OSMOSIS

Scalable Delivery of Real-Time Streaming Media in Ad-Hoc Overlay Networks

Azer Bestavros

Joint work with

Shudong Jin, Abhishek Sharma, and Ibrahim Matta

Web and interNetworking Group

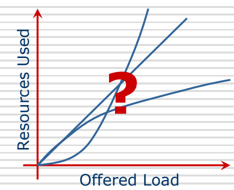
http://www.cs.bu.edu/groups/wing

Old Dominion University

April 18, 2005

Scalable Content Delivery: Why?

- Need to manage resource usage as demand goes up
  - Server load:
    - CPU, memory, etc.
  - Network load:
    - Bytes, Byte-Hops, etc.



- Also need to worry about QoS to clients
  - Delay, response time, jitter, *etc.*

Scalable Content Delivery: How?

Replicate It!

Scalable Content Delivery: How?

- Replicate it from the client side
  - Client caching/prefetching, proxy caching, cooperative caching, server selection, *etc.*
- Replicate it from the server side
  - Servers on steroids, server farms, reverse proxy caching and CDNs, *etc.*
- Replicate it in the network
  - Network caches, multicast, anycast, traffic engineering, *etc.*

Scalable Content Delivery: How?

- Two flavors of replication
  - Caching: Replicate the artifact by storing it at an alternate location (server, CDN, proxy, or client)
  - Multicast: Duplicate the constituent packets of the artifact en route to multiple destinations

Multicast is to synchronous requests what caching is for asynchronous requests

- Hybrids are also possible
  - Use coding to multicast to asynchronous clients [Bestavros:RTAS'96][ByersEtAl:Sigcomm'98][RostByersBestavros:WCW'01]

Scalable Content Delivery: What?

- Static bulk content
  - Early focus of scalable content delivery work
  - Moderate savings (~ 40% max) possible
  - Diminishing % of today's web transactions
- Dynamic bulk content
  - Need to worry about freshness of content
  - Fairly straightforward…

Case closed for bulk content replication!

Scalable Content Delivery: What?

- Dynamic "tailor-made" content
  - Not a unidirectional content exchange!
  - Replicate assembly process vs content
  - Complicated by issues of consistency, coherence, trust, security, code safety, *etc.*

Wide open!

On Consistency and Coherence

June 21^{st}
03:25 am

Why Focus on Streaming Delivery?

- Streaming content
  - Emerging as largest sink of net resources
  - Potential for savings is huge due to more predictable access patterns
  - RT QoS of delivery to client is key
  - Complicated by the bursty nature of network and server conditions
  - Further complicated by the increasingly peer-to-peer nature of content delivery

A gold mine!

Talk Outline

- Motivation
- Taxonomy of Streaming Media Delivery
- OSMOSIS: Play-and-Relay Delivery
  - Network Scalability
    - Effect of Limited Client Bandwidth
    - Effect of Limited Client Memory
  - Server Scalability
    - Effect of Content Popularity
    - Effect of Client Arrival/Departure Dynamics
- Implementation Sketch
- Conclusion

Streaming Delivery: A Taxonomy

- Content Source:
  - Stored vs Live
- Content Encoding:
  - Fixed vs Layered
- Service:
  - Immediate vs Delayed
- Request Arrivals:
  - Synchronous vs Asynchronous
- Playout:
  - Sequential vs Random

Stored	Live
Fixed	Layered
Immediate	Delayed
Synchronous	Asynchronous
Sequential	Random

Multicast: Basic

Pros & Cons

+ Perfect $O(1)$ server scalability
− Depends on network support of multicast
− Cannot support asynchronous request arrivals
? Network scalability $O(n^{1-1/H})$ depends on topology

Stored	Live
Fixed	Layered
Immediate	Delayed
Synchronous	Asynchronous
Sequential	Random

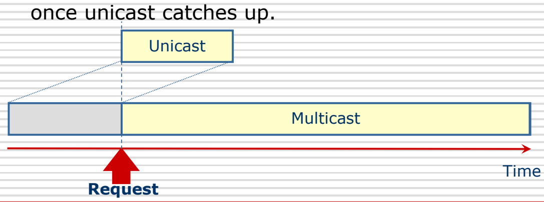
References:

- [JinBestavros:BUCS'02-04]

Multicast: Stream Merging

Client opens two connections

- A temporary unicast channel on which it receives content from beginning of stream.
- A multicast channel on which it receives whatever is being sent. This content is cached and played once unicast catches up.



