INF 5071 – Performance in Distributed Systems

Distribution – Part I

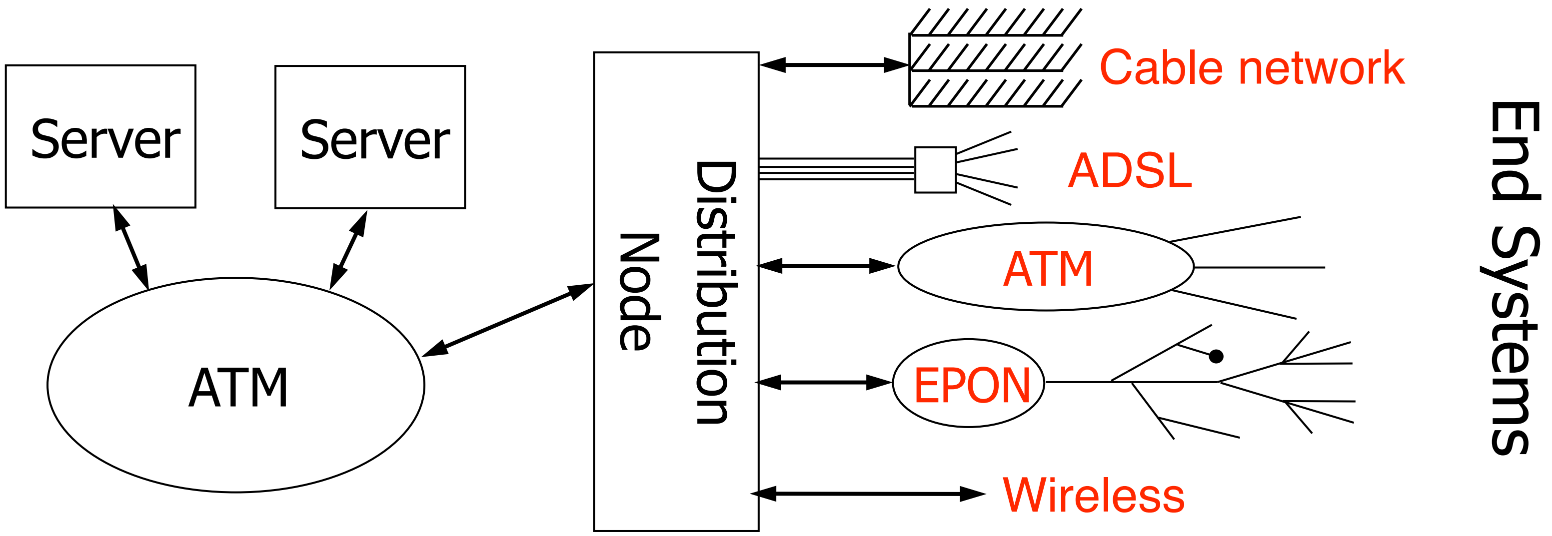
October 9, 2009

Distribution

- Why does distribution matter?
  - Digital replacement of earlier methods

- What can be distributed?

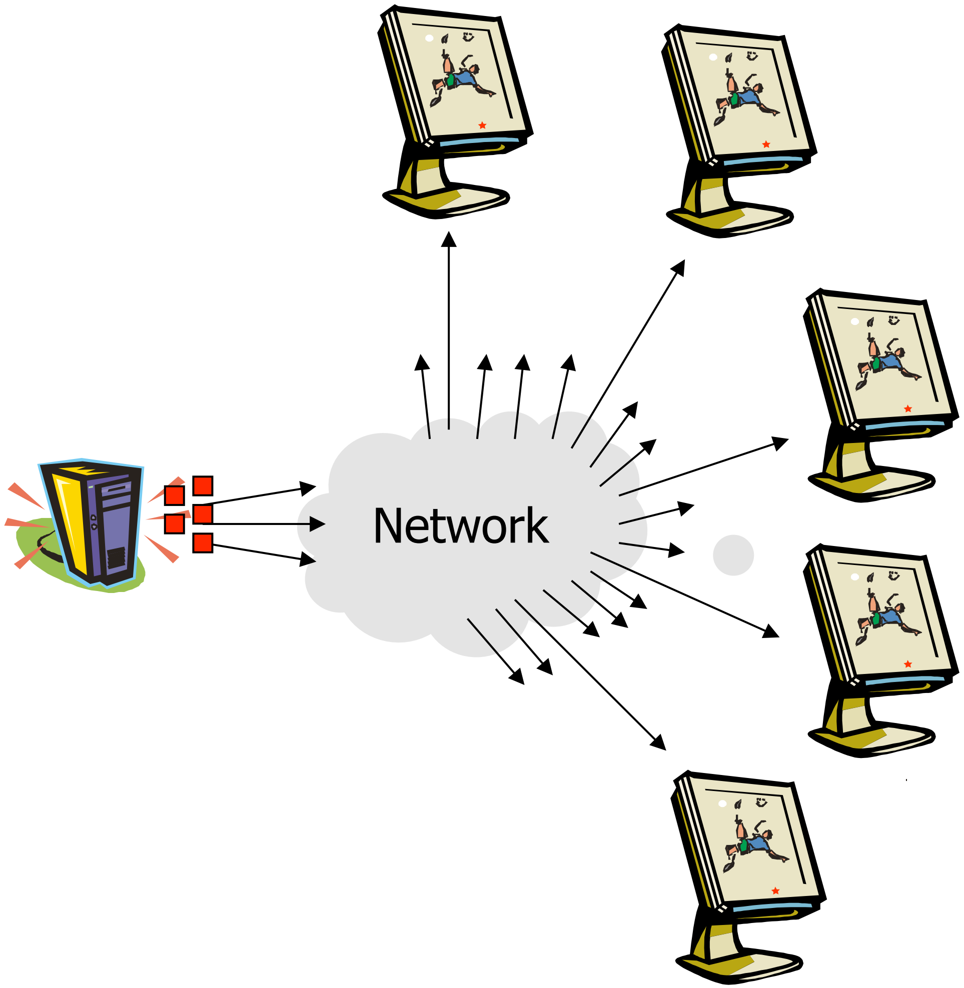
Distribution Network Approaches



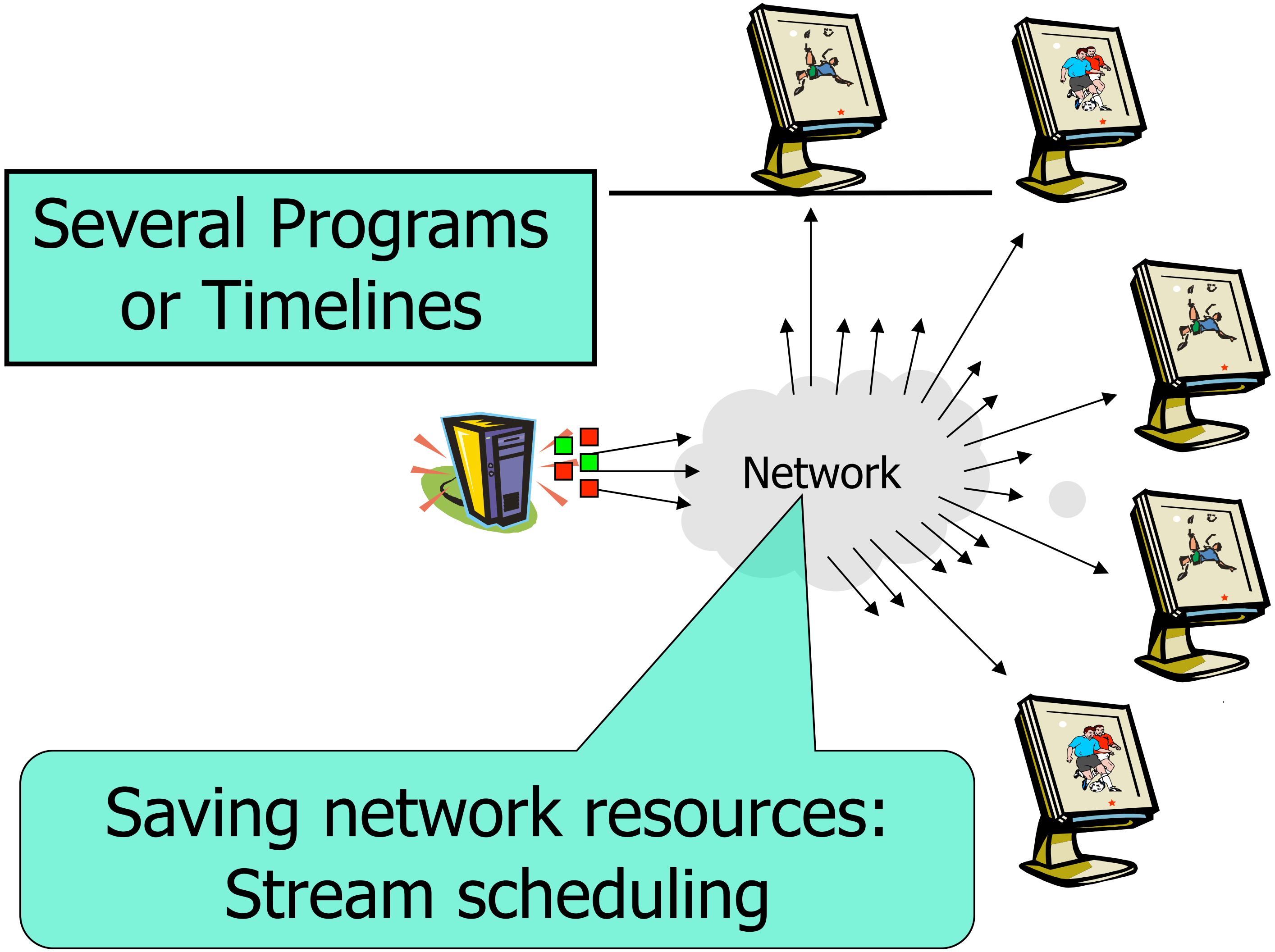
- Wide-area network backbones
  - FDM/WDM
  - 10GB Ethernet
  - ATM
  - SONET

- Local Distribution network
  - Wired
    - HFC (Hybrid Fiber Coax - cable)
    - ADSL (Asymmetric Digital Subscriber Line)
    - EPON (Ethernet Based Passive Optical Networks)
  - Wireless
    - IEEE 802.11
    - WiMax

Delivery Systems Developments



Delivery Systems Developments



Optimized delivery scheduling

- Background/Assumption:
  - Performing all delivery steps for each user wastes resources
  - Scheme to reduce (network & server) load needed
  - Terms
    - Stream: a distinct multicast stream at the server
    - Channel: allocated server resources for one stream
    - Segment: non-overlapping pieces of a video
  - Combine several user requests to one stream
- Mechanisms
  - Type I: Delayed on-demand delivery
  - Type II: Prescheduled delivery
  - Type III: Client-side caching

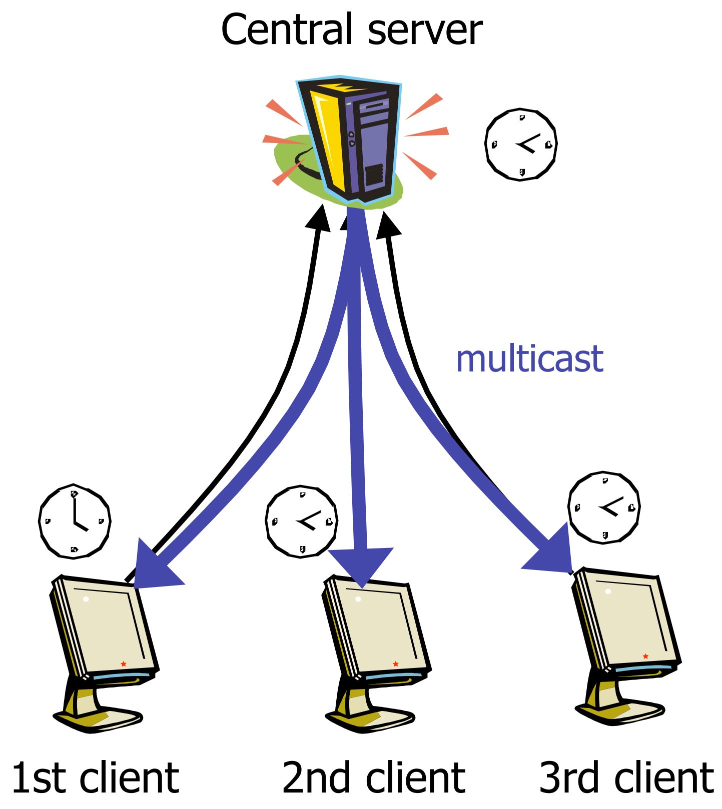
Type I:
Delayed On Demand Delivery

Optimized delivery scheduling

- Delayed On Demand Delivery

[Dan, Sitaram, Shahabuddin 1994]

  - Collecting requests
  - Joining requests
  - Batching
    - Delayed response
    - Collect requests for same title
    - Batching Features
      - Simple decision process
      - Can consider popularity
    - Drawbacks
      - Obvious service delays
      - Limited savings

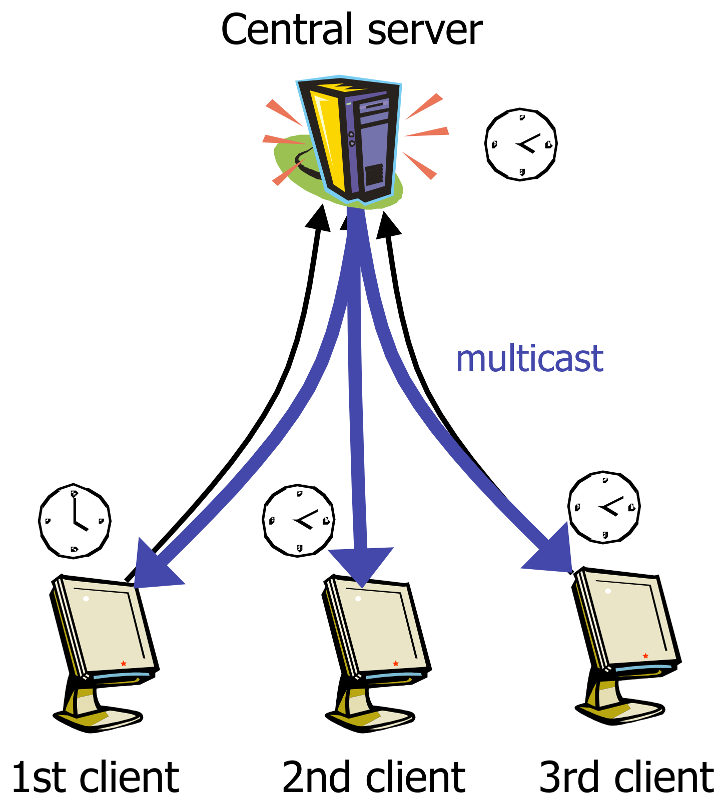


Optimized delivery scheduling

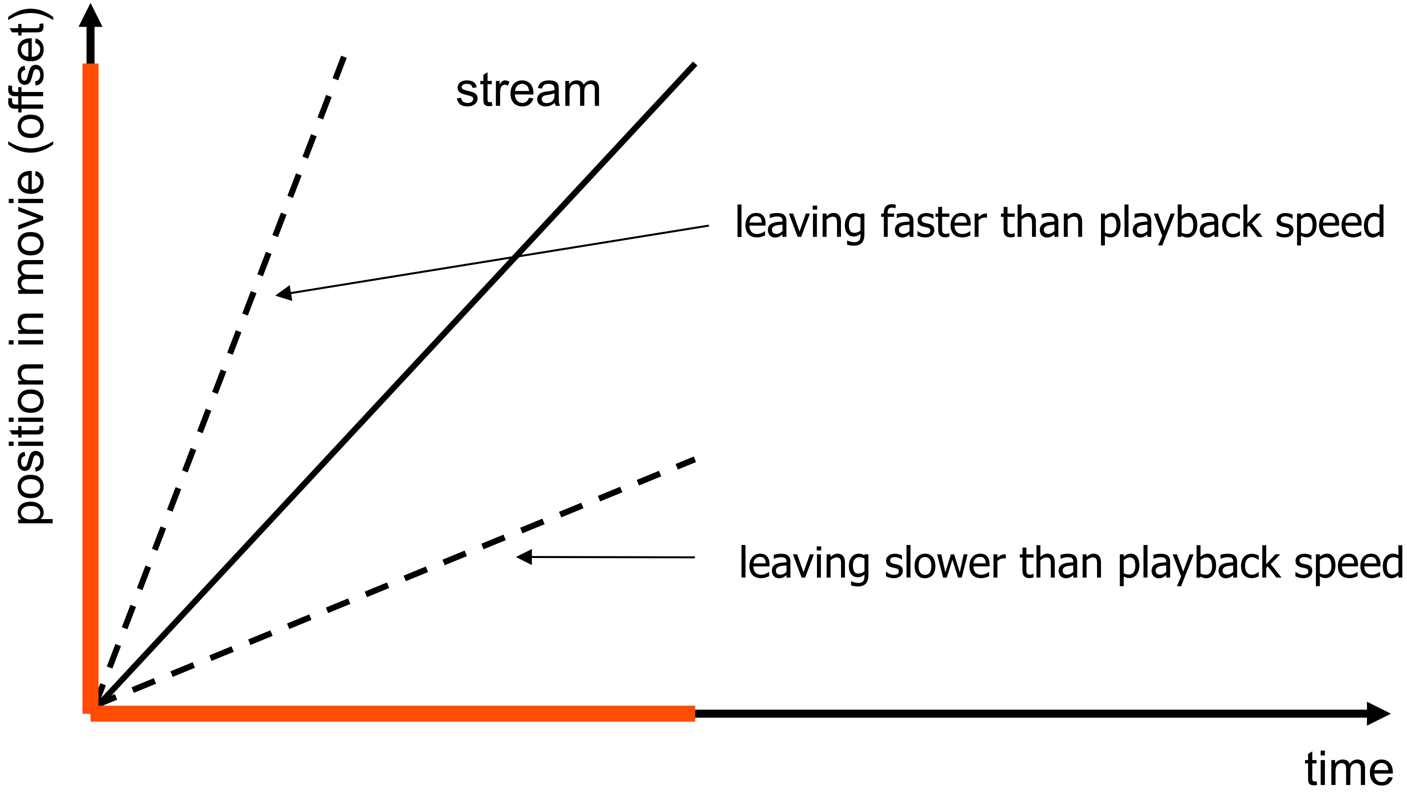
- Delayed On Demand Delivery
  - Collecting requests
  - Joining requests

