NSYNC

Network Synchronization for Low-latency Peer-to-Peer Streaming Overlay Construction

Shudong Jin

Case Western Reserve University

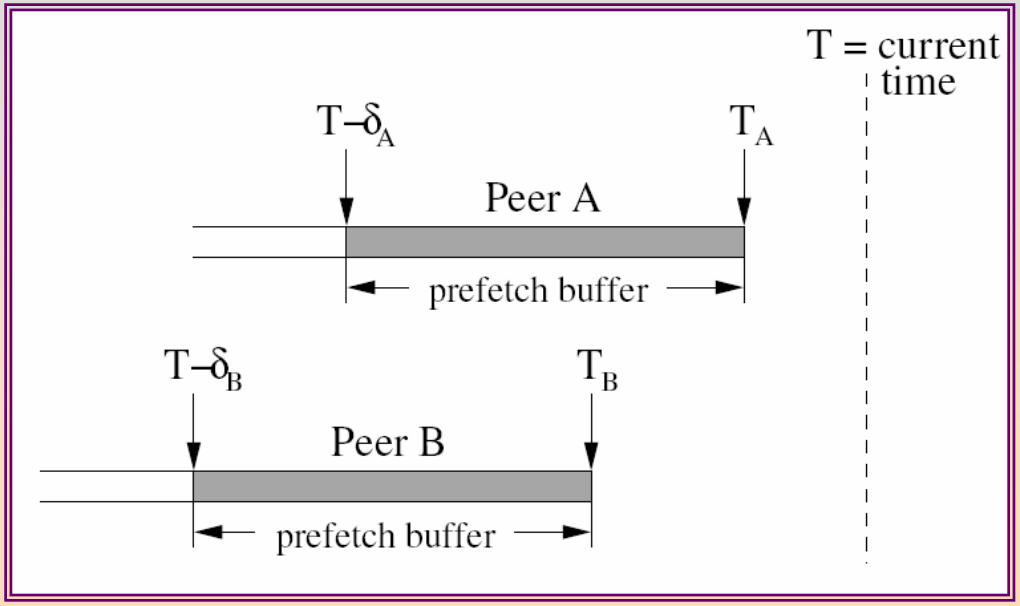
NEONet Workshop, March 1, 2006

Peer-to-Peer Streaming

- Peer-to-peer systems versus client/server systems
  - Eliminate the performance bottleneck problem
  - Eliminate the single point-of-failures
- Media streaming: a media stream is played while being received.
  - Video-on-demand (stored media such as movies)
  - Conference broadcast, live shows, sports events
  - Interactive games, real-time conferencing
- How to construct an overlay network for peer-to-peer streaming?
  - High-quality media streams
  - Low peer-perceived latency

