# Papers to be presented today

- **Distributed Video Streaming over the Internet** T Nguyen and A. Zakhor
- **On Peer-to-Peer Media Streaming** Dongyan Xu, Mohamed Hefeeda, Susanne Hambrusch, Bharat Bhargava
- **Distributing Streaming Media Content Using Cooperative Networking** V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai

# A Broader picture

**These paper address the problem of media transport**

- **With a client-server approach**
- **With a peer-to-peer approach**
- **With a hybrid approach**

**These papers have new and interesting algorithms**

# Distributed Video Streaming over the Internet

- Problem
- Assumptions
- Solution
- Rate allocation algorithm
- Packet partition algorithm
- Overall mechanism
- Evaluation
- Results
- Critique

# Problem

Design a framework for transport protocol of video streaming with following features

- receiver simultaneously receives video stream from multiple senders
- the framework should be TCP friendly
- loss should be minimum

# Assumptions

- Available bandwidth from all senders is more than that required by the receiver
- Routes from a client to senders do not share a common congestion link
- Packet loss and delay due to congestion are bottleneck instead of physical bandwidth at the last hop

# Solution

- Propose a TCP friendly framework
- A rate estimating algorithm being run at the receiver
- A distributed algorithm that runs on each sender to partition packets
- Optimal with respect to loss

# Rate Allocation Algorithm

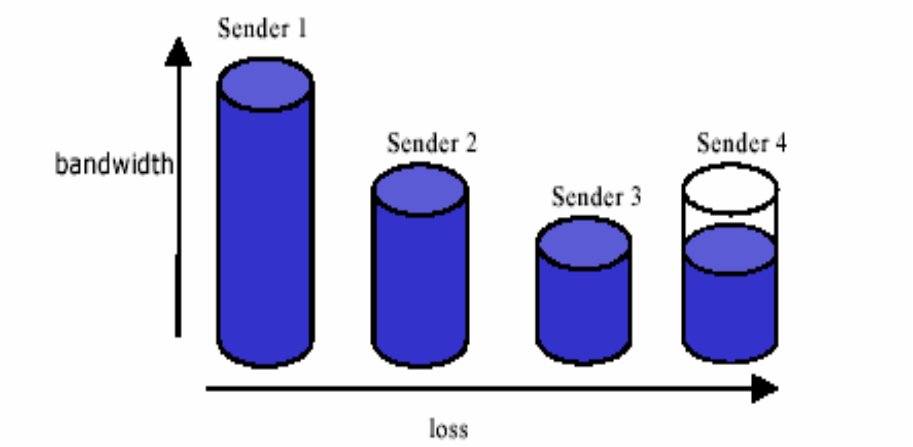
- Receiver estimates bandwidth, *B(i,t)* for each sender using a TFRC
- Receiver computes the optimal sending rate, *S(i,t)* for each sender with minimum loss rate as the criterion

$$F(t) = \sum_{i=1}^{N} L(i,t)S(i,t)$$

$$\text{subject to} \quad \begin{cases} 0 \le S(i,t) \le B(i,t) \\ \sum_{i=1}^{N} S(i,t) = S_{req}(t) \end{cases}$$

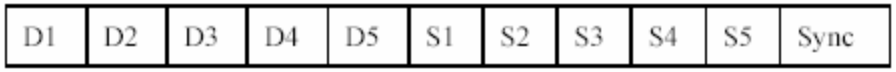
# Rate Allocation Algorithm

- Senders sorted (increasingly) based on loss rate
- Start with first sender and assign it's rate
- Continue assigning rate to successive senders until the sum of available bandwidth exceeds reqd rate



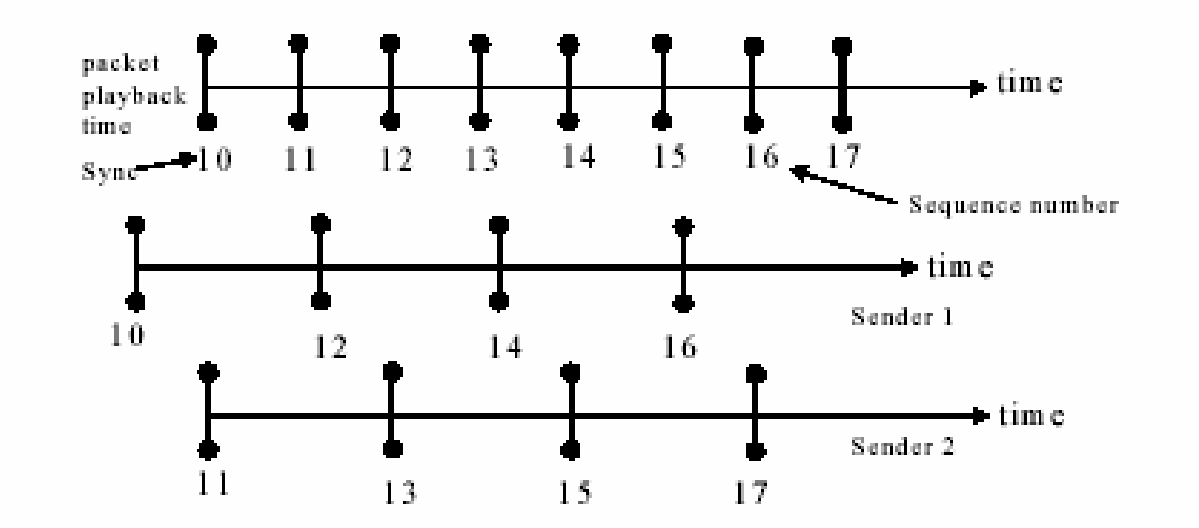
# Packet Allocation Algorithm

- Receiver sends control packet to all the senders




- All senders simultaneously run the algorithm
- Estimated difference between the arrival time and the playback time is considered
- Each sender decides which packet it should send
- Each sender also estimates the packets which would be send by other senders