  - Batching
    - Delayed response
  - Content Insertion
    - E.g. advertisement loop
  - Piggybacking
    - "Catch-up streams"
    - Display speed variations

  - Typical
    - Penalty on the user experience
    - Single point of failure



Graphics Explained



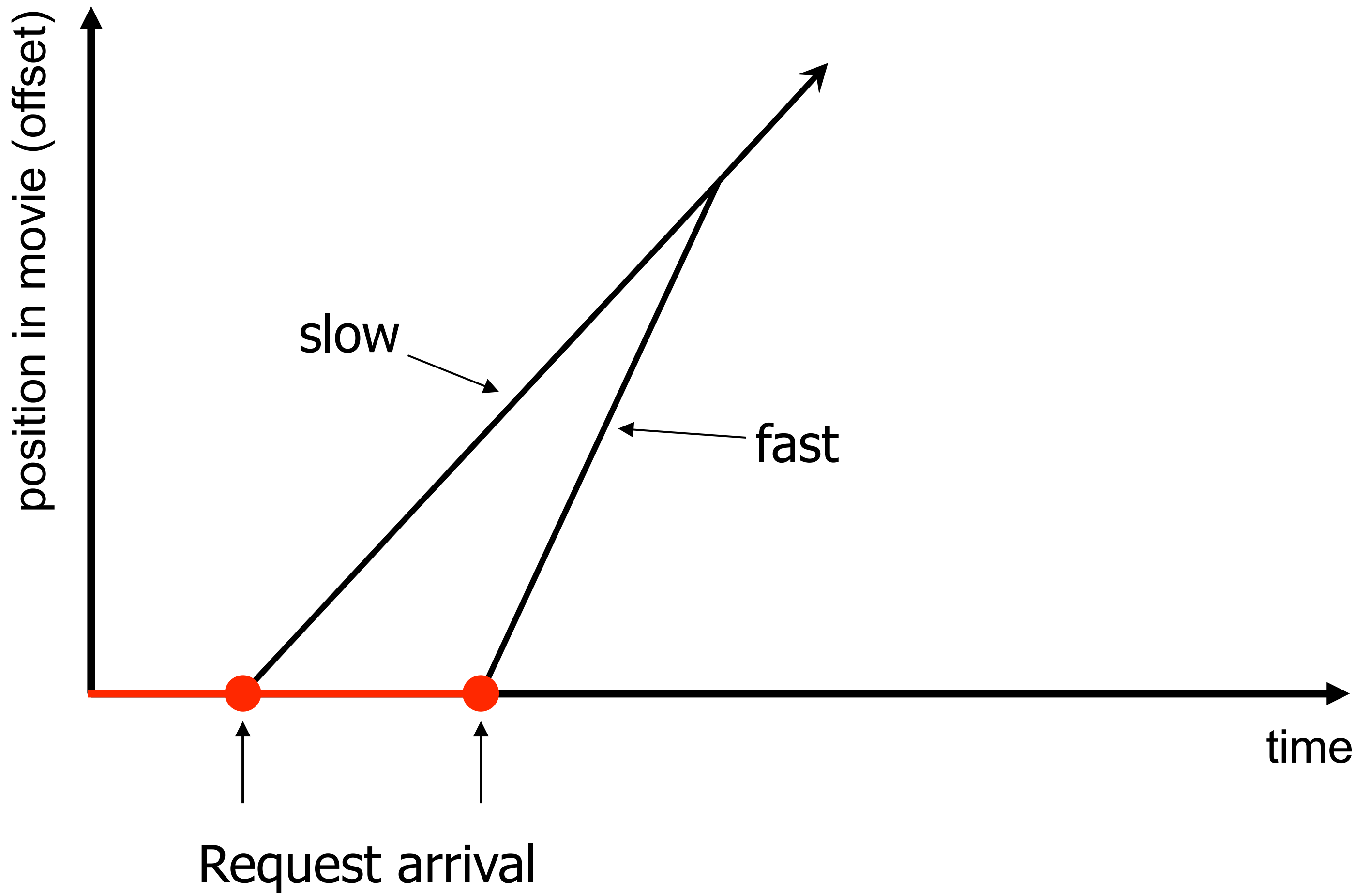
- Y - the current position in the movie
  - the temporal position of data within the movie that is leaving the server
- X - the current actual time

Piggybacking

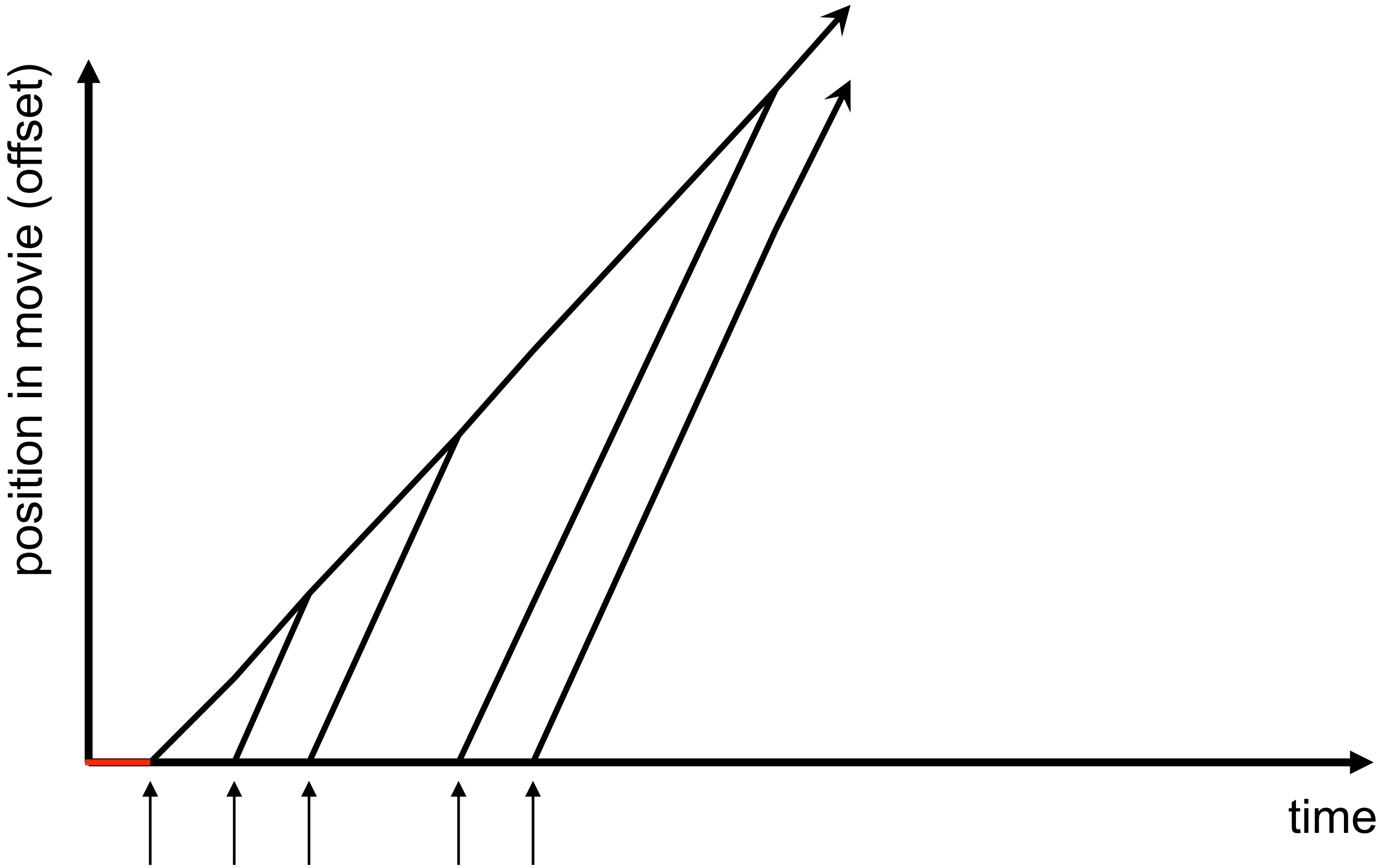
[Golubchik, Lui, Muntz 1995]

- Save resources by joining streams
  - Server resources
  - Network resources
- Approach
  - Exploit limited user perception
  - Change playout speed
    - Up to +/- 5% are considered acceptable
- Only minimum and maximum speed make sense
  - i.e. playout speeds
    - 0
    - +10%

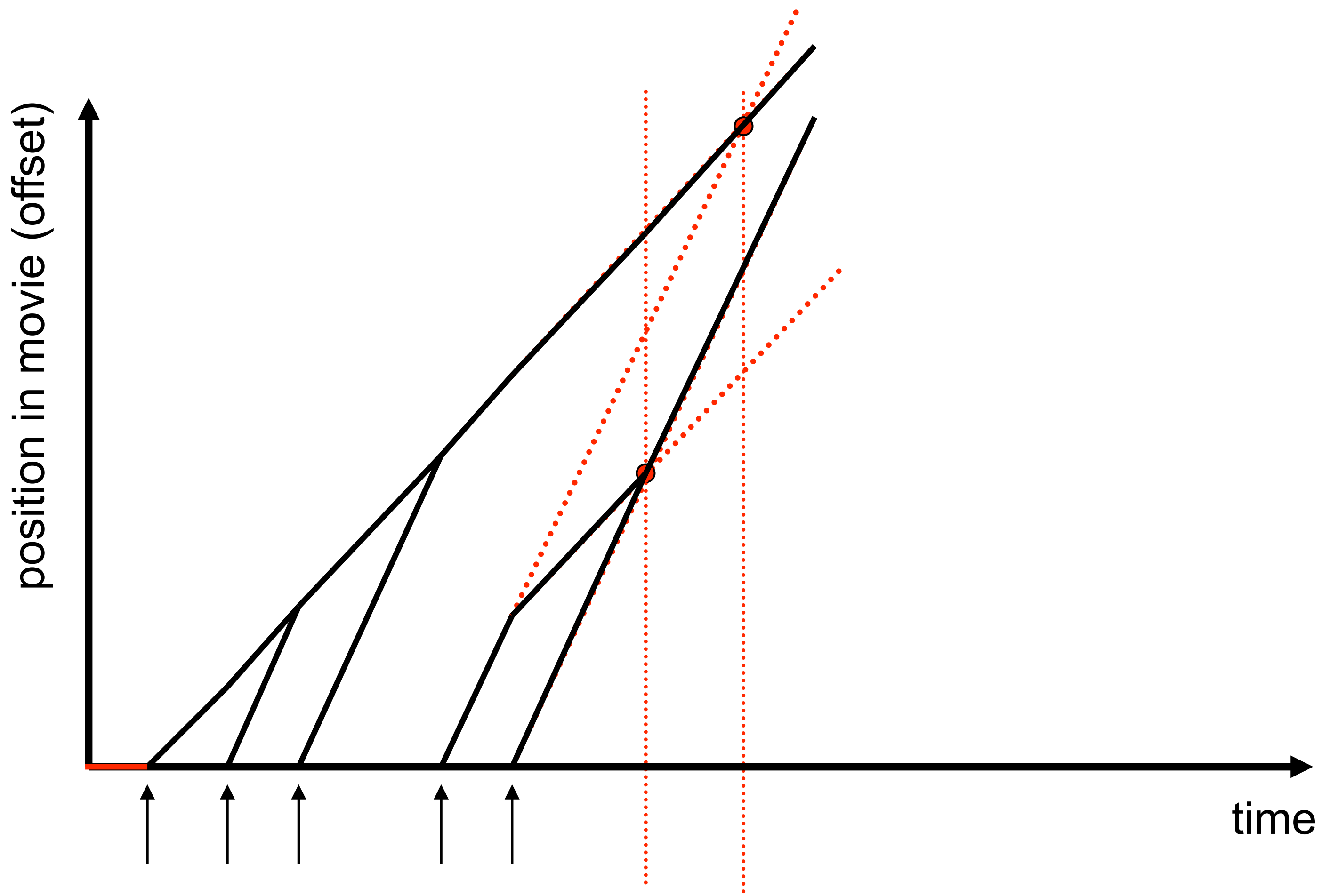
Piggybacking



Piggybacking



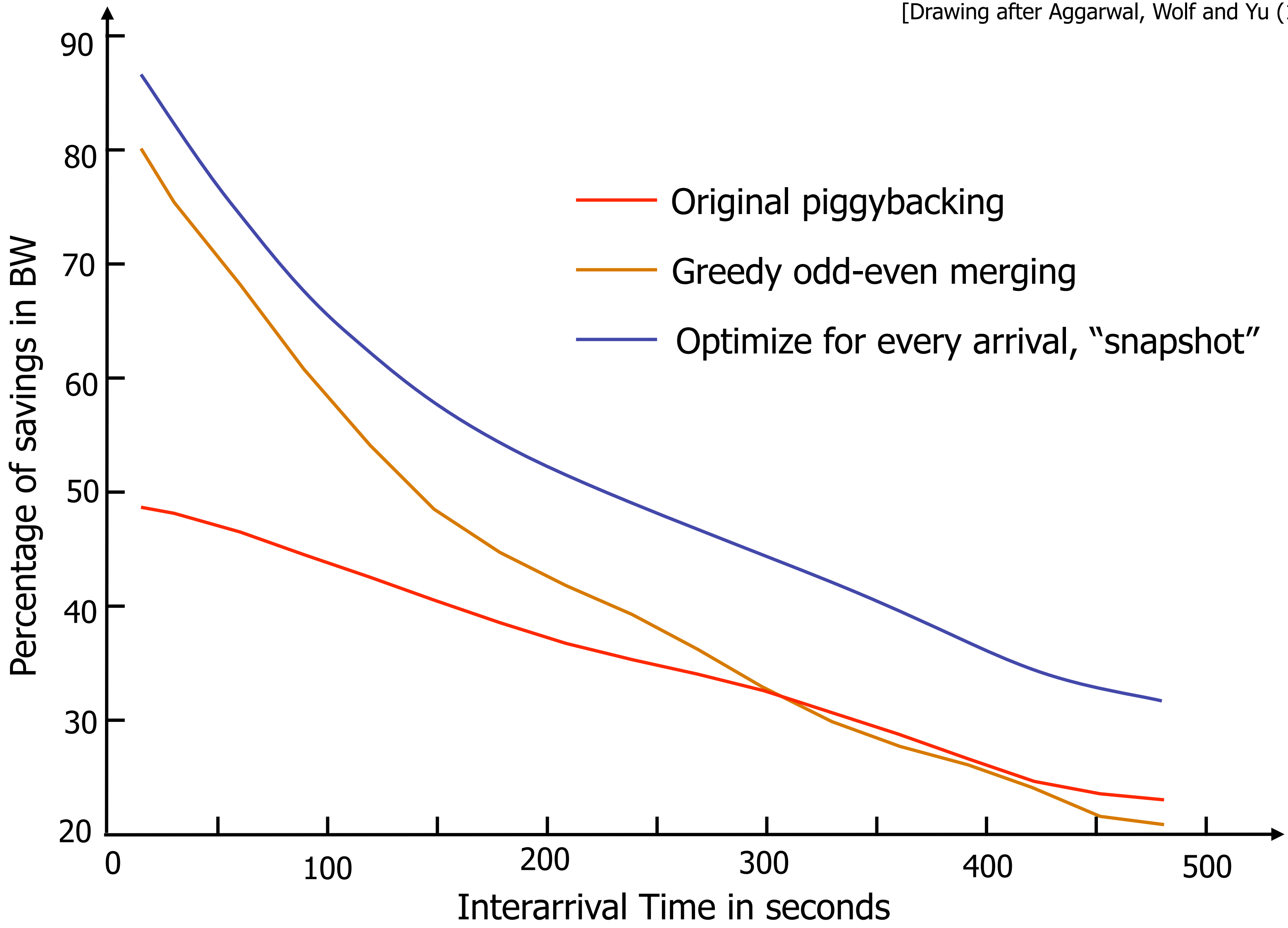
Adaptive Piggybacking



[Aggarwal, Wolf, Yu 1996]

