Distributed Video Systems

Chapter 6

Issues in Video Transmission and Delivery

Part 3 - Batching, Caching, and Piggybacking

Jack Yiu-bun Lee

Department of Information Engineering

The Chinese University of Hong Kong

Contents

5.1 Introduction

- VoD technologies have been available for many years, why VoD services are still not popular?
  - It's expensive and not economically viable.
- How can cost be reduced?
  - By evolution of faster computer hardware, higher bandwidth network for the same price.
  - By taking advantage of economy of scales, i.e. using commodity hardware platforms like the PC.
    - E.g. parallel servers.
  - By intelligent ways of reducing the system requirement.
    - E.g. batching, caching, and piggybacking.

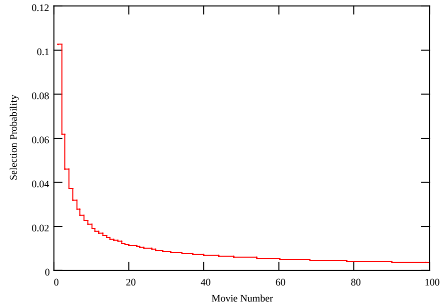
5.2 Principles

- Observation
  - In real-world applications, a large proportion of VoD users watch only a small number of popular movies.
  - Studies from traditional video rental services show that the movie popularity is Zipf distributed:

$$q_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \text{ for } i = 1 \ldots n$$

where $f_i = 1/i^{1-\theta}$

with $\theta = 0.271$.

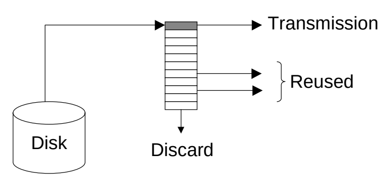


5.2 Principles

- Motivation
  - The movie popularity is highly skewed.
  - Many users are likely to watch the same movies.
  - Why not let the users share **it**?
- Share What?
  - Server
    - Share retrieved video data at the server by caching.
  - Network
    - Share transmitted video data by multicasting.
  - Client
    - Share received video data by buffering.

5.3 Caching At Server

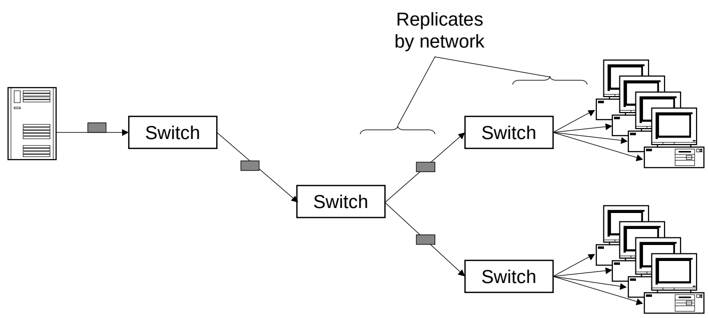
- Principle
  - Keep retrieved video data in a cache for some time in case another user wants the same piece of data.



- Problems
  - How long/much to keep the retrieved video data?
  - Keep all retrieved data or only selected data?
  - What is the tradeoffs in delay, and buffer?
  - Can the gain offsets the cost incurred?

5.4 Multicasting and Batching

- Principle
  - Transmitting a video by multicast enables the system to serve more than one viewers using only one-channel's worth of resources at the server and part of the network.

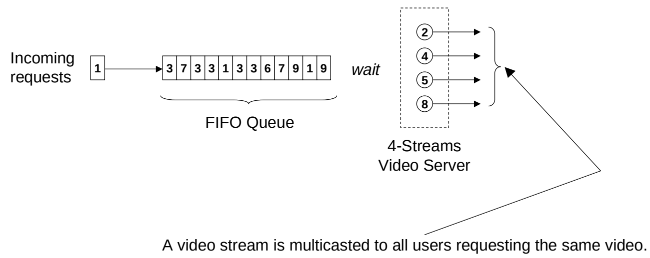


5.4 Multicasting and Batching

- Problem
  - Video playback at different clients are unlikely to be synchronized.
  - Hence simply sharing a multicast video session isn't going to be very effective.
- Solutions
  - Tradeoff Delay (e.g. Batching, NVoD)
  - Tradeoff Buffer (e.g. Split and Merge)
  - Tradeoff Quality (e.g. Piggybacking)
  - Any combinations of the above.

