dPAM: A Distributed Prefetching Protocol for Scalable Asynchronous Multicast in P2P System

from INFOCOM 2005

presented by wdchiang

Outline

- Introduction
- Pre-Fetch-And-Relay
- Server Bandwidth Requirement
  - Analysis
- Performance Evaluation
- Conclusion

Introduction

- Motivating Application
  - Directing all requests to the “source server” is neither scalable nor practical.
  - Using asynchronous multicast approaches requiring multicast capabilities (e.g. periodic broadcasts) is not practical.
  - Assuming that every node is capable (or willing) to store the entire feed for future access by other nodes is not warranted.

Introduction

- Leveraging Local Storage for Scalable Asynchronous Multicast in P2P Systems
  - In cache-and-relay approach, cache content could be used by other nodes requesting the feed within some bounded delay.
  - Problem:
    - When a node leaves the system, any other nodes receiving the feed from that node are disconnected.
  - Problem:
    - Such disconnected nodes must be treated as new arrivals, which in turn presents added load to the server.

Introduction

- Cache-And-Relay
  - Assume that each client is able to buffer the streamed content for a certain amount of time after playback by overwriting its buffer in a circular manner.

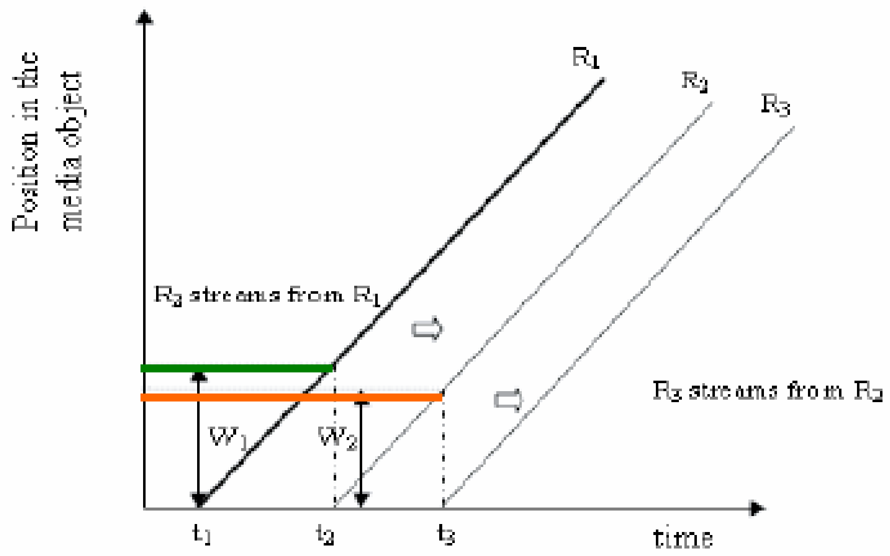


Fig. 1. Overlay-based asynchronous streaming

Introduction

- Cache-And-Relay
  - Stream patching technique:
    - In Fig. 2, if the download rate is greater than the playback rate, R2 can open two simultaneous streams and starts downloading from R1 at playback rate as well as from server for missing “H”.

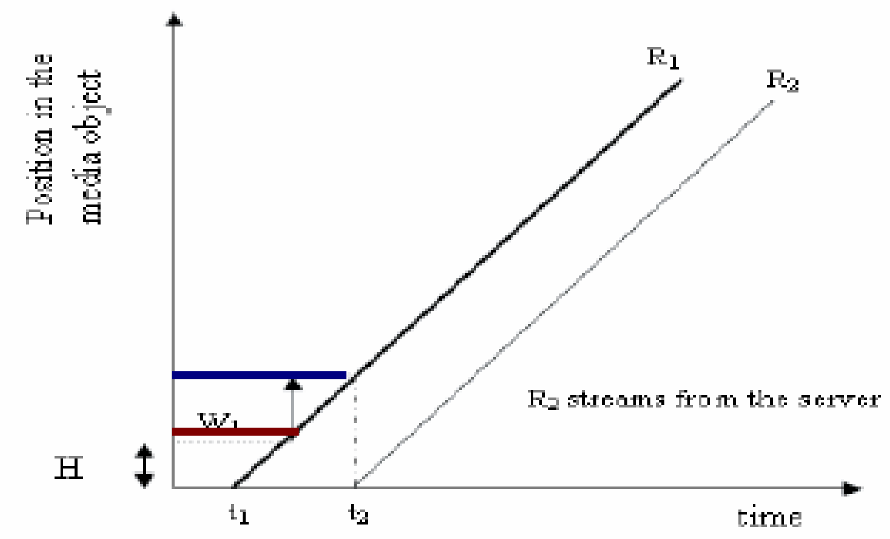


Fig. 2. Overlay-based aynchronous streaming

Introduction

- Cache-And-Relay
  - In the event of a client's departure
    - all the clients downloading from the buffer of the departing client will have to switch their streaming session either to some other client or the server