Performance

[Drawing after Aggarwal, Wolf and Yu (1996)]

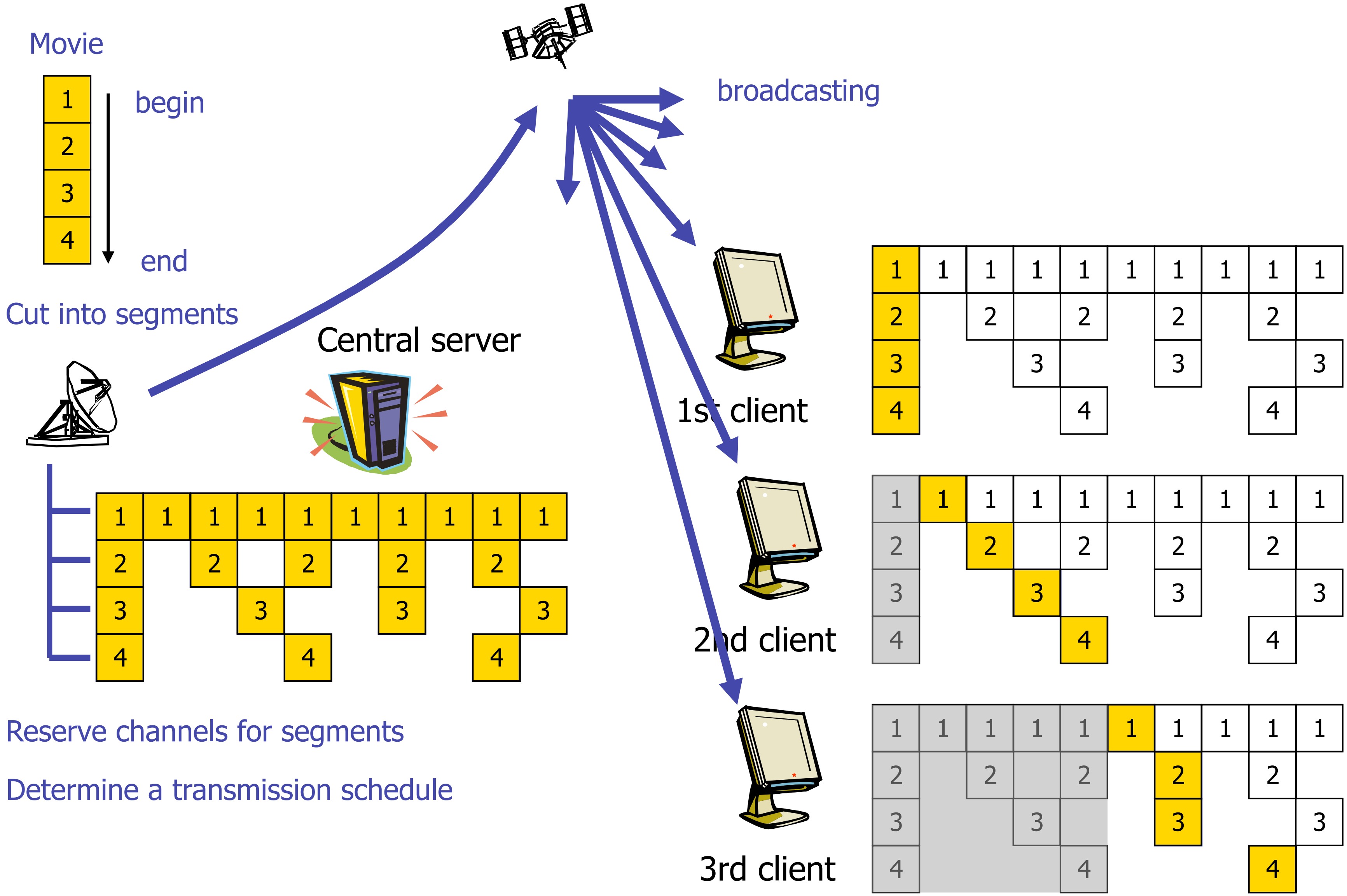


Type II:
Prescheduled Delivery

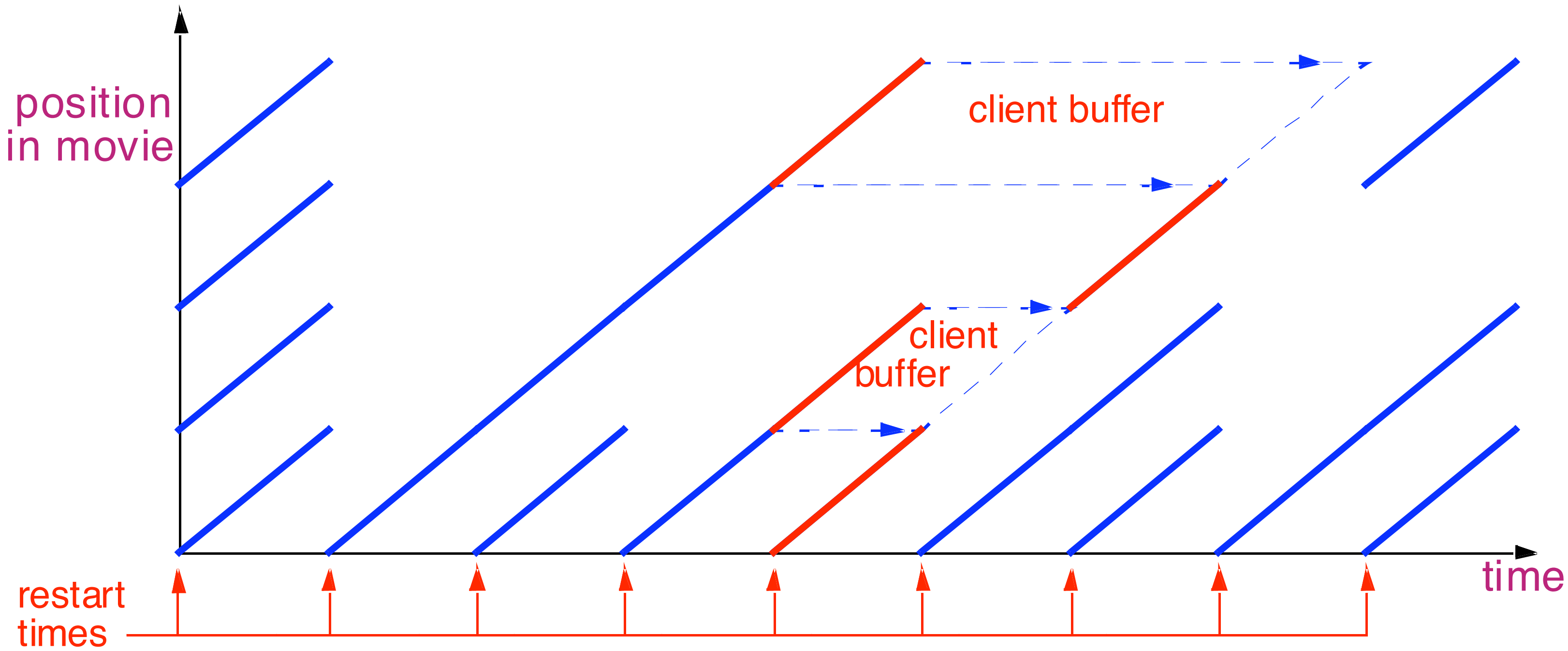
Optimized delivery scheduling

- Prescheduled Delivery
  - No back-channel
  - Non-linear transmission
  - Client buffering and re-ordering
  - Video segmentation
  - Examples
    - Staggered broadcasting, Pyramid b., Skyscraper b., Fast b., Pagoda b., Harmonic b., …
  - Typical
    - Good theoretic performance
    - High resource requirements
    - Single point of failure

Optimized delivery scheduling



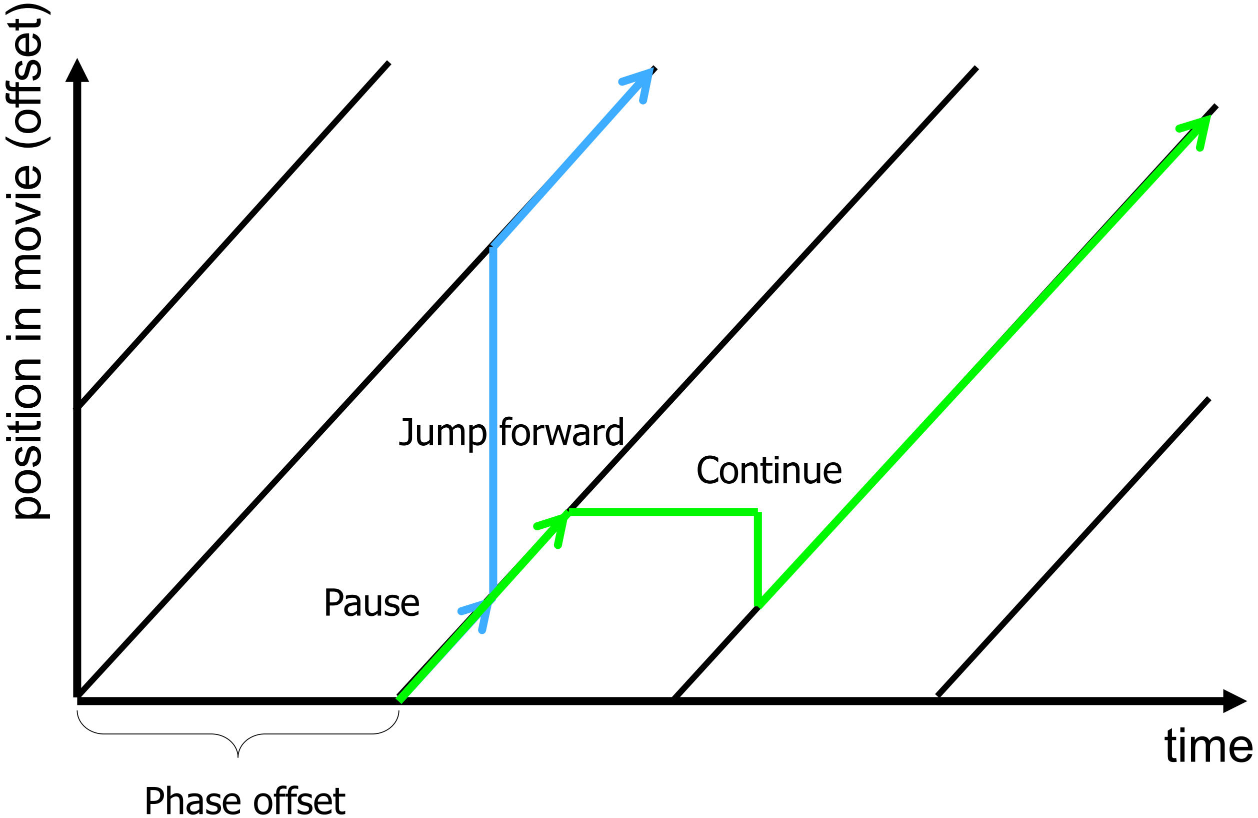
Prescheduled Delivery



- Arrivals are not relevant
  - users can start viewing at each interval start

Staggered Broadcasting

[Almeroth, Ammar 1996]



- Near Video-on-Demand
  - Applied in real systems
  - Limited interactivity is possible (jump, pause)
  - Popularity can be considered → change phase offset

Pyramid Broadcasting

[Viswanathan, Imielinski 1996]

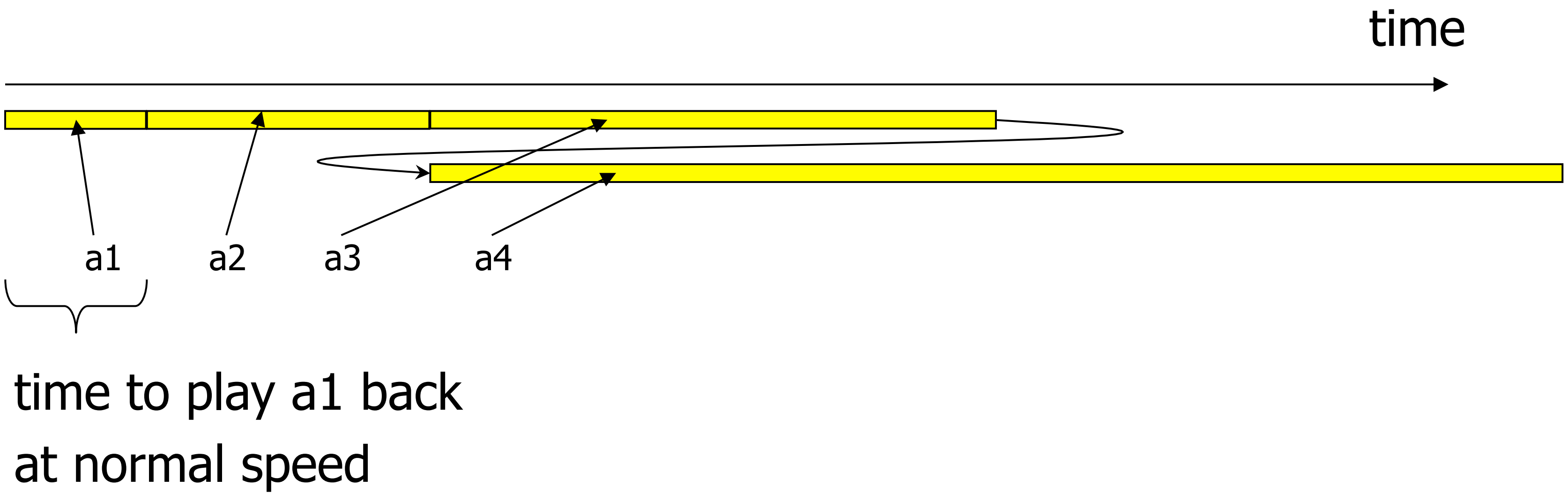
- Idea
  - Fixed number of HIGH-bitrate channels C_i with bitrate B
  - Variable size segments $a_1 \dots a_n$
  - One segment repeated per channel
  - Segment length is growing exponentially
  - Several movies per channel, total of m movies (constant bitrate 1)
- Operation
  - Client waits for the next segment a_1 (on average ½ len(d_1))
  - Receives following segments as soon as linearly possible
- Segment length
  - Size of segment a_i:

    $$len(a_i) = \alpha^{i-1} \cdot len(a_1)$$
  - α is limited
  - $\alpha > 1$ to build a pyramid
  - $\alpha \leq B/m$ for sequential viewing
  - $\alpha = 2.5$ considered good value
- Drawback
  - Client buffers more than 50% of the video
  - Client receives all channels concurrently in the worst case

Pyramid Broadcasting

- Pyramid broadcasting with B=4, m=2, α=2
- Movie a

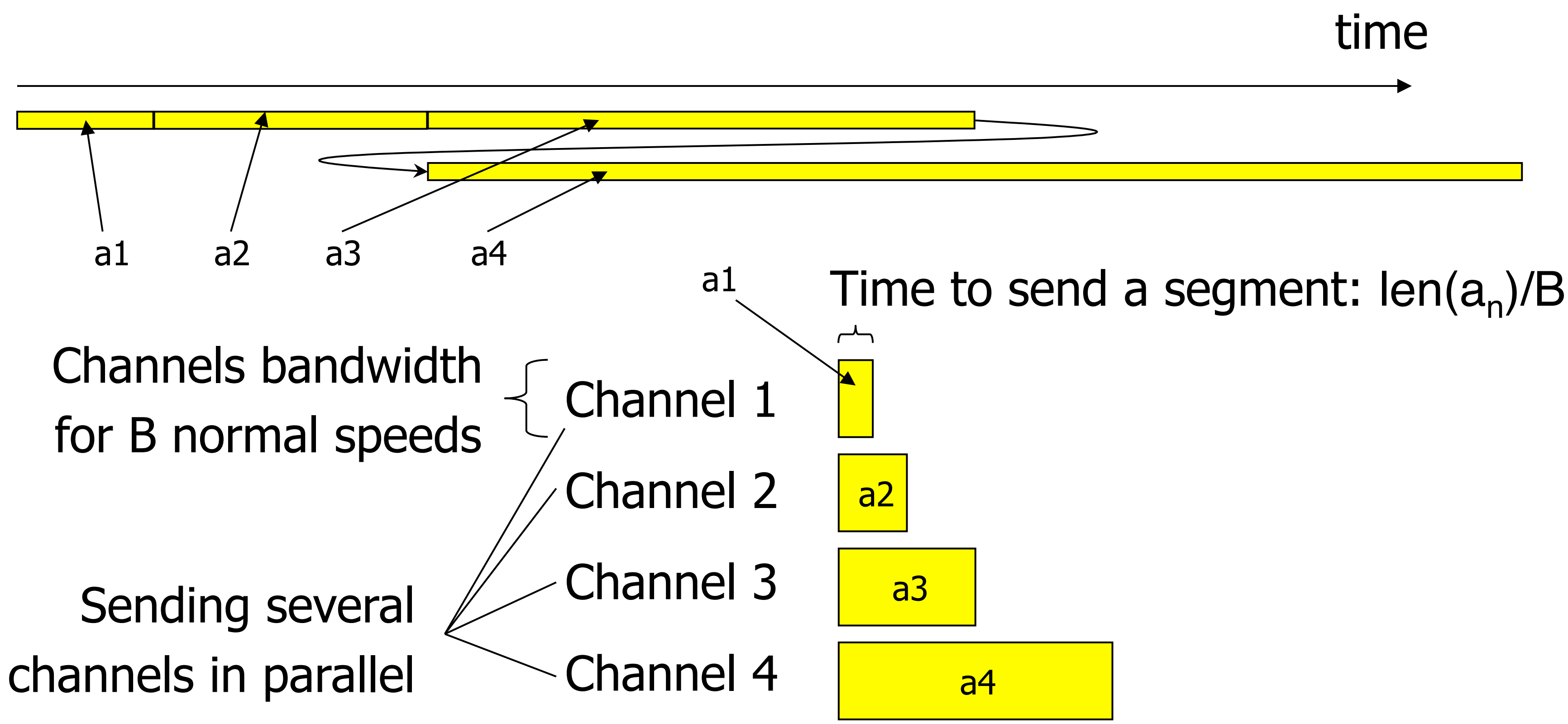
$$len(a4) = \alpha \cdot len(a3) = \alpha^2 \cdot len(a2) = \alpha^3 \cdot len(a1)$$



