

INF5071 – Performance in distributed systems

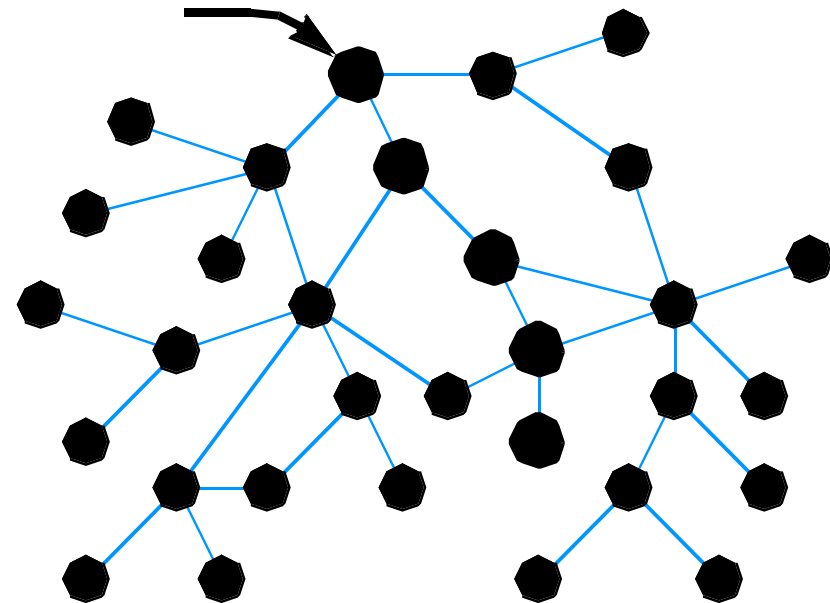


Distribution – Part II

27/10 – 2006

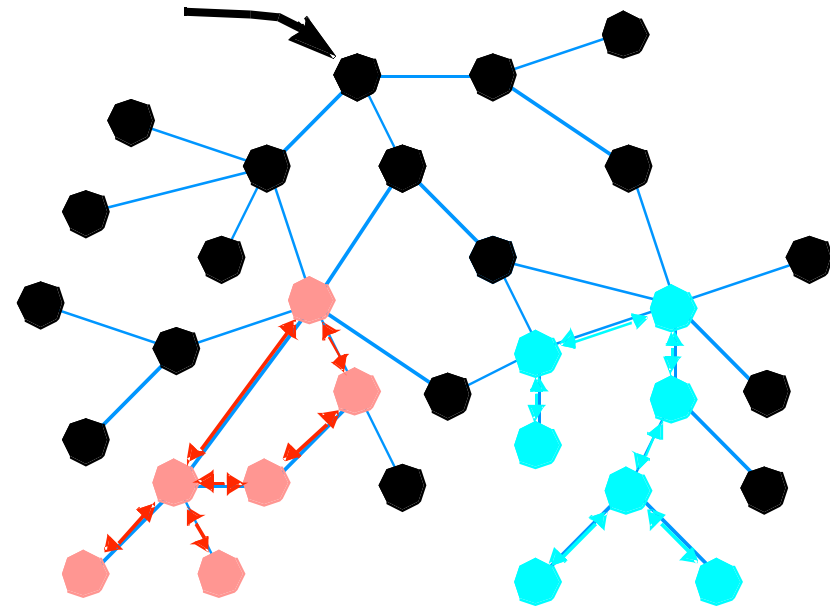
Type IV – Distribution Systems

- Combine
 - Types I, II or III
 - Network of servers
- Server hierarchy
 - Autonomous servers
 - Cooperative servers
 - Coordinated servers
- “Proxy caches”
 - Not accurate ...
 - Cache servers
 - Keep copies on behalf of a remote server
 - Proxy servers
 - Perform actions on behalf of their clients



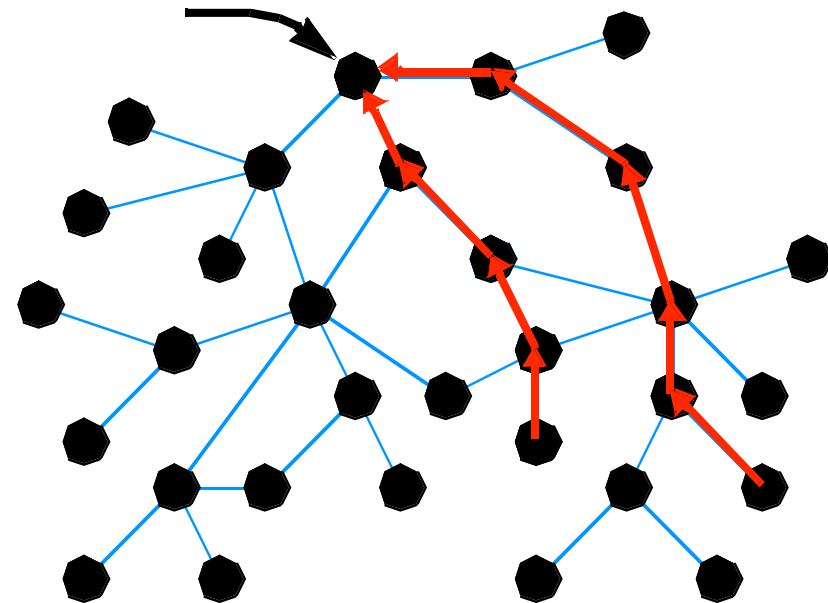
Type IV – Distribution Systems

- Combine
 - Types I, II or III
 - Network of servers
- Server hierarchy
 - Autonomous servers
 - Cooperative servers
 - Coordinated servers
- “Proxy caches”
 - Not accurate ...
 - Cache servers
 - Keep copies on behalf of a remote server
 - Proxy servers
 - Perform actions on behalf of their clients



Type IV – Distribution Systems

- Combine
 - Types I, II or III
 - Network of servers
- Server hierarchy
 - Autonomous servers
 - Cooperative servers
 - Coordinated servers
- “Proxy caches”
 - Not accurate ...
 - Cache servers
 - Keep copies on behalf of a remote server
 - Proxy servers
 - Perform actions on behalf of their clients



Type IV – Distribution Systems

- Variations
 - Gleaning
 - Autonomous, coordinated possible
 - In komssys
 - Proxy prefix caching
 - Coordinated, autonomous possible
 - In Blue Coat (which was formerly Cacheflow, which was formerly Entera)
 - Periodic multicasting with pre-storage
 - Coordinated
 - The theoretical optimum

Gleaning

- *Webster's Dictionary: from Late Latin glennare, of Celtic origin*

 *to gather grain or other produce left by reapers*

 *to gather information or material bit by bit*

- Combine *patching* with *caching* ideas

- non-conflicting benefits of caching and patching

- Caching

- reduce number of end-to-end transmissions
- distribute service access points
- no single point of failure
- true on-demand capabilities