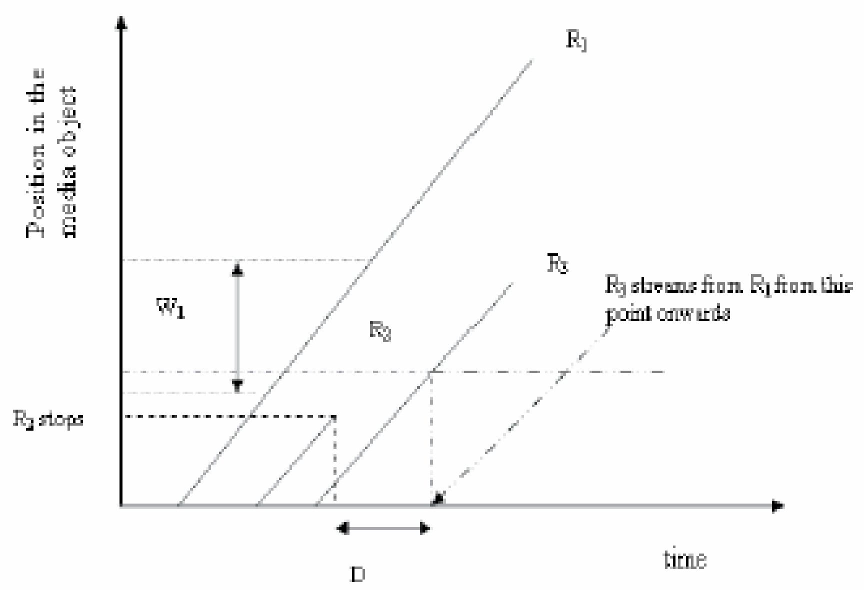


Fig. 3. Delay in finding a new download source

Pre-Fetch-And-Relay

- Pre-Fetch
  - When the download is greater than the playback rate, a client can *pre-fetch* content to its buffer.
  - Pre-fetching content can help achieve a better playback quality in overlay multicast.
  - In the case of *Cache-and-relay*, R3 will have to open a stream from the server as soon as it realizes that R2 has departed and continue downloading from the server for D seconds.
  - In the case of *Pre-fetch-and-relay*, if the time required to play out the pre-fetch content is larger than the delay, the playback at R3 would not be disrupted upon R2's departure.

Pre-Fetch-And-Relay

- Control Parameters
  - α = Download rate/Playback rate
    - without loss of generality, take the playback rate to be 1 byte/second.
    - download rate is equal to α byte/second
    - assume $\alpha > 1$
  - Tb
    - The time it takes to fill the buffer at the download rate.
    - The actual buffer size at a client is α*Tb
  - β = Future Content/Past Content

Pre-Fetch-And-Relay

- Constraints in the case of an arrival
  - Theorem 1.

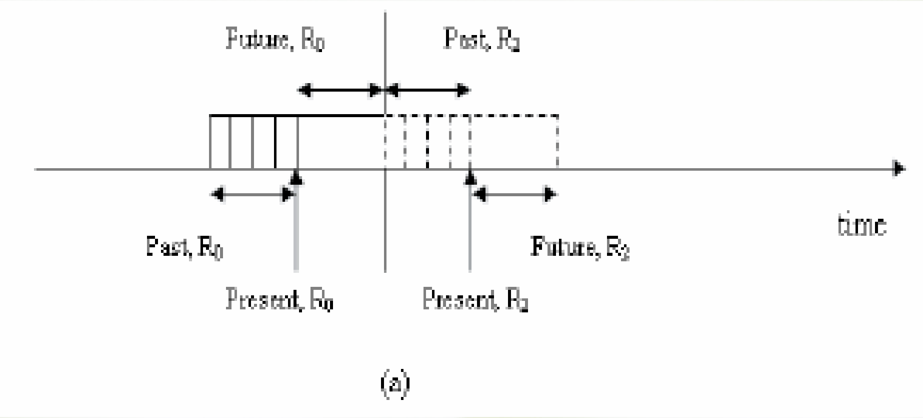
    A newly arrived client R0 can download from the buffer of R1 if the following conditions are satisfied:
    - The inter-arrival time between R0 and R1 is less than Tb, or
    - If the inter-arrival time between R0 and R1 is greater than Tb, then α should be greater than or equal to 2, R1 must be overwriting the content in its buffer at the playback rate and the size of the content missing from R1's buffer should be less than or equal to α*Tb