Multicast: Stream Merging

Pros & Cons

Stored	Live
Fixed	Layered
Immediate	Delayed
Synchronous	Asynchronous
Sequential	Random

\+ Good $O(log(n))$ server scalability
\+ Fairly straightforward
– Depends on network support of multicast
– Requires doubling of client bandwidth
– Bad server scalability $O(n^{0.5})$ under non-sequential access
? Network scalability depends on topology

References:

- [EagerVernon:TKDE'01]
- [MahantiEtAl:Sigcomm'01]
- [JinBestavros:Sigmetrics'02]
- [JinBestavros:BUCS'02-04]

Multicast: Periodic Broadcasting

Server Protocol

- Set up $O(\log(S))$ multicast channels and transmit a given segment on each channel periodically

Client Protocol

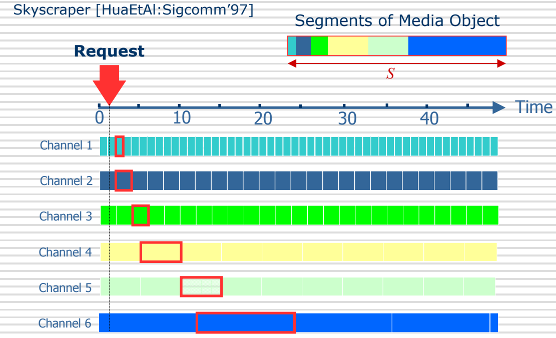
- Step through successive multicast channels (joining at most two at a time) to fetch the entire object

Object Segmentation

- Small earlier segments allow a small start-up delay d. Large later segments keep # of channels low

Multicast: Periodic Broadcasting

Skyscraper [HuaEtAl:Sigcomm'97]



Multicast: Periodic Broadcasting

Pros & Cons

Stored	Live
Fixed	Layered
Immediate	Delayed
Synchronous	Asynchronous
Sequential	Random

\+ Good O(log(n/d)) server scalability
– Depends on network support of multicast
– Requires doubling of client bandwidth
– Bad server scalability $O(n^{0.5})$ under non-sequential access
– Object segmentation and client overhead
? Network scalability depends on topology

References:

- [EagerVernon:MIS'01]
- [MahantiEtAl:Sigcomm'01]
- [JinBestavros:Sigmetrics'02]
- [JinBestavros:BUCS'02-04]

Multicast: Broadcasting & Merger

Server Protocol

- Set up a S/x multicast channels and transmit the object on channel i periodically with phase shift $i*x$

Client Protocol

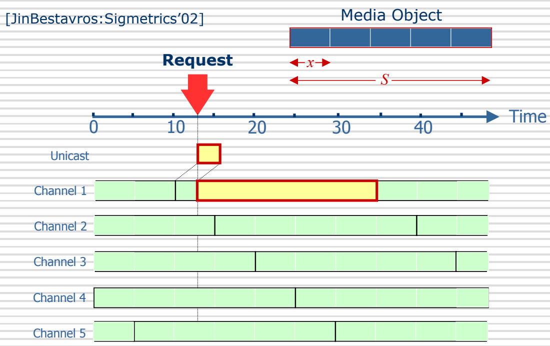
- Join the most recently started multicast and fetch the missed portion using dedicated unicast

Phase Shift

- The value of x is chosen so as to optimize server scalability

Multicast: Broadcasting & Merger

[JinBestavros:Sigmetrics'02]



Multicast: Broadcasting & Merger

Pros & Cons

+ Best possible $O(n^{0.5})$ server scalability
+ Fairly straightforward
+ Lends itself to layered encoding
− Depends on network support of multicast
− Requires doubling of client bandwidth
? Network scalability depends on topology

Stored	Live
Fixed	Layered
Immediate	Delayed
Synchronous	Asynchronous
Sequential	Random

References:
- [JinBestavros:Sigmetrics'02]