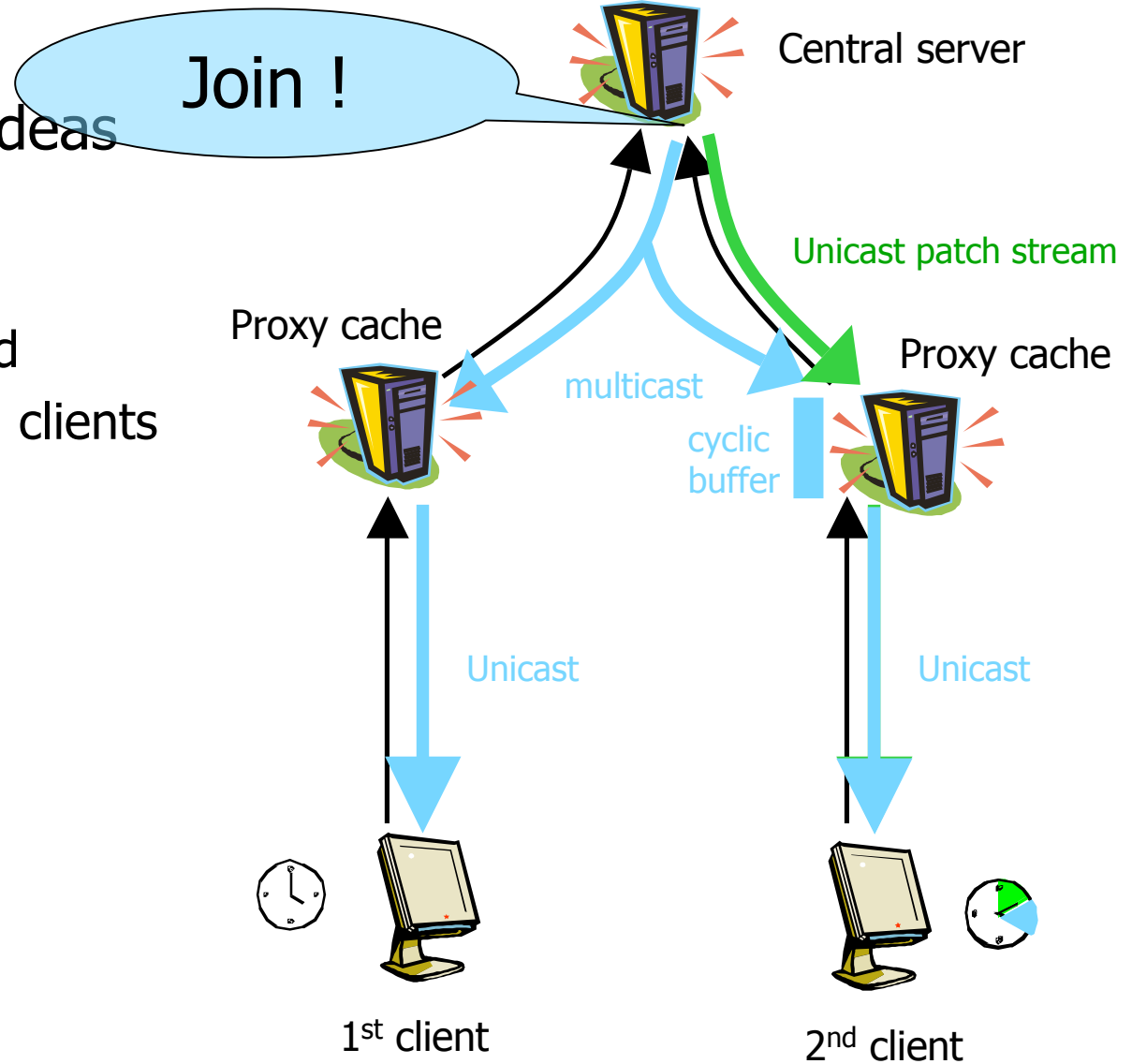
- Patching

- shorten average streaming time per client
- true on-demand capabilities



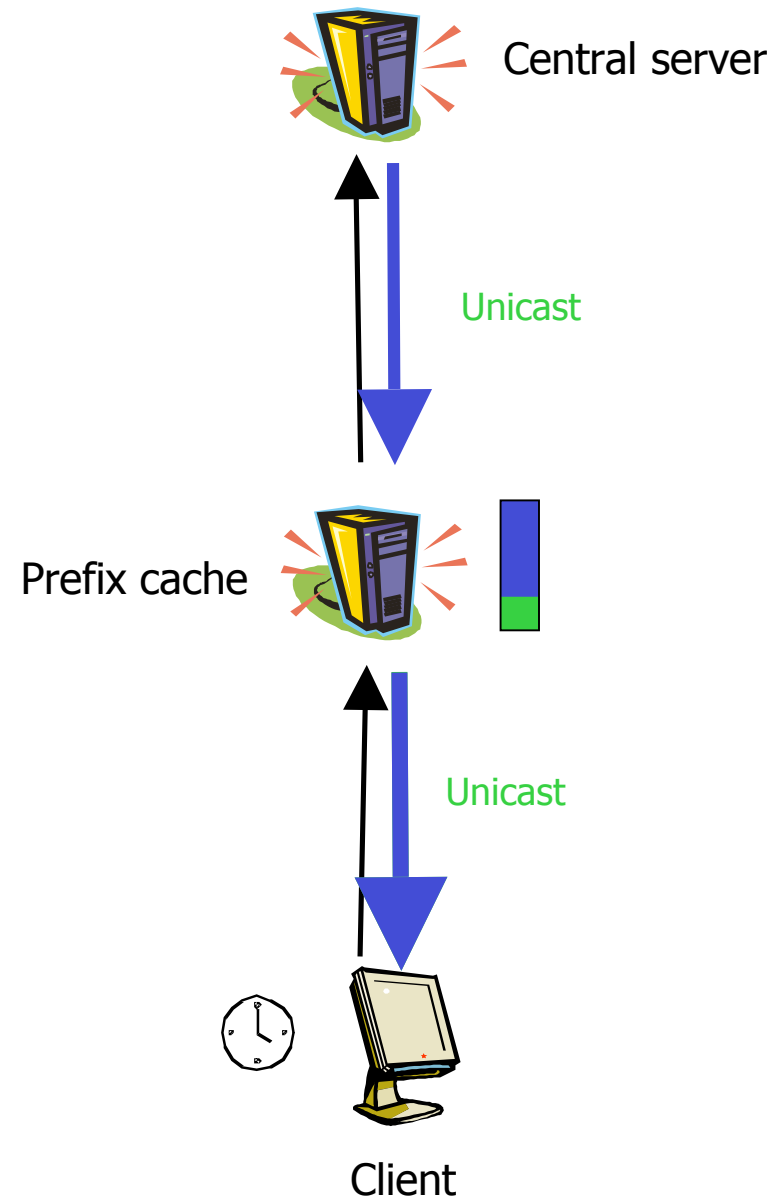
Gleaning

- Combines Patching & Caching ideas
 - Wide-area scalable
 - Reduced server load
 - Reduced network load
 - Can support standard clients



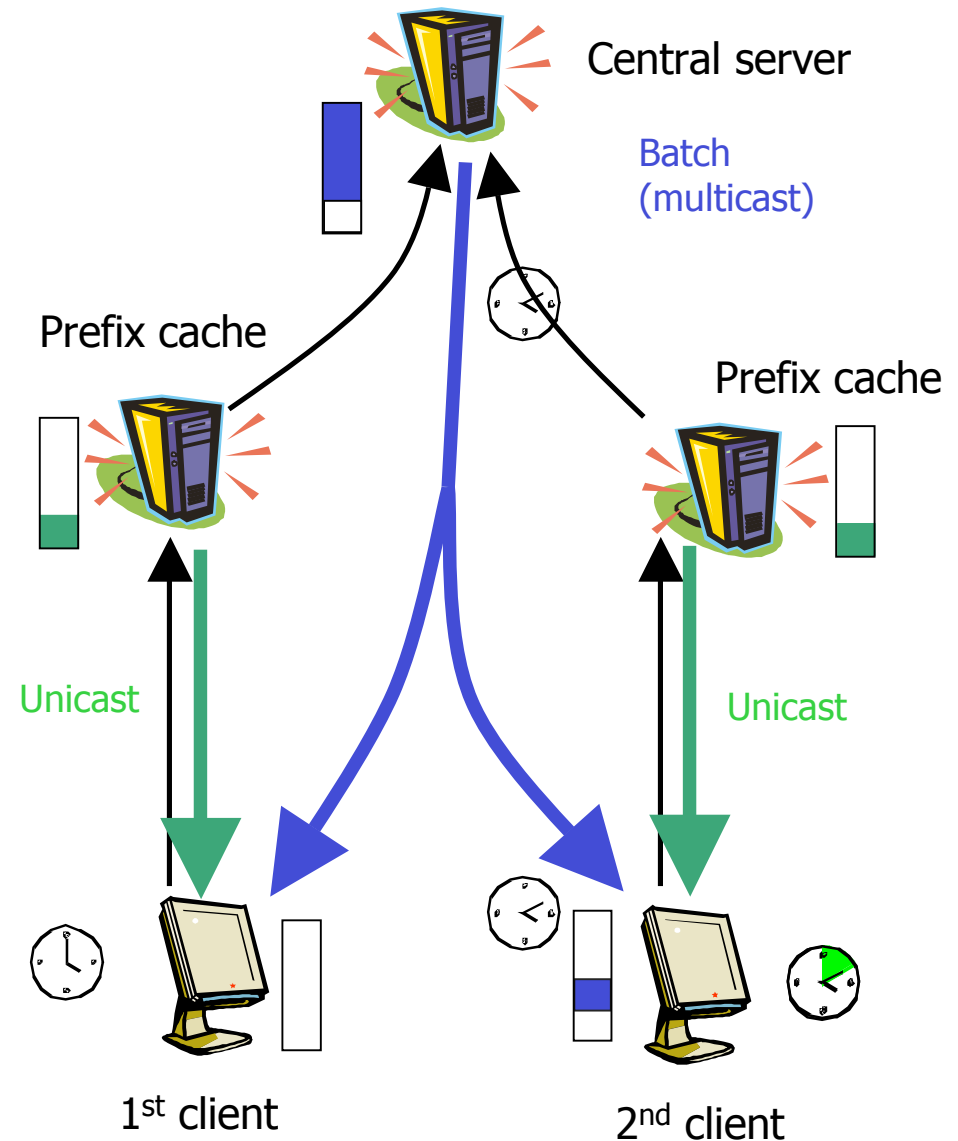
Proxy Prefix Caching

- Split movie
 - Prefix
 - Suffix
- Operation
 - Store prefix in prefix cache
 - Coordination necessary!
 - On demand
 - Deliver prefix immediately
 - Prefetch suffix from central server
- Goal
 - Reduce startup latency
 - Hide bandwidth limitations, delay and/or jitter in backbone
 - Reduce load in backbone



MCache

- One of several Prefix Caching variations
- Combines Batching and Prefix Caching
 - Can be optimized per movie
 - server bandwidth
 - network bandwidth
 - cache space
 - Uses multicast
 - Needs non-standard clients

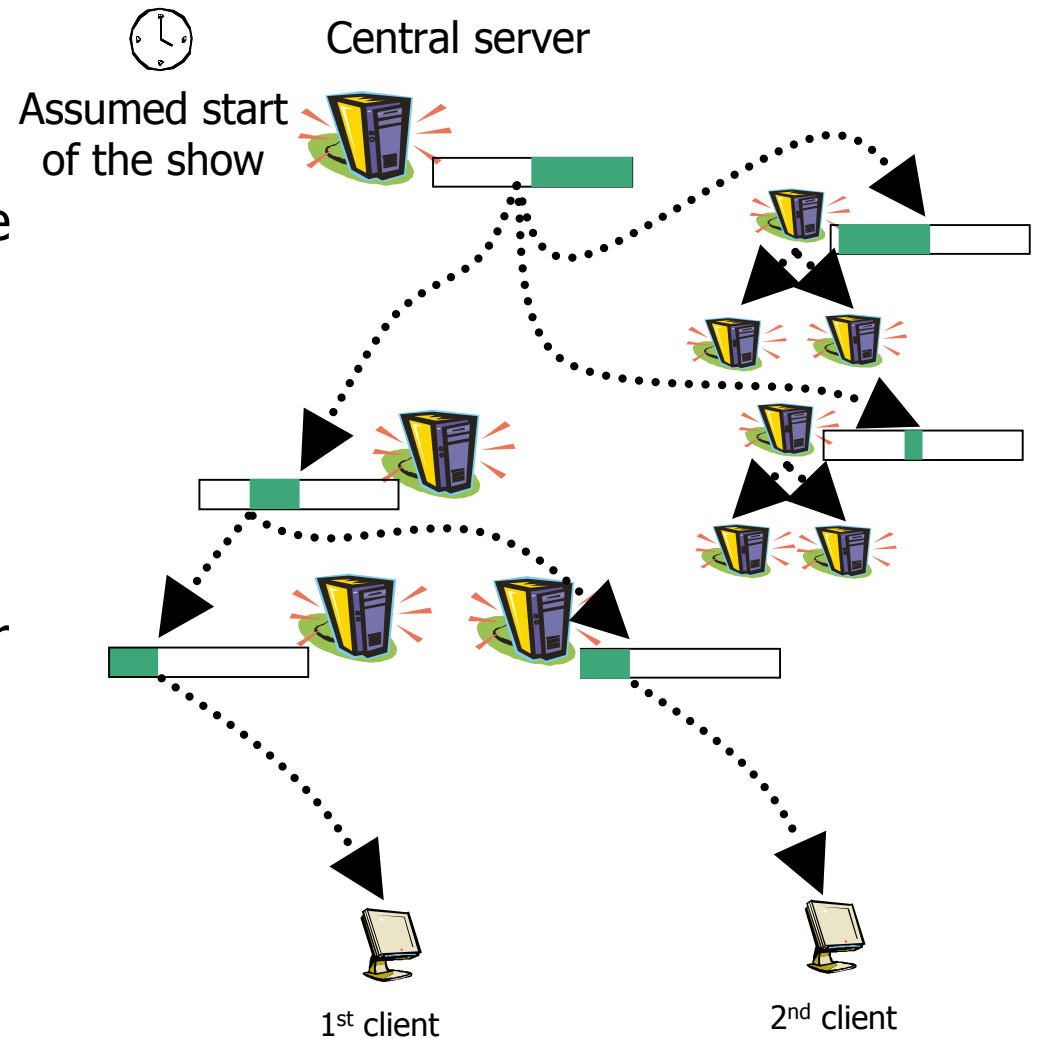


Proxy Prefix Caching

- Basic version
 - Practical
 - No multicast
 - Not optimized
 - Aimed at large ISPs
 - Wide-area scalable
 - Reduced server load
 - Reduced network load
 - Can support standard clients
 - Can partially hide jitter
- Optimized versions
 - Theoretical
 - Multicast
 - Optimized
 - Optimum is constantly unstable
 - jitter and loss is experienced for each client !

