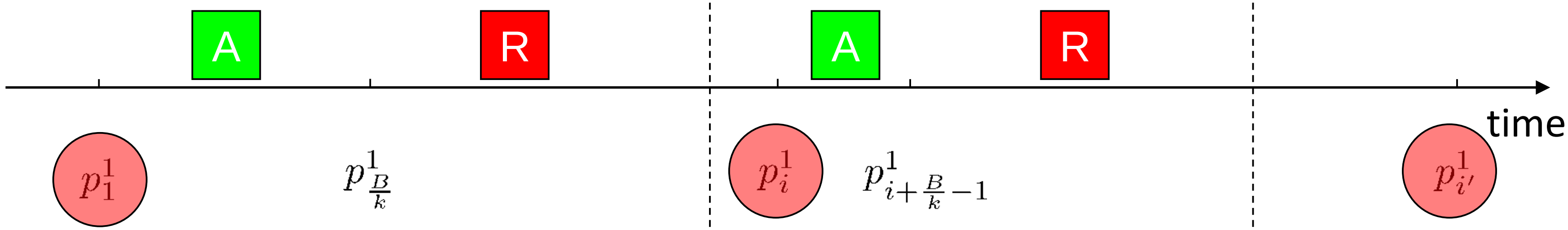


p_i^1 is the first 1-packet arriving after reject period

Example: *SPA* for k=2



- Consider level 1, i.e., 1-packets

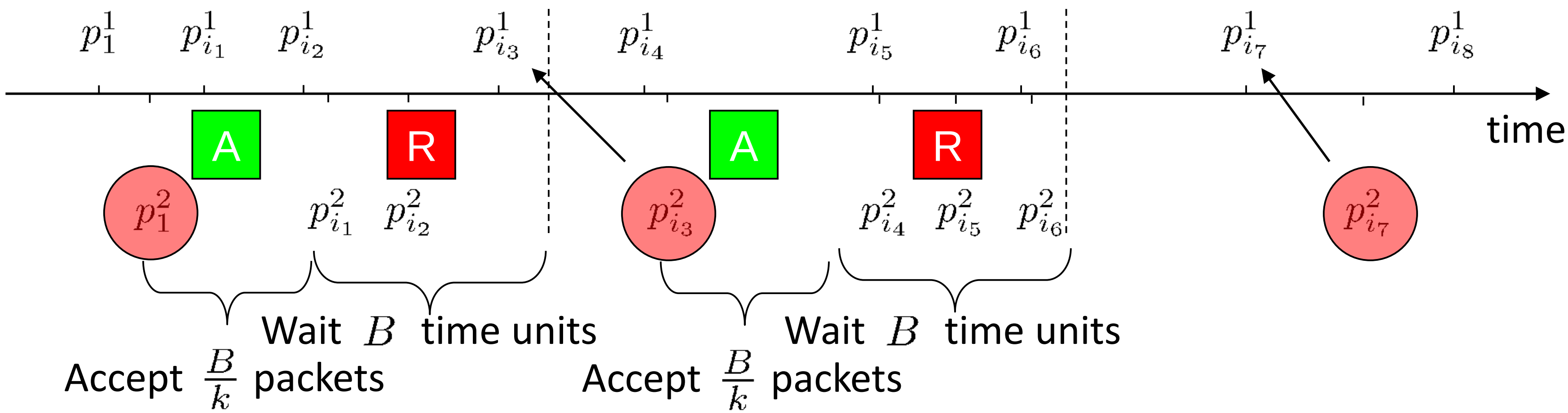


- 1-sync frame indices: $1, i, i', \ldots$
- Accepts $\dfrac{B}{k}$ first 1-packets after every 1-sync
  – Specifically, has sufficient buffer space