End-System Multicast

- Native support for multicast is not paramount, making all previous techniques of limited practical use!
- Solution: End-System Multicast
  - Duplication of packets done by applications at the receivers themselves and not by the network
  - All communication is between the "end systems" and is unicast
  - Efficient "emulation" of native multicast depends on "small world" properties of underlying network topology [JinBestavros:BUCS'02-04]

Streaming Delivery: Summary

	Minimize Net Cost		Minimize Server Load	
Synchronous Access	Native IP Multicast	End-system Multicast	End-system Multicast	Native IP Multicast
Asynchronous Access	?	OSMOSIS		Periodic Broadcast & Patching
		No IP Multicast Infrastructure		

Talk Outline

- Motivation
- Taxonomy of Streaming Media Delivery
- OSMOSIS: Play-and-Relay Delivery
  - Network Scalability
    - Effect of Limited Client Bandwidth
    - Effect of Limited Client Memory
  - Server Scalability
    - Effect of Content Popularity
    - Effect of Client Arrival/Departure Dynamics
- Implementation Sketch
- Conclusion

OSMOSIS: Play-and-Relay Delivery

Target Settings and Applications

- *Supporting asynchrony and VCR functionality*
  - Internet: End-System Multicast
  - P2P Networks: Sharing of Live Streams
- *Relaying live clips from unique vantage points*
  - In-Situ Networks: Public Safety and Battlefield
  - Ad-Hoc Overlays: Entertainment Applications

From "Content Sharing" to "Osmosis"

- Current P2P content sharing assumes
  - No central server available or desirable
  - A peer *can store entire object*
  - Access to content comprises
    - Finding object (e.g., using DHT techniques *a la* CHORD, CAN, Tapestry, Viceroy, …)
    - Downloading object from a peer (or more)
- What if a peer cannot store entire object?
  - Say hello to delivery via OSMOSIS!

Ad-Hoc Overlays for RT Streaming

- Think of a "read-once" Gnutella network, in which a client:
  - comes in at a point in time and space
  - requests a RT stream (e.g., movie)
  - reads stream (e.g., watches movie)
  - does not care (or can't) store object
  - but willing to keep part of object in cache
- Need to devise an ad-hoc peer-to-peer delivery network to enable an efficient solution to above problem

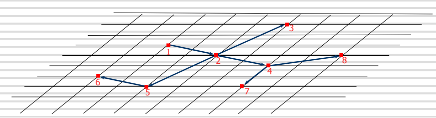
OSMOSIS: Assumptions

- Client has (very) limited storage capacity, compared to size of object
  - VoD on a PDA (e.g., watching a movie, a soccer game, or getting a battlefield conditions update)
  - Ad-hoc delivery of "almost" live RT content (e.g., video distribution using an overlay of wireless PDAs)
  - Scalable diffusion of sensor data (e.g., multicast of sensor data in distributed embedded systems)
- Communication cost ~ function of distance
  - # hops (a la Internet)
  - Physical distance (a la wireless)

OSMOSIS: Play-and-Relay

General Idea

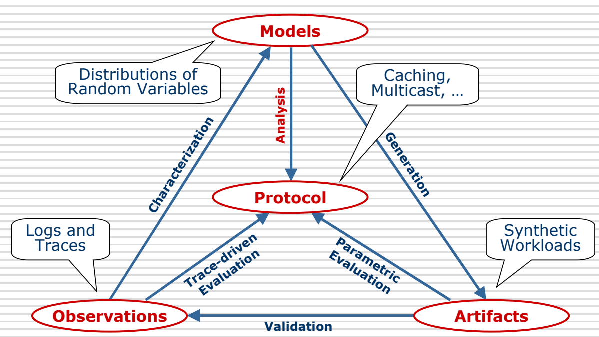
- Clients keep streams in caches after playout
- Later clients receive the objects from close-by peers
- No multicast functionality support from network layer
- Clients have abundant bandwidth and local caches



Talk Outline

- Motivation
- Taxonomy of Streaming Media Delivery
- OSMOSIS: Play-and-Relay Delivery
  - Network Scalability
    - Effect of Limited Client Bandwidth
    - Effect of Limited Client Memory
  - Server Scalability
    - Effect of Content Popularity
    - Effect of Client Arrival/Departure Dynamics
- Implementation Sketch
- Conclusion

A Word on Research Methodology



OSMOSIS: Network Scalability

Objective:

Characterize total network link cost ***L(n)*** versus OSMOSIS tree size ***n***.