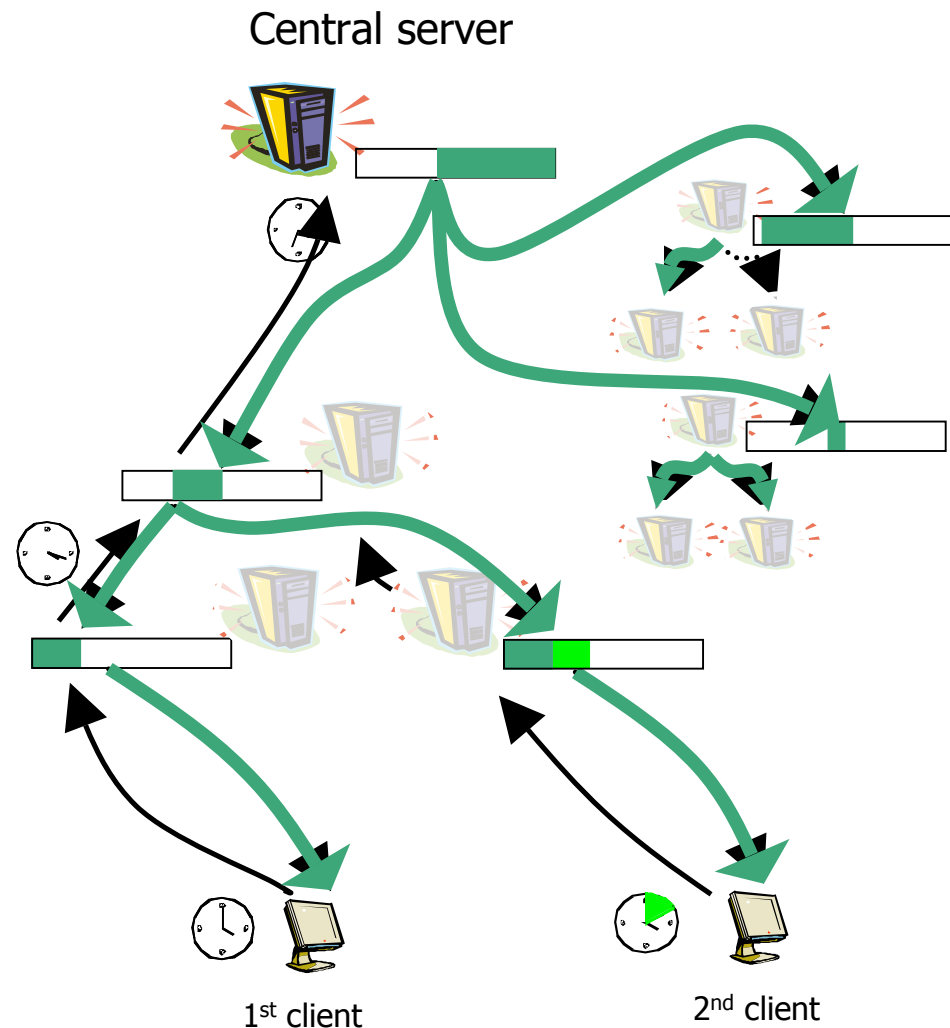
Periodic Multicasting with Pre-Storage

- Optimize storage and network
 - Wide-area scalable
 - Minimal server load achievable
 - Reduced network load
 - Can support standard clients
- Specials
 - Can optimize network load per subtree
- Negative
 - Bad error behaviour



Periodic Multicasting with Pre-Storage

- Optimize storage and network
 - Wide-area scalable
 - Minimal server load achievable
 - Reduced network load
 - Can support standard clients
- Specials
 - Can optimize network load per subtree
- Negative
 - Bad error behaviour



Type IV – Distribution Systems

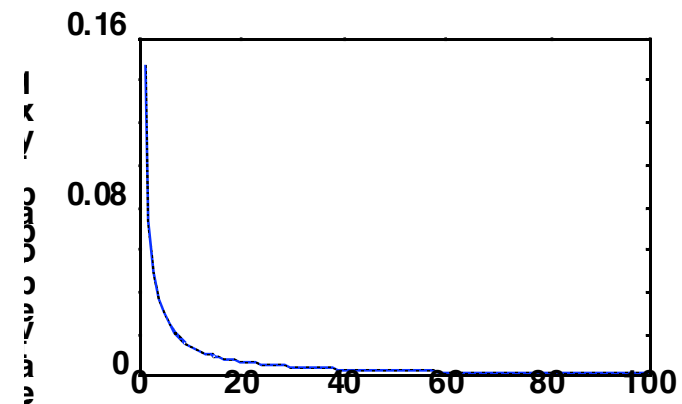
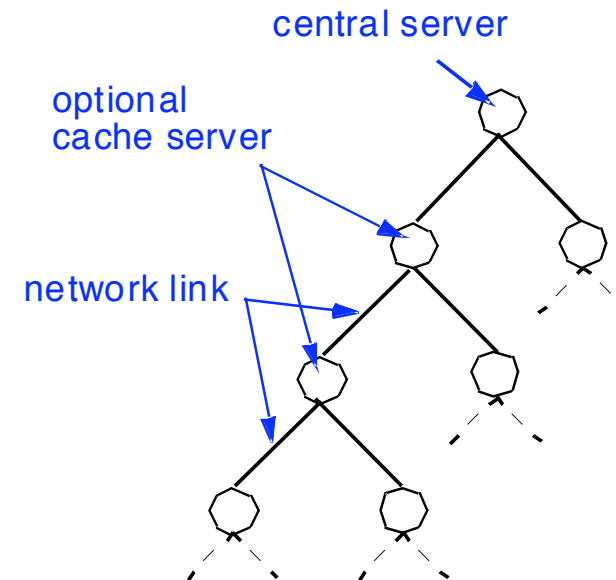
- Autonomous servers
 - Requires decision making on each proxy
 - Some content must be discarded
 - Caching strategies
- Coordinated servers
 - Requires central decision making
 - Global optimization of the system
- Cooperative servers
 - No quantitative research yet



Autonomous servers

Simulation

- Binary tree model allows
 - Allows analytical comparison of
 - Caching
 - Patching
 - Gleaning
- Considering
 - optimal cache placement per movie
 - basic server cost
 - per-stream costs of storage, interface card, network link
 - movie popularity according to Zipf distribution



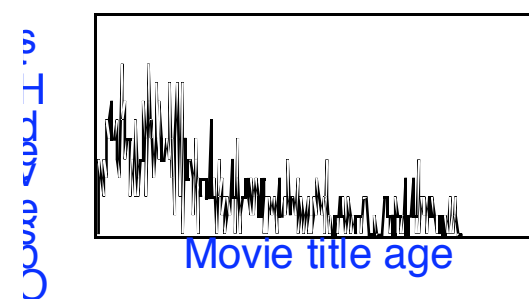
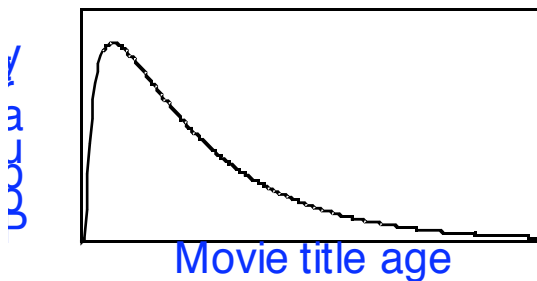
Simulation

- Example
 - 500 different movies
 - 220 concurrent active users
 - basic server: \$25000
 - interface cost: \$100/stream
 - network link cost: \$350/stream
 - storage cost: \$1000/stream
- Analytical comparison
 - 👍 demonstrates potential of the approach
 - 👎 very simplified

Caching	Caching Unicast transmission	4664 Mio \$
λ -Patching	No caching Client side buffer Multicast	375 Mio \$
Gleaning	Caching Proxy client buffer Multicast	276 Mio \$

Simulation

- Modeling
 - User behaviour
 - Movie popularity development
 - Limited resources
 - Hierarchical topology
- Individual user's
 - Intention
 - depends on user's time (model randomly)
 - Selection
 - depends on movies' popularity
- Popularity development



Caching Strategies

- FIFO: First-in-first-out
 - Remove the oldest object in the cache in favor of new objects
- LRU: Least recently used strategy
 - Maintain a list of objects
 - Move to head of the list whenever accessed
 - Remove the tail of the list in favor of new objects
- LFU: Least frequently used
 - Maintain a list distance between last two requests per object
 - Distance can be time or number of requests to other objects
 - Sort list: shortest distance first
 - Remove the tail of the list in favor of new objects

Caching Strategies

- Considerations
 - limited uplink bandwidth
 - quickly exhausted
 - performance degrades immediately when working set is too large for storage space
 - conditional overwrite strategies
 - can be highly efficient
- ECT: Eternal, Conditional, Temporal