Pyramid Broadcasting

- Pyramid broadcasting with B=4, m=2, α=2
- Movie a

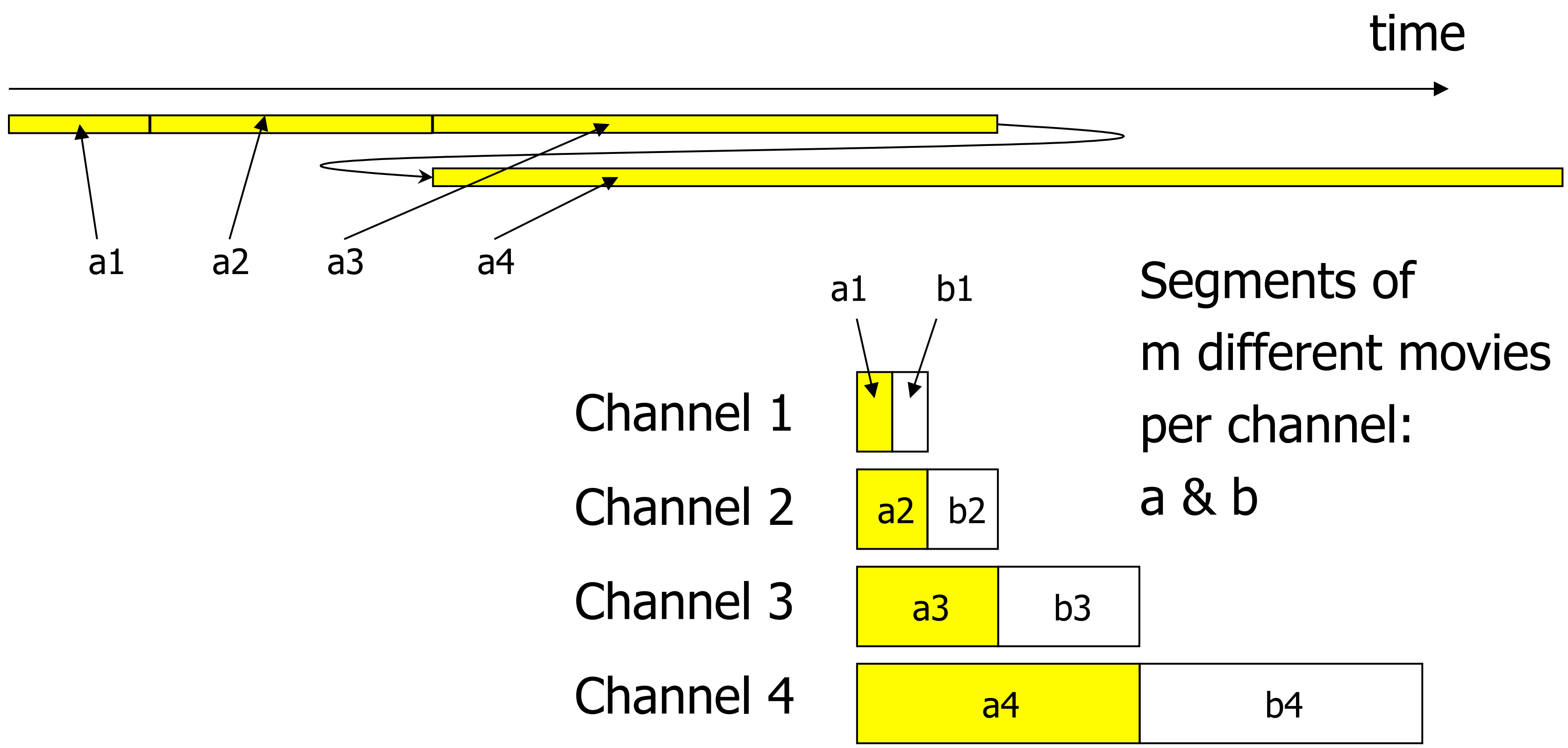
$$len(a4) = \alpha \cdot len(a3) = \alpha^2 \cdot len(a2) = \alpha^3 \cdot len(a1)$$



Pyramid Broadcasting

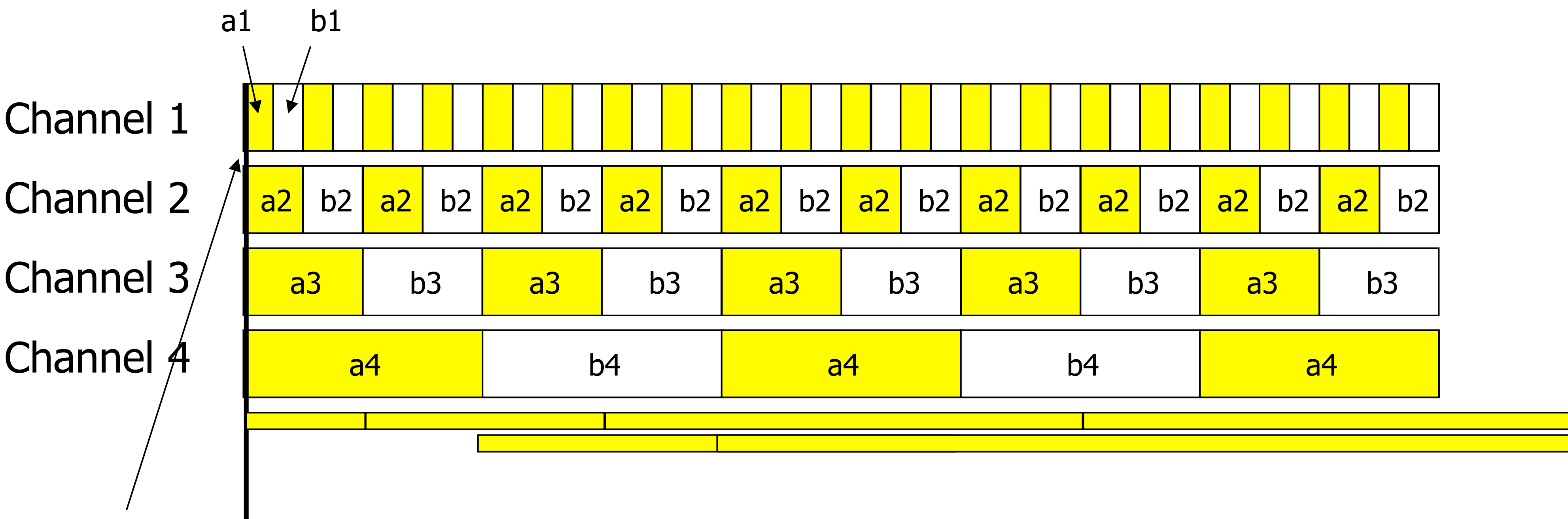
- Pyramid broadcasting with B=4, m=2, α=2
- Movie a

$$len(a4) = \alpha \cdot len(a3) = \alpha^2 \cdot len(a2) = \alpha^3 \cdot len(a1)$$



Pyramid Broadcasting

- Pyramid broadcasting with B=4, m=2, α=2

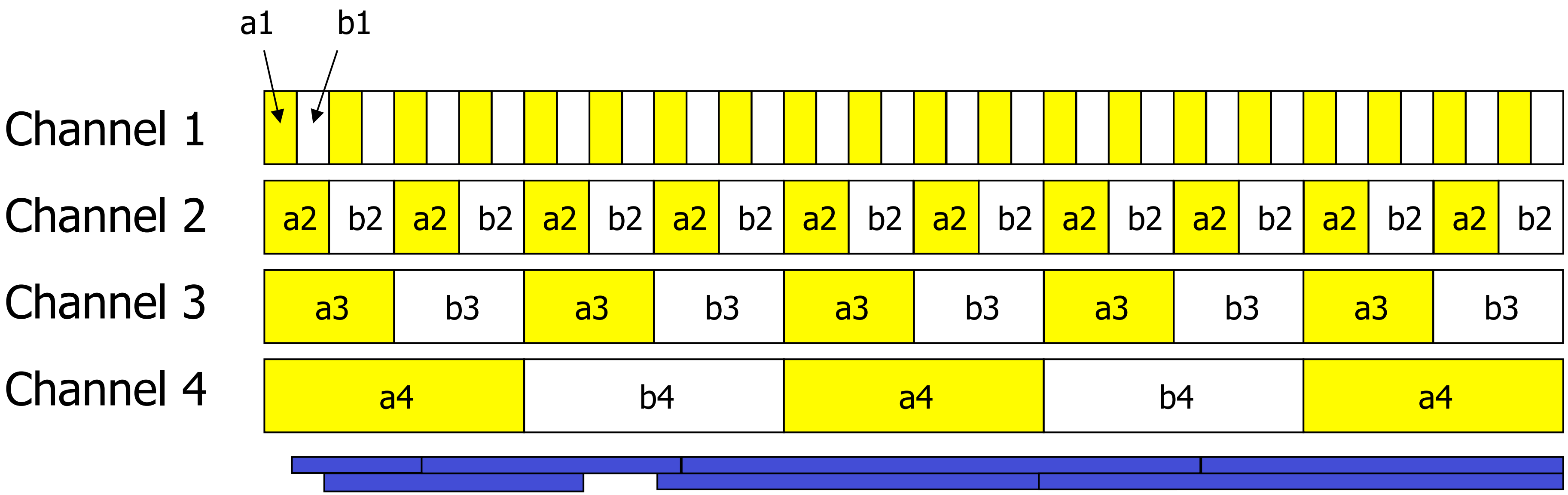


request for a arrives

client starts receiving and playing a1
client starts receiving and playing a2
client starts receiving a3 client starts playing a3
client starts receiving a4 client starts playing a4

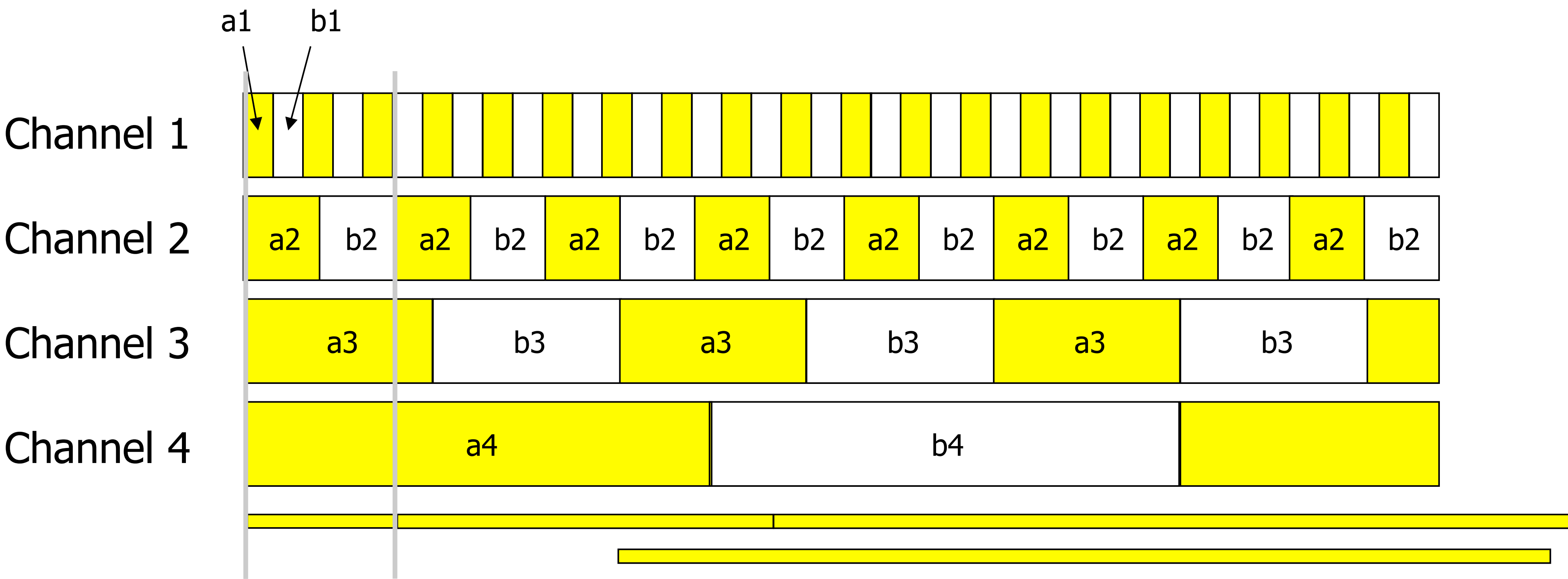
Pyramid Broadcasting

- Pyramid broadcasting with B=4, m=2, α=2



Pyramid Broadcasting

- Pyramid broadcasting with B=5, m=2, α=2.5

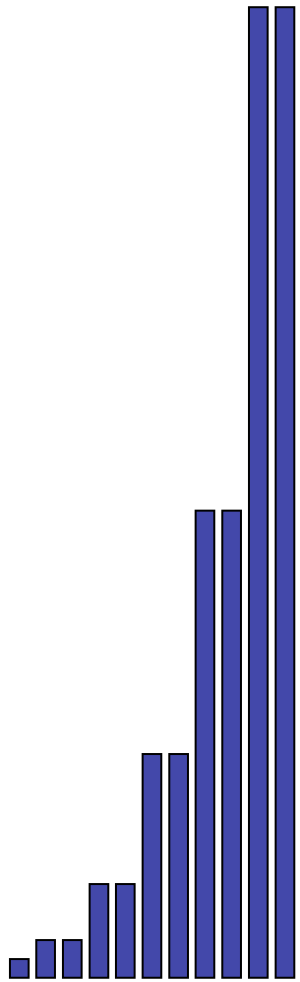


- Choose m=1
  - Less bandwidth at the client and in multicast trees
  - At the cost of multicast addresses