Example: *SPA* for k=2

p_i^j : Frame f_i's j-packet

- Consider level 2, i.e., 2-packets

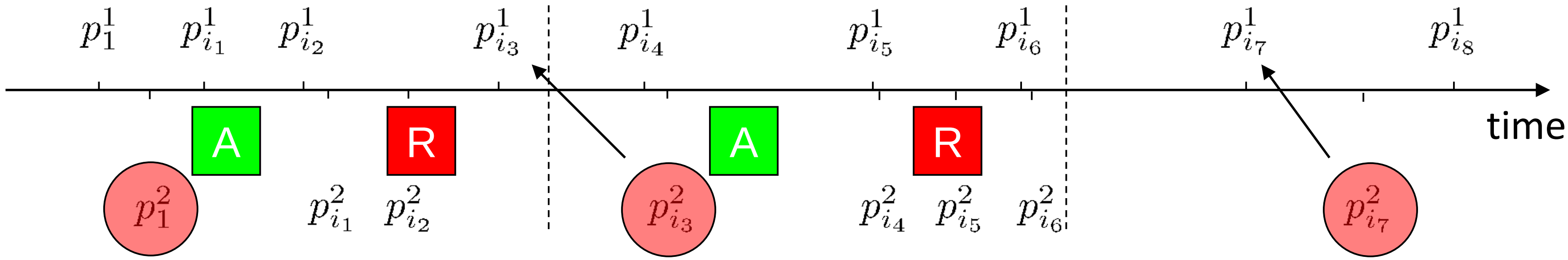


$p_{i_3}^2$ is the first 2-packet of a 1-sync arriving after reject period

Example: *SPA* for k=2

p_i^j : Frame f_i's j-packet

- Consider level 2, i.e., 2-packets



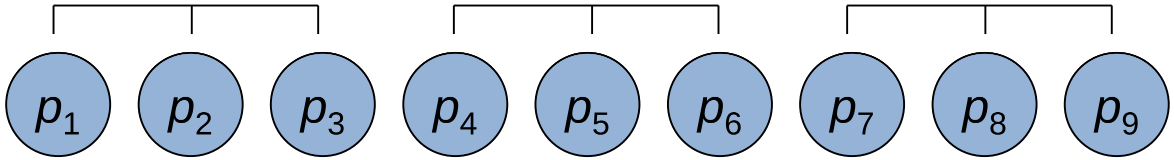
- 2-sync indices $\subseteq$ 1-sync indices
- Accepts $\frac{B}{k}$ first 2-packets after every 2-sync
  - Specifically, has sufficient buffer space

Analysis of *SPA*

- Analysis ingredients:
  - For any two consecutive k-syncs i, i'
    SPA delivers $\frac{B}{k}$ out of frames $\{f_i, \ldots, f_{i'-1}\}$
  - For any two consecutive k-syncs i, i'
    OPT delivers at most $2kB + B$ out of frames $\{f_i, \ldots, f_{i'-1}\}$
    - Intuition: at most $\frac{B}{k} + 2B$ out of every level
- Theorem: *SPA* is $(2k^2 + k)$-competitive
- Design criteria:
  - *No regret*: algorithm never preempts
  - *Ensure progress*: the first $\frac{B}{k}$ frames are always delivered

Coming Closer to Reality

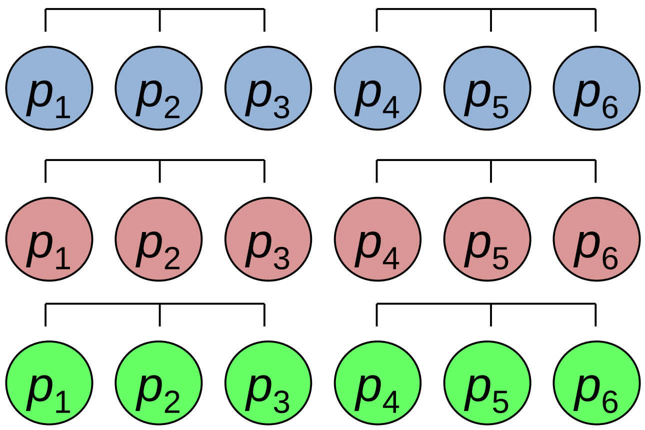
- Real-life traffic is hardly order-respecting
  - Network traffic consists of multiple *flows/streams*
  - No reason to assume any order is maintained for frames of different streams
- Modeling a single stream:

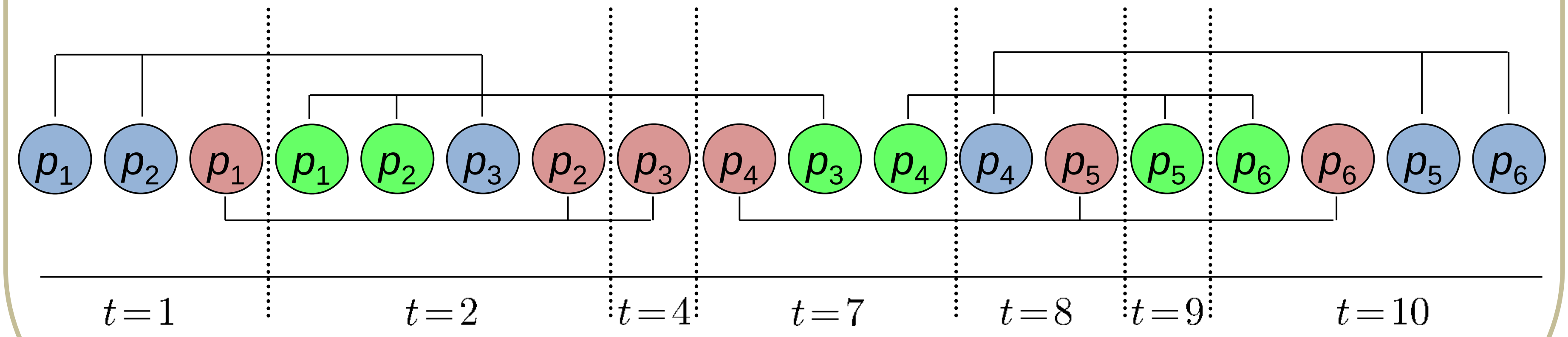


  - Packet stream is partitioned into blocks of k packets
  - Each block is a single frame
  - Packets of a stream arrive in order

Multiple Streams

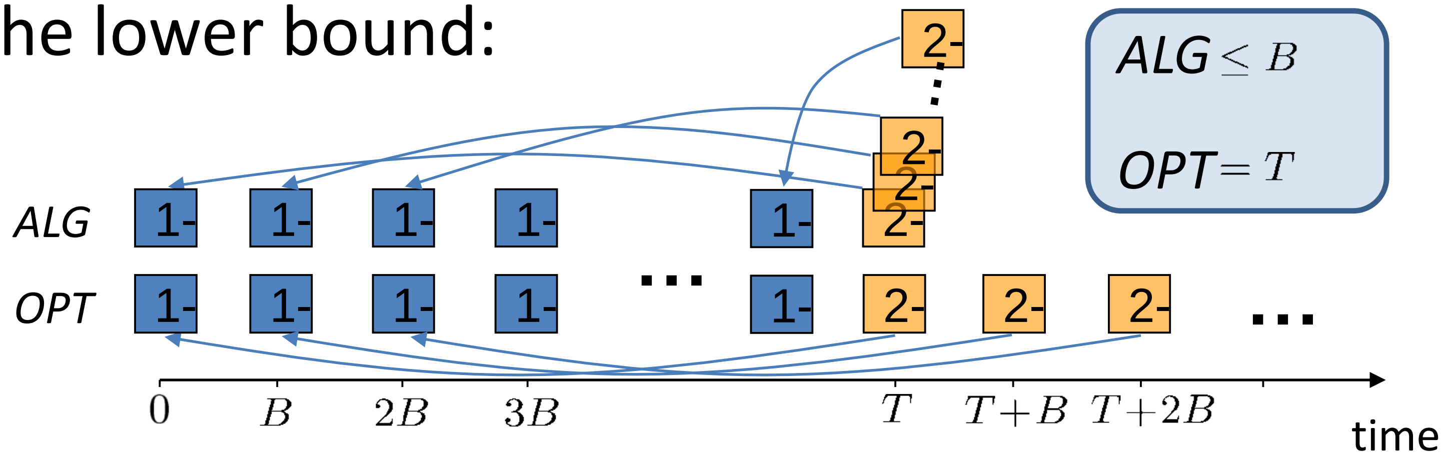
- An aggregate of M streams
- Bursty





Lower Bounds

- Recall the lower bound:



- M streams implies $T = O(M)$
  - Immediately gives a lower bound of $\Omega\left(\frac{M}{B}\right)$
- Can be generalized to show a lower bound of $\Omega\left(\frac{kM}{B}\right)$

The Return of the Greedy Approach

- At any time t:
  - Every stream has at most one “live” frame
  - We define a ranking on all streams by (lexicographically):