LFU

Forget object statistics when removed

Cache all requested objects

Log # requests or time between hits

ECT

Remember object statistics forever

Compare requested object and replacement candidate

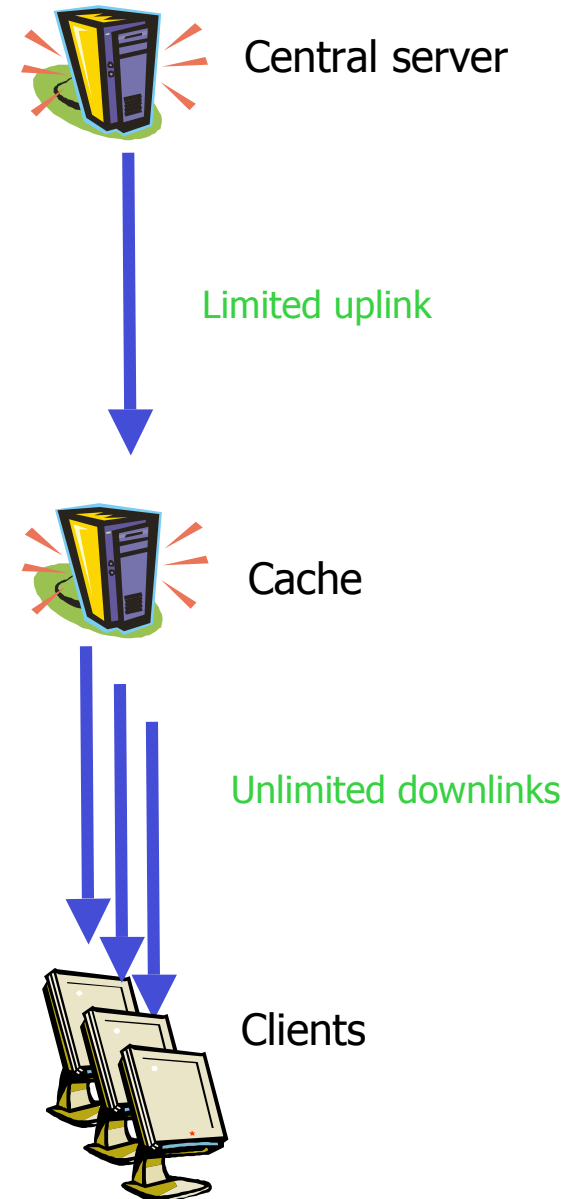
Log times between hits



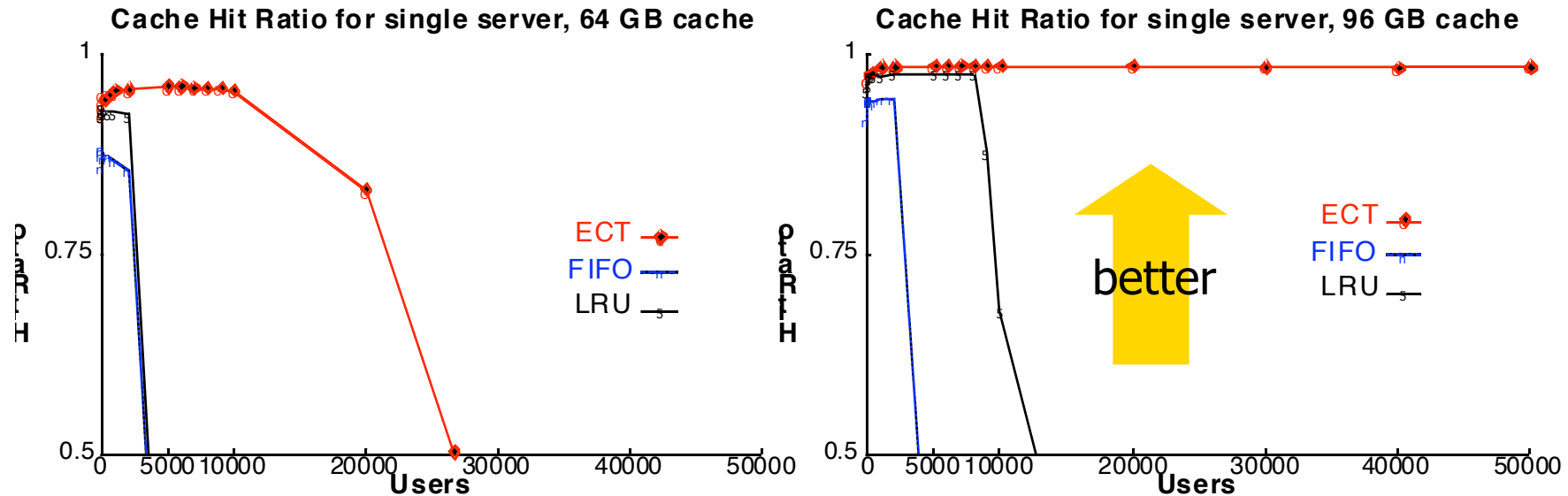
Simulation

■ Movies

- 500 movies
- Zipf-distributed popularity
- 1.5 MBit/s
- 5400 sec
- File size ~ 7.9 GB



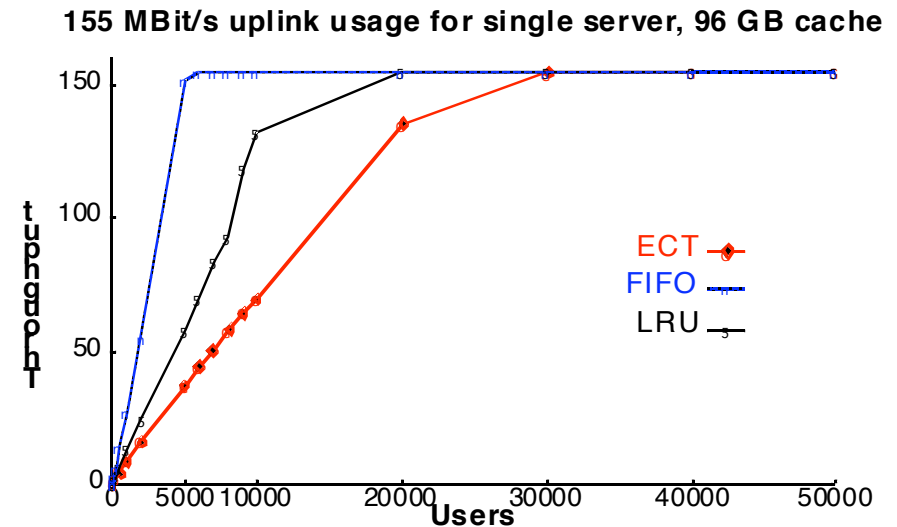
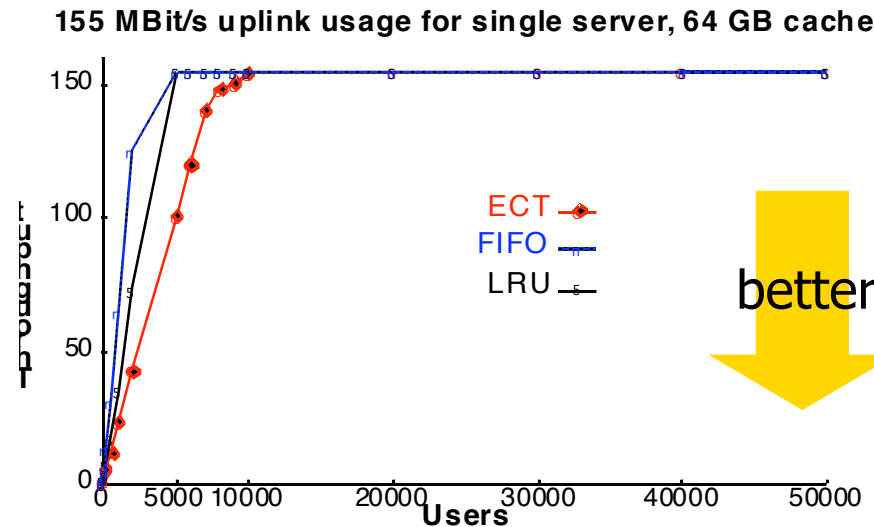
Effects of caching strategies on user hit rates



■ Hit ratio

- dumb strategies (almost) do not profit from cache size increases
- intelligent strategies profit hugely from cache size increases
- strategies that use conditional overwrite outperform other strategies massively
 - doesn't have to be ECT

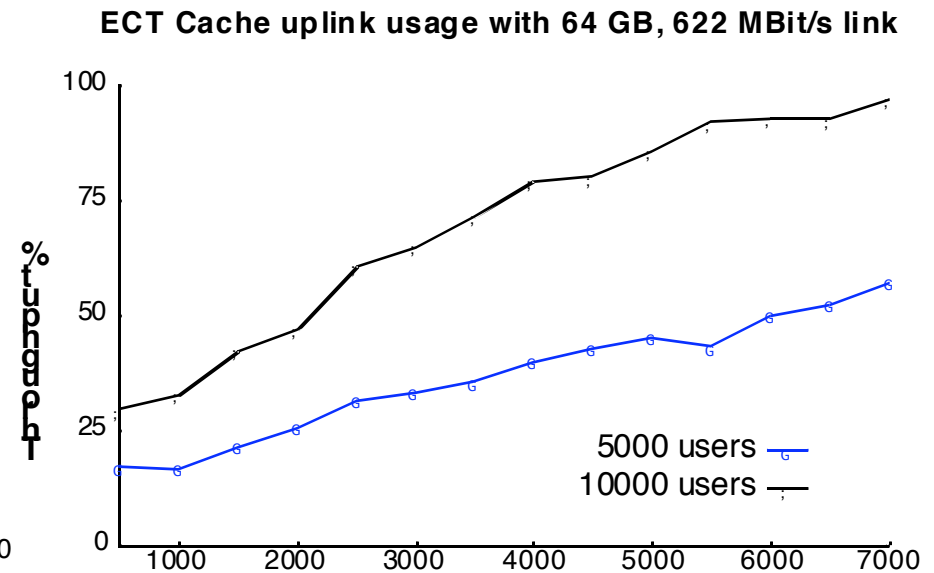
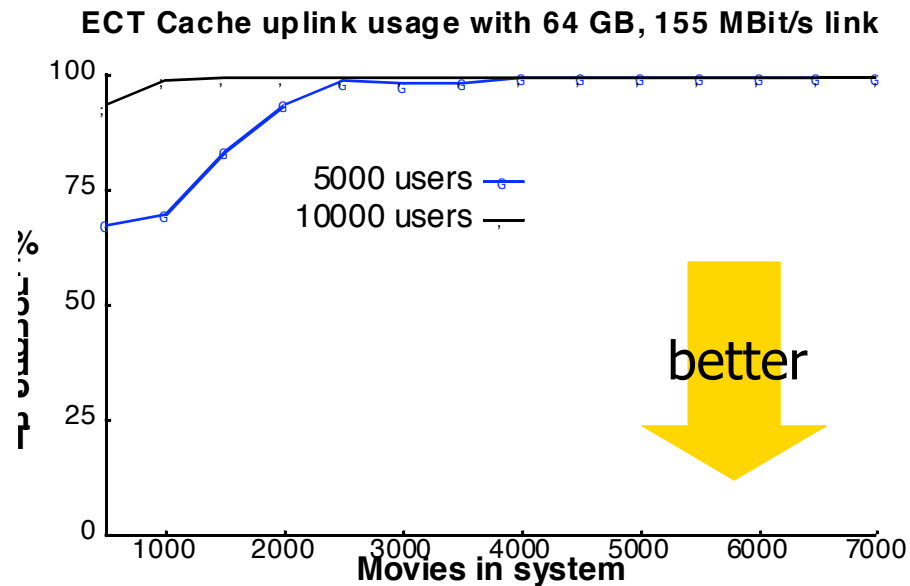
Effects of caching strategies on throughput



■ Uplink usage

- profits from small cache increase greatly if there is a strategy
- conditional overwrite reduces uplink usage

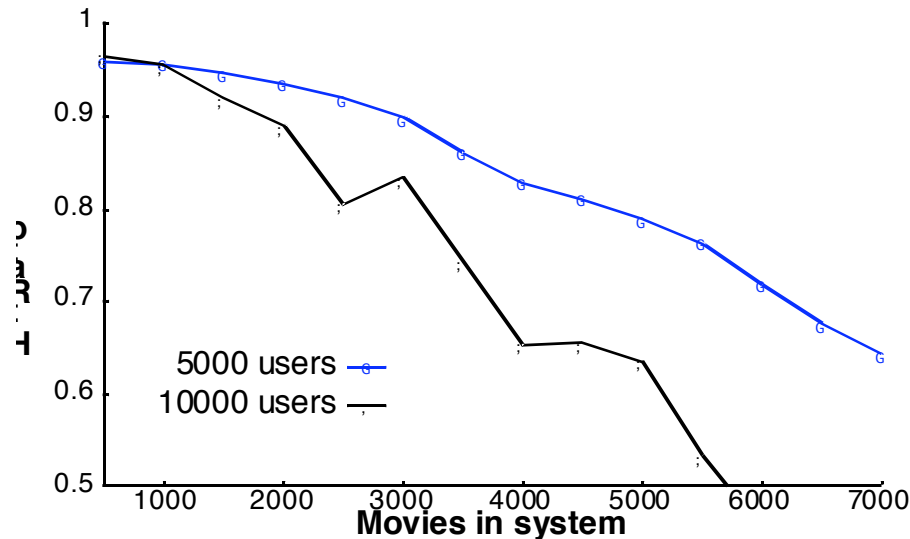
Effects of number of movies on uplink usage