Pre-Fetch-And-Relay

- Constraints in the event of a departure
  - Assume that
    - R0 is streaming from R1's buffer
    - R1 leaves the network at time t=td
    - Available buffer size is α*Tb bytes
    - (α*Tb) * [1/(1+ β)] bytes for past content
    - (α*Tb) * [β /(1+ β)] bytes for future content
    - Playback rate is equal to 1 byte/second

Pre-Fetch-And-Relay

- Constraints in the event of a departure
  - If α =1, then after R1's departure, R0 can download from another client R2's buffer iff the content buffers overlaps (partially).



Pre-Fetch-And-Relay

- Constraints in the event of a departure
  - Any client that is ahead of R0, in terms of playing out the stream, would have some content that R0 needs to download missing from its buffer and hence, unsuitable for R0 to download from.

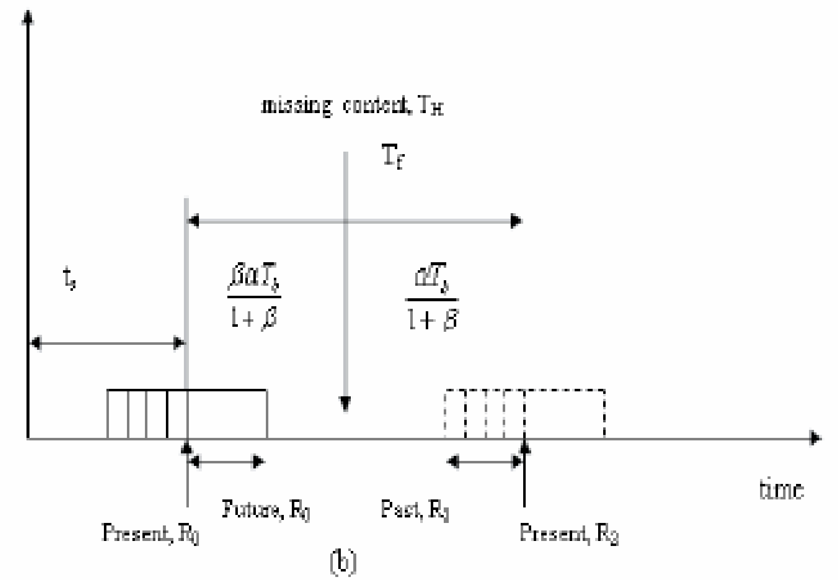


Fig. 4. Buffers of R_0 and R_2

Pre-Fetch-And-Relay

- Constraints in the event of a departure
  - Assume that the "missing" content is TH bytes
  - If α >1, R0 can open two simultaneous streams, one from the server and the other from R2, and terminate its stream from the server after it has downloaded the "missing" content.
  - The following constraints must be satisfied by the size of the "missing" content, TH byes, for R0 to able to stream from R2's buffer:
    - Constraint imposed due to α
    - Constraint imposed by the size of the buffer

Pre-Fetch-And-Relay

- Constraints in the event of a departure
  - Constraint imposed due to α
    - The time taken by R0 to play out the pre-fetched content in its buffer and the "missing" content, TH bytes, is equal to $(\alpha * Tb) * [\beta / (1+\beta)] + TH >= TH / (\alpha - 1)$ ....(1)
    - If $\alpha >= 2$, the condition (1) is always satisfied.
    - The streaming patching can be used in the case of a departure even when $1 < \alpha < 2$ if a client has sufficient pre-fetched content.
  - Constraint imposed by the size of the buffer
    - $TH <= (\alpha * Tb) * [1/(1+\beta)]$ ....(2)