$$r_t(m) = (w_t(m), m)$$

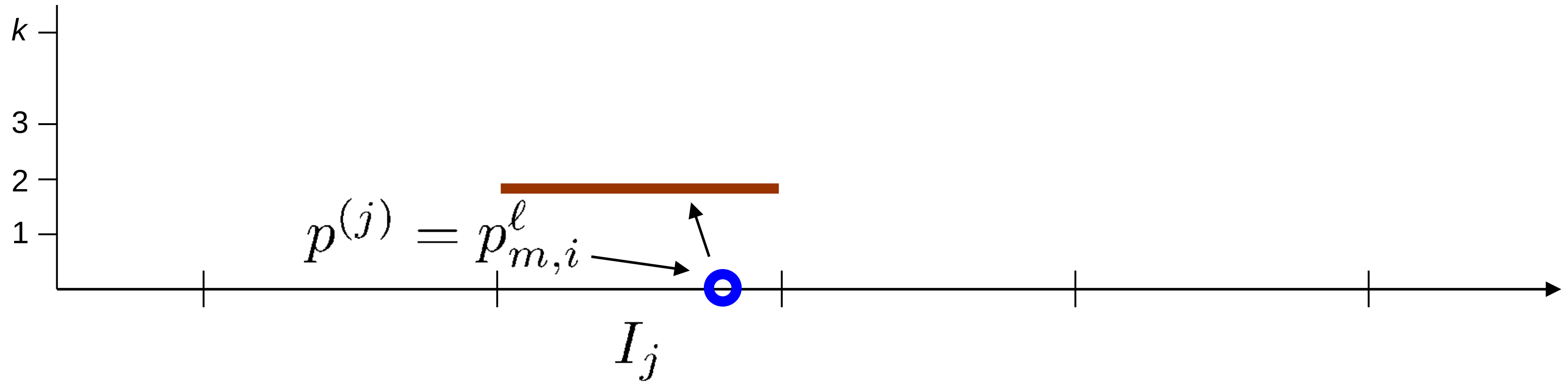
how many packets of the live frame have been sent (the “*weight*”)

tie-breaking stream index

- Algorithm *WeightPriority* (*WP*)
  - Upon overflow, drop lowest ranking frames first

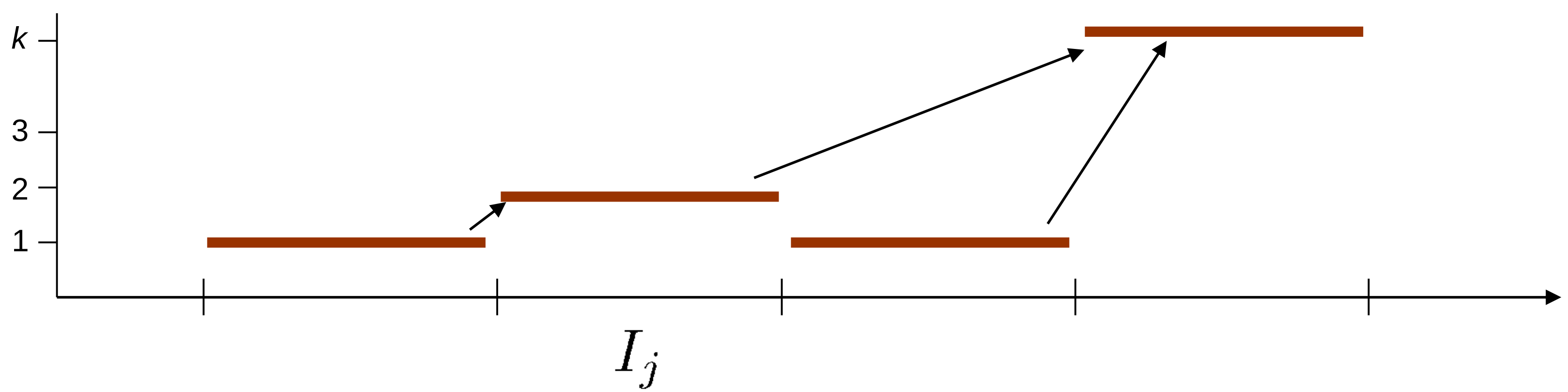
Analysis of *WP* (in a nutshell)

- Map frames delivered by *OPT* to ones delivered by *WP*
  - Partition time into intervals $I_1, I_2, I_3, \ldots$
  - Map every overflow in I_j to a packet delivered by *WP* in I_j
    - Implicitly implies a maximum weight for every interval



Analysis of *WP* (in a nutshell)