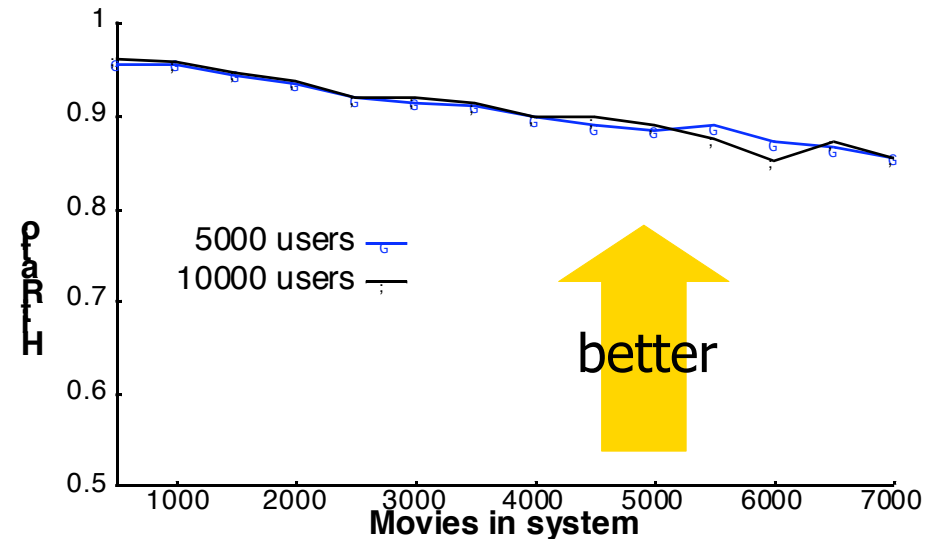
- In spite of 99% hit rates
 - Increasing the number of users will congest the uplink
 - Note
 - scheduling techniques provide no savings on low-popularity movies
 - identical to unicast scenario with minimally larger caches

Effects of number of movies on hit ratio

ECT Cache hit ratio with 64 GB, 155 MBit/s link

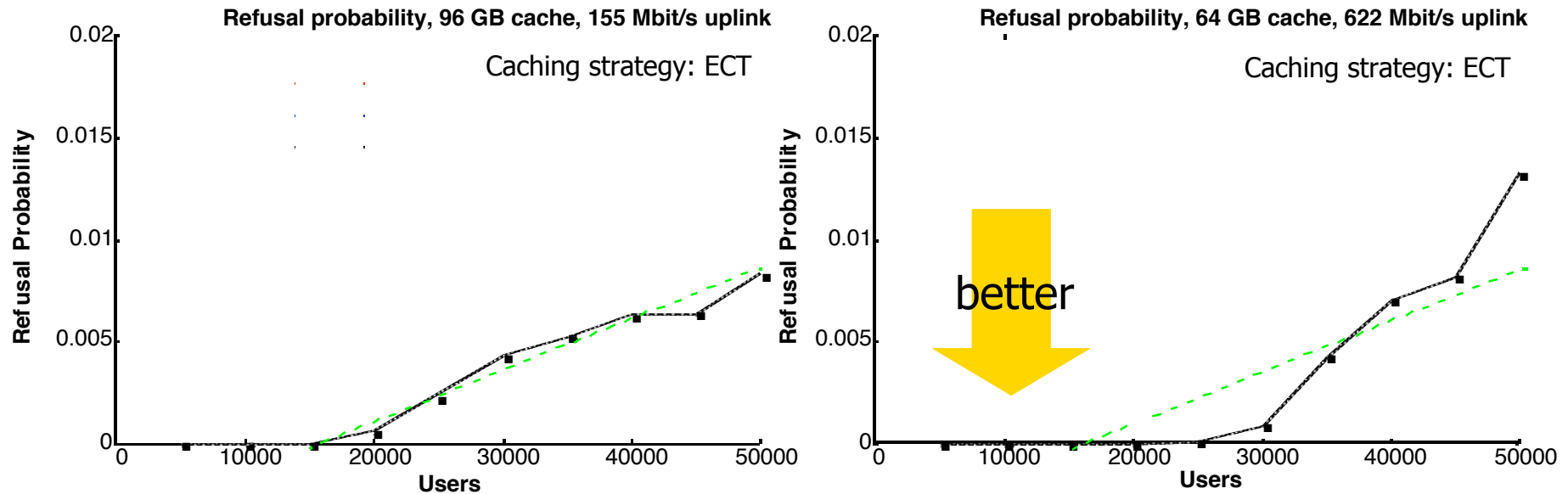


ECT Cache hit ratio with 64 GB, 622 MBit/s link



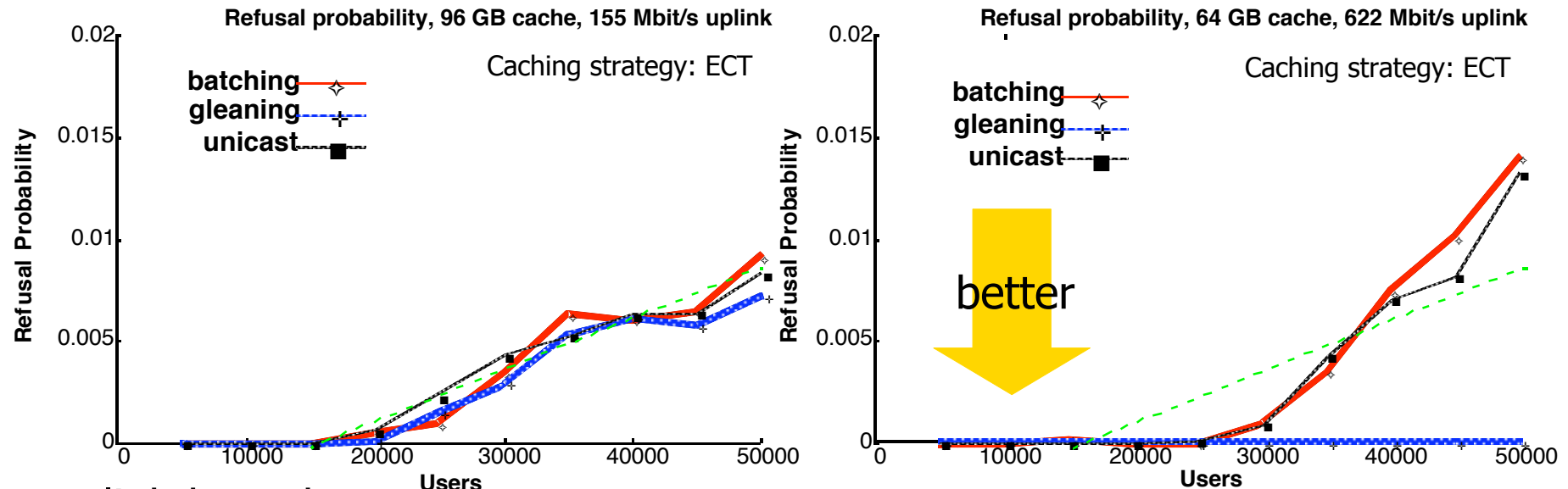
- Limited uplink bandwidth
 - Prevents the exchange of titles with medium popularity
 - Unproportional drop of efficiency for more users
 - Strategy can not recognize medium popularity titles

Effects of user numbers on refusal probabilities



- Refusal in this simulation
 - Network is admission controlled – users have to wait for their turn
 - Users wait up to 5 minutes – afterwards count one refusal
- Scenarios
 - Left: cache ~12 movies, uplink capacity ~103 streams
 - Right: cache ~8 movies, uplink capacity ~415 streams
- Uplink-bound scenario
 - No bottleneck between cache server and clients
 - Moderately popular movies can exhaust uplink bandwidth quickly
 - Unicast gains a lot from large caches

Effects of user numbers on refusal probabilities



Uplink-bound scenario

- Shows that low-popularity movies are accessed like unicast by all techniques
- Patching techniques with infinite window can exploit multicast
- Collecting requests does not work

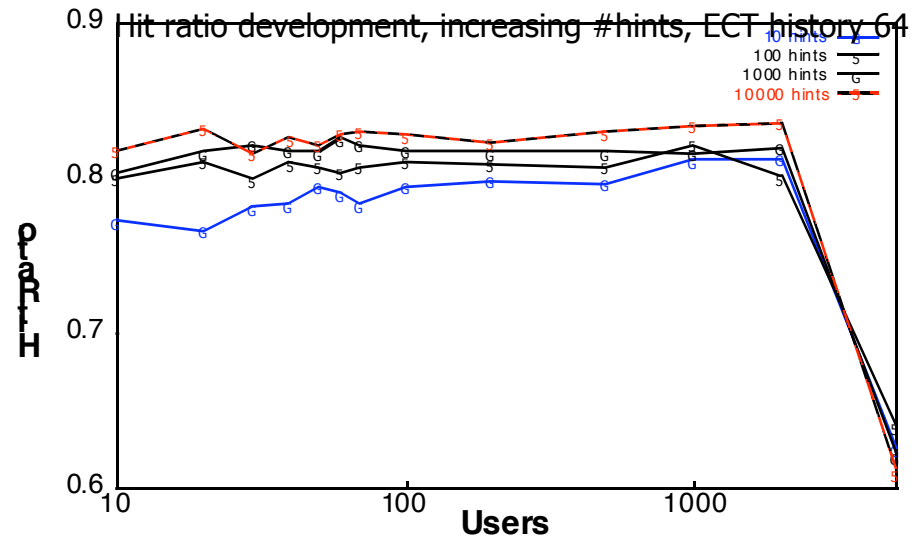
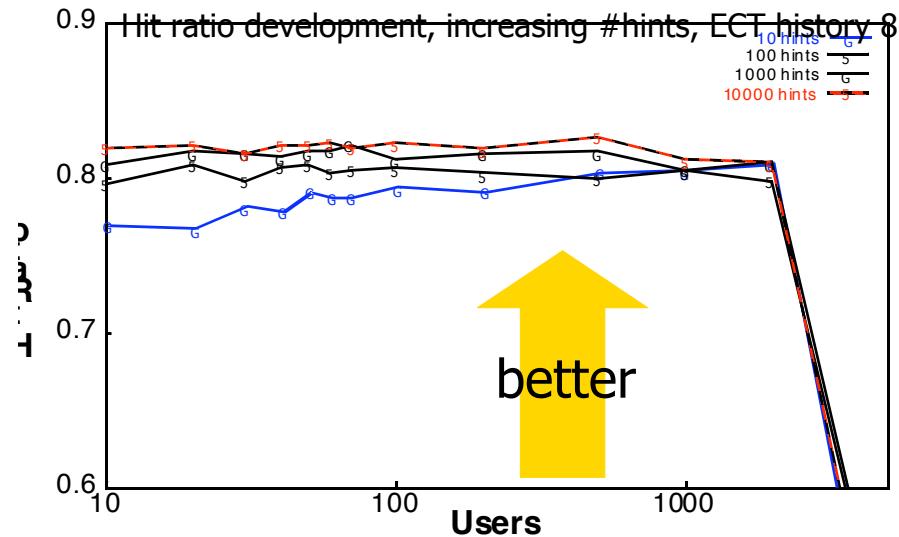
Cache size

- Is *not very relevant for patching techniques*
- Is very relevant for full-title techniques

Bandwidth effect of daytime variations

- Change popularity according to time-of-day
- Two tests
 - Popularity peaks and valleys uniformly distributed
 - Complete exchange of all titles
 - Spread over the whole day
 - Popularity peaks and valleys either at 10:00 or at 20:00
 - Complete exchange of all titles
 - Within a short time-frame around peak-time
- Astonishing results
 - For ECT with all mechanisms
 - Hardly any influence on
 - hit rate
 - uplink congestion
 - Traffic is hidden by delivery of low-popularity titles