# Packet Allocation Algorithm




- Top one is receiver and bottom two are senders
- Both senders get the control packet
- Sender 1 decides to start sending packets from 10
- Sender 2 decides to start sending packets from 11

# Algorithm

How often would a receiver estimate the sender's bandwidth

- Periodically sample B(i,t)
- If B(i,t) > S(i,t) +w, increment counter by one
- If B(i,t) < S(i,t) - w, decrement counter by one
- If counter > Gamma, update S(i,t)

What Value to take for the synchronization sequence Number

- minimum of estimates of latest packets, k"(j) sent by all senders
- K"(j) could be k(j) + 2 S*D(j)
- This slows down all the senders, so that those who lag can catch up

# Overall Mechanism

- Receiver estimates bandwidth for selected servers
- Receiver runs rate allocation algorithm
- Receiver sends control packet to each one of them
- Senders run packet allocation algorithm and start sending packets
- Periodically receiver updates bandwidth and sending rate

# Evaluation

**NS based simulation**

- Scenario1:Both senders use TFRC
- Scenario 2: senders don't use TFRC

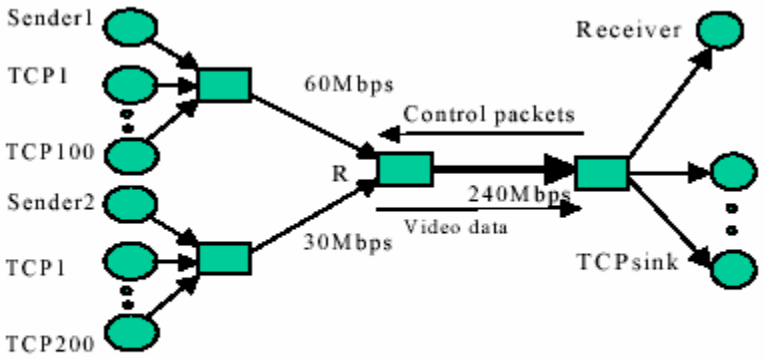



Figure 8: *Simulation configuration*

**Internet based experiment**

Senders at Indiana and Sweden send data to UC Berkeley

# Results

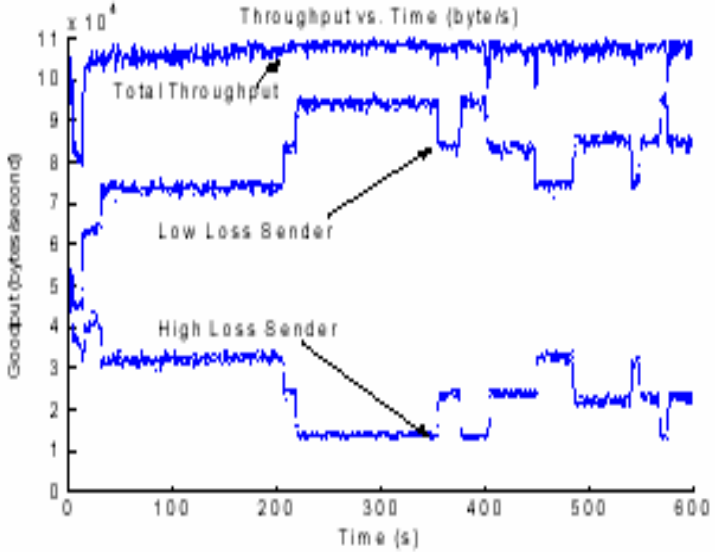



Figure 11: *Throughputs of two senders in adaptive scenario one*

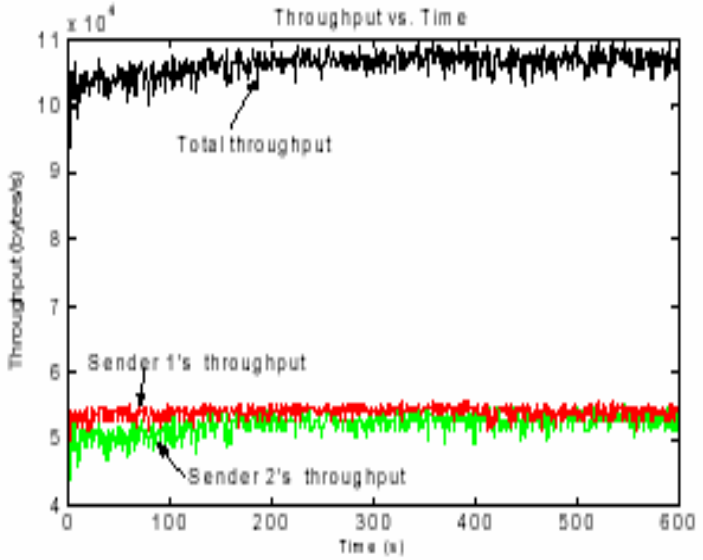



Figure 12: *Throughputs of two senders in non-adaptive scenario two*

# Results

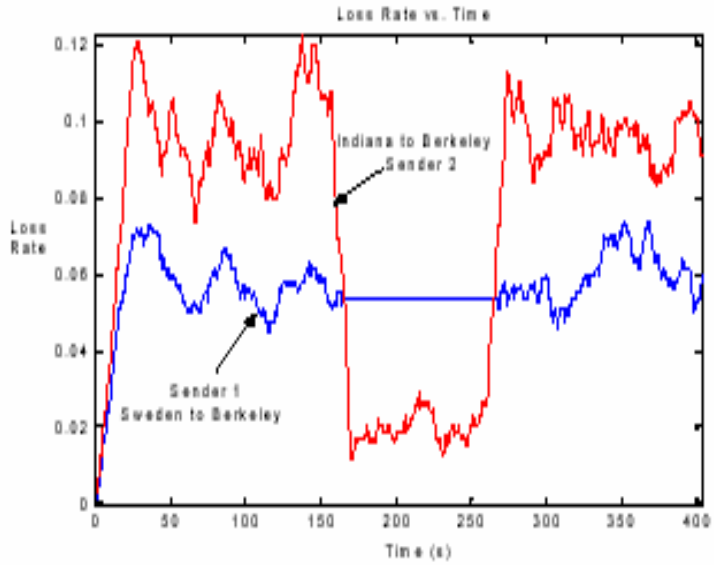



Figure 15: *Loss rate vs. time, streaming from Sweden and Indiana to Berkeley.*

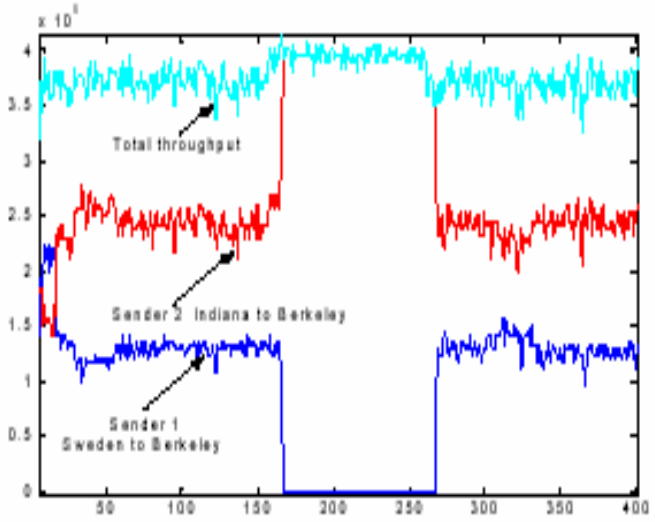



Figure 16: *Throughput vs. time for simultaneous streaming from Sweden and Indiana to Berkeley.*

# Critique

- When the estimated bandwidth goes beyond the band $S(i,t) \pm w$, the counter should be increased
- The scope of this solution is limited and is not suitable for live streaming
- Need to inculcate error control [they refer in future work]

# On Peer-to-Peer Media Streaming

- Problem
- Assumptions