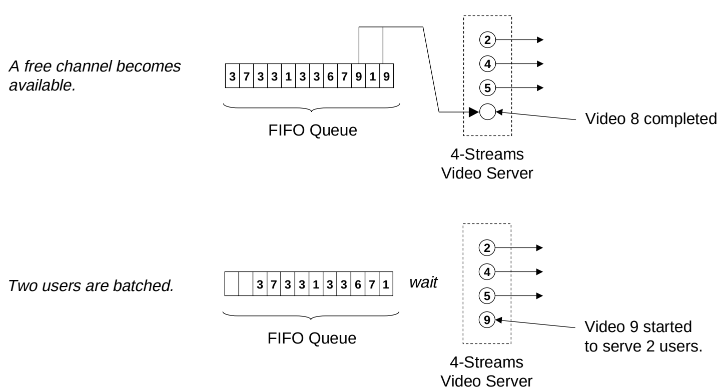
5.5 Batching

- Principle
  - Force a user to wait until it synchronizes with a multicast session.
- Algorithm 1: First-Come-First-Serve (FCFS)
  - Incoming requests are queued:



5.5 Batching

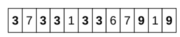
- Algorithm 1: First-Come-First-Serve (FCFS)
  - The HOL request and all requests for the same video title are then served when a channel becomes available.



5.5 Batching

Jack Y.B. Lee

- Algorithm 1: First-Come-First-Serve (FCFS)
  - ◆ Advantage
    - Fairness (unpopular videos will not be denied service)
  - ◆ Disadvantage
    - Does not consider batching efficiency.
    - Example
      - – FCFS assigns the available channel to video 9 with 2 waiting users while there are 5 users waiting for video 3.

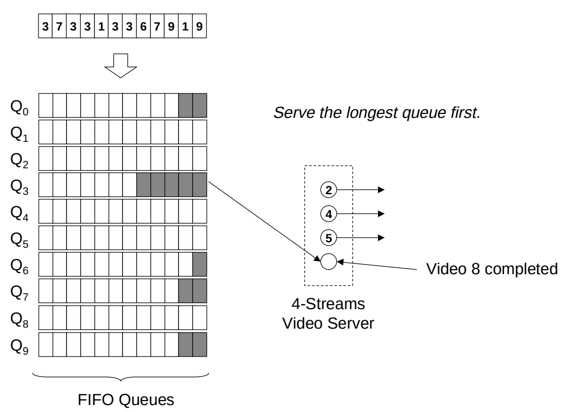


      - – In terms of batching efficiency, the system should serve video 3 instead of video 9.

5.5 Batching

Jack Y.B. Lee

- Algorithm 2: Maximal Queue Length (MQL)
  - ◆ One FIFO queue per video title.



- Algorithm 2: Maximal Queue Length (MQL)
  - Advantage
    - Improved batching efficiency.
  - Disadvantage
    - No consideration for waiting time and fairness;
    - Users may leave the queue (turned away) if the waiting time is too long.
- Performance Measures
  - Turn-away probability
  - Average response time
  - Fairness
    - By comparing the turn-away probability for a particular video title with the system-wide turn-away probability.