Server Bandwidth Requirement

- Analysis
  - Consider the case of a single CBR media distribution
  - The client requests are generated according to a Poisson process with rate λ
  - The time spent by a client downloading the stream is exponentially distributed with rate μ

Server Bandwidth Requirement

- Analysis
  - Arrivals: a new arrival, R0, would have to download from the server in either of the following two cases:
    - The inter-arrival time between R0 and the arrival immediately preceding R0, say R1, is greater than W;
      - where W = Tb if $1 < \alpha < 2$ or
      - $W = (\alpha * Tb) + (\alpha * Tb) * [1/(1+\beta)]$ if $\alpha >= 2$
    - Suppose R0 arrives at time t=t0. If all the users that arrived during the interval TD=[t0-W, t0) have already departed, R0 would have to download from the server.

Server Bandwidth Requirement

- Analysis
  - Arrivals:
    - Case 1: $P\{ w > W \} = e^{-\lambda W}$
    - Case 2:
      - Let yi be the time spent by client Ri downloading the stream before it depart.
      - Let the inter-arrival time between user Ri and R0 be wi.
      - If yi <= wi, Ri would have departed by the time R0 arrives.
      - Let A represent the event that R0 has to download from the server:

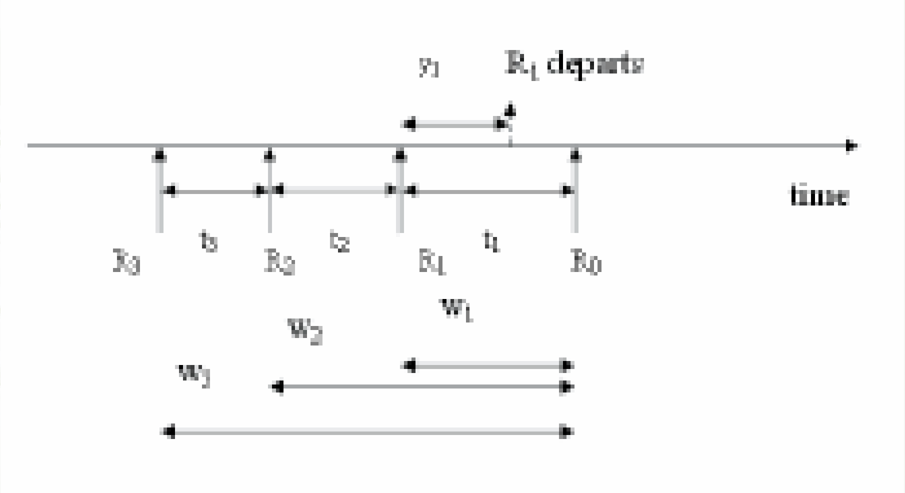
        $P\{ \text{Event } A | N=n, y_i, w_i \} = P\{N=n\} * \prod_{(i=1 \text{ to } n)} P\{ y_i <= w_i \} * P\{w_i\}$

Server Bandwidth Requirement

- Analysis
  - Arrivals
    - Case 2
      - Since $w_i \in [0, W)\ \forall i = 1, \ldots, n$
      - P{ Event A }

        $$= \sum_{n=1}^{\infty} \int_0^W \cdots \int_0^W P\{N=n\} * \prod_{i=1}^{n} P\{ y_i \le w_i \} * P\{ w_i \}\, dw_1 \ldots dw_n$$
      - P{ a new arrival goes to the server }

        = P{ w>W } + P{ Event A }



Server Bandwidth Requirement

- Analysis
  - Departures
    - Let R1 depart at time t=td such that by this time R0 has been downloading and playing out the stream for a duration of ts time units

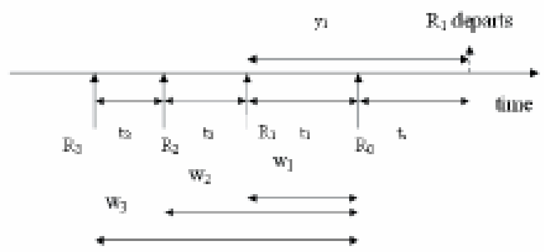


Fig. 6. Departure of R_1

Server Bandwidth Requirement

- Analysis
  - Departures
    - The difference, in terms of number of bytes, between the "present" of R0 and R2, represented by Tf, is
      - Tf
        = "future" content at R0 + "missing" content + "past" content at R2
        = (α *Tb)*[β /(1+ β)] + TH + (α *Tb) * [1/(1+ β)]
      - the maximum value of Tf is

$$T_f = \begin{cases} (\alpha \times T_b)\left(\frac{\beta-\alpha+2}{(1+\beta)(2-\alpha)}\right) & \text{if } \alpha \le \left(\frac{2+\beta}{1+\beta}\right) \\ (\alpha \times T_b)\left(\frac{2+\beta}{1+\beta}\right) & \text{otherwise} \end{cases}$$