Skyscraper Broadcasting

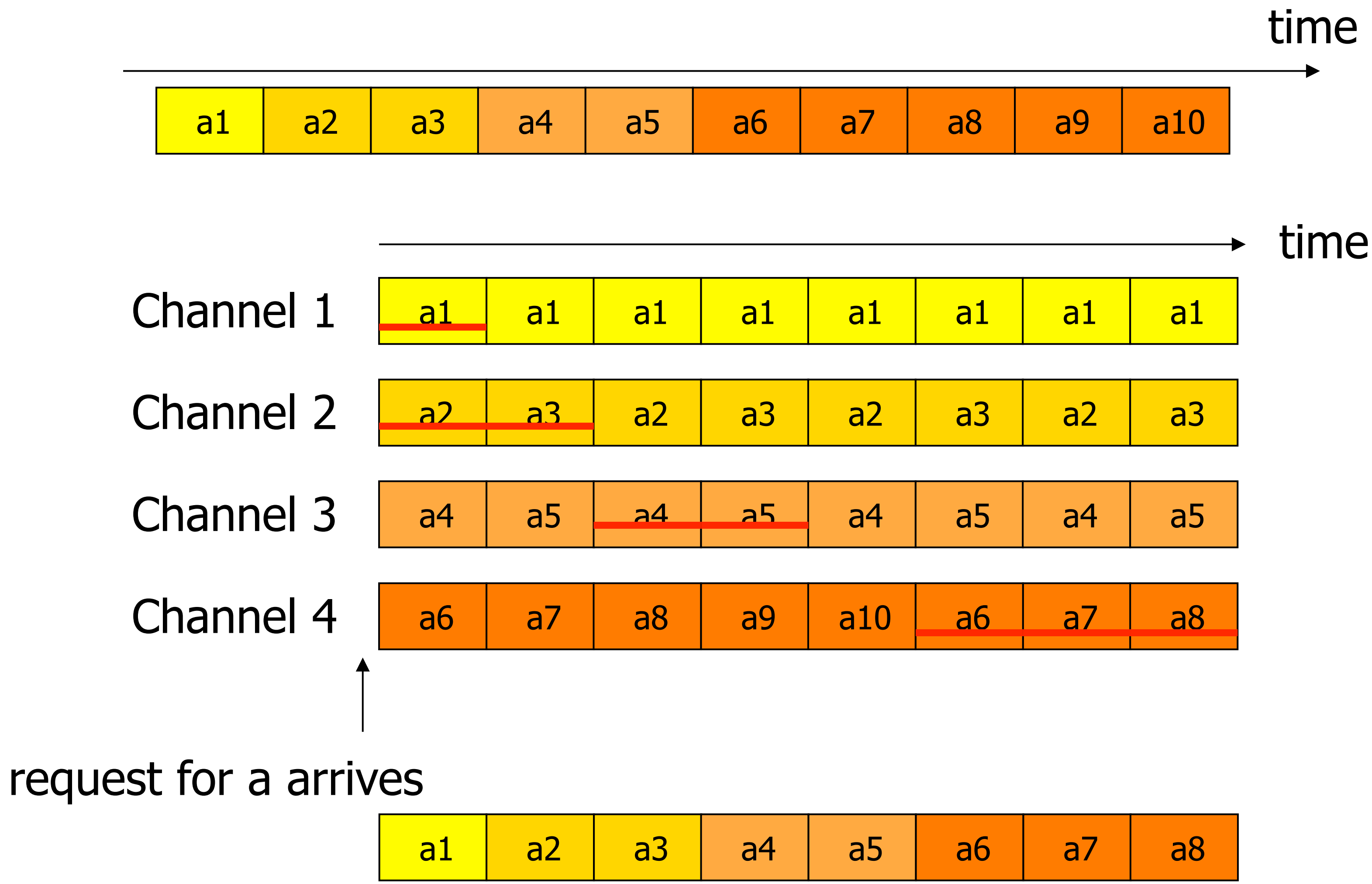
[Hua, Sheu 1997]

- Idea
  - Fixed size segments
  - More than one segment per channel
  - Channel bandwidth is playback speed
  - Segments in a channel keep order
  - Channel allocation series
    - *1,2,2,5,5,12,12,25,25,52,52, ...*
  - Client receives at most 2 channels
  - Client buffers at most 2 segments



- Operation
  - Client waits for the next segment a1
  - Receive following segments as soon as linearly possible

Skyscraper Broadcasting



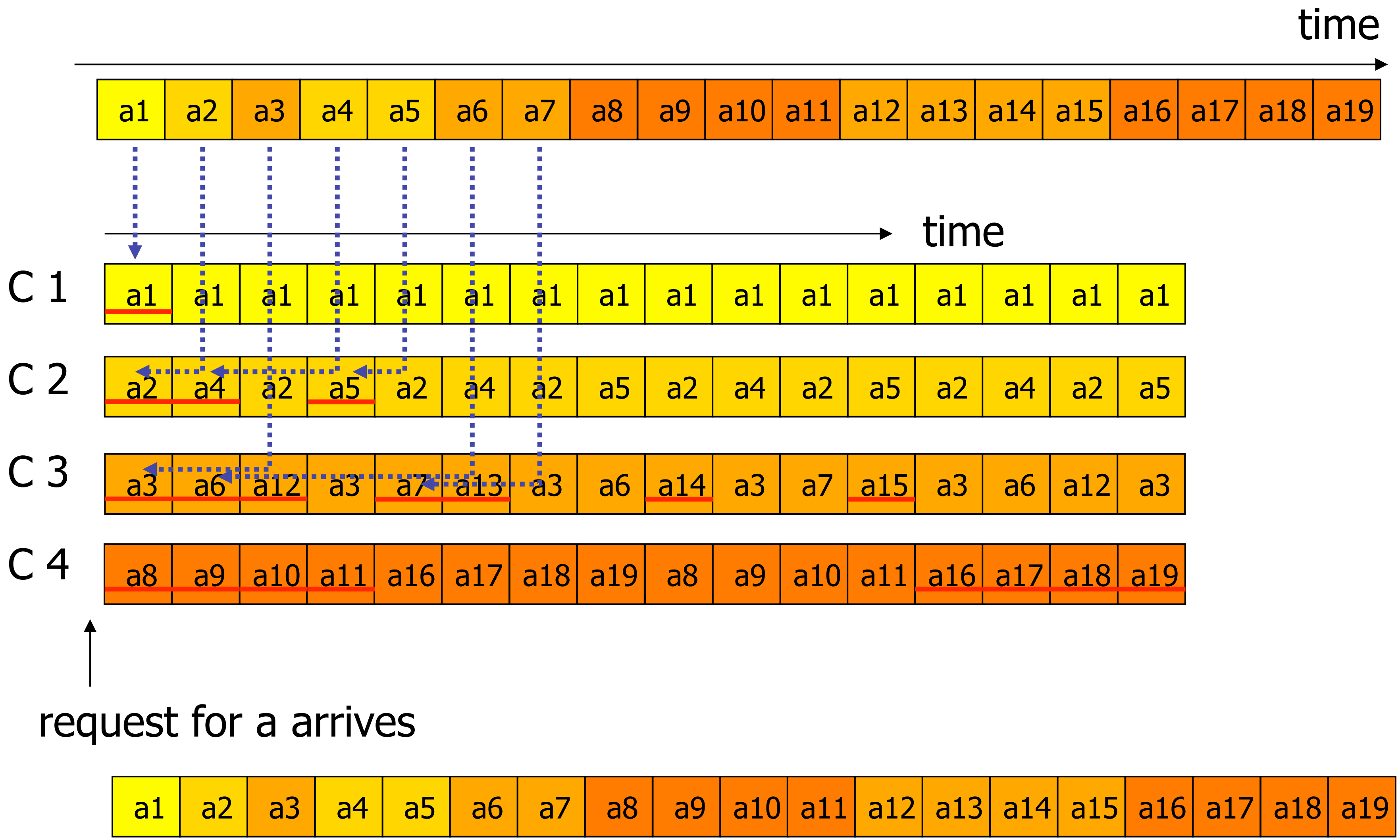
Skyscraper Broadcasting

Pagoda Broadcasting

[Paris, Carter, Long 1999]

- Pagoda Broadcasting
  - Channel allocation series
    - *1,3,5,15,25,75,125*
  - Segments are ***not*** broadcast linearly
  - Consecutive segments appear on pairs of channels
  - Client must receive up to 7 channels
    - For more channels, a different series is needed !
  - Client must buffer 45% of all data
  - Based on the following
    - Segment 1 – needed every round
    - Segment 2 – needed at least every 2nd round
    - Segment 3 – needed at least every 3rd round
    - Segment 4 – needed at least every 4th round
    - …

Pagoda Broadcasting



Pagoda Broadcasting

Harmonic Broadcasting

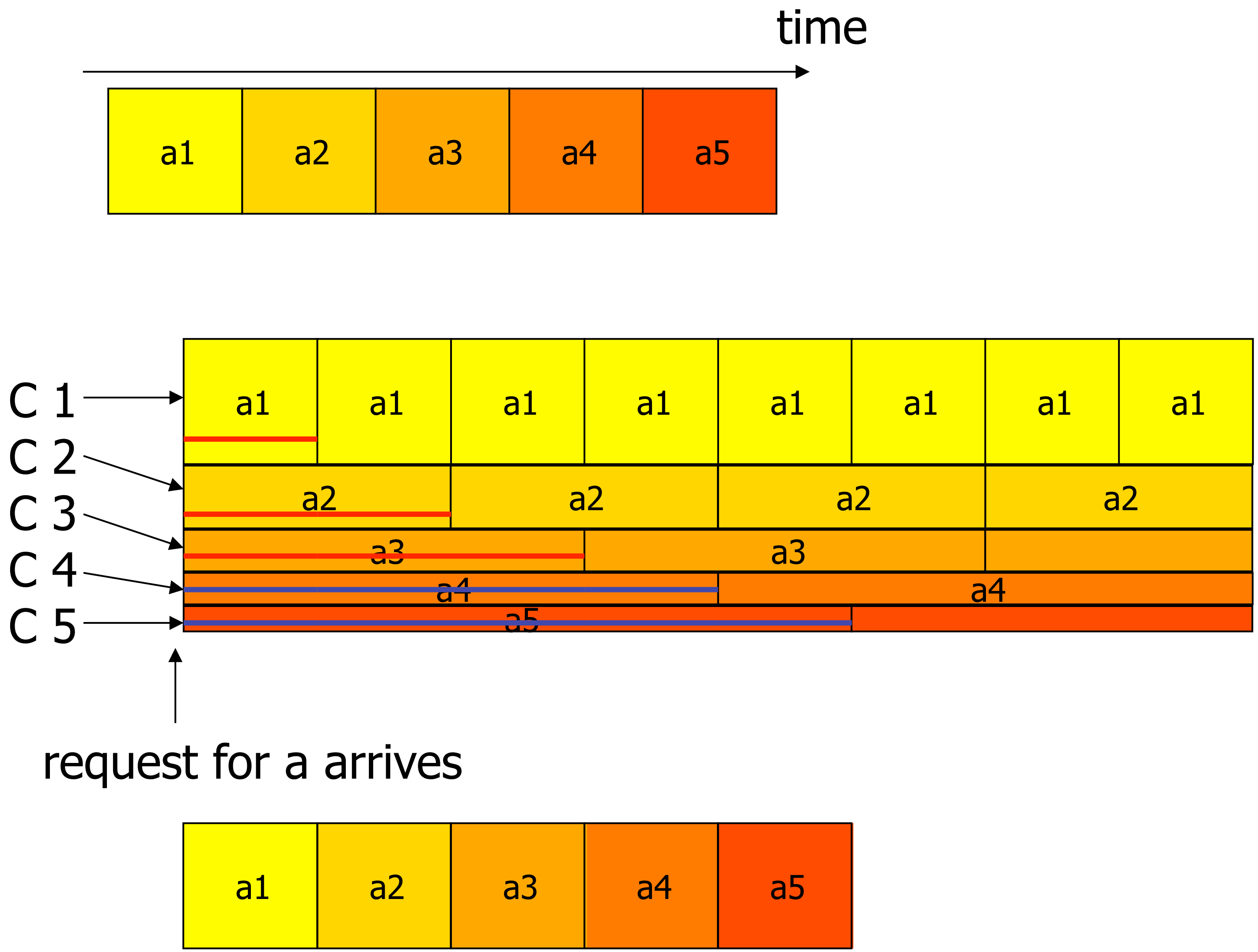
[Juhn, Tseng 1997]

- Idea
  - Fixed size segments
  - One segment repeated per channel
  - Later segments can be sent at lower bitrates
  - Receive all other segments concurrently
  - Harmonic series determines bitrates
    - Bitrate(a_i) = Playout-rate(a_i)/i
    - Bitrates 1/1, 1/2, 1/3, 1/4, 1/5, 1/6, …

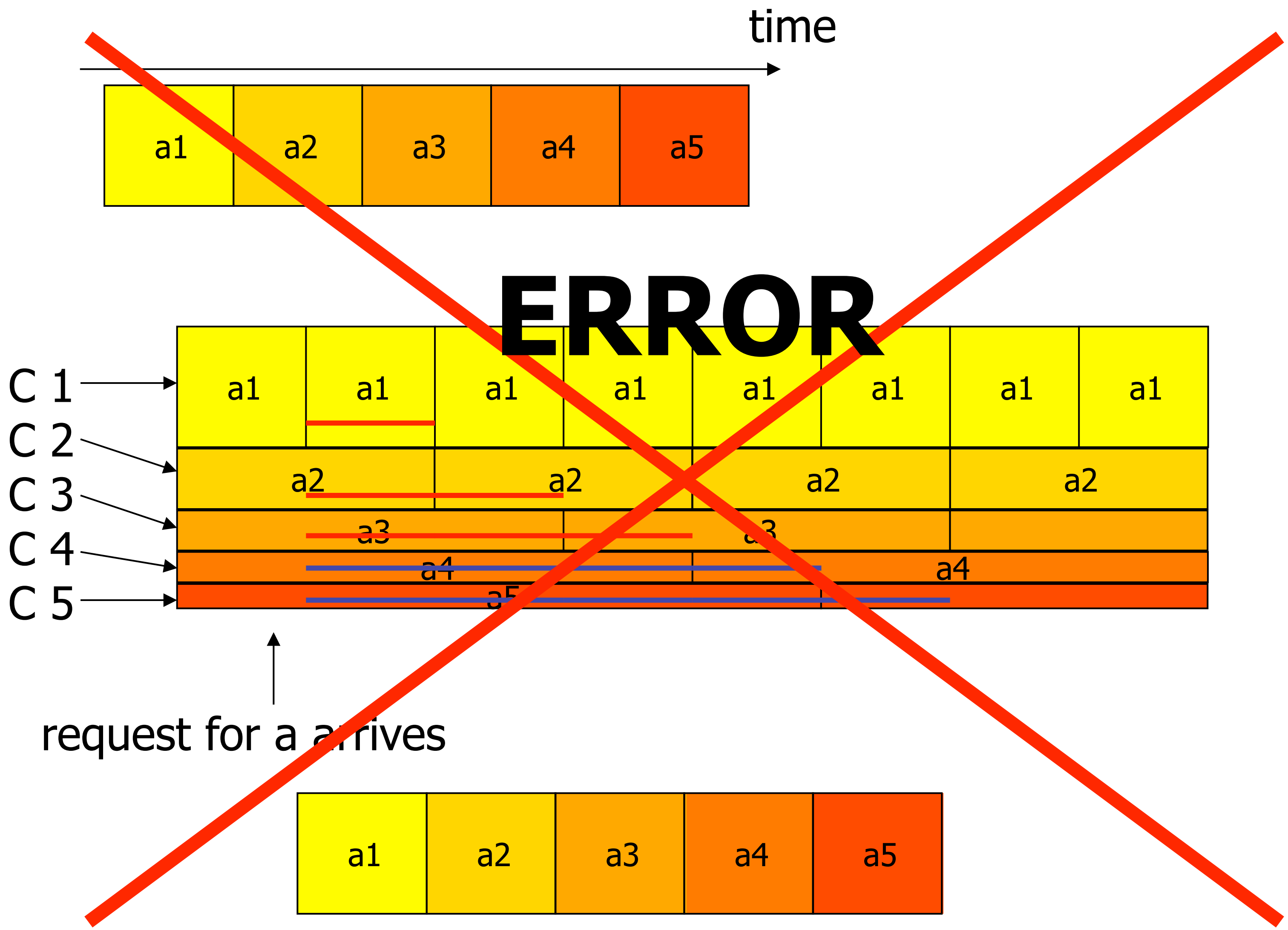
- Consideration
  - Size of a_1 determines client start-up delay
  - Growing number of segments allows smaller a_1
  - Required server bitrate grows very slowly with number of segments

- Drawback
  - Client buffers about 37% of the video for >=20 channels
  - (Client must re-order small video portions)
  - Complex memory cache for disk access necessary

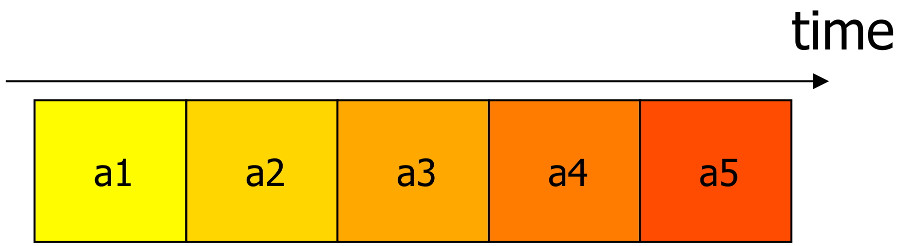
Harmonic Broadcasting

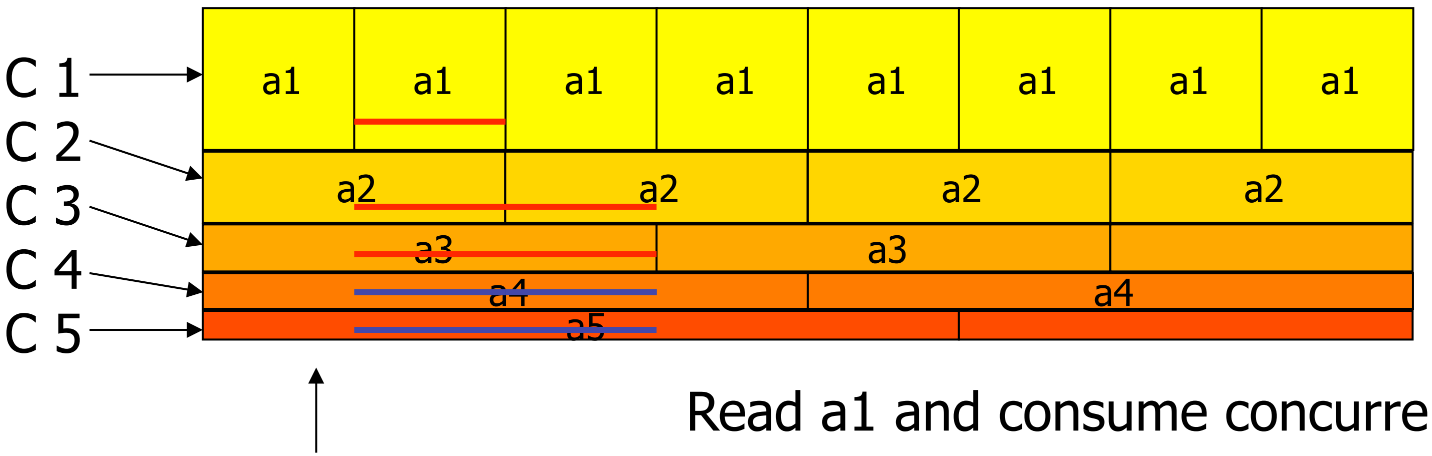


Harmonic Broadcasting



Harmonic Broadcasting

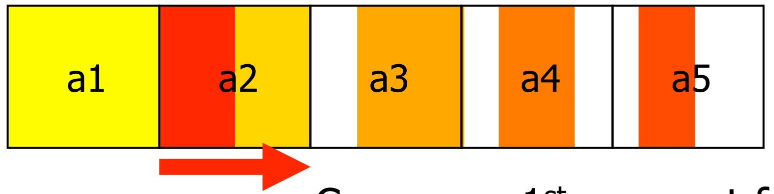




request for a arrives

Read a1 and consume concurrently ☺

Read rest of a2 and consume concurrently ☹



Consumes 1st segment faster than it is received !!!