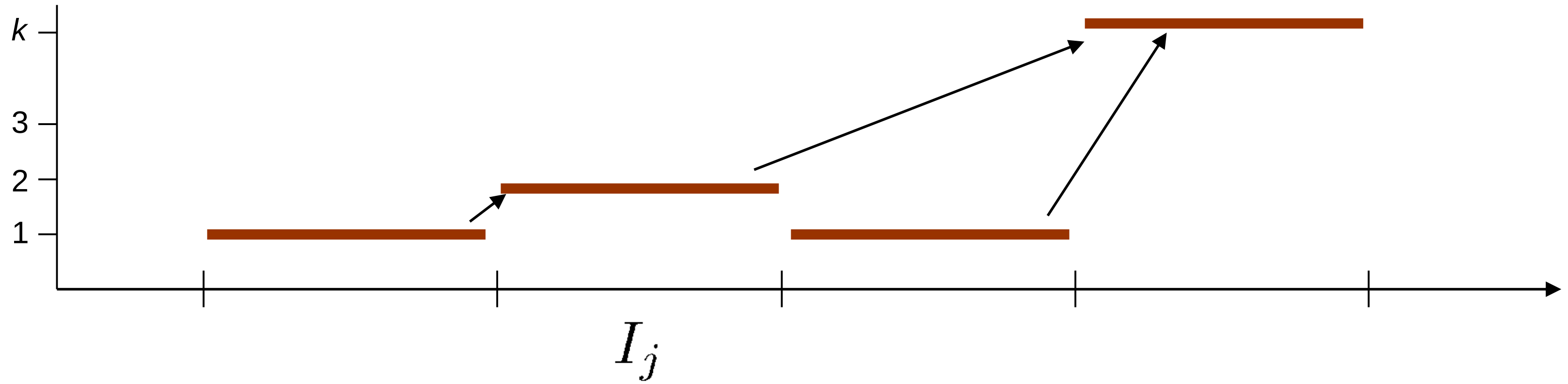
- Map frames delivered by *OPT* to ones delivered by *WP*
  - Partition time into intervals $I_1, I_2, I_3, \ldots$
  - Map every overflow in I_j to a packet delivered by *WP* in I_j
    - Implicitly implies a maximum weight for every interval
  - Map every I_j to a strictly-higher weight interval



Analysis of *WP* (in a nutshell)

- Show there exists an α:
  - *OPT* handles at most α frames in each interval I_j
  - At most α intervals are mapped to any interval I_j

- Results in a k-height α-ary tree
- Choosing $\alpha \sim kMB$ turns out to be a good choice
- Theorem: *WP* is $O((2kMB)^{k+1})$-competitive.



Analysis of *WP* (in a nutshell)

- Show there exists an α:
  - *OPT* handles at most α frames in each interval I_j
  - At most α intervals are mapped to any interval I_j

- Results in a k-height α-ary tree
- Choosing $\alpha \sim kMB$ turns out to be a good choice
- Theorem: *WP* is $O((2kMB)^{k+1})$-competitive.

- Design criteria: due to ranking

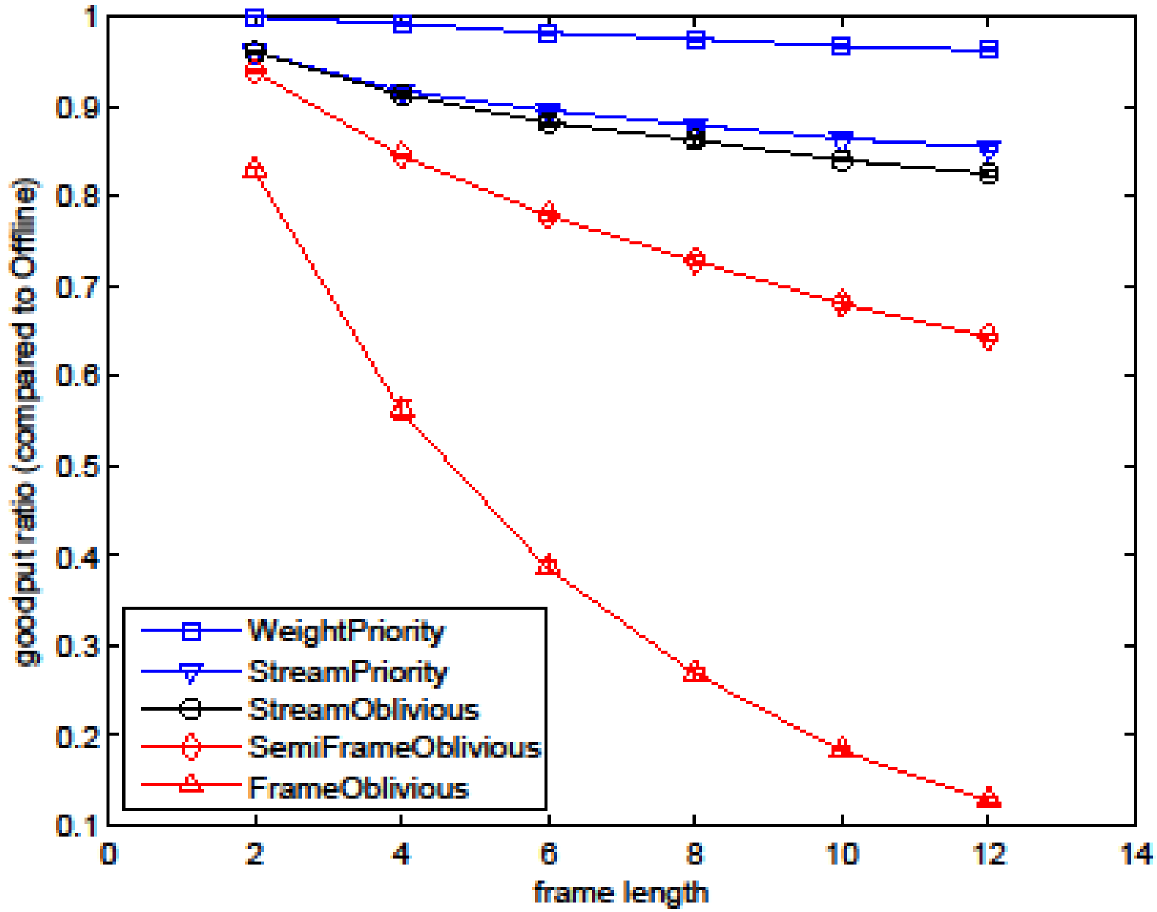
$$r_t(m) = (\underbrace{w_t(m)}_{\text{Ensure progress}}, \underbrace{m}_{\text{No regret}})$$