- Solution
- Media distribution algorithm
- Admission control algorithm
- Overall mechanism
- Evaluation
- Results
- Critique

# Problem

**Designing a framework for media distribution on a p2p system**

- How to assign media data to multiple peers
- How to fast amplify the streaming capacity of the p2p system

# Assumptions

- The p2p system is self-growing
- p2p media streaming is *server-less*
- Peers are heterogeneous in their out-bound capacity
- Multiple senders can send data to one receiver

# Solution

Two very interesting algorithms addressing both the problems

- OTS_$_{p2p}$ for computing the optimal media data assignment
- DAC_$_{p2p}$ for differentiated admission control protocol

# p2p streaming model

- Requesting/Supplying peers
- Supplying peers have out-bound bandwidth $R\_{out}(P\_s)$ belonging to set $\{R_0/2, R_0/4, R_0/2^3, .., R_0/2^N\}$
- Peer with capacity $R_0/2^N$ belongs to class N
- Media distribution consists of small sequential segments of equal sizes [CBR]
- Capacity of the system is

$$C_{sys}(t) = \lfloor \frac{\sum_{P_a \in P_a(t)} (R_{out}(P_a))}{R_0} \rfloor$$

# Media distribution algorithm, $OTS_{p2p}$

- Run by requesting peer
- After computing media distribution, it notifies each sender
- Algorithm
  - Suppose, *m* supplying peers arranged in descending order of their $R_{out}$
  - Compute assignment of first $2^n$ packets, where n=m-1
  - Repeat itself every $2^n$ segments

# Media distribution algorithm, $OTS_{p2p}$

**Theorem 1** *Given a set of $m$ supplying peers $\{P_s^1, P_s^2 ... P_s^m\}$ and a requesting peer $P_r$, if we have $R_0 = R_{in}(P_r) = \sum_{i=1}^{m} R_{out}(P_s^i)$, then Algorithm $OTS_{p2p}$ will compute an optimal media data assignment, which achieves the minimum buffering delay in the consequent peer-to-peer streaming session. The minimum buffering delay is $T_{buf}^{min} = m * \delta t$.*

*where delta t is playback time*

- A typical assignment is, for m=4
- n would be 3 with **[$R_0/2^1$,$R_0/2^2$ ,$R_0/2^3$ ,$R_0/2^3$]**
  - $P_s^1$ 7, 3, 1, 0
  - $P_s^2$ 6, 2
  - $P_s^3$ 5
  - $P_s^4$ 4