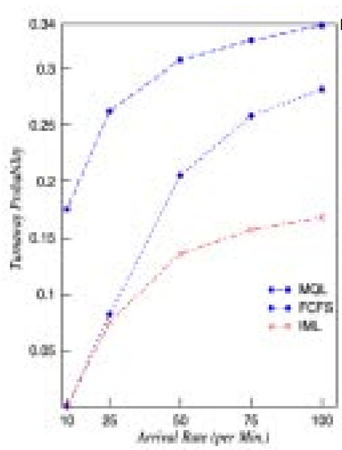
- Algorithm 3: Wait Tolerance Batching (WTB)
  - Video titles are classified into two types:
    - hot videos (i.e. popular) and cold (unpopular) videos.
  - Max_Batch Schemes
    - A video title is available for scheduling only if some of its requests have waiting time exceeded a batching threshold.
    - If there are no eligible videos, unused channels remain idle.
    - There is a minimum waiting time on all requests.
    - The objective is to maximize batching.
    - The batching threshold is chosen based on the wait tolerance.

- Algorithm 3: Wait Tolerance Batching (WTB)
  - Max_Batch Schemes (Video Selection Strategies)
    - Max_Batch with Maximum Queue Length (BMQ)
      - Attempts to increase batching effect;
      - No consideration for length of waiting time.
    - Max_Batch with Minimum Loss (BML)
      - Assumes a waiting user will depart after reaching the batching threshold;
      - Based on the requests arrival times of each queue, the expected number of departures can be calculated because stream completions are known (assumed no VCR control).

- Algorithm 3: Wait Tolerance Batching (WTB)
  - Min_Idle Schemes
    - Batching is performed on hot videos only.
    - Cold videos are always eligible for scheduling.
    - No minimum wait time.
    - The objective is to reduce response time and decrease loss of viewers for cold videos.
    - Schemes similar to BMQ and BML can also be devised for Min_Idle to form IMQ and IML.

- Performance Results
  - System Parameters
    - 100 videos, average length 120 minutes;
    - No VCR control;
    - 800-streams server;
    - 50 video request arrivals per minute.
  - Performance Measures
    - Turnaway probability
    - Average response time
    - Fairness

- Performance Results
  - Turnaway Probability versus Arrival Rate:



MQL

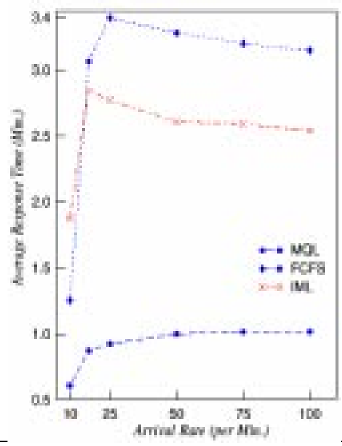
FCFS

- **Only IML shown as others have similar results.**
- **IML has lower turnaway probability.**

5.5 Batching

Jack Y.B. Lee

- Performance Results
  - Average Response Time versus Arrival Rate:



FCFS

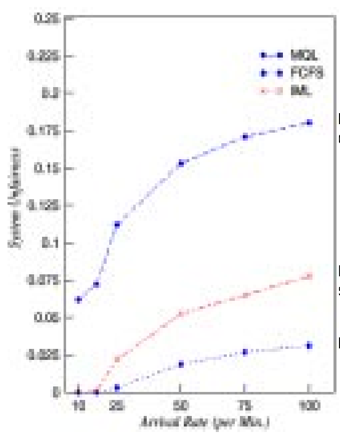
IML

MQL is lowest, at the cost of driving away 25% users.

5.5 Batching

Jack Y.B. Lee

- Performance Results
  - System Unfairness versus Arrival Rate:



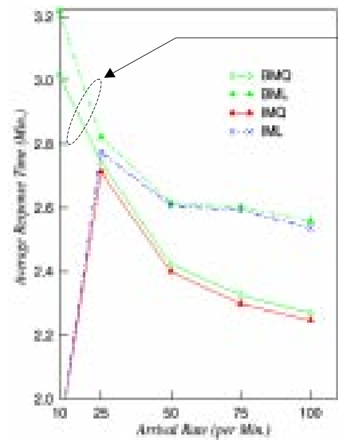
MQL is most unfair because cold videos easily got denied service completely.