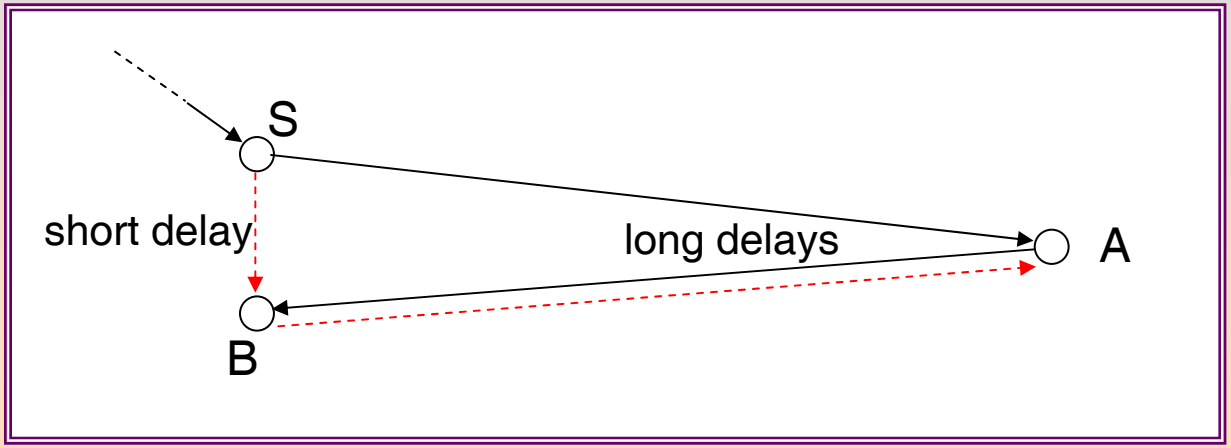
Problem One: Long Latency

- Peers need to buffer a portion of the media stream
  - Peer B receives media data from peer A.
  - A long latency may be found unnecessary later
- If this latency is not trimmed, the average latency of the system could be unbounded (early peers leave and new peers join)
  - Find peers with low-latency as parent, but …



Problem Two: Inefficient Overlay

- There is a strict partial ordering of the peers
  - It doesn't make sense for an earlier peer to receive media data from a later peer
- This partial ordering limits the space for overlay network construction (and potentially results in low quality)



NSYNC Objectives

- NSYNC: Network Synchronization for peer-to-peer streaming overlay construction
- NSYNC is
  - A set of primitives to help construct and optimize streaming overlay
  - A method that can be used in other peer-to-peer streaming systems
- NSYNC is not
  - A peer-to-peer system with its own overlay network construction

NSYNC Ideas

- The principles
  - To capitalize on the flexibilities of streaming applications
  - To exploit temporal locality and geographical proximity
- Flexibilities of streaming applications
  - Increase/decrease the speed of playing a media stream
- Temporal and geographical
  - Peers that are geographically close to each other can be temporally close too.
  - If not, we can make that happen!

NSYNC Assumptions

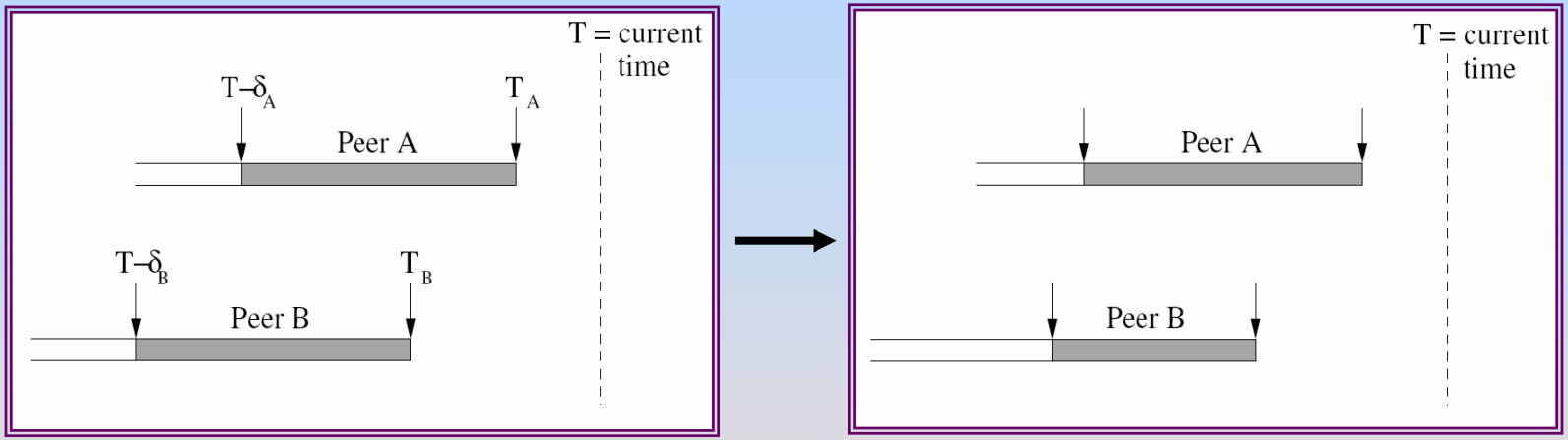
NSYNC Primitives

- Provided primitives?
  - increase the speed of playing a media ((1+Δ) times the normal rate)
  - decrease the speed of playing a media
  - increase the speed of receiving a media
  - decrease the speed of receiving a media?
- Combined in use with other operations
  - switch to another peer as a parent
  - reverse client/server roles