# Admission control algorithm, DAC_p2p : The Need

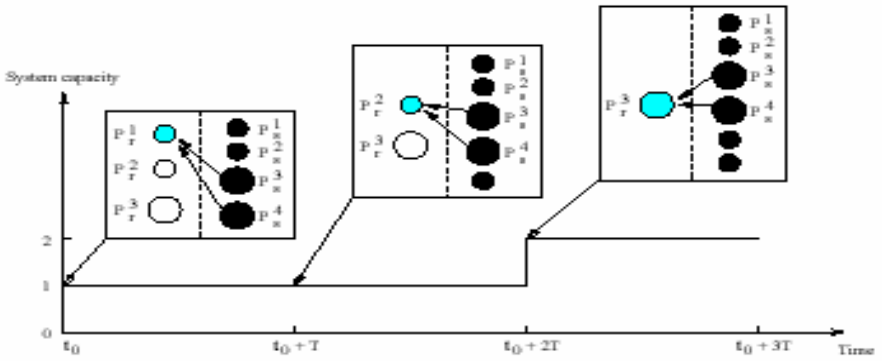
**p2p systems grow dynamically**

**System consists of**

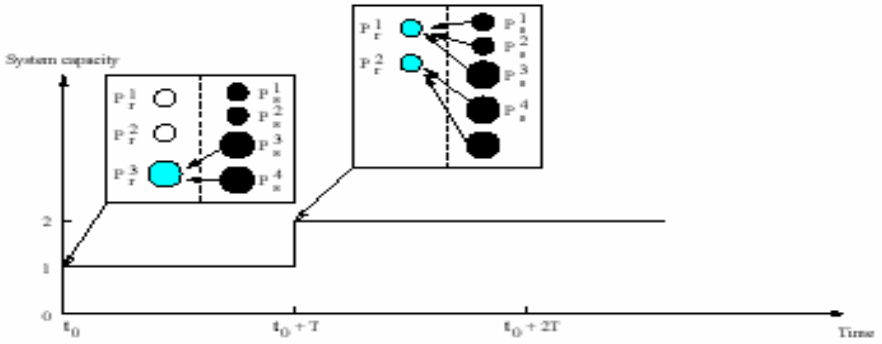
- ❖2 class 1 suppliers,
- ❖2 class 2 suppliers,
- ❖2 class 2 receivers and
- ❖1 class 1 receiver

**Capacity is lower integer of (1/2+1/2+1/4+1/4) =1**

**How to amplify system capacity with least delay**




(a) Admitting $P_r^1 \rightarrow P_r^2 \rightarrow P_r^3$




(b) Admitting $P_r^3 \rightarrow (P_r^1 + P_r^2)$

# Desirable features of DAC_p2p

- **Favor higher class of requesting peers**
  - ***-decisions are probabilistic***
- **The lower class of requesting peers should not be starved**
  - ***-a system of reminder***
- **Should enforce in a distributed fashion**
- **Higher bandwidth pledged by the requesting peer, larger chances of it's admission [incentive based scheme]**

# Admission control algorithm, DAC_$_{p2p}$

- Each supplying peer, Ps maintains a admission probability vector for each requesting peer

  <Pr[1], Pr[2], Pr[3] … Pr[1]>

- When Ps becomes a supplying peer, it initializes
  - Pr[i]=1.00 for 1≤ i ≤ k,
  - Pr[i]=$1/2^{i-k}$ for k≤ i ≤ N

  where k is the class of Ps
- If Ps was idle, it will elevate the lower probabilities i.e.

  Pr[i]=2*Pr[i] for k≤ i ≤ N

# Admission control algorithm, $DAC\_{p2p}$

**If Ps has just finished serving a session**

- **If it didn't server anyone from the favored class it elevates the lower probabilities**

  $$Pr[i]=2*Pr[i] \text{ for } k \leq i \leq N$$

- **if received request when Ps was busy and if it keeps a reminder, then if k' is the maximum class of among the reminder set, then it elevates**
  - $Pr[i]=1.0$ for $1 \leq i \leq k'$
  - $Pr[i]=1/2^{i-k'}$ for $k' \leq i \leq N$

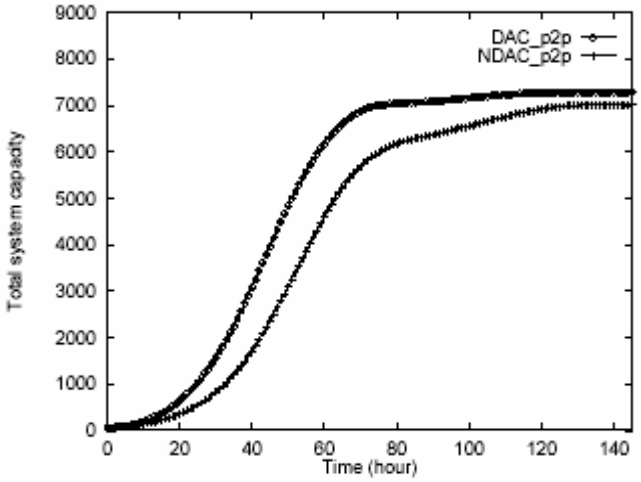
# Overall Mechanism

- Requesting peer, Pr does a lookup and finds M of the supplying peers
- If the request is granted, then Pr runs media distribution algorithm
- Else it backs off
- Pr converts to supplying peer
- Maintains/Updates a probability vector for each requesting peers
- Applies $DAC_{\_p2p}$ for processing future requests