Hint-based Caching



- Idea
 - Caches consider requests to neighbour caches in their removal decisions
- Conclusion
 - Instability due to uplink congestion can not be prevented
 - Advantage exists and is logarithmic as expected
 - Larger hint numbers maintain the advantage to the point of instability
 - Intensity of instability is due to ECT problem
 - ECT inherits IRG drawback of fixed-size histograms

Simulation: Summary

- High relevance of population sizes
 - complex strategies require large customer bases
- Efficiency of small caches
 - 90:10 rule-of-thumb reasonable
 - unlike web caching
- Efficiency of distribution mechanisms
 - considerable bandwidth savings for uncached titles
- Effects of removal strategies
 - relevance of conditional overwrite
 - unlike web caching, paging, swapping, ...
- Irrelevance of popularity changes on short timescales
 - few cache updates
compared to many direct deliveries

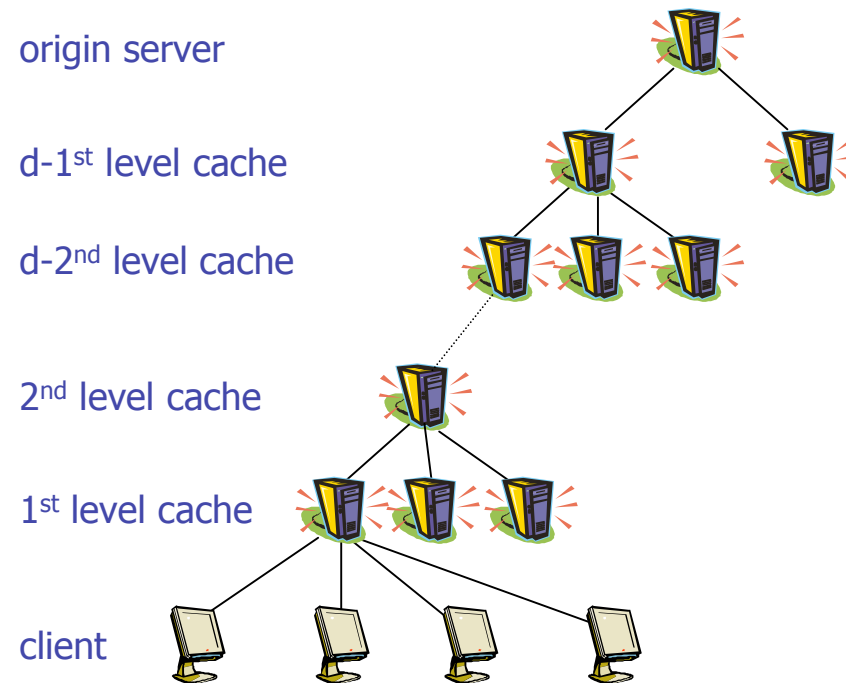




Coordinated servers

Distribution Architectures

- Combined optimization
 - Scheduling algorithm
 - Proxy placement and dimensioning

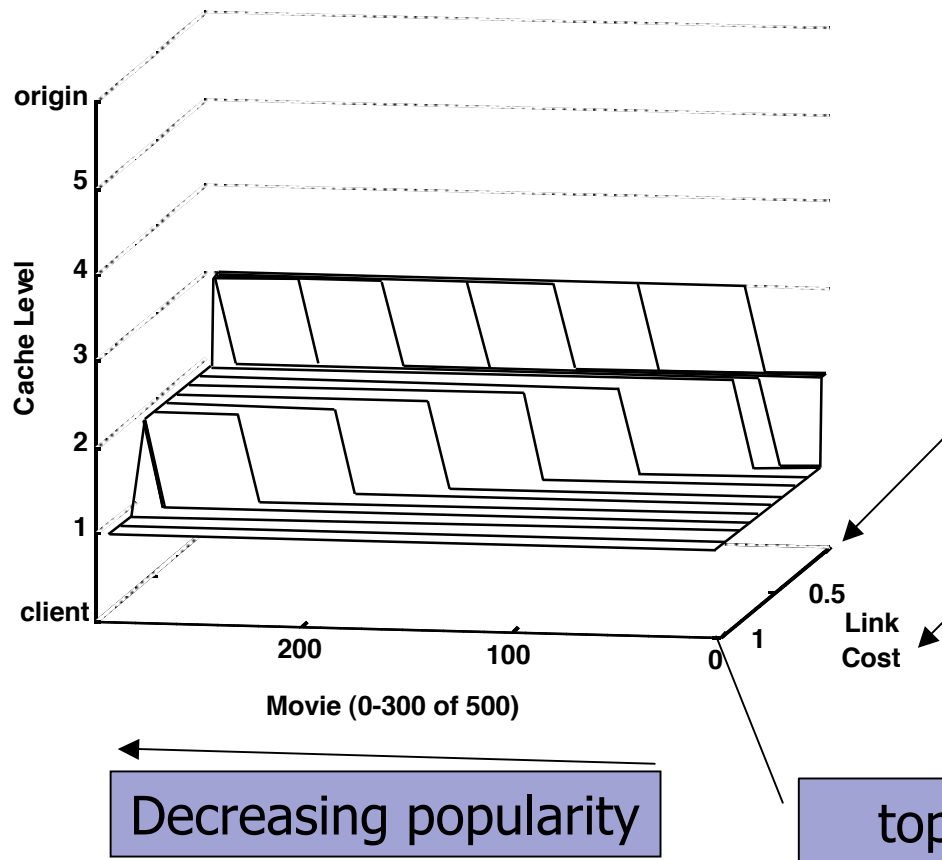


Distribution Architectures

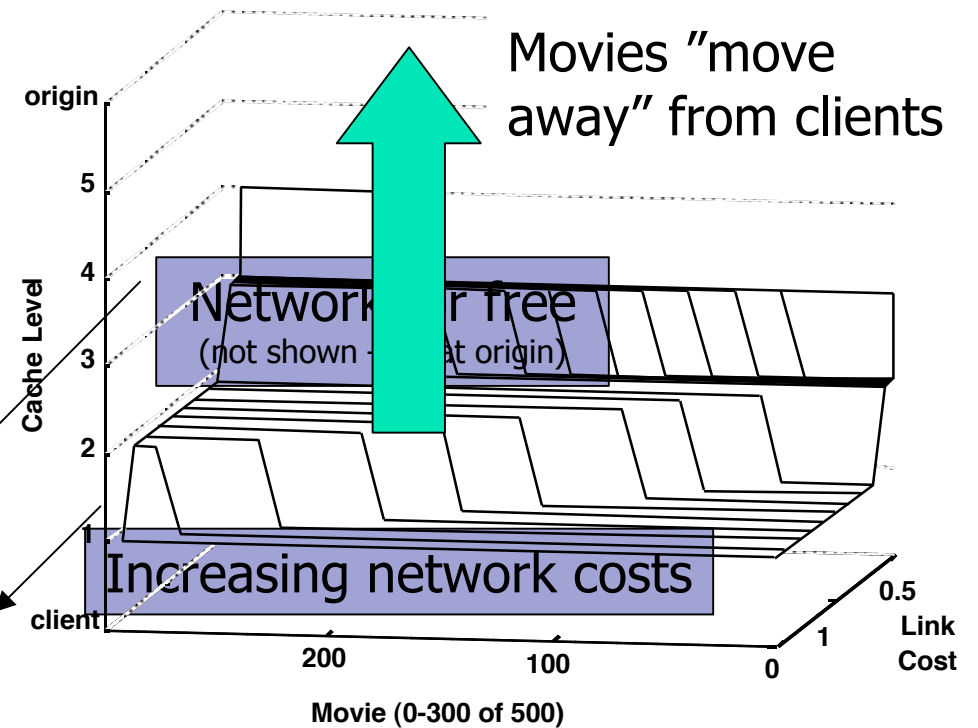
- Combined optimization
 - Scheduling algorithm
 - Proxy placement and dimensioning
- No problems with simple scheduling mechanisms
- Examples
 - Caching with unicast communication
 - Caching with greedy patching
 - Patching window in greedy patching is the movie length

Distribution Architectures

Caching



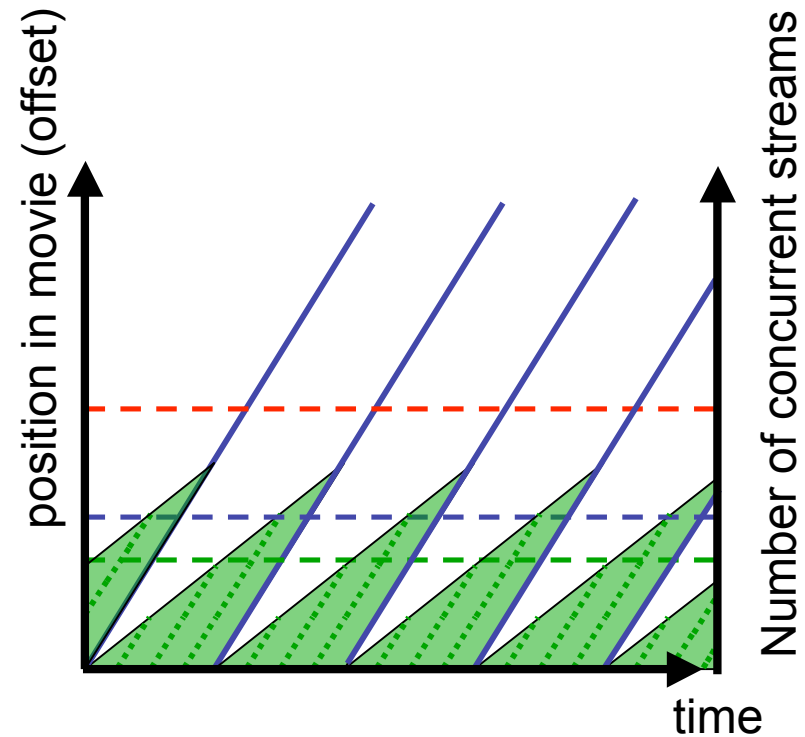
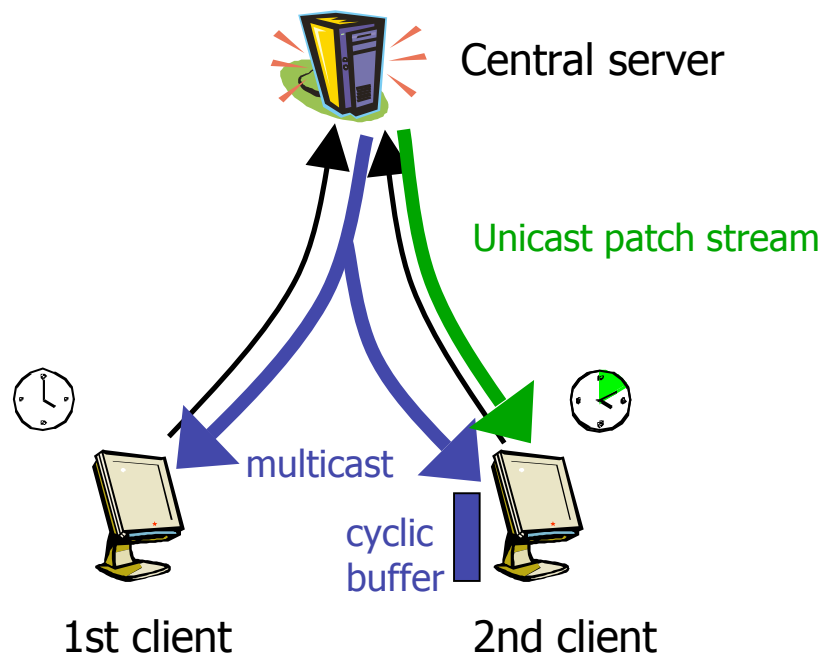
Caching and Greedy Patching