IML has much better fairness than MQL while still has best batching efficiency.

FCFS

5.5 Batching

- Performance Results
  - Average Response Time versus Arrival Rate:



High response time at low load due to Max_Batch's minimum waiting time.

BML & IML has higher response time.

BMQ & IMQ has lower response time.

5.5 Batching

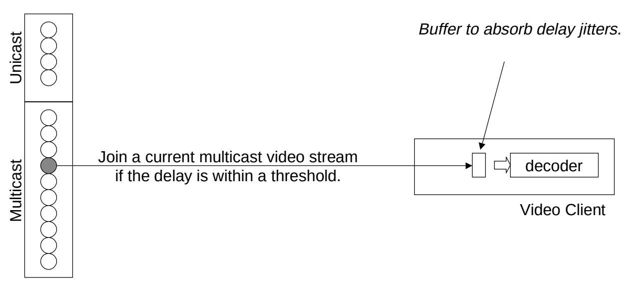
- Conclusions
  - The four schemes (BML, BMQ, IML, and IMQ) generally outperform the FCFS and MQL schemes.
  - Using MQL in selecting streams (i.e. BMQ, IMQ) yields smaller response time (at the expense of fairness) while keeping the throughput close to that provided by Minimal-Loss schemes (i.e. BML, IML).
- Remarks
  - No VCR actions is allowed.
  - The average response time is in 2~3 minutes. Hence this is not really true VoD, but is in fact a near VoD only.
  - The turn-away probability is fairly high (>10%), leading to a somewhat unsatisfactory service in practice.
  - Note that 50 requests/min x 120 min = 6000 customers served by a 800-streams server.

5.6 Batching with Bridging

- Principle
  - Absorb the playback time differences by buffering at the client; or at an intermediate node.
  - Some capacity of the video server is for multicast, while the rest is for unicast.
  - The unicast channels are used to fill the gap between the time difference of the multicasted stream and the requested video stream.
- Advantages
  - Batching can be done by multicasting.
  - True-VoD service with VCR control can be provided without the excessive delay incurred by simple batching.

5.6 Batching with Bridging

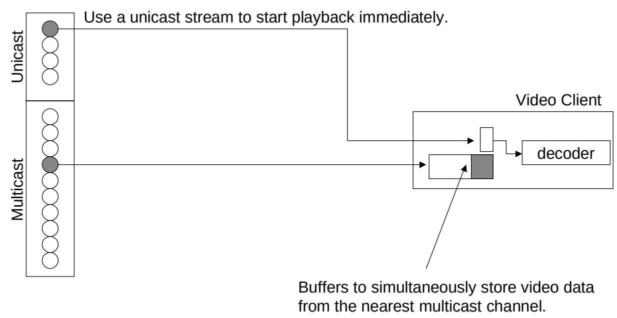
- Buffering At Client
  - Starting a new video session:



5.6 Batching with Bridging

Jack Y.B. Lee

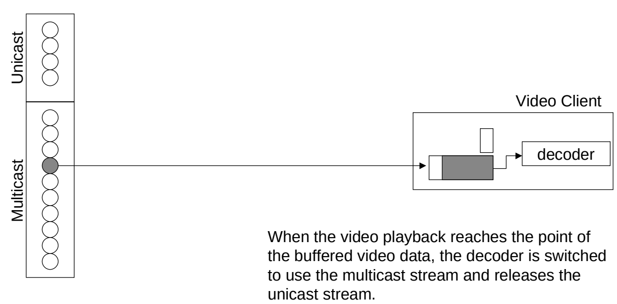
- Buffering At Client
  - Starting a new video session:



5.6 Batching with Bridging

Jack Y.B. Lee

- Buffering At Client
  - Starting a new video session:



- Buffering At Client
  - Observations
    - The unicast streams are not occupied for the entire duration of the movie, but for only a short time to bridge the gap between the playback schedule and the multicast schedule.
    - The buffers are essentially used to introduce delays into the multicast stream. By varying the amount of buffered data, the amount of delay can be controlled.
    - VCR functions can be supported by treating them as new sessions.
    - So the tradeoff in delay is compensated by the tradeoff in buffers (and some unicast streams).