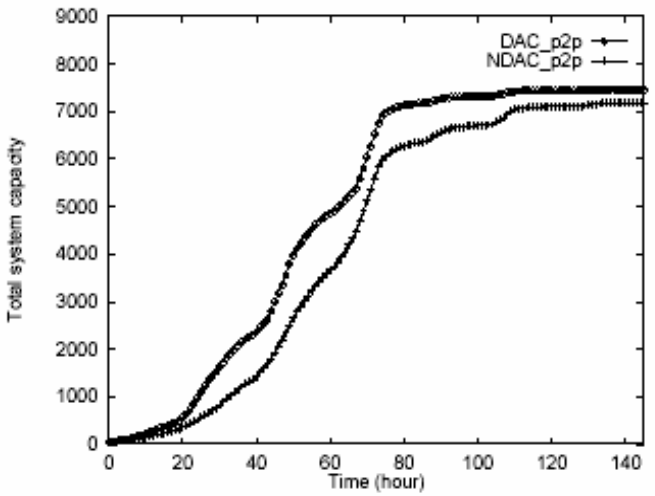
# Evaluation

- p2p system consists of 50K nodes coupled with another 100 “seed” nodes
- 50K requesting nodes belong to classes 1,2,3 and 4
- User 4 patterns
    1. Constant arrivals
    2. Gradually increasing, then gradually decreasing arrival
    3. Bursty arrivals followed by lower and constant arrivals
    4. Periodic bursty arrivals with low and constant arrivals between bursts
- Compare DAC with NDAC