Harmonic Broadcasting: Bugfix

[By Paris, Long, …]

- Delayed Harmonic Broadcasting
  - Wait until a_1 is fully buffered
  - All segments will be completely cached before playout
  - Fixes the bug in Harmonic Broadcasting

or

- Cautious Harmonic Broadcasting
  - Wait an additional a_1 time
  - Starts the harmonic series with a_2 instead of a_1
  - Fixes the bug in Harmonic Broadcasting

Prescheduled Delivery Evaluation

- Techniques
  - Video segmentation
  - Varying transmission speeds
  - Re-ordering of data
  - Client buffering

- Advantage
  - Achieves server resource reduction

- Problems
  - Tends to require complex client processing
  - May require large client buffers
  - Is incapable (or not proven) of working with user interactivity
    - Current research to work with VCR controls
  - Guaranteed bandwidth required

Type III:
Client Side Caching

Optimized delivery scheduling

- Client Side Caching
  - On-demand delivery
  - Client buffering
  - Multicast complete movie
  - Unicast start of movie for latecomers (patch)

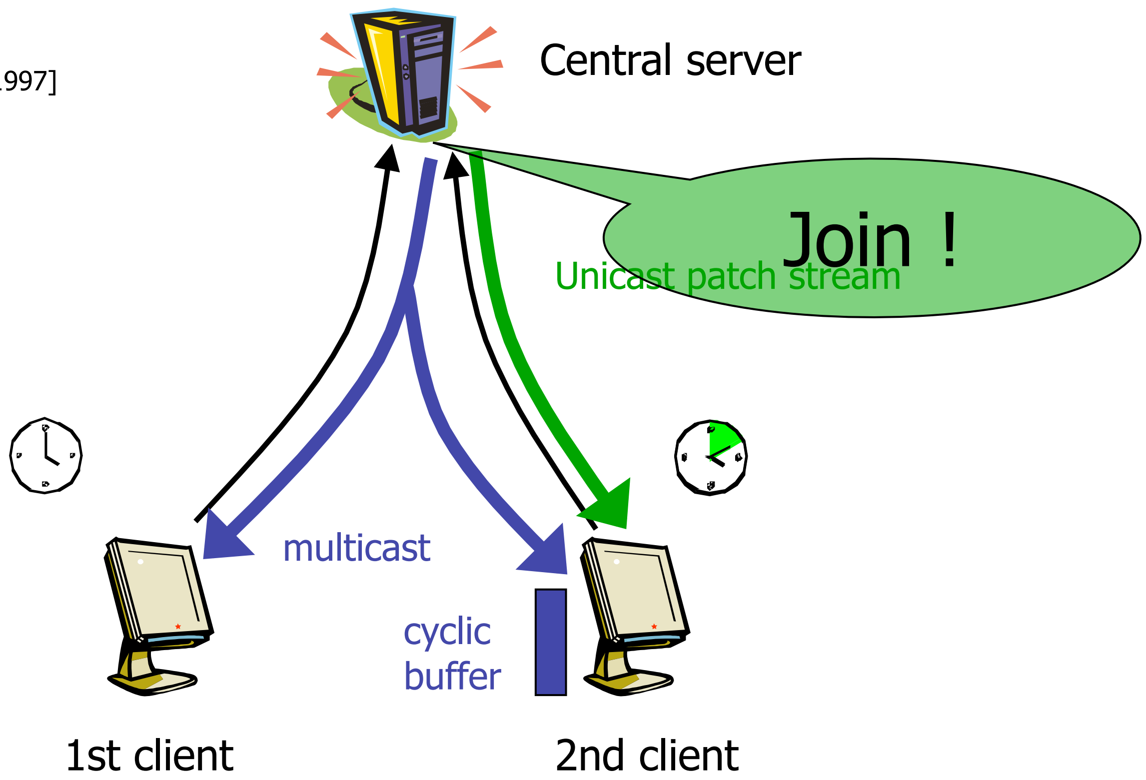
- Examples
  - Stream Tapping, Patching, Hierarchical Streaming Merging, …

- Typical
  - Considerable client resources
  - Single point of failure

Optimized delivery scheduling

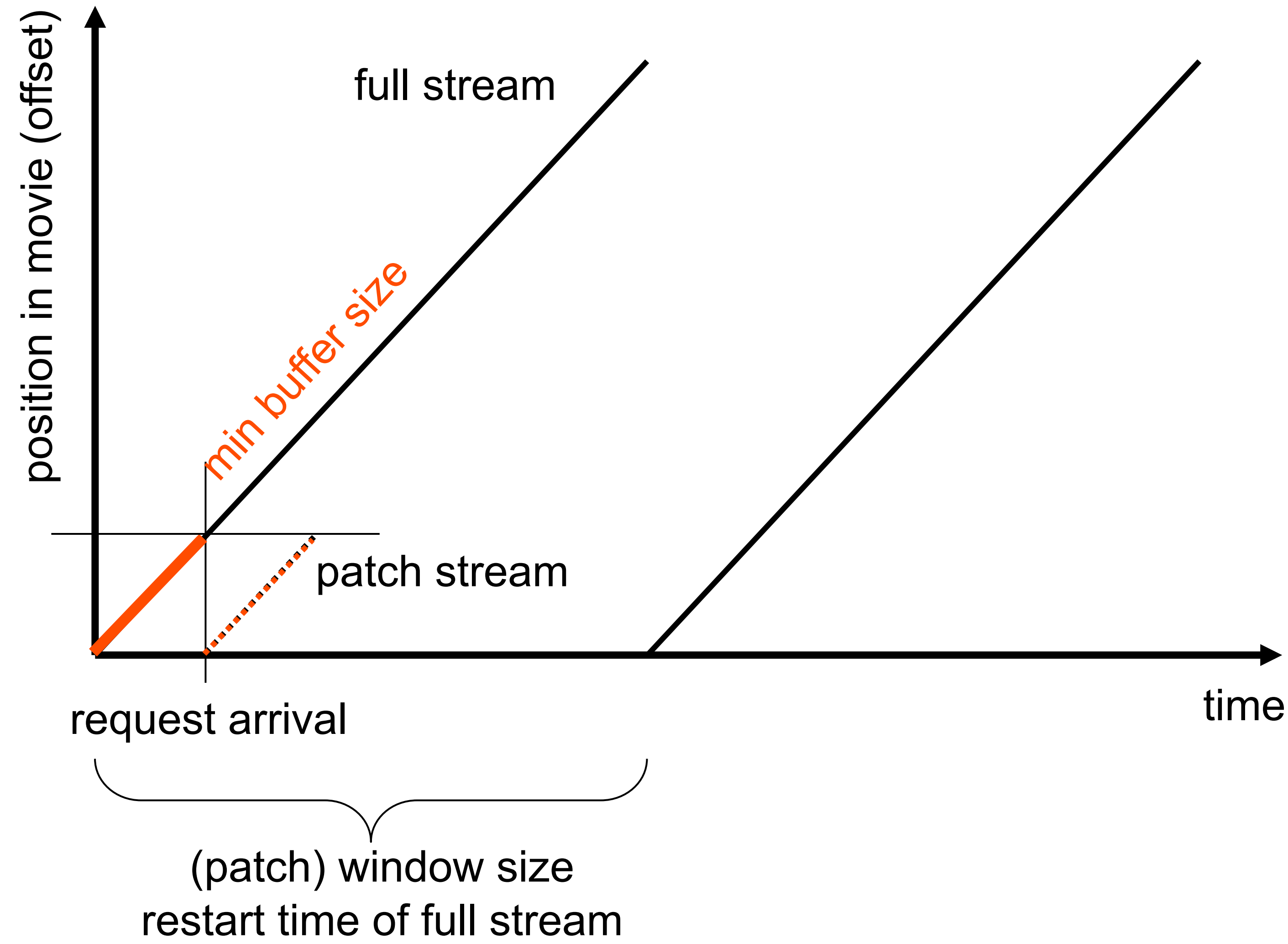
- Patching

[Hua, Cai, Sheu 1998,
also as Stream Tapping Carter, Long 1997]