Additional Algorithms

- *FrameOblivious*
  - drop overflowing packets (ignore dependencies)
  - no state information req., no buffer scan
- *SemiFrameOblivious*
  - Same as FrameOblivious, but
  - Scans buffer to remove packets of dropped frames
- *StreamOblivious*
  - Same as *WP*, but rank is solely based on weight (arbitrary tie-break)
- *StreamPriority*
  - Same as *WP*, but rank is solely based on stream index

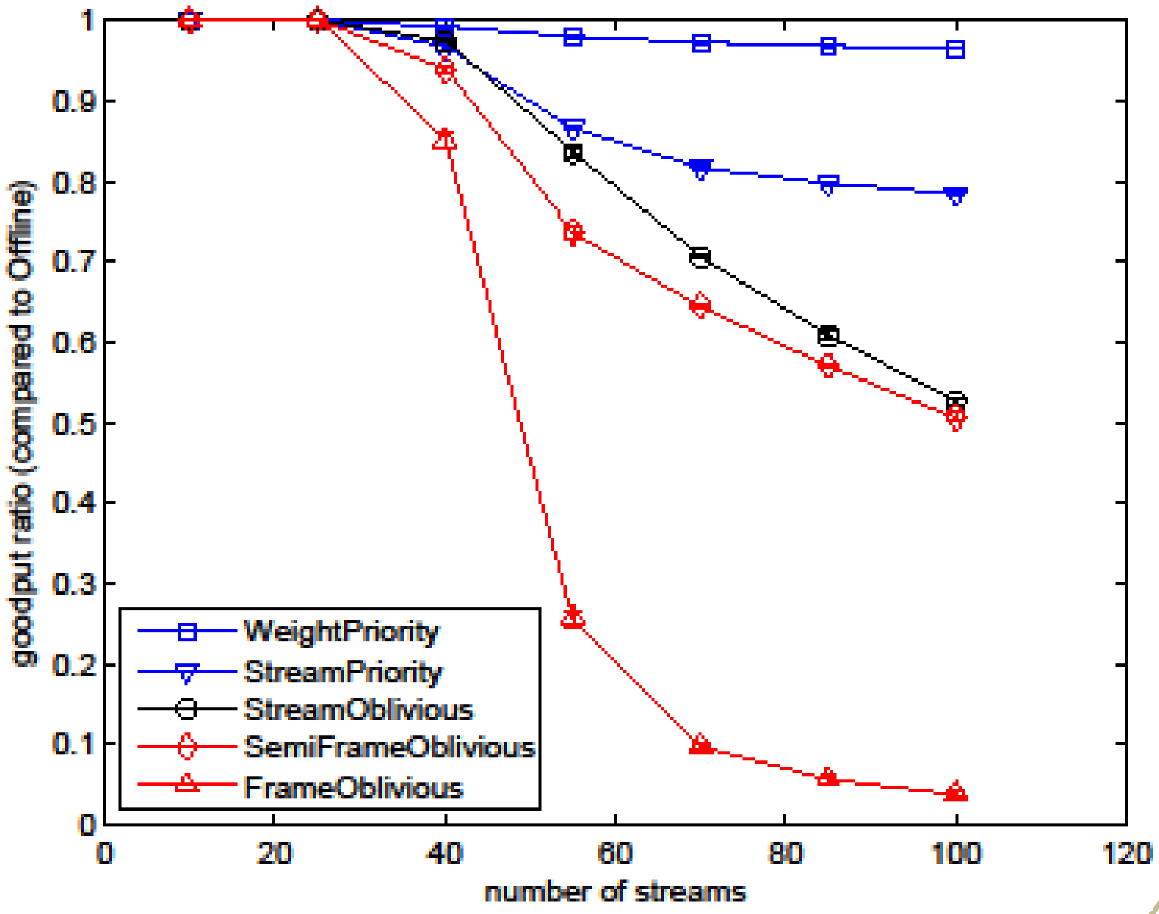
Effect of increasing *k*

- Each stream:
  - MMPP source
  - Av. rate: 0.025
- 50 streams
  - Av. agg. rate: 1.25
- 200 frames/stream
- B=12