# Results



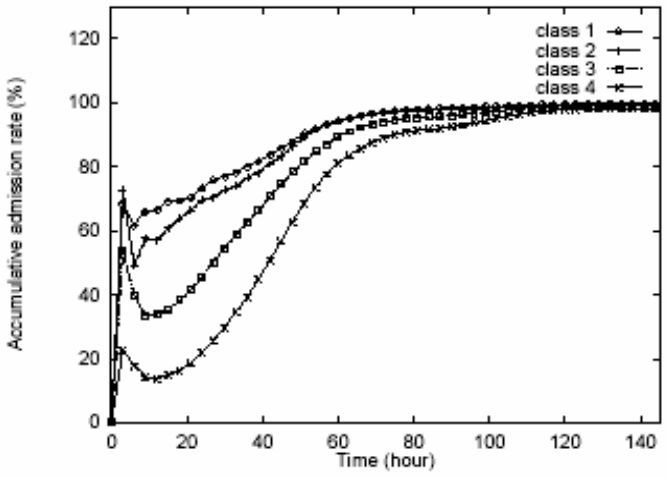

(a) Arrival Pattern 2



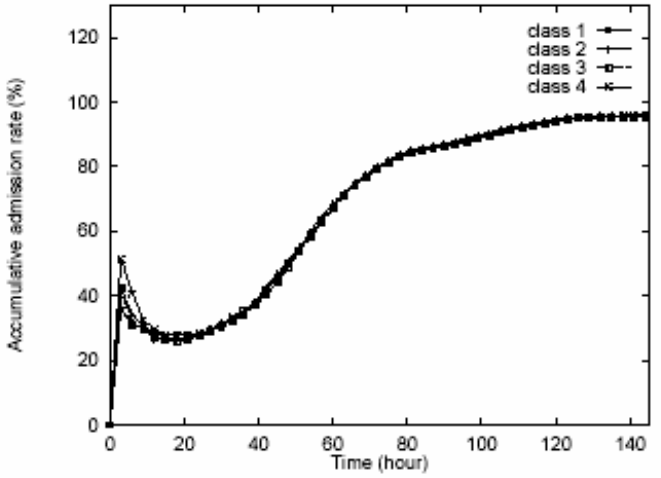

(b) Arrival Pattern 4

**Figure 4. System capacity amplification using $DAC_{p2p}$ and $NDAC_{p2p}$**

# Results



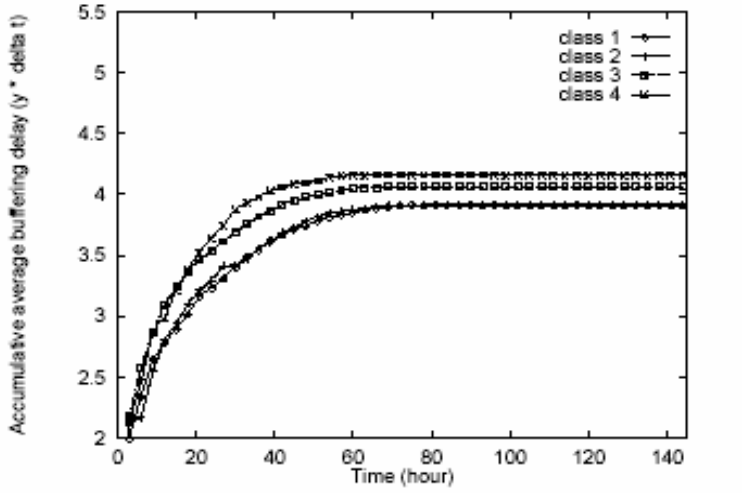

(a) Arrival Pattern 2, using $DAC_{p2p}$



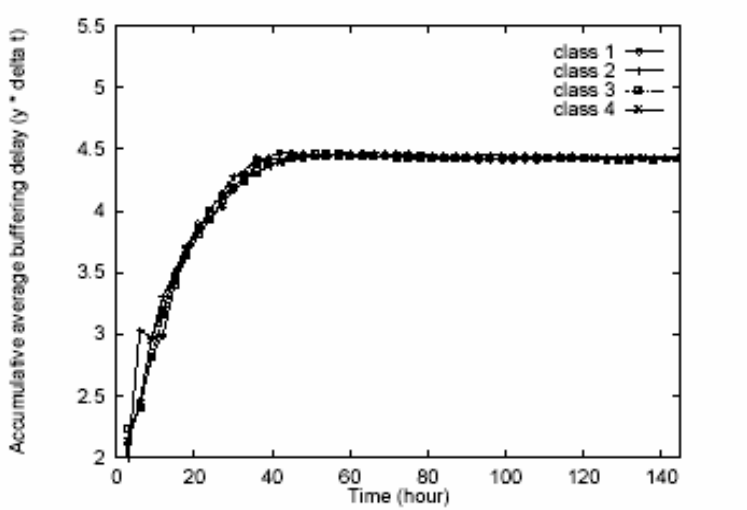

(b) Arrival Pattern 2, using $NDAC_{p2p}$

**Figure 5. Per-class accumulative request admission rate**

# Results



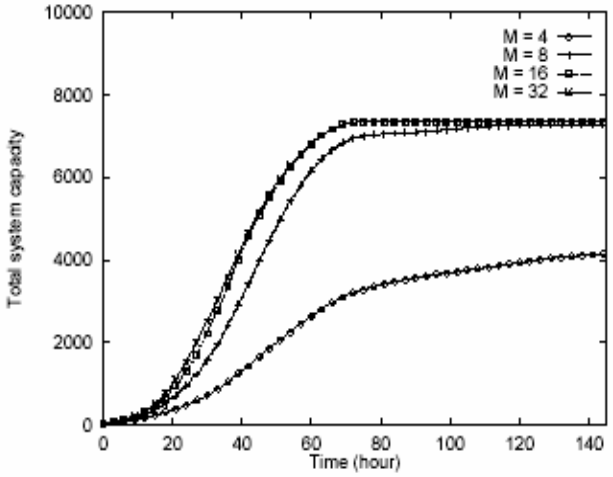

(a) Arrival Pattern 2, using $DAC_{p2p}$



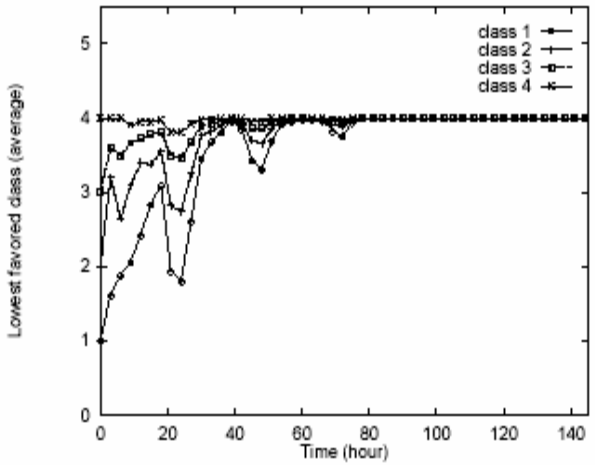

(b) Arrival Pattern 2, using $NDAC_{p2p}$

**Figure 6. Per-class accumulative average buffering delay (the actual delay is $y * \delta t$)**

# Results




(a) Impact of $M$




(a) Arrival pattern 4

**Impact of M on system**

**Lowest class of requesting peers favored by each class of supplying peers (averaged every 3 hours)**

# Critique

- Definition of Capacity
- Exponential back off
- Loss rate of links is not taken into account
- Error control mechanism

# Distributing Streaming Media Content Using Cooperative Networking

- Problem
- Assumptions
- Solution
- Multiple Description Coding
- Tree Management
- CoopNet: Live Streaming
- CoopNet: On-demand Streaming
- Feasibility Analysis
- Results Live Streaming
- Results On-demand Streaming
- Critique

# Problem

- Distribution of media content in a scalable way
- There are short periods in which the server could get overwhelmed by requests
- The mode could be live or on-demand

# Solution

- Cooperative Networking (CoopNet) is proposed
- Clients cooperate to distribute the content, when the server gets overloaded
- Tightly coupled with encoding
- CoopNet complements client-server approach
- CoopNet uses p2p lookup services
- CoopNet complements existing application layer multicast

# Assumptions

- Encoding is done as Multiple Description Coding (MDC)
- The frequent nature of node departure/arrival is the important bottleneck

# Multiple Description Coding

- The audio/video signal is encoded into M>1 separate streams (or descriptions)
- Any subset can be decoded into a signal with distortion
- The more the descriptions, m the lower the distortion, D(R) [higher quality]
- Expected distortion is

$$\sum^{M}_{m=0} p(m)D(R_m)$$

# Tree Management

- The server creates M tree (rooted at itself) where M is number of description
- Server has full knowledge of the topology of all distribution trees
- Each node attaches itself to a parent node from each tree
- When packet loss reaches a threshold, the child confirms it with parent

# CoopNet: Live Streaming

- The server creates M tree (rooted at itself) where M is number of description
- Server identifies nodes which are “nearer”
- Starting at the server, the client comes down the tree, till it gets parent nodes which have spare bandwidth
- Server responds with a list of designated parents

# CoopNet: On-demand Streaming

- The server creates M tree (rooted at itself) where M is number of descriptions
- For every URL, server maintains a fixed list of IP addresses of clients
- A new client gets a subset of IP addresses
- The new client selects from among those clients [such that it connects each of M trees]
- A peer from the list might have only a segment of the content
- The requesting peer does handshake to know this
- The requesting peer uses greedy algorithm

# Feasibility Analysis

For M=16 and each tree has 4 children on an average

- Memory: 10 MB
- Network Bandwidth: 8Mbps
- CPU: with 40ns memory cycle, 390 memory accesses per insertino