Server Bandwidth Requirement

- Analysis
  - Departures
    - On R1's departure R0 can download from only those clients that arrived at most Tf time units before R0
    - If all the clients Ri, i=2,…,n have departed by the time td = t0+ts, R0 would have no option but to download from the server on R1's departure.
    - Let Event B represent the situation that R0 downloads from the server on R1's departure.
      - P{ Event B | N=n, ts, yi, wi }
        = P{ N=n }P{ts} Π(i=2 to n)P{wi}P{yi<= ts+wi}

$$P\{\text{Event B}\} = \int_0^{\infty} \int_0^{T_f} (Expression)\ dw_2 dt_s \qquad (6)$$

where $Expression^3$ is:

$$\left(\frac{e^{-\lambda T_f}(\lambda T_f)^2}{2!}\right)(\mu e^{-\mu t_s})\left(\frac{\lambda(\lambda w_2)e^{-\lambda w_2}}{1!}\right)(1 - e^{-\mu(t_s+w_2)})$$

Server Bandwidth Requirement

- Analysis
  - Server Load
    - P{ Event S } = P{ w>W } + P{ Event A } + P{ Event B}
    - the average number of client requests that download the stream from the server is λ * P{ Event S }
  - Incorporating the delays
    - If D <= (α *Tb)*[β /(1+ β)] , R0 can absorb the delay without any disruption of its playback.
    - Assuming that the delay is uniformly distributed

Performance Evaluation

- Simulation Model

Figure	Buffer size	β	$(1/\mu)$	Delay
8	5	100,000	1000	No
9	10	100,000	1000	No
10	20	100,000	1000	No
11	10	100,000	100	No
12	10	1	1000	Yes
13	10	100,000	1000	Yes

Performance Evaluation

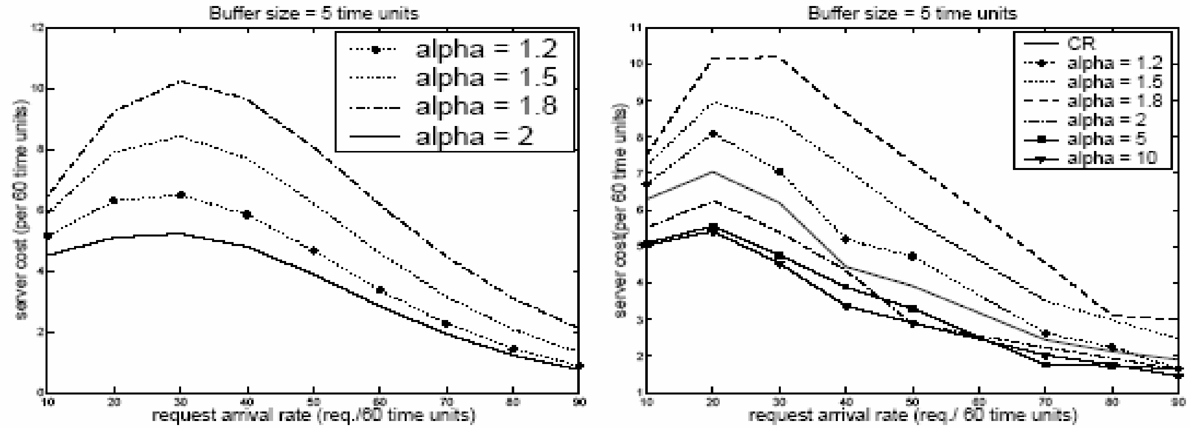


Fig. 8. (left) Analysis; (right) Simulation. Mean download time = 1000 time units, buffer size = 5 bytes