- $L(n)$ measures the cost (hop distance) to serve n concurrent clients, normalized by the cost to serve a single client, *i.e.*, $L(1)=1$
- $L(n)$ characterizes the scaling behavior in terms of network link cost

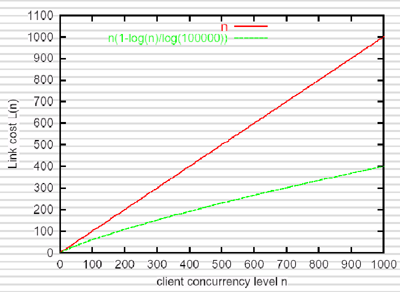
Canonical Network Topologies

Neighborhood Expansion (NE) function $E(d)$ is the fraction of graph vertices reachable in d hops

- Random graphs have exponential expansion
  $E(d) \sim e^{d}$
- Mesh (2D, 3D, …) graphs have power-law expansion
  $E(d) \sim d^{a}$
- How about small-world graphs (a.k.a. Internet like)?
  $E(d) \sim e^{d} \rightarrow d^{a}$

Random Networks: NE ~ Exp

$$L(n) \sim n(1-\ln(n)/\ln(N))$$



Mesh Networks: NE ~ Power

$$L(n) \sim n^{1-1/H}$$

where H is the power-law exponent

Efficient for small H (*e.g.*, $1 < H < 2$)

As H increases $L(n)$ approaches unicast

Real Networks: Internet

- Small World graphs
  - Average Path Length: Short (~ random)
  - Clustering Coefficient: High (>> random)

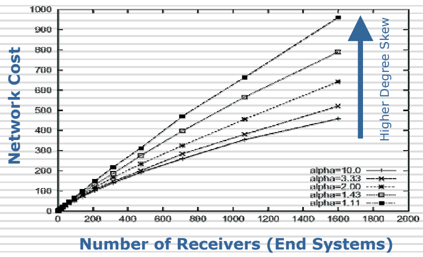
Graphs	AS2000	AS2001	Lucent	Scan+Lucent
L	3.655	3.627	10.02	8.803
C	0.4399	0.4578	0.1001	0.0996
L_{random}	6.721	6.797	10.49	11.38
C_{random}	0.00035	0.00025	0.000022	0.000007

Understanding Internet Topology

- Internet topology is small world. Why?
  - Local Connectivity:
    "Love thy neighborhood (routers)" phenomenon
  - Preferential Connectivity:
    "Rich (routers) get richer" phenomenon
- Both phenomena present with different relative strengths in network topologies at the router and AS levels

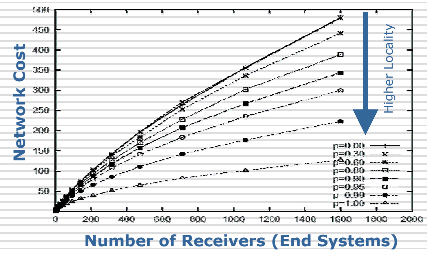
OSMOSIS: Impact of Topology

Preferential Connectivity ➔ Worse Scalability

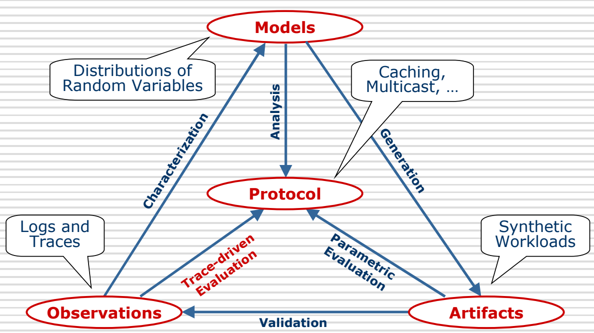


OSMOSIS: Impact of Topology

Localized Connectivity ➔ Better Scalability

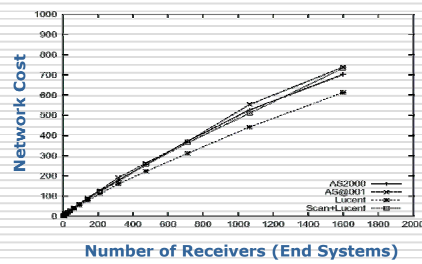


A Word on Research Methodology



OSMOSIS: Impact of Topology

Empirical results based on Internet measurements



How About Ad-Hoc Networks?