# Evaluation – Live Streaming

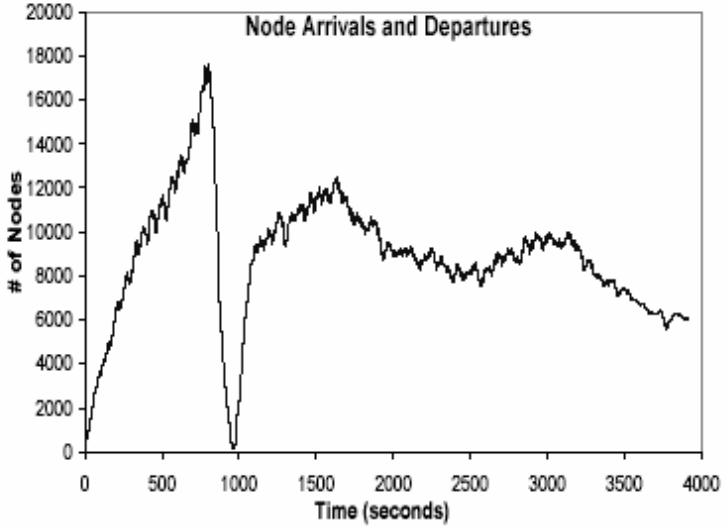



Figure 6: Number of clients and departures.

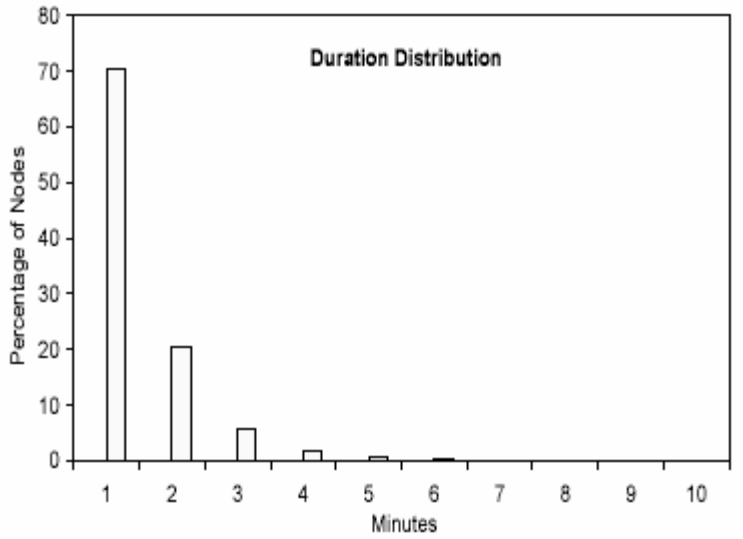



Figure 7: Duration distribution.

**MSNBC trace of 9/11**

# Evaluation – Live Streaming

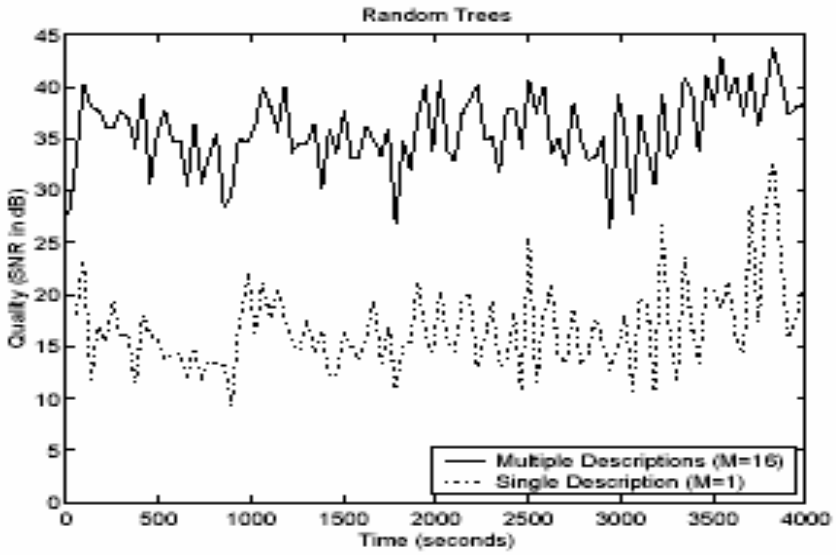



Figure 9: Random Tree Experiment: The SNR over time for the MDC and SDC cases. At each time instant, we compute the average SNR over all clients.