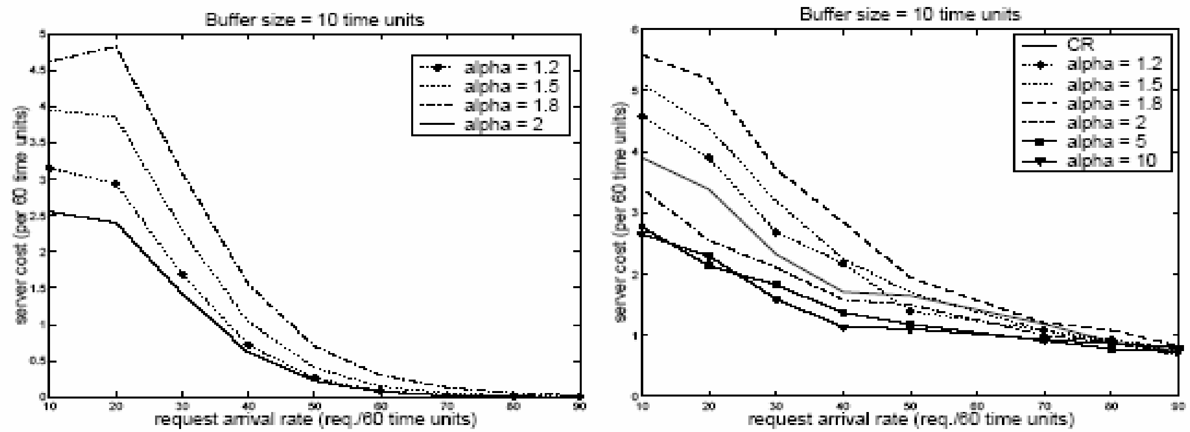


Fig. 9. (left) Analysis; (right) Simulation. Mean download time = 1000 time units, buffer size = 10 bytes

Performance Evaluation

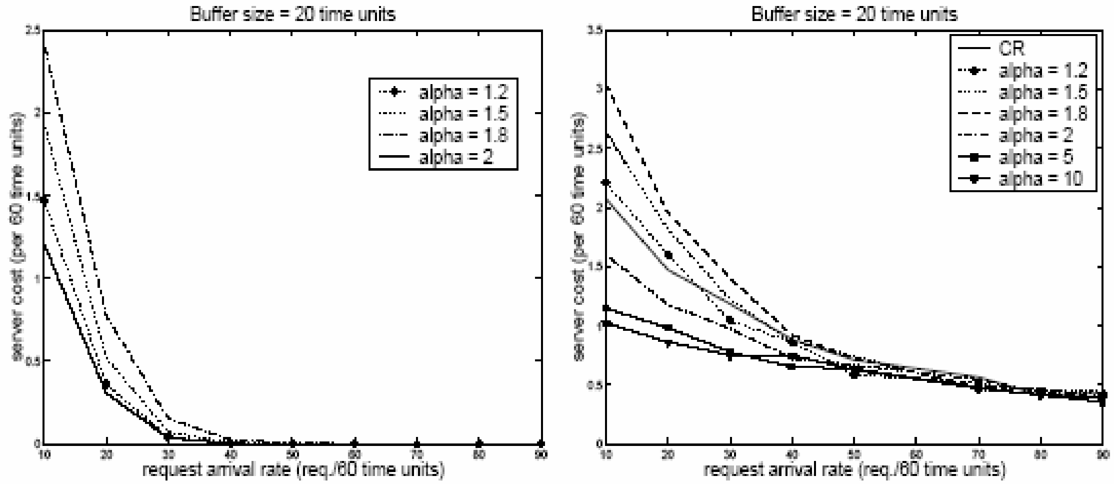


Fig. 10. (left) Analysis; (right) Simulation. Mean download time = 1000 time units, buffer size = 20 bytes