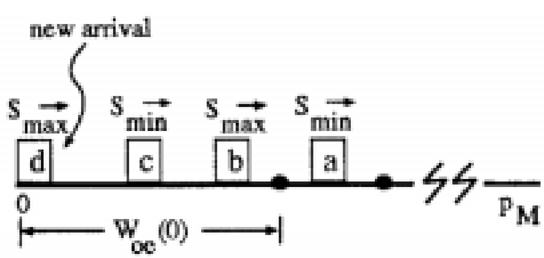
- Buffering At Client
  - Challenges
    - How many channels should be reserved for unicast, and how many for multicast?
    - How to assign multicast channels to video titles? Static or dynamic?
    - VCR functions could be blocked if all unicast channels are occupied.
    - A video session could also be blocked after VCR interaction if all channels become occupied unless the video title is being remulticasted periodically.
    - Is the scheme (and batching in general) effective for popularity models other than Zipf?
    - Can the gain offset the additional cost in buffering?

5.7 Piggybacking

- Principle
  - Adjust the playback rates of in-progress video sessions to merge them into a single stream.
  - This works for network transmission as well as disk I/O.
- Adjusting Playback Rates
  - Display rate variations up to 5% is not perceivable by the viewer.
  - Display rate can be increased by discarding frames periodically and decreased by stuffing frames or adding interpolated frames.
  - The adjustment can be made online in real-time; or off-line by storing multiple versions of the video.

5.7 Piggybacking

- Merging Policies
  - Odd-Even Reduction Policy
    - If catch-up is possible, playback at max. rate; otherwise playback at min. rate.
    - Merging is done on two streams only.

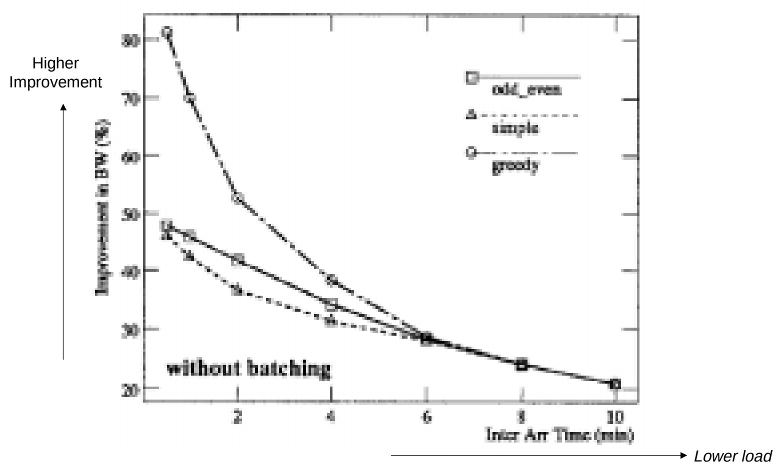


5.7 Piggybacking

- Merging Policies
  - Simple Merging Policy
    - Attempts to form merging groups so that more than two sessions can be merged into a single session.
  - Greedy Merging Policy
    - Attempts to perform merging not only at startup, but continue to merge on-going sessions and groups to form larger groups.
  - Limited Merging Policy
    - Taking into storage overhead and attempts merging only for up to a certain distance (rather than the entire length of video).