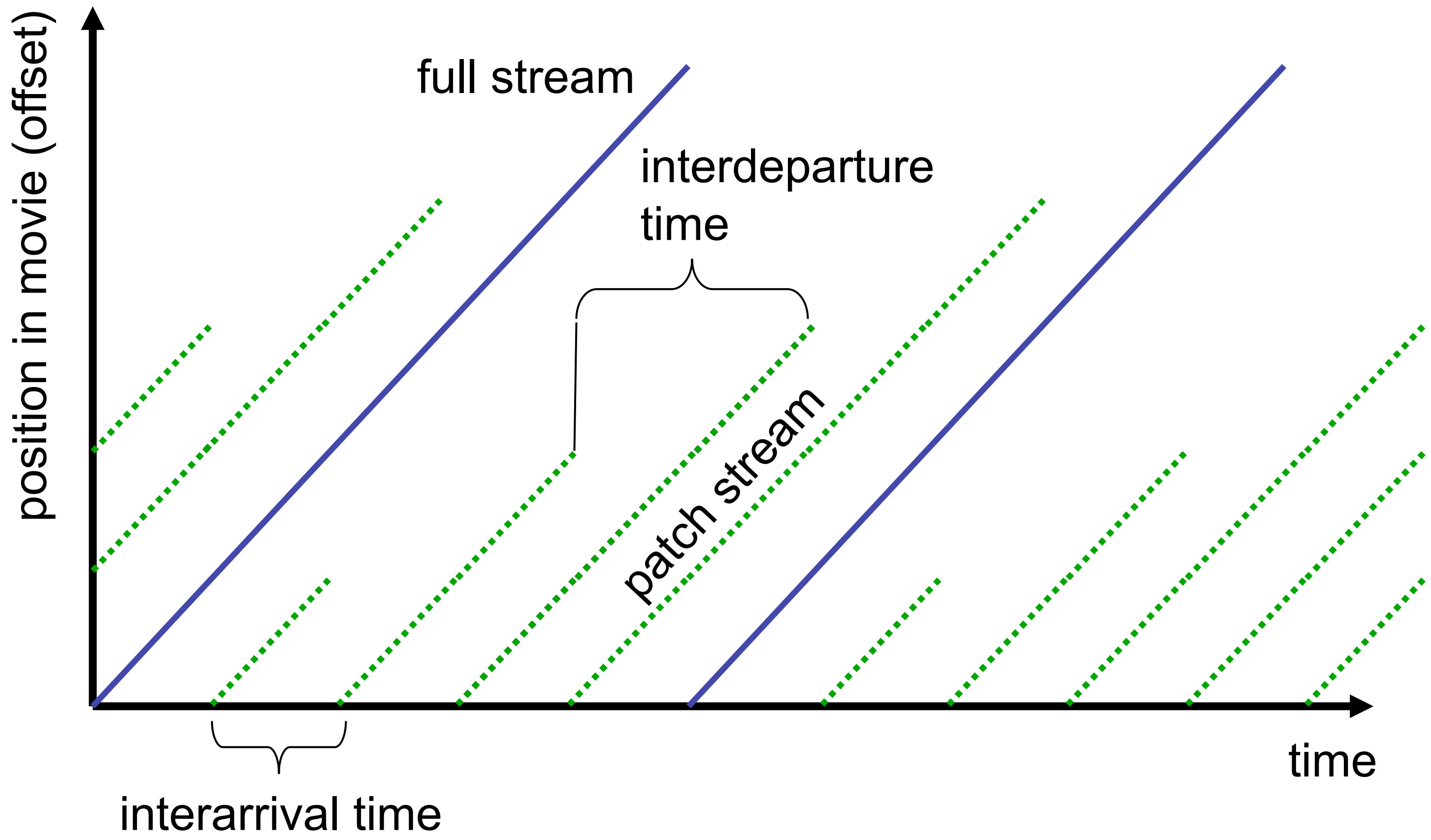


- Server resource optimization is possible

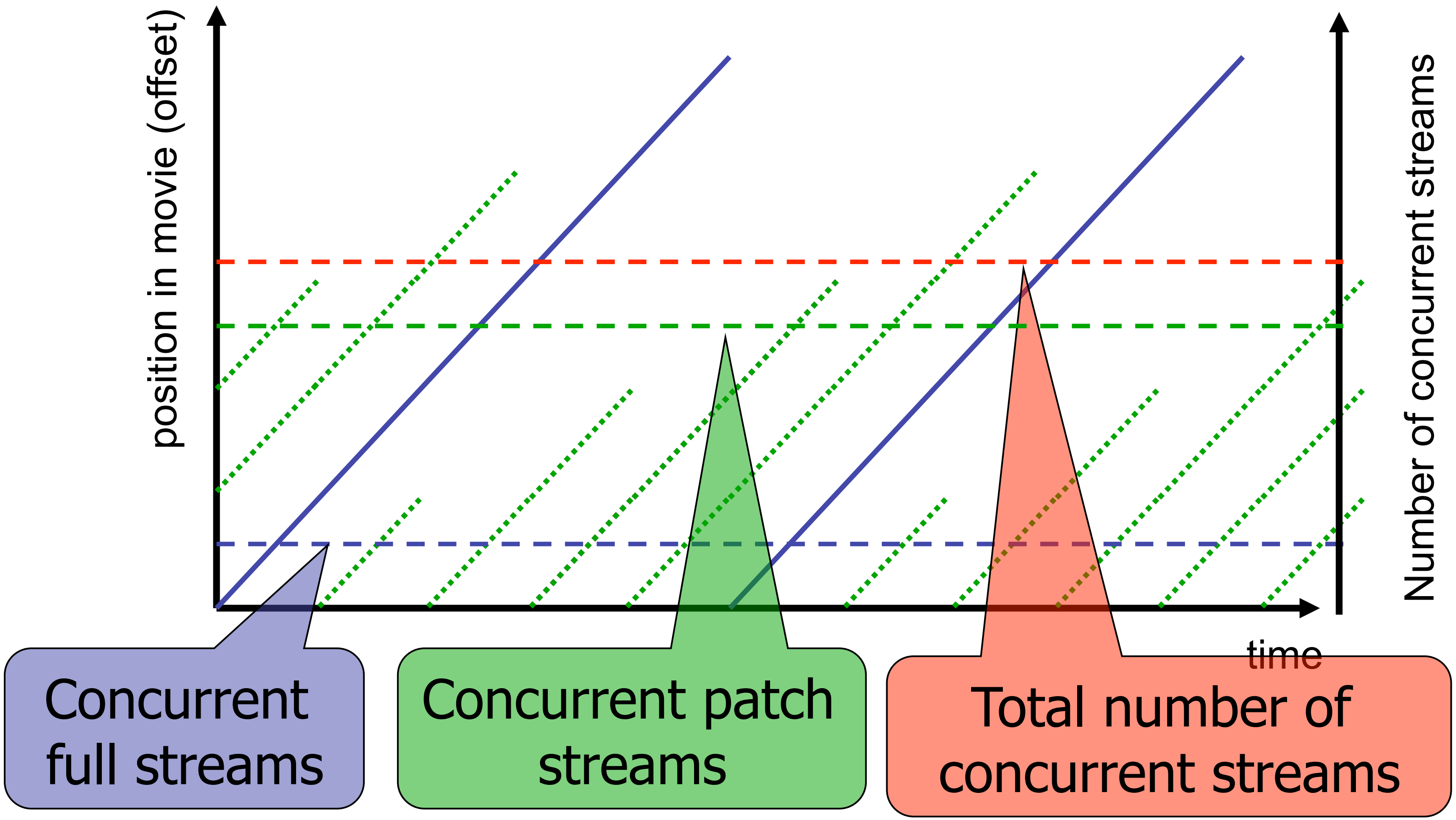
Optimized delivery scheduling



Optimized delivery scheduling

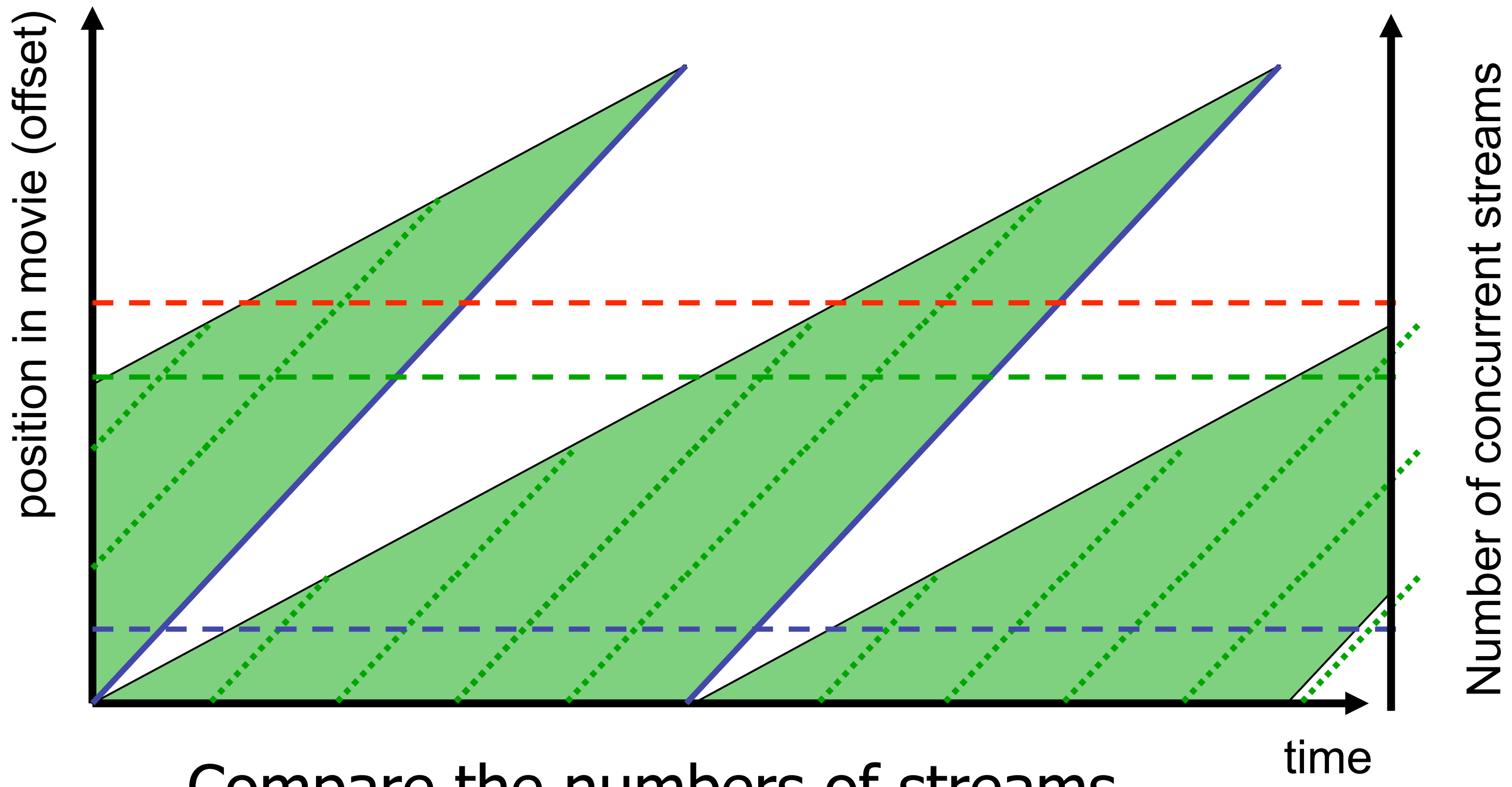


Optimized delivery scheduling



The average number of patch streams is constant if the arrival process is a Poisson process

Optimized delivery scheduling

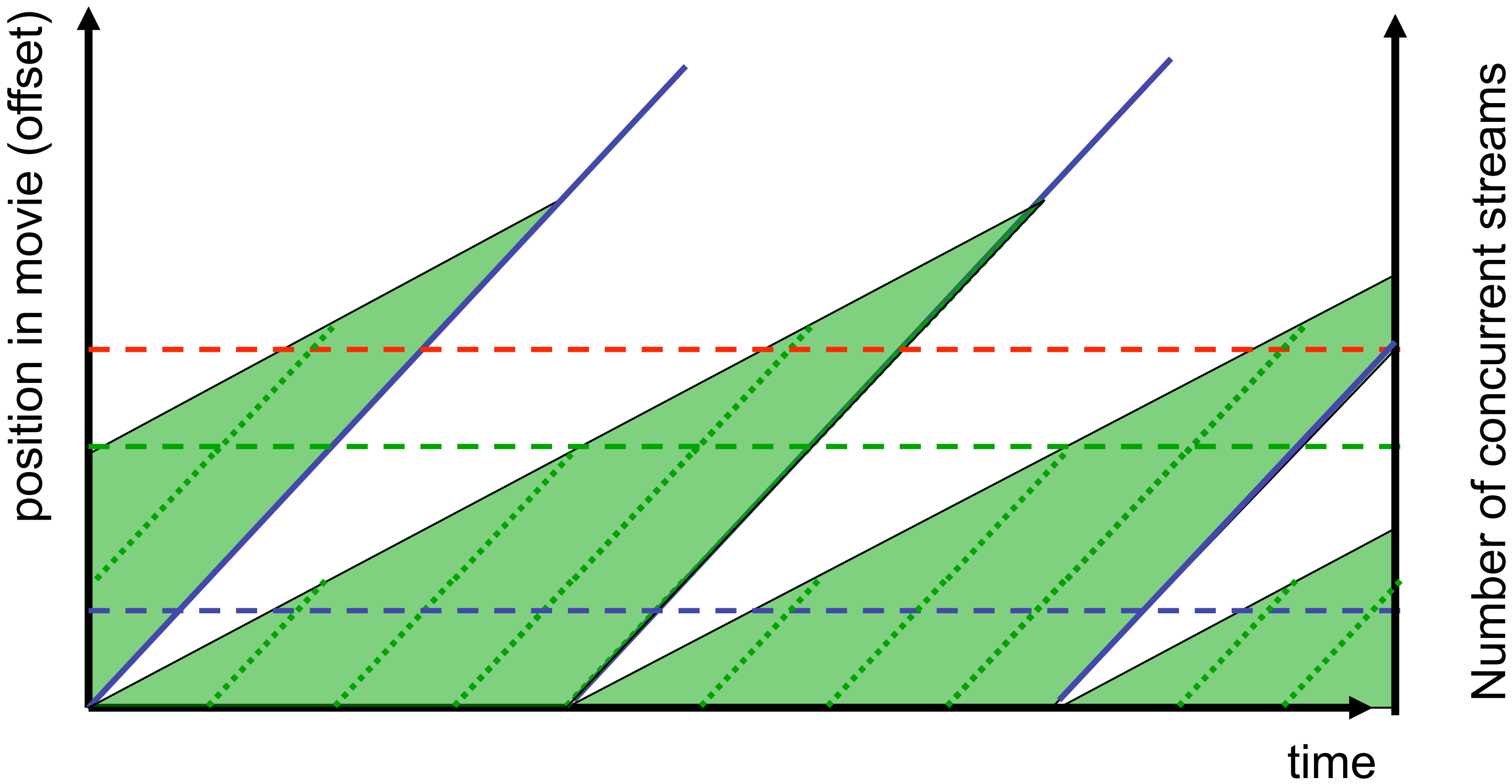


Compare the numbers of streams

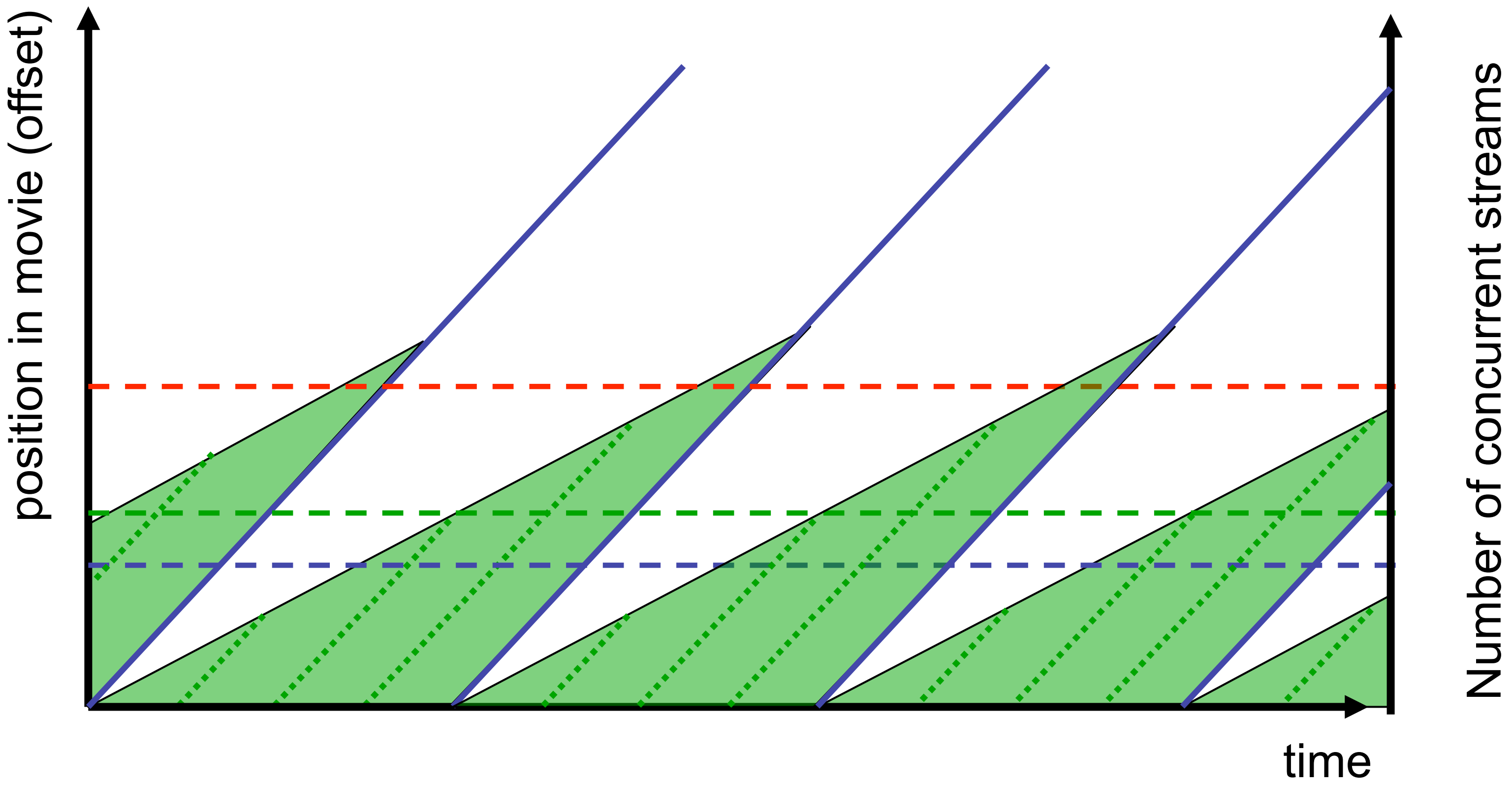
Shown patch streams are just examples

But always: patch end times on the edge of a triangle

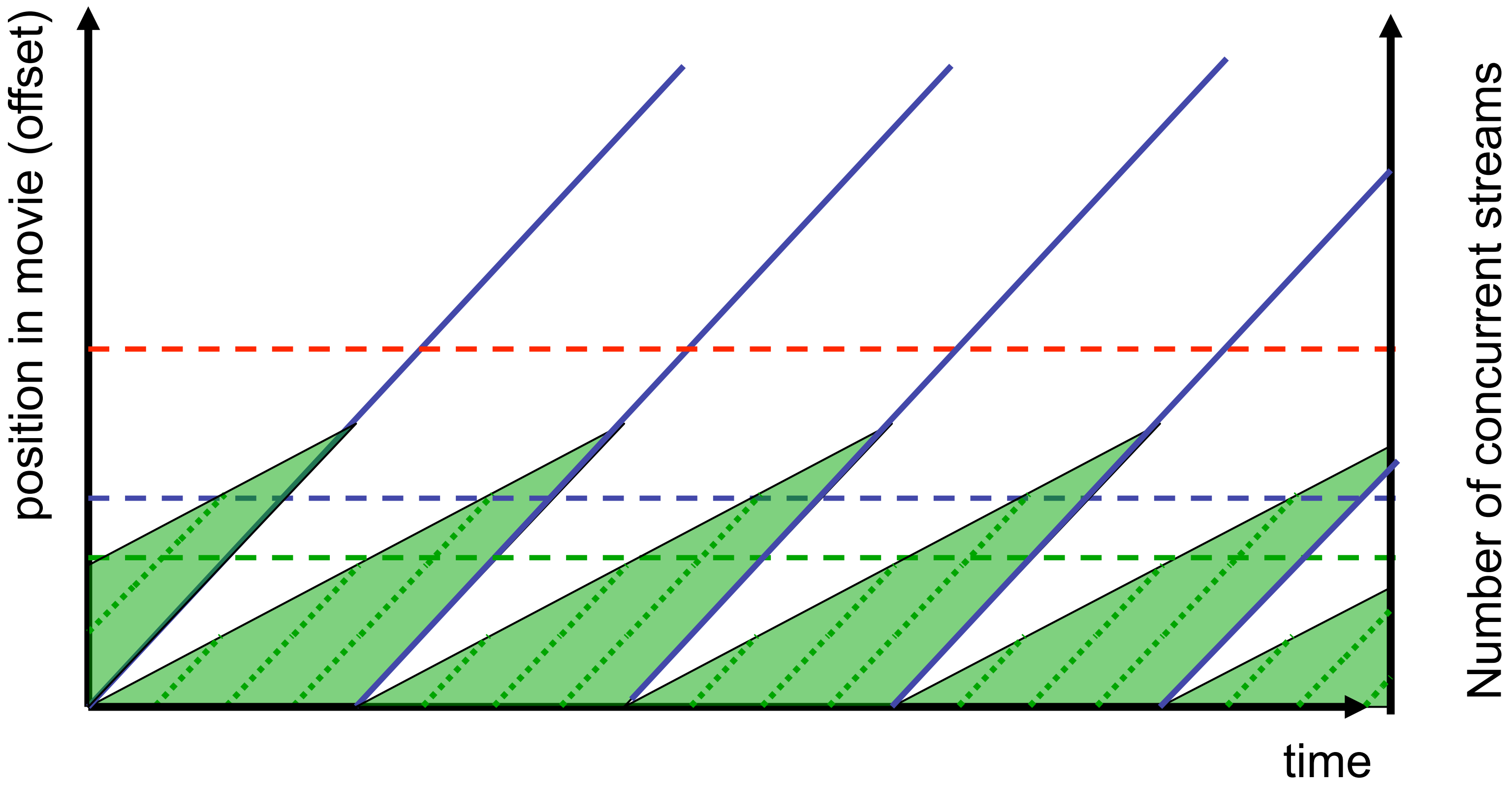
Optimized delivery scheduling



Optimized delivery scheduling



Optimized delivery scheduling



Optimized delivery scheduling

- Minimization of server load
- Minimum average number of concurrent streams
- Depends on
  - F — movie length
  - Δ_U — expected interarrival time
  - Δ_M — patching window size
  - C_U — cost of unicast stream at server
  - C_M — cost of multicast stream at server
  - S_U — setup cost of unicast stream at server
  - S_M — setup cost of multicast stream at server

Optimized delivery scheduling

- Optimal patching window size
  - For identical multicast and unicast setup costs
  - For different multicast and unicast setup costs

$$\Delta_M = \sqrt{2 \cdot F \cdot \Delta_U}$$

$$\Delta_M = \sqrt{2 \cdot \frac{S_M + C_M F}{C_U} \cdot \Delta_U}$$

Patching window size

Movie length

Interarrival time

- Servers can estimate Δ_U
  - And achieve massive saving

Unicast	7445 Mio $
Greedy patching	3722 Mio $
λ-patching	375 Mio $

Based on prices by Deutsche Telekom, IBM and Real Networks from 1997

HMSM

[Eager, Vernon, Zahorjan 2001]

- Hierarchical Multicast Stream Merging
- Key ideas
  - Each data transmission uses multicast
  - Clients accumulate data faster than their playout rate
    - multiple streams
    - accelerated streams
  - Clients are merged in large multicast groups
  - Merged clients continue to listen to the same stream to the end
- Combines
  - Dynamic skyscraper
  - Piggybacking
  - Patching