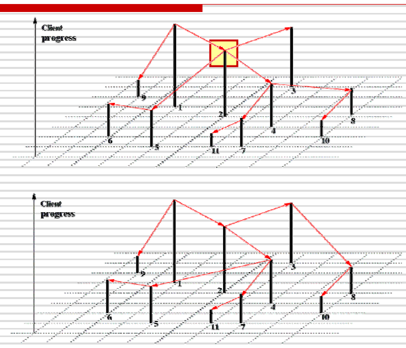
- Likely to be closer to low-dimensional mesh networks. Why?
  - Wireless/radio networks promote locality.
  - No concept (or opportunity) for preferential connectivity.

➔ Excellent scalability for OSMOSIS!

OSMOSIS with b/w Constraints

- Clients have limited bandwidth to receive and send objects (e.g., 2 upload streams)
- Optimal solution is NP-hard (reduced to degree-constrained spanning tree)
- We use a greedy solution: each client receives the object from the nearest ongoing peer who still has abundant bandwidth

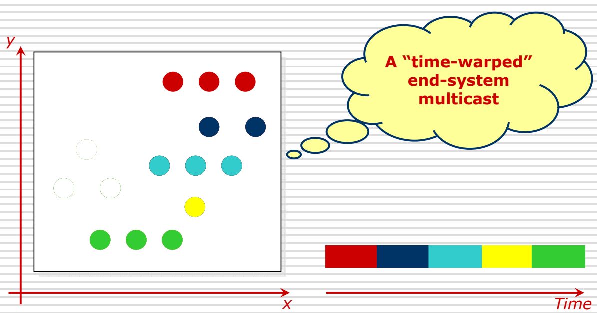
OSMOSIS with b/w Constraints



OSMOSIS with Limited Cache

- Clients have limited cache to keep partial objects
- For now, we use a simple FIFO replacement: a sliding window of the last *s* seconds (or *b* bytes) from the stream ➔ "Cache & Relay"
- A new client receives object from closest peer caching the starting prefix of the object and with enough bandwidth

Content is Always "In Transit"

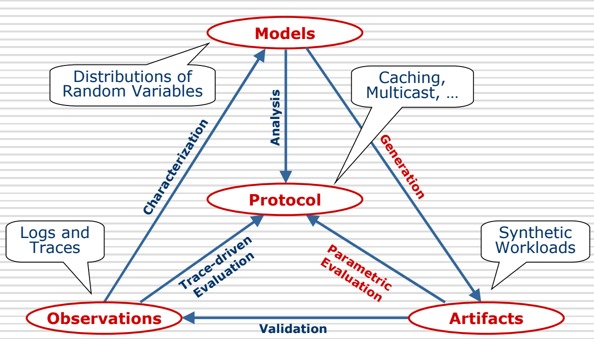


OSMOSIS Evaluation

Goals:

- Validate the theoretically derived network cost
- Study the effect of different network topologies
- Study the effect of limited client bandwidth
- Study the effect of limited client cache

A Word on Research Methodology

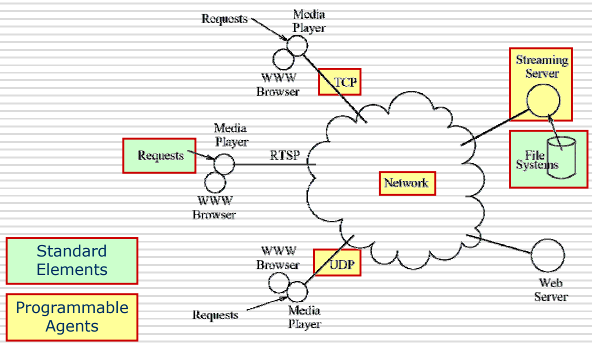


GISMO Workload Generator

- GISMO: A toolkit to generate synthetic streamed media workloads [JinBestavros:PER'02]
- GISMO generates
  - A set of "placeholder" streaming media objects, which can be installed on servers
  - Requests to these objects, initiated by clients subject to a prescribed access model