**MSNBC trace of 9/11**

# Evaluation – Live Streaming

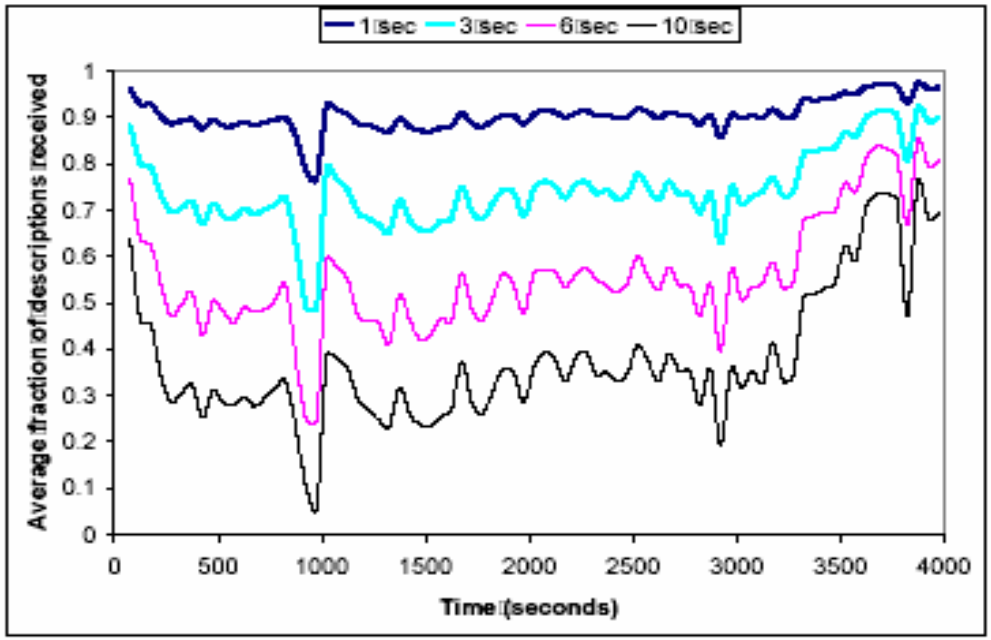
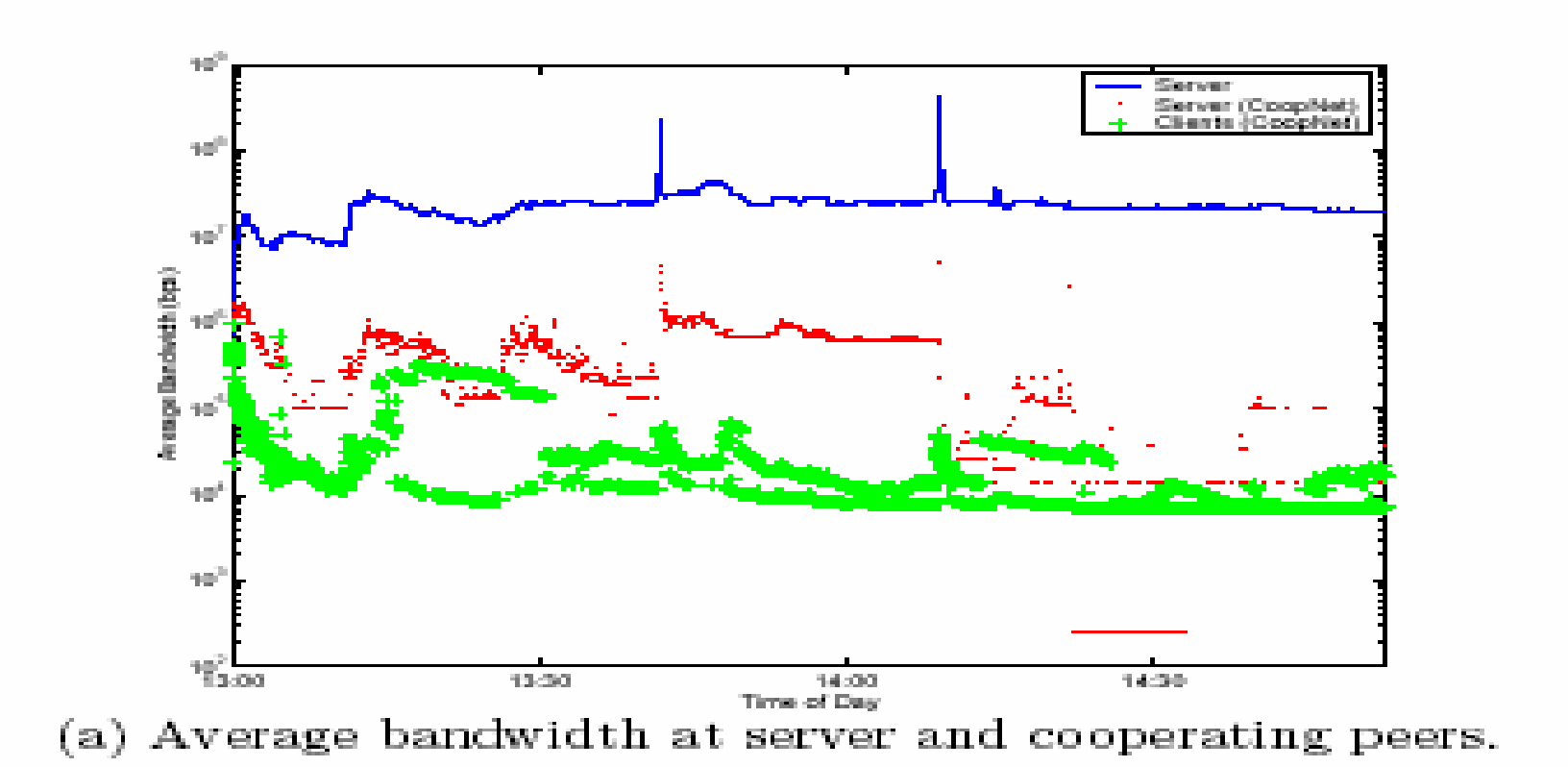



Figure 12: The average fraction of descriptions received for various repair times.

**MSNBC trace of 9/11**

# Evaluation – On demand Streaming




(a) Average bandwidth at server and cooperating peers.

**MSNBC trace of 9/11**

# Critique

- When would the server decide to switch to CoopNet needs more explanation
- Error handling mechanism
- Seems fairly limited to MDC only
- In feasibility analysis, there are only 64 nodes, which is hardly a significant load for server
- Presentation definitely could have been better