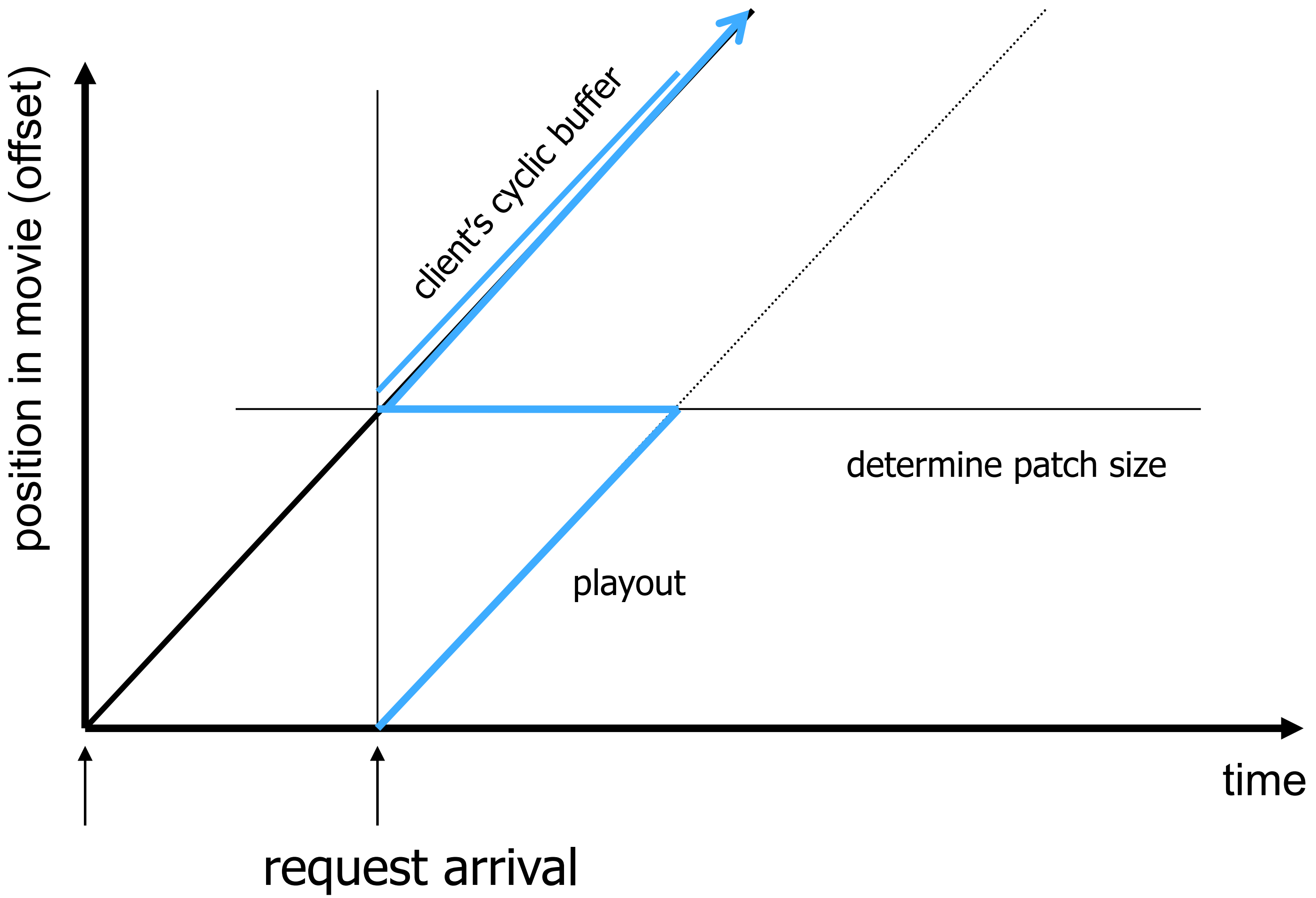
HMSM

- Always join the closest neighbour

- HMSM(n,1)
  - Clients can receive up to n streams in parallel
- HMSM(n,e)
  - Clients can receive up to n full-bandwidth streams in parallel
  - but streams are delivered at speeds of e, where e « 1
- Basically
  - HMSM(n,1) is another recursive application of patching

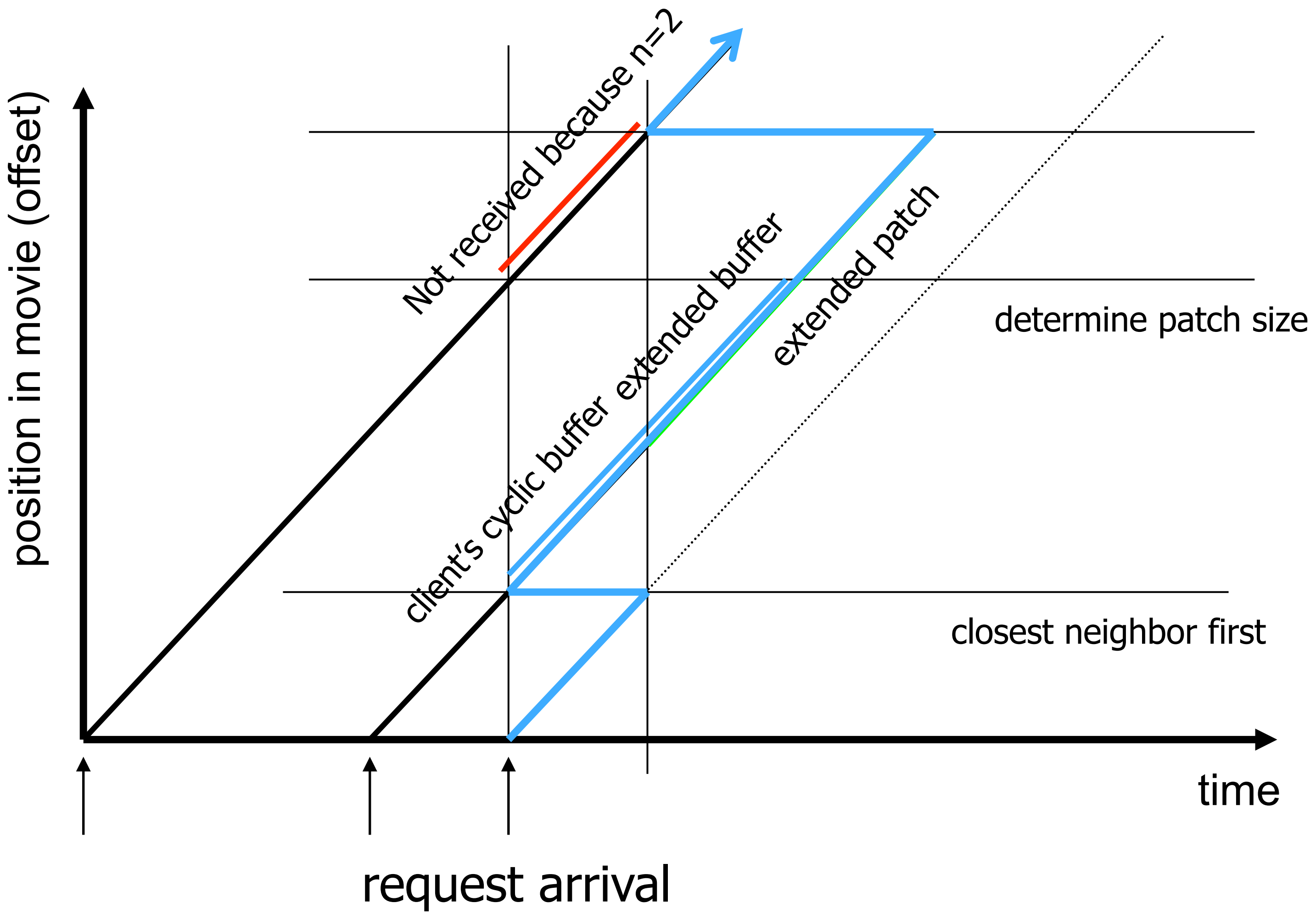
HMSM(2,1)

HMSM(2,1): max 2 streams, 1 playout speed each

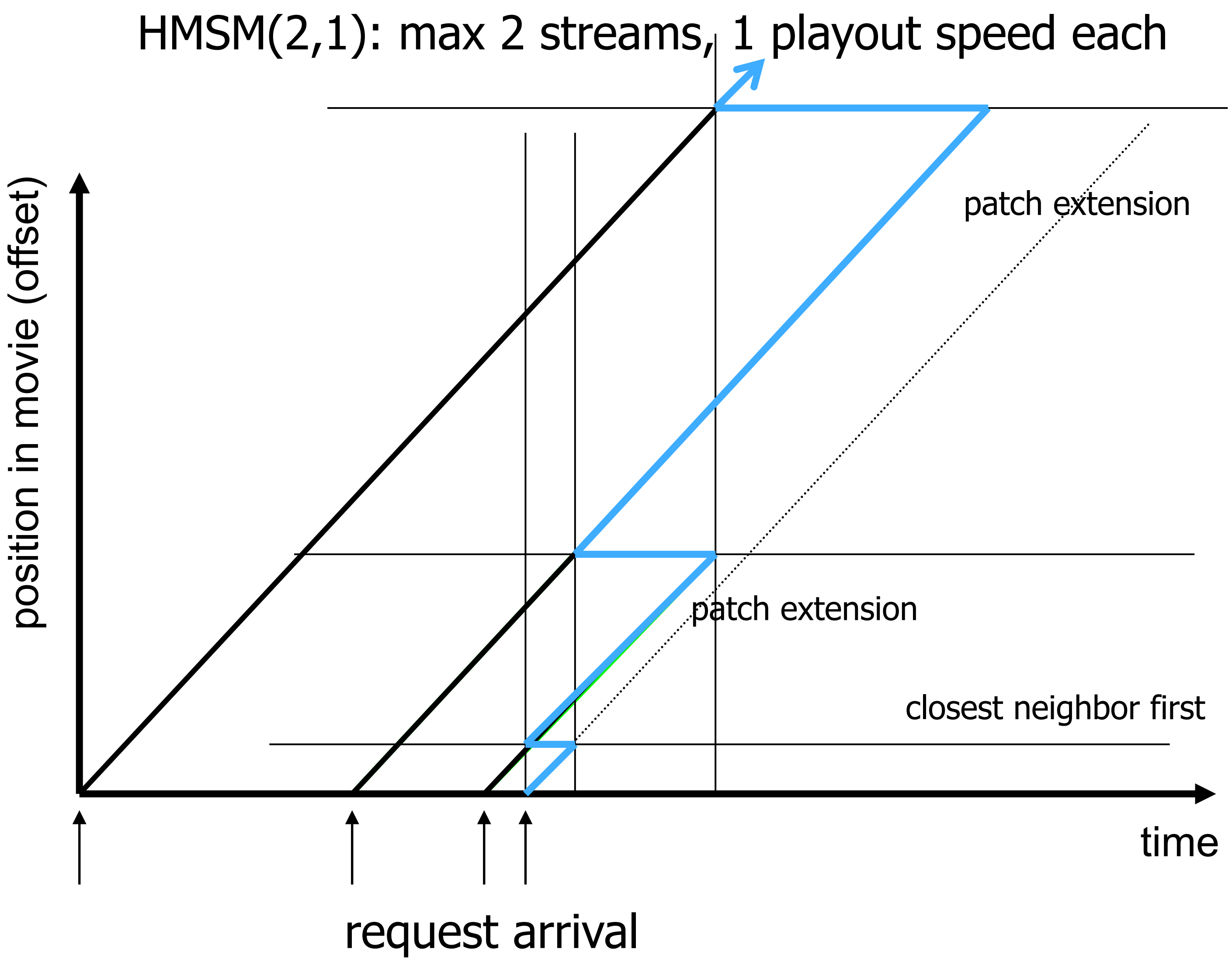


HMSM(2,1)

HMSM(2,1): max 2 streams, 1 playout speed each



HMSM(2,1)


HMSM(2,1): max 2 streams, 1 playout speed each
position in movie (offset)
patch extension
patch extension
closest neighbor first
time
request arrival

Client Side Caching Evaluation

- Techniques
  - Video segmentation
  - Parallel reception of streams
  - Client buffering
- Advantage
  - Achieves server resource reduction
  - Achieves True VoD behaviour
- Problems
  - Optimum can not be achieved on average case
  - Needs combination with prescheduled technique for high-popularity titles
  - May require large client buffers
  - Are incapable (or not proven) to work with user interactivity
  - Guaranteed bandwidth required

Overall Evaluation

- Advantage
  - Achieves server resource reduction
- Problems
  - May require large client buffers
  - Incapable (or not proven) to work with user interactivity
  - Guaranteed bandwidth required
- Fixes
  - Introduce loss-resistant codecs and partial retransmission
  - Introduce proxies to handle buffering
  - Choose computationally simple variations

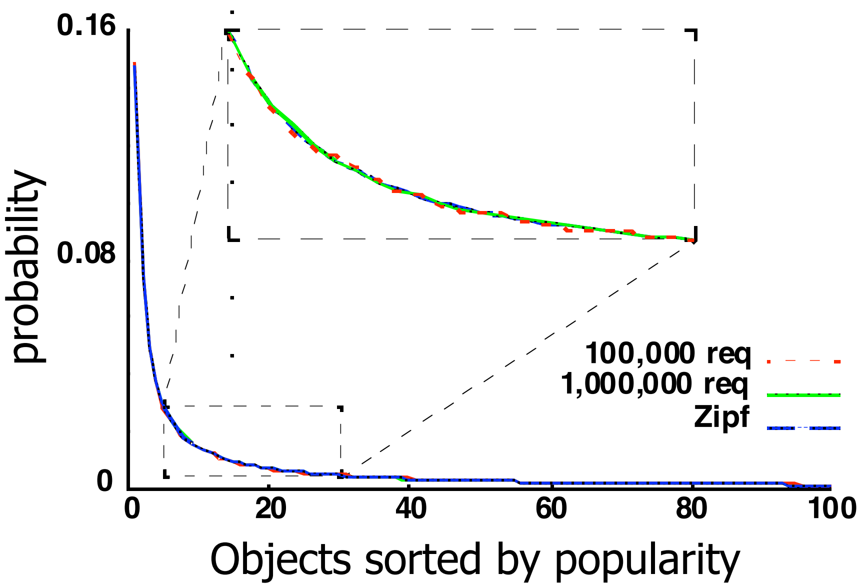
Zipf-distribution

The typical way or modeling access probability

Zipf distribution and features

- Popularity
  - Estimate the popularity of movies (or any kind of product)
  - Frequently used: Zipf distribution

$$z(i) = \frac{C}{i^{\varsigma}} \qquad C = 1 / \sum_{n=1}^{N} \frac{1}{n^{\varsigma}}$$



- **Danger**
  - Zipf-distribution of a process
    - can only be applied while popularity doesn't change
    - is only an observed property
    - a subset of a Zipf-distributed dataset is no longer Zipf-distributed

Some References

1. Thomas D.C. Little and Dinesh Venkatesh: "Prospects for Interactive Video-on-Demand", IEEE Multimedia 1(3), 1994, pp. 14-24
2. Asit Dan and Dinkar Sitaram and Perwez Shahabuddin: "Scheduling Policies for an On-Demand Video Server with Batching", IBM TR RC 19381, 1993
3. Rajesh Krishnan and Dinesh Venkatesh and Thomas D. C. Little: "A Failure and Overload Tolerance Mechanism for Continuous Media Servers", ACM Multimedia Conference, November 1997, pp. 131-142
4. Leana Golubchik and John C. S. Lui and Richard R. Muntz: "Adaptive Piggybacking: A Novel Technique for Data Sharing in Video-on-Demand Storage Servers", Multimedia Systems Journal 4(3), 1996, pp. 140-155
5. Charu Aggarwal, Joel Wolf and Philipp S. Yu: “On Optimal Piggyback Merging Policies for Video-on-Demand Systems”, ACM SIGMETRICS Conference, Philadelphia, USA, 1996, pp. 200-209
6. Kevin Almeroth and Mustafa Ammar: "On the Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service", IEEE JSAC 14(6), 1996, pp. 1110-1122
7. S. Viswanathan and T. Imielinski: "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting", Multimedia Systems Journal 4(4), 1996, pp. 197--208
8. Kien A. Hua and Simon Sheu: "Skyscraper Broadcasting: A New Broadcasting Scheme for
9. Metropolitan Video-on-Demand Systems", ACM SIGCOMM Conference, Cannes, France, 1997, pp. 89-100
10. L. Juhn and L. Tsend: "Harmonic Broadcasting for Video-on-Demand Service", IEEE Transactions on Broadcasting 43(3), 1997, pp. 268-271
11. Carsten Griwodz and Michael Liepert and Michael Zink and Ralf Steinmetz: "Tune to Lambda Patching", ACM Performance Evaluation Review 27(4), 2000, pp. 202-206
12. Kien A. Hua and Yin Cai and Simon Sheu: "Patching: A Multicast Technique for True Video-on Demand Services", ACM Multimedia Conference, Bristol, UK, 1998, pp. 191-200
13. Derek Eager and Mary Vernon and John Zahorjan: "Minimizing Bandwidth Requirements for On-Demand Data Delivery", Multimedia Information Systems Conference 1999
14. Jehan-Francois Paris: http://www.cs.uh.edu/~paris/