Distribution Architectures

- Combined optimization
 - Scheduling algorithm
 - Proxy placement and dimensioning
- Problems with complex scheduling mechanisms
- Examples
 - Caching with λ -patching
 - Patching window is optimized for minimal server load
 - Caching with gleaning
 - A 1st level proxy cache maintains the “client buffer” for several clients
 - Caching with MPatch
 - The initial portion of the movie is cached in a 1st level proxy cache

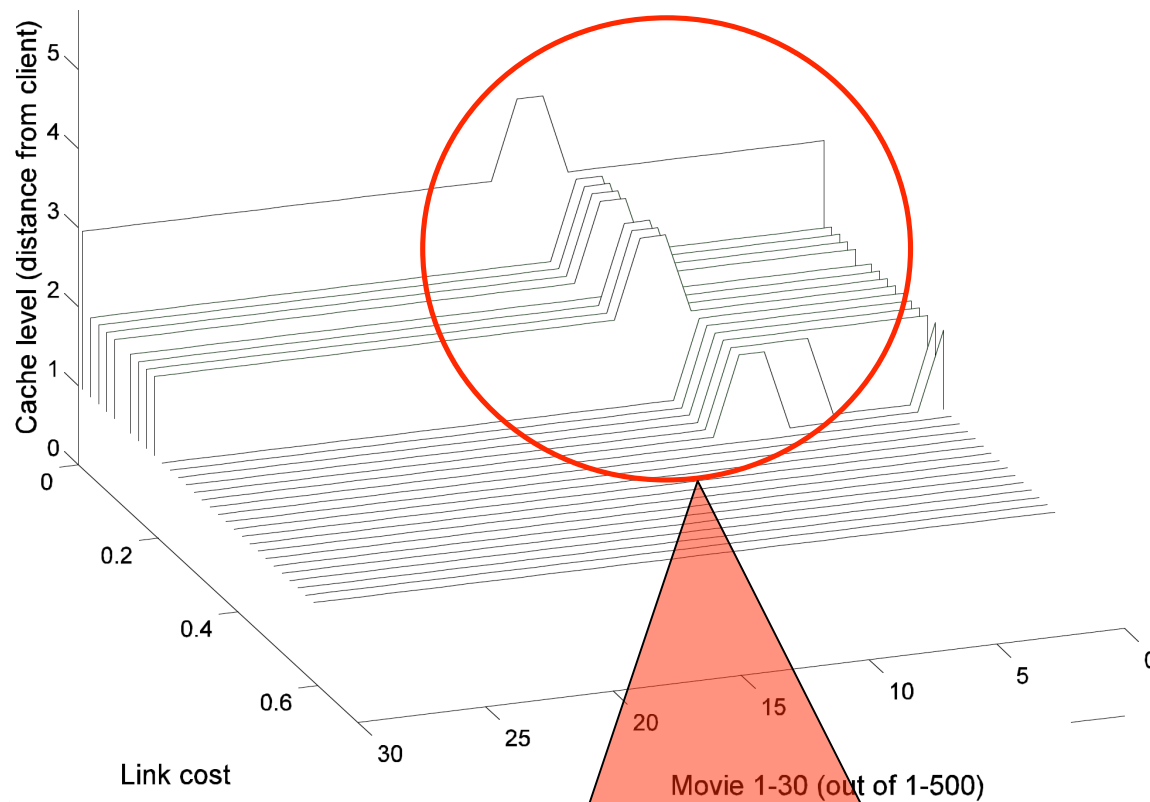
Distribution Architectures: λ -Patching



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

Distribution Architectures: λ -Patching

- Placement for λ -patching



Popular movies may be more distant to the client

Distribution Architectures: λ -Patching

- Failure of the optimization
 - Implicitly assumes perfect delivery
 - Has no notion of quality
 - User satisfaction is ignored
- Disadvantages
 - Popular movies further away from clients
 - Longer distance
 - Higher startup latency
 - Higher loss rate
 - More jitter
 - Popular movies are requested more frequently
 - **Average** delivery quality is lower