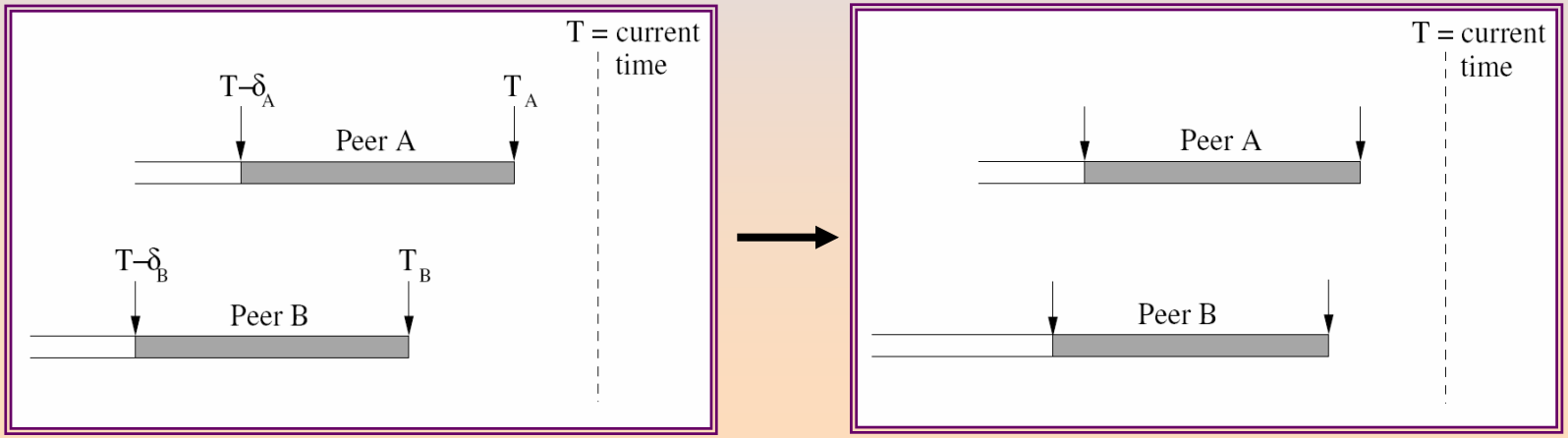
NSYNC for Peer Catching

- Scenarios
  - Peer B receives a media stream from peer A. Peer B finds the lag behind A is unnecessarily large. For example,
    - The path is less congested, and has few delay jitters
    - A's other three children left
  - Peer B's previous parent left and B locates A as the new parent who is well ahead of B.
- Let us use NSYNC to construct a *catching* process