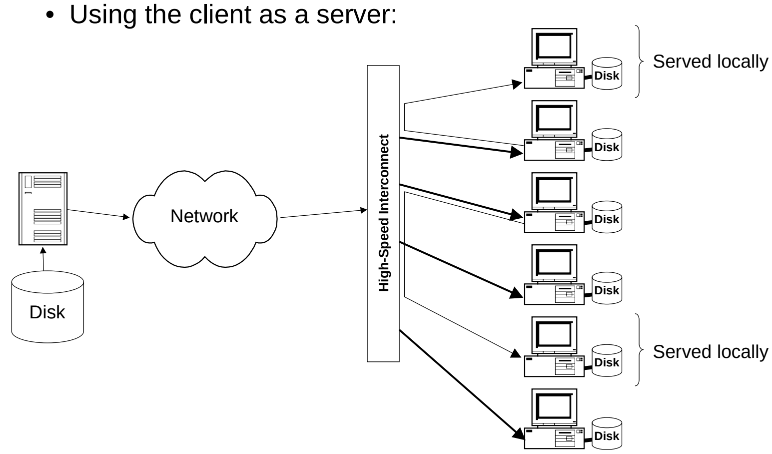
5.7 Piggybacking

- Performance Results
  - BW Improvement versus Inter-arrival Time:



5.8 Buffering

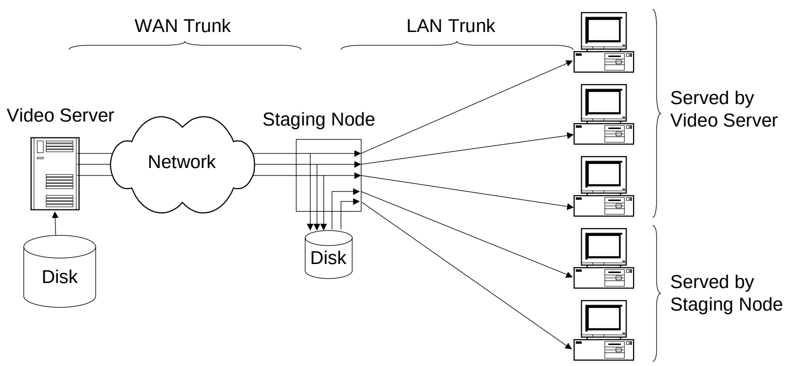
- Principle
  - Try to reuse received video data to serve other users.
    - Using the client as a server:



Jack Y.B. Lee

5.8 Buffering

- Principle
  - Try to reuse received video data to serve other users.
    - Using a staging node as a cache:



References

Jack Y.B. Lee

Part of this chapter's materials are based on:

[1] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," Proc. 2nd ACM International Conference on Multimedia '94, 1994, pp.15-23.

[2] H. Shachnai, and P.S. Yu, "Exploring Wait Tolerance in Effective Batching for Video-on-Demand Scheduling," Proc. 8th Israeli Conference on Computer Systems and Software Engineering, 1997, pp.67-76.

[3] L. Golubchik, J.S.D. Lui, and R. Muntz, "Reducing I/O Demand in Video-on-Demand Storage Servers," ACM Performance Evaluation Review, vol.23(1), May 1995, pp.25-36.

Other useful references:

[4] H.K. Park, and H.B. Ryou, "Multicast Delivery for Interactive Video-on-Demand Service," Proc. 12th International Conference on Information Networking, 1998, pp.46-50.

[5] M.Y.Y.Leung, J.C.S.Lui, and L.Golubchik, "Buffer and I/O Resource Pre-allocation for Implementing Batching and Buffering Techniques for Video-on-Demand Systems," Proc. 13th International Conference on Data Engineering, 1997, pp.344-353.

[6] E.L.Abram-Profeta, and K.G.Shin, "Providing Unrestricted VCR Functions in Multicast Video-on-Demand Servers," Proc. IEEE International Conference on Multimedia Computing and Systems, 1998, pp.66-75.

[7] S.W.Carter and D.D.E.Long, "Improving Video-on-Demand Server Efficiency Through Stream Tapping," Proc. 6th International Conference on Computer Communications and Networks, 1997, pp.200-207.

[8] W. Liao, and V.O.K. Li, "The Split and Merge Protocol for Interactive Video-on-Demand," IEEE Multimedia, vol.4(4), October 1997, pp.51-62.