Effect of increasing *M*

- Each stream:
  - MMPP source
  - Av. rate: 0.025
- 200 frames/stream
- *B*=12, *k*=6



Summary

- New model for traffic with packet dependencies
- Competitive algorithms (and lower bounds):

Traffic Model	Lower Bound	Upper Bound	Conclusion
Arbitrary	Unbounded		Too General
Order-resp.	$\Omega(k)$	$O(k^2)$	Too restrictive
M Streams	$\Omega\left(\frac{kM}{B}\right)$	$O((2kMB)^{k+1})$	Manageable, Realistic

Later improved to $O(k)$

Later improved to $O(kMB)$ (deterministic) and $O(M)$ (randomized)

Additional Models and Questions

- Competitive ratio gaps (new algorithms?)
- Using forward-error-correction (FEC)
  - Suffices to deliver *d*-of-*k*
- Recouple scheduling
  - "Price of FIFOness"
- Other extensions and QoS considerations
  - Fairness, increasing/diminishing returns, variable frame size / value, delay
- Many more
  - Inter-frame dependencies, randomness, …

Most results are for the case of no buffers:
a single packet admitted and delivered in every time slot

Focus on *scheduling,*
some results on uniform delays

References

- A. Kesselman, B. Patt-Shamir and G. Scalosub, “Competitive Buffer Management with Packet Dependencies”, Theoretical Computer Science, 489--490, pp. 75-87, 2013.
- G. Scalosub, J. Liebeherr and P. Marbach, “Buffer Management for Aggregated Streaming Data with Packet Dependencies”, IEEE Transactions on Parallel and Distributed Systems 24(3), pp. 439-449, 2013.
- Yishay Mansour, Boaz Patt-Shamir, and Dror Rawitz, “Overflow management with multipart packets”, Computer Networks, 56(15), pp. 3456–3467, 2012.
- Y. Emek, M. M. Halldórsson, Y .Mansour, B. Patt-Shamir, J. Radhakrishnan, and D. Rawitz, “Online set packing and competitive scheduling of multi-part tasks”, SIAM Journal on Computing, 41(4), pp. 728–746, 2012.
- J. Kawahara, K. M. Kobayashi, and S. Miyazaki, “Better bounds for online k-frame throughput maximization in network switches”, TCS, 657(B), pp. 173-190, 2017.
- G. Scalosub, “Towards Optimal Buffer Management for Streams with Packet Dependencies”, Computer Networks, 129, pp. 207-214, 2017.
- M. Markovitch and G. Scalosub, “Bounded Delay Scheduling with Packet Dependencies”. Computer Communications, 182, pp. 98-109, 2022.

Network QoS
371-2-0213

Lecture 13

Gabriel Scalosub

Outline

Course Agenda

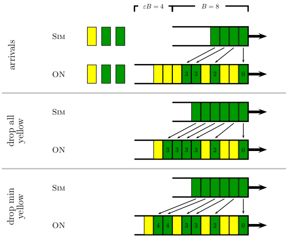
- Design and analysis of protocols for providing *Quality-of-Service* (QoS):
  - currently used protocols, and
  - new designs and models
- Main problems in networks arise from drops/delays, e.g.:
  - applications: bumpy audio/video
  - network: increased congestion due to TCP retransmissions
- Network resources (BW, buffers, ...) shared by all the users
- Most current networks: best-effort
  - all packets were created equal
  - FIFO scheduling
  - no admission control
- QoS support must be active about resource allocation
  - how should resources be allocated?