http://csr.bu.edu/gismo

GISMO: Components



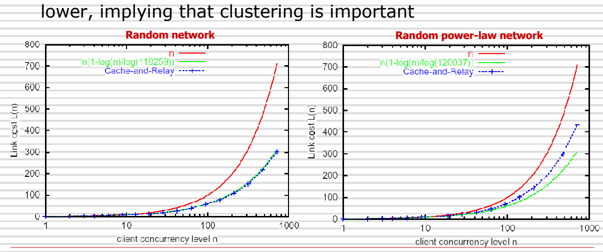
Simulation Settings

Networks	# of Nodes	Mean Degree	Description
Random network	119,259	3.19	Uniform probability of having an edge between 2 nodes (ER model)
Random Power-law network	120,037	3.18	Power-law degree, but edges are created using a random matching
Small-world network	120,000	3.11	Power-law but preference for short edges → stronger clustering
Real router-level network	112,669	3.21	A router-level Internet map (1999 lucent) available to public

Workload:
Poisson arrival process with client joining at random point in network

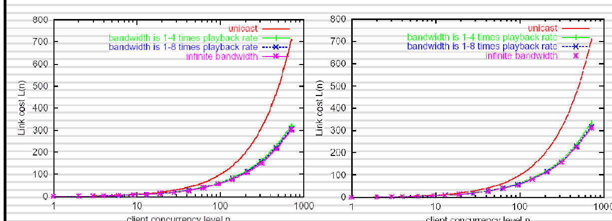
Network Cost Validation

- Using random network, link cost is predicted by our analysis
- Using power-law random network, link cost is higher than that using random network
- Using small-world network (not shown here), link cost is lower, implying that clustering is important



Effect of Limited Bandwidth

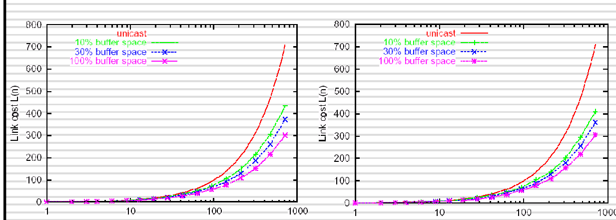
- Client bandwidth is chosen uniformly in a range
- With limited bandwidth, the greedy approach is still effective (close to that with infinite bandwidth)



(a) Random network (c) Small-world power-law network

Effect of Limited Cache Size

- Each client has limited bandwidth (as before) and limited cache
- OSMOSIS still reduces network cost, significantly
- There are yet better cache management (other than FIFO)



(a) Random network (d) Router-level Internet map

Talk Outline

- Motivation
- Taxonomy of Streaming Media Delivery
- OSMOSIS: Play-and-Relay Delivery
  - Network Scalability
    - Effect of Limited Client Bandwidth
    - Effect of Limited Client Memory
  - Server Scalability
    - Effect of Content Popularity
    - Effect of Client Arrival/Departure Dynamics
- Implementation Sketch
- Conclusion

OSMOSIS: Server Scalability

Assumptions:

- There is one server which stores the entire stream.
- Client will go to server only if it cannot find content in any of the OSMOSIS caches
- Retrieval must be instantaneous—no delays are allowed in playout

Theoretically speaking, OSMOSIS could yield constant server load!

OSMOSIS: Server Scalability

- Why Access the Server?
  - Content not found in any peer cache
    - ➔ *OSMOSIS peer density is not enough*
      - Arrival rate is not high enough to keep stream "alive"
      - Caches are too small compared to stream length
  - Recover from upstream node departures
    - ➔ *OSMOSIS peer churn is too high*
      - Clients do not stick to stream long enough
      - Access is non sequential due to VCR functionality

OSMOSIS: Prefetch & Relay

- **Idea:**

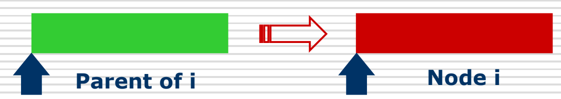
  Reduce load on server by buffering *future* content (as opposed to past) to allow clients to tolerate upstream churn

- **How?**

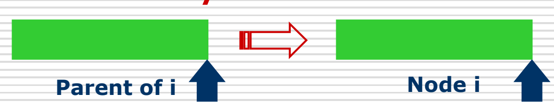
  Use additional download bandwidth to "fast forward" the buffer so that all buffered content is from future!