Distribution Architectures: Gleaning

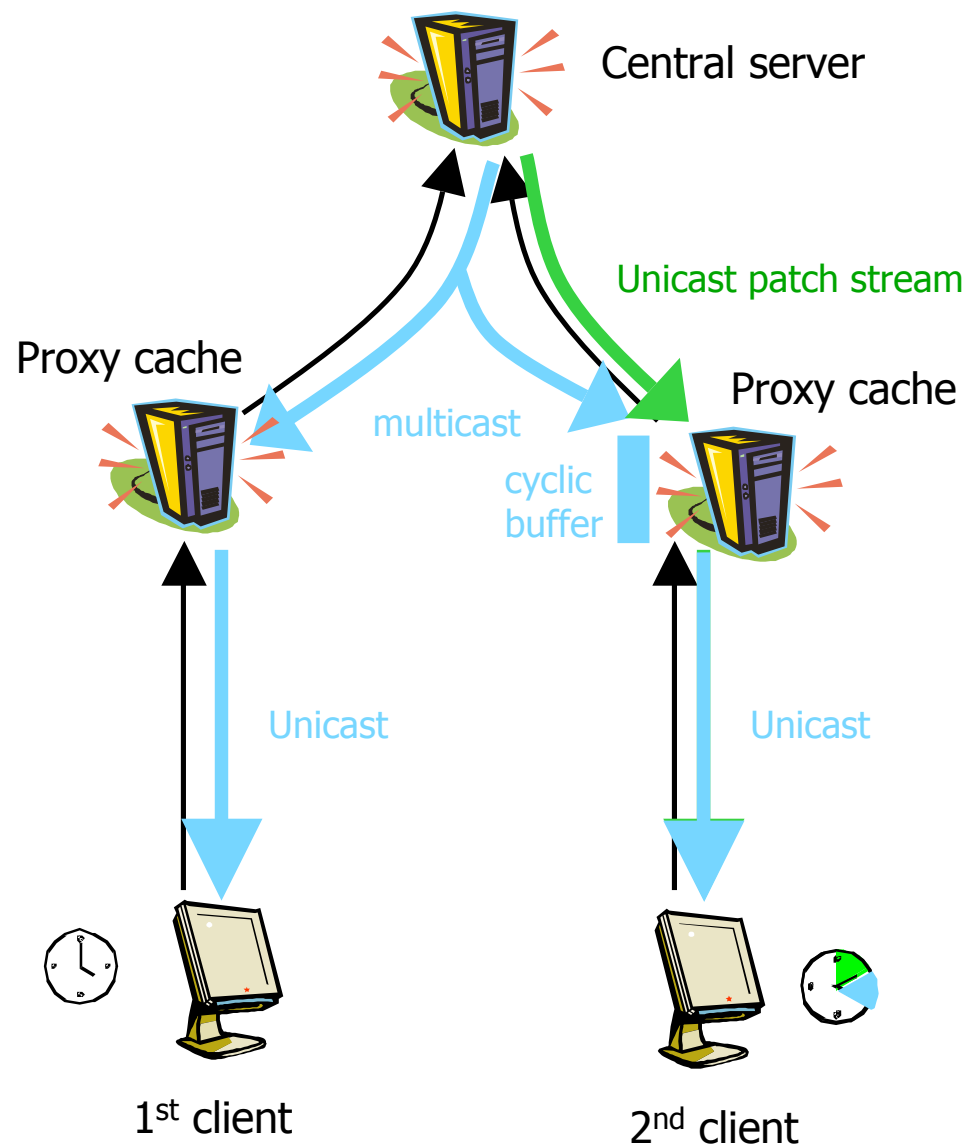
■ Placement for gleaning

— Combines

- Caching of the full movie
- Optimized patching
- Mandatory proxy cache

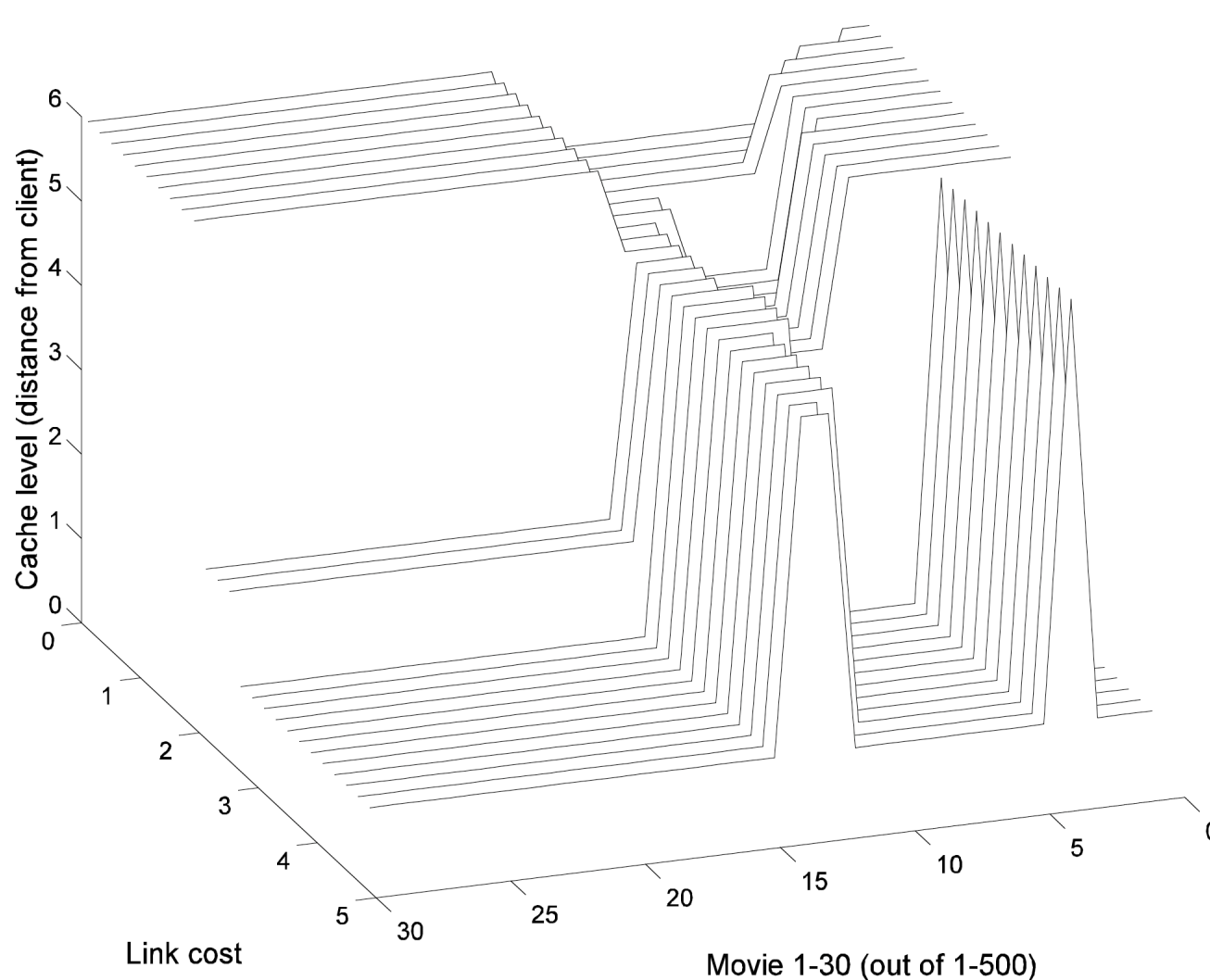
— 2 degrees of freedom

- Caching level
- Patch length



Distribution Architectures: Gleaning

- Placement for gleaning



Distribution Architectures: MPatch

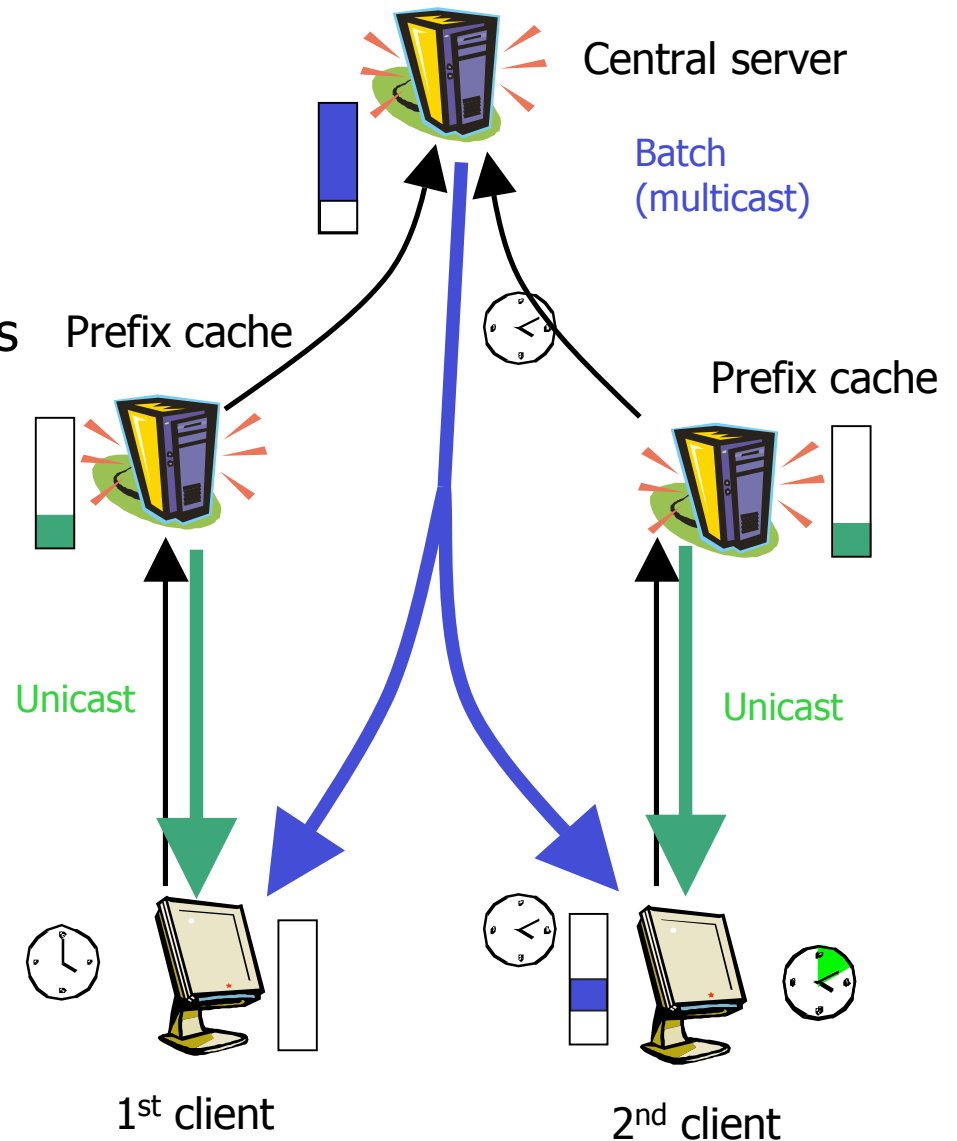
Placement for MPatch

— Combines

- Caching of the full movie
- Partial caching in proxy servers
- Multicast in access networks
- Patching from the full copy

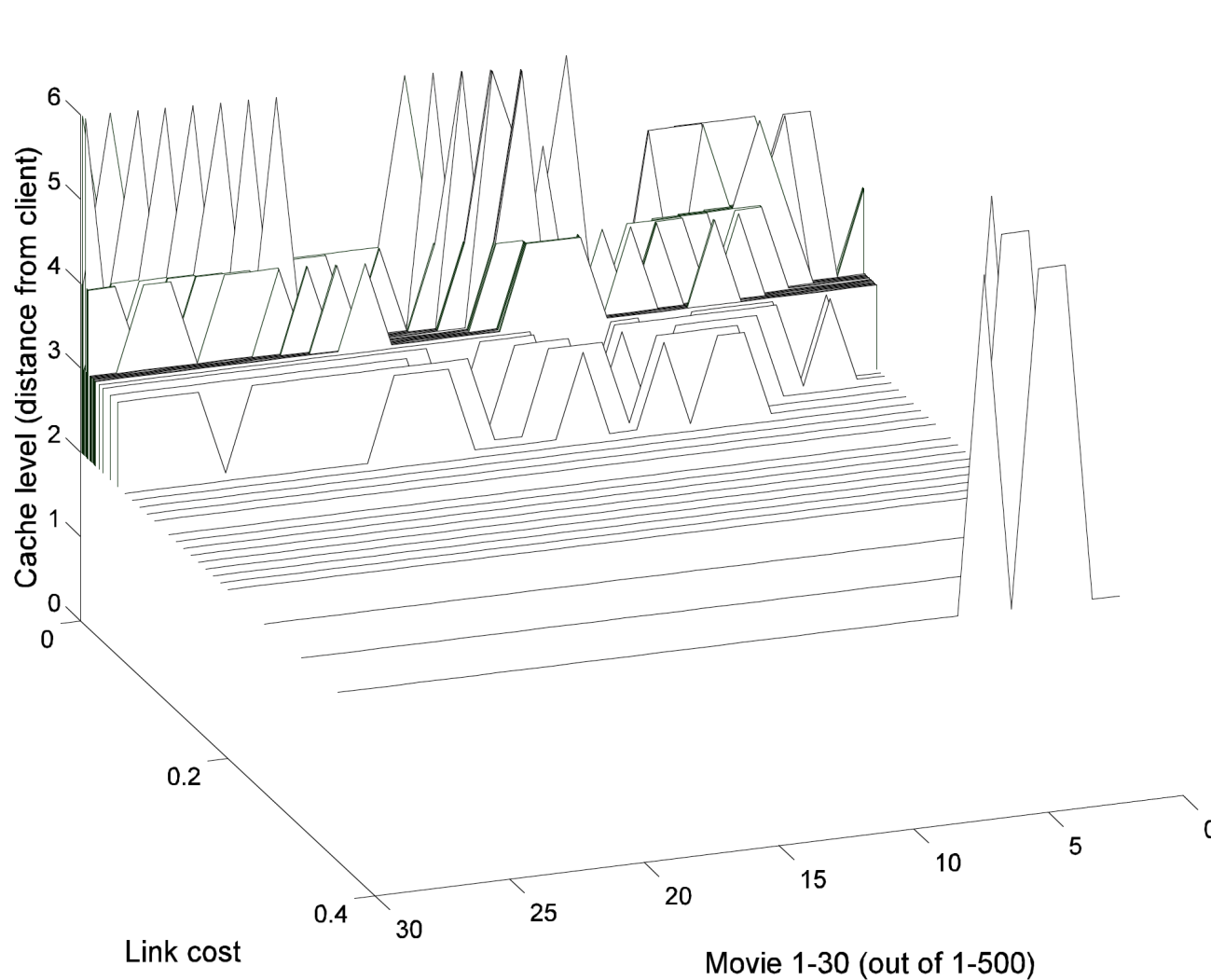
— 3 degrees of freedom

- Caching level
- Patch length
- Prefix length



Distribution Architectures: MPatch

- Placement for MPatch



Approaches

- Current approach does not consider quality
 - Penalize distance in optimality calculation
 - Sort
- Penalty approach
 - Low penalties
 - Doesn't achieve order because actual cost is higher
 - High penalties
 - Doesn't achieve order because optimizer gets confused
- Sorting
 - Trivial
 - Very low resource waste



Distribution Architectures

- Combined optimization
 - Scheduling algorithm
 - Proxy placement and dimensioning
 - Impossible to achieve optimum with autonomous caching
- Solution for complex scheduling mechanisms
- A simple solution exists:
 - Enforce order according to priorities
 - (simple sorting)
 - Increase in resource use is marginal

References

-  S.-H. Gary Chan and Fourad A. Tobagi: "Distributed Server Architectures for Networked Video Services", IEEE/ACM Transactions on Networking 9(2), Apr 2001, pp. 125-136
-  Subhabrata Sen and Jennifer Rexford and Don Towsley: "Proxy Prefix Caching for Multimedia Streams", Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY, USA, Mar 1999, pp. 1310-1319
-  Sridhar Ramesh and Injong Rhee and Katherine Guo: "Multicast with cache (mcache): An adaptive zero-delay video-on-demand service", Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), Anchorage, Alaska, USA, Apr 2001
-  Michael Bradshaw and Bing Wang and Subhabrata Sen and Lixin Gao and Jim Kurose and Prashant J. Shenoy and Don Towsley: "Periodic Broadcast and Patching Services - Implementation, Measurement, and Analysis in an Internet Streaming Video Testbed", ACM Multimedia Conference (ACM MM), Ottawa, Canada, Sep 2001, pp. 280-290
-  Bing Wang and Subhabrata Sen and Micah Adler and Don Towsley: "Proxy-based Distribution of Streaming Video over Unicast/Multicast Connections", Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), New York, NY, USA, Jun 2002
-  Carsten Griwodz and Michael Zink and Michael Liepert and Giwon On and Ralf Steinmetz, "Multicast for Savings in Cache-based Video Distribution", Multimedia Computing and Networking (MMCN), San Jose, CA, USA, Jan 2000
-  Carsten Griwodz and Michael Bär and Lars C. Wolf: "Long-term Movie Popularity in Video-on-Demand Systems", ACM Multimedia Conference (ACM MM), Seattle, WA, USA, Nov 1997, pp. 340-357
-  Carsten Griwodz: "Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure", PhD thesis, Darmstadt University of Technology, Darmstadt, Germany, Apr 2000