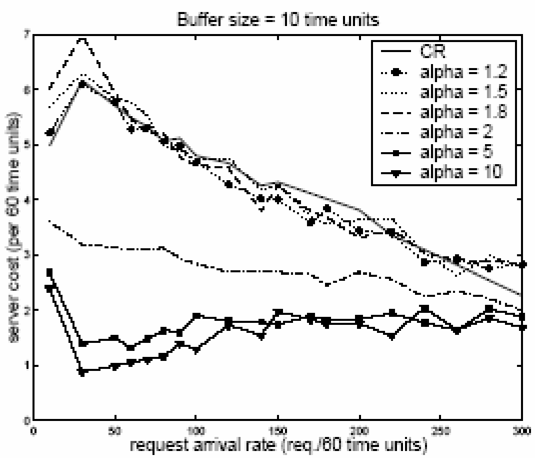


Fig. 11. Mean download time = 100 time units, buffer size = 10 bytes

Performance Evaluation

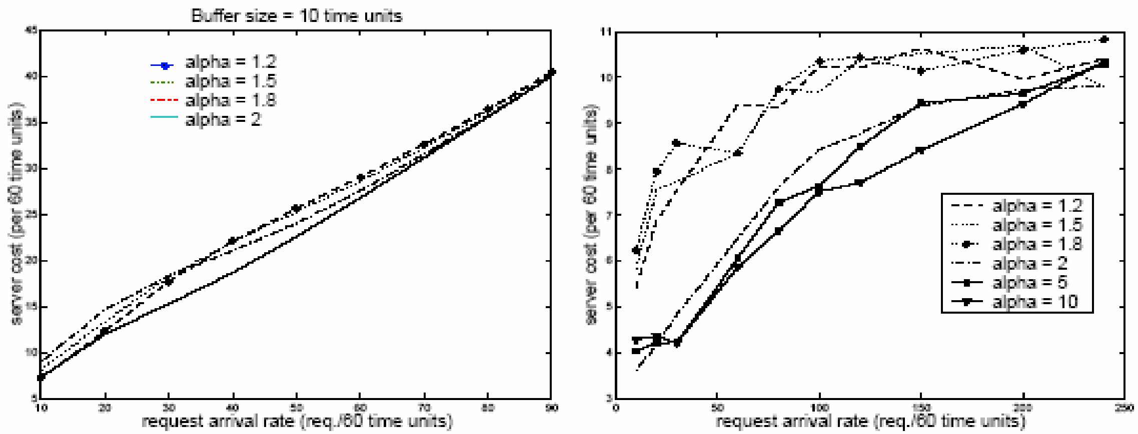


Fig. 12. (left) Analysis; (right) Simulation. With DELAY, Mean download time = 1000 time units, buffer size = 10 bytes, $\beta = 1$

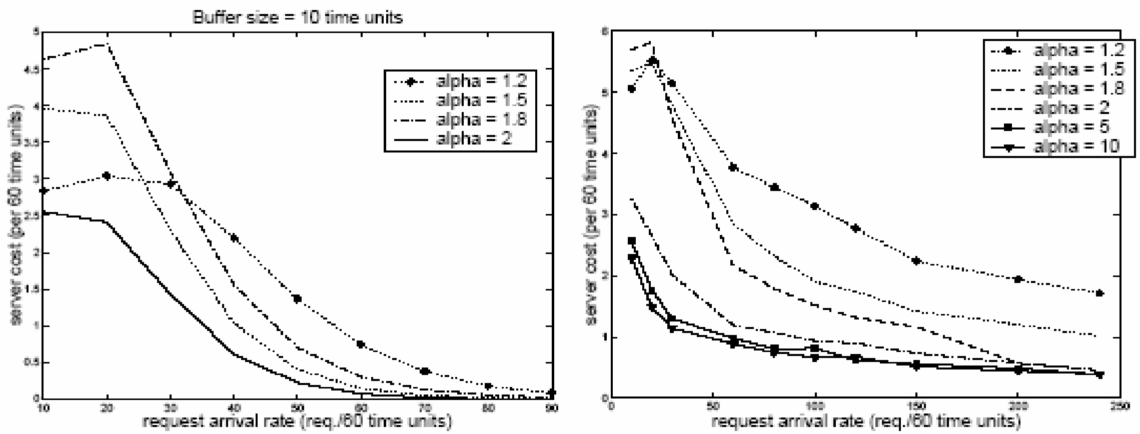


Fig. 13. (left) Analysis; (right) Simulation. With DELAY, Mean download time = 1000 time units, buffer size = 10 bytes, $\beta = 100,000$

Conclusions

- A more effective use of the local storage at P2P nodes must involve pre-fetching.
- This "look ahead" buffer capability provides each node with an opportunity to recover from the premature departures of its source.
- If download rate is sufficiently greater than the playback rate($\alpha > 2$), the distributed pre-fetching scheme significantly reduces the load on the server as it effectively increases the capacity of the P2P system.