OSMOSIS: Cache or Prefetch?

Cache & Relay

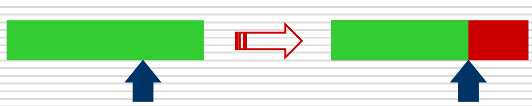


Prefetch & Relay



OSMOSIS: Cache or Prefetch?

In General: Play & Relay



Why both "cache" and "prefetch"?

- Keeping "past" content reduces the load on server due to client arrivals ➔ Caching is good
- Keeping "future" content reduces the load on server due to client departures ➔ Prefetching is good
- Tradeoff is needed! Clients will keep a fraction β of their buffers for future content.

OSMOSIS: Parameters

- Download Bandwidth (α)

  Download bandwidth is α times the playout rate

- Arrival Process (λ)

  Poisson with rate λ

- Prefetch Ratio (β)

  Fraction β of buffer is used for prefetching

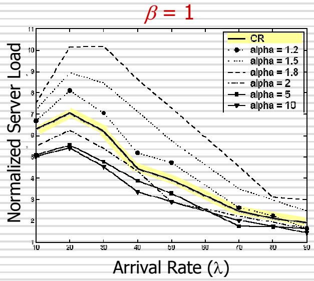
- On-Time (θ)

  We assume a client watches stream for θ% of stream

How do the above parameters affect the load on the server?

OSMOSIS: Server Scalability

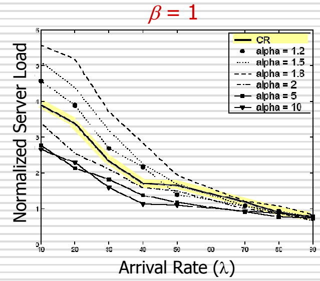
Cache = 5% of Stream Size
On-Time = 100% of Stream
$\beta = 1$



- When buffer is too small prefetching is counter productive!
- Need bandwidth at least twice the playout rate for prefetching to be better than caching.

OSMOSIS: Server Scalability

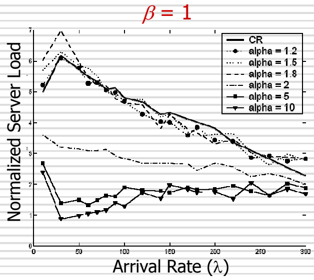
Cache = 10% of Stream Size
On-Time = 100% of Stream
$\beta = 1$



- With proper buffer sizing prefetching reduces server load by 25%
- As expected, with enough peer density, server load becomes constant, even with limited bandwidth and buffer size

OSMOSIS: Server Scalability

Cache = 10% of Stream Size
On-Time = 5% of Stream
$\beta = 1$



- Prefetching is very effective in reducing server load due to client churn!

Talk Outline

- Motivation
- Taxonomy of Streaming Media Delivery
- OSMOSIS: Play-and-Relay Delivery
  - Network Scalability
    - Effect of Limited Client Bandwidth
    - Effect of Limited Client Memory
  - Server Scalability
    - Effect of Content Popularity
    - Effect of Client Arrival/Departure Dynamics
- Implementation Sketch
- Conclusion

dPAM Architecture

- A client needs to locate the peer with the proper content for it to prefetch
  - Content ID and time when playout started uniquely constitute a "name"
  - Use controlled flooding or DHT techniques to locate peer to be used for prefetching
  - Implemented for World Cup 2002!
- Other optimizations
  - Need techniques to ensure load balancing
  - Adapt OSMOSIS parameters (e.g., β)
  - Support for multiple servers

Conclusion

- OSMOSIS is a scalable approach to serve asynchronous clients
  - Network cost depends on the topological properties of the network (random network, power-law, clustering *etc.*)
  - Server cost depends on popularity of content and client churn rate
  - Effective even when clients have limited bandwidth and cache

Related Links

http://www.cs.bu.edu/groups/wing

http://www.cs.bu.edu/~best