Current Concepts of QoS

- Integrated Services (IntServ)
  - per-flow resource reservation
    - routing protocol finds E2E path depending on flow demand
    - protocol: RSVP
  - handicaps:
    - accounting (e.g., inter-AS), scalability (per-flow state)
- Differentiated Services (DiffServ)
  - better than best-effort
  - dividing traffic into small number of forwarding classes
    - employ policing, marking per class
    - traffic prioritization: scheduling/buffer management at routers
  - handicaps:
    - no reservation, hard to provide deterministic QoS guarantees
- Traffic engineering
  - optimize usage of network resources
  - sophisticated routing
    - needs "global" information: topology, traffic (mostly intra-AS)
    - usually MPLS-based

Routing and Buffer Management

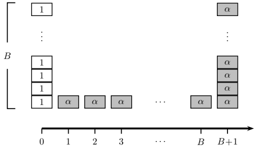
- AAP routing
  - competitive model for path reservation
  - models many of IntServ's aspects
- Committed and excess buffer mgmt.
  - competitive model for satisfying SLA
  - models aspects of managing a single AF PHB class



Feasibility and lag of algorithm

Routing and Buffer Management

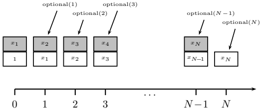
- AAP routing
  - competitive model for path reservation
  - models many of IntServ's aspects
- Committed and excess buffer mgmt.
  - competitive model for satisfying SLA
  - models aspects of managing a single AF PHB class
- Weight-based FIFO buffer mgmt.
  - competitive model for managing multiple AF/EF PHBs



Analysis of Greedy is tight

Routing and Buffer Management

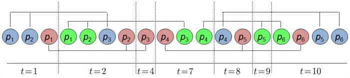
- AAP routing
  - competitive model for path reservation
  - models many of IntServ's aspects
- Committed and excess buffer mgmt.
  - competitive model for satisfying SLA
  - models aspects of managing a single AF PHB class
- Weight-based FIFO buffer mgmt.
  - competitive model for managing multiple AF/EF PHBs
- Bounded-delay scheduling and buffer mgmt.
  - TTL-based competitive model for managing delay-sensitive traffic



Asymptotic LB of $\phi \sim 1.618$

Routing and Buffer Management

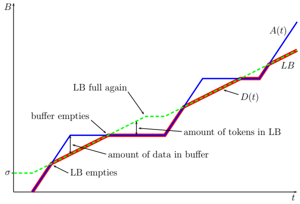
- AAP routing
  - competitive model for path reservation
  - models many of IntServ's aspects
- Committed and excess buffer mgmt.
  - competitive model for satisfying SLA
  - models aspects of managing a single AF PHB class
- Weight-based FIFO buffer mgmt.
  - competitive model for managing multiple AF/EF PHBs
- Bounded-delay scheduling and buffer mgmt.
  - TTL-based competitive model for managing delay-sensitive traffic
- Packet dependencies
  - packet level decisions vs. data-frame level effect



packets vs. frames

Network Calculus and Scheduling

- Network Calculus
  - Arrival and service curves
  - Envelopes and regulators
  - Prove IntServ delay bounds



Arrival and service curves

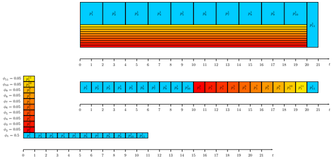
Network Calculus and Scheduling

- Network Calculus
  - Arrival and service curves
  - Envelopes and regulators
  - Prove IntServ delay bounds
- Classic machine scheduling
  - Makespan, interval scheduling

bad ways to sort intervals

Network Calculus and Scheduling

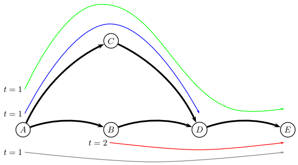
- Network Calculus
  - Arrival and service curves
  - Envelopes and regulators
  - Prove IntServ delay bounds
- Classic machine scheduling
  - Makespan, interval scheduling
- Cornerstones of packet scheduling
  - FIFO, PQ, RR, WFQ (and beyond)



WFQ

Network Calculus and Scheduling

- Network Calculus
  - Arrival and service curves
  - Envelopes and regulators
  - Prove IntServ delay bounds
- Classic machine scheduling
  - Makespan, interval scheduling
- Cornerstones of packet scheduling
  - FIFO, PQ, RR, WFQ (and beyond)
- Adversarial Queueing Theory (AQT)
  - scheduling across a network
  - stability of protocols, traffic-patterns, and topologies



AQT arrival model

Many Interesting Research Directions

- Buffer management
  - e.g., packet dependencies, closing UB/LB gaps, other models
- Scheduling
  - interplay between scheduling and buffer management
  - it shouldn't all be FIFO

- Many/most of networking/systems research topics are related to QoS
  - latency, bandwidth, drop, differentiation
  - constraints / objectives
  - "local" or system-wide perspectives
- Also working on *caching* problems
  - what to cache, when, size/value awareness, cost models, ...
  - local / distributed, self-adjusting
- You're all welcome to stop by and chat anytime
  - also about research... ☺

Next Semester (Undergraduate Course)

Thank You!

sgabriel@bgu.ac.il