Catching Process



With normal receiving rate

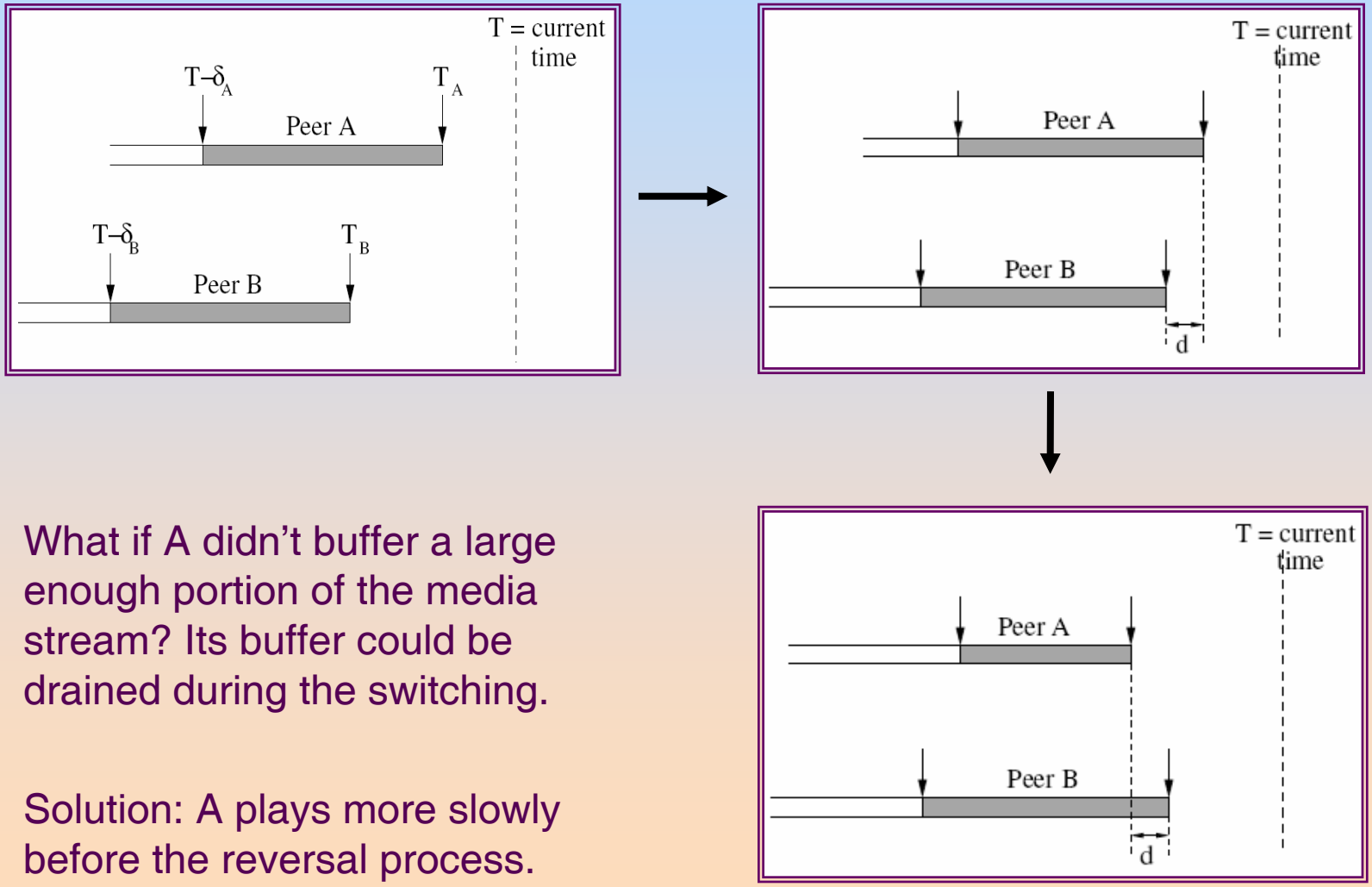


With (1+Δ) times normal receiving rate

NSYNC for Peer Reversal

- Scenarios:
  - Initially peer B receives a media stream from A, who receives the media stream from S. They find it is better for B to receive it from S and relay it to A.
- Let us use NSYNC to construct a *reversal* process for A and B

Reversal Process



What if A didn't buffer a large enough portion of the media stream? Its buffer could be drained during the switching.

Solution: A plays more slowly before the reversal process.

Open Questions: Implementation

- What primitives should be provided in NSYNC?
- What meaningful operations (and processes) can be constructed using the primitives?
- Who, when, and how to decide what processes should be started to improve the performance and optimize the network?

Open Questions: Algorithms

- Graph algorithms and scheduling? Examples:
  - In a directed acyclic graph where T(child)<T(parent), how can we schedule a sequence of caching processes to minimize the latency as quick as possible? Maybe easy.
  - How can we schedule a sequence of reversal processes to minimize the average latency of all peers? Assume one peer can be involved in one reversal process at one time. Could be very difficult.
- Even simple problems become difficult, considering the decentralized nature of the networks.

Open Questions: Deployment

- Which peer-to-peer systems really like to have NSYNC? Likely it depends on the type of streaming applications
- Are there any technical barriers for the deployment of NSYNC? For example, some media players may not support a Δ increase of the playing speed.
- Are there any non-technical barriers?