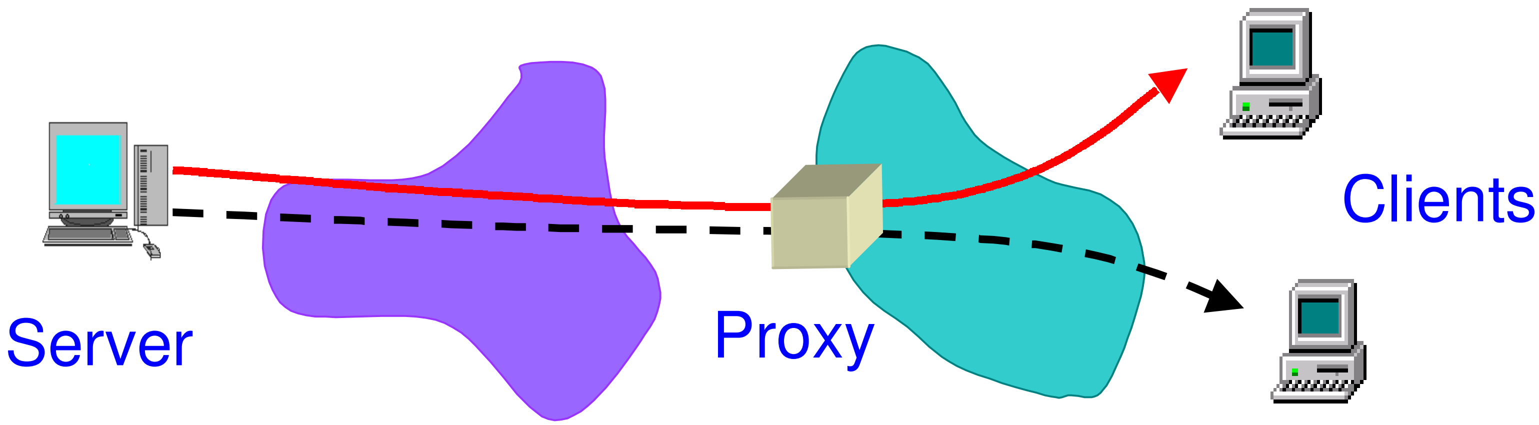
Design Considerations for an RTSP-Based Prefix Caching Proxy Service for Multimedia Streams

Stephane Gruber, Jennifer Rexford,
and Andrea Basso

AT&T Labs - Research



Outline

- Motivation for prefix caching
- Relationship to multimedia protocols
- Practical challenges at the proxy
- Prototype implementation
- Conclusions and future work

Multimedia Data vs. Traditional Web Content

- Multimedia applications
  - Web-initiated multimedia streaming
  - Microsoft, Real-Networks, QuickTime, …
  - growing popularity as access bandwidth increases
- Multimedia streams
  - large response size
  - high bandwidth
  - timing requirements
  - different set of protocols

Performance Problems

- Multiple round-trip delays
  - connection set-up, session description, control ops
- Start-up latency
  - accumulate initial frames at client to tolerate jitter
- Variable throughput on server-client path
  - absence of performance guarantees in IP networks
- Degraded audio/image quality
  - lost/late packets disrupt service or reduce quality

Proxy caching? Multimedia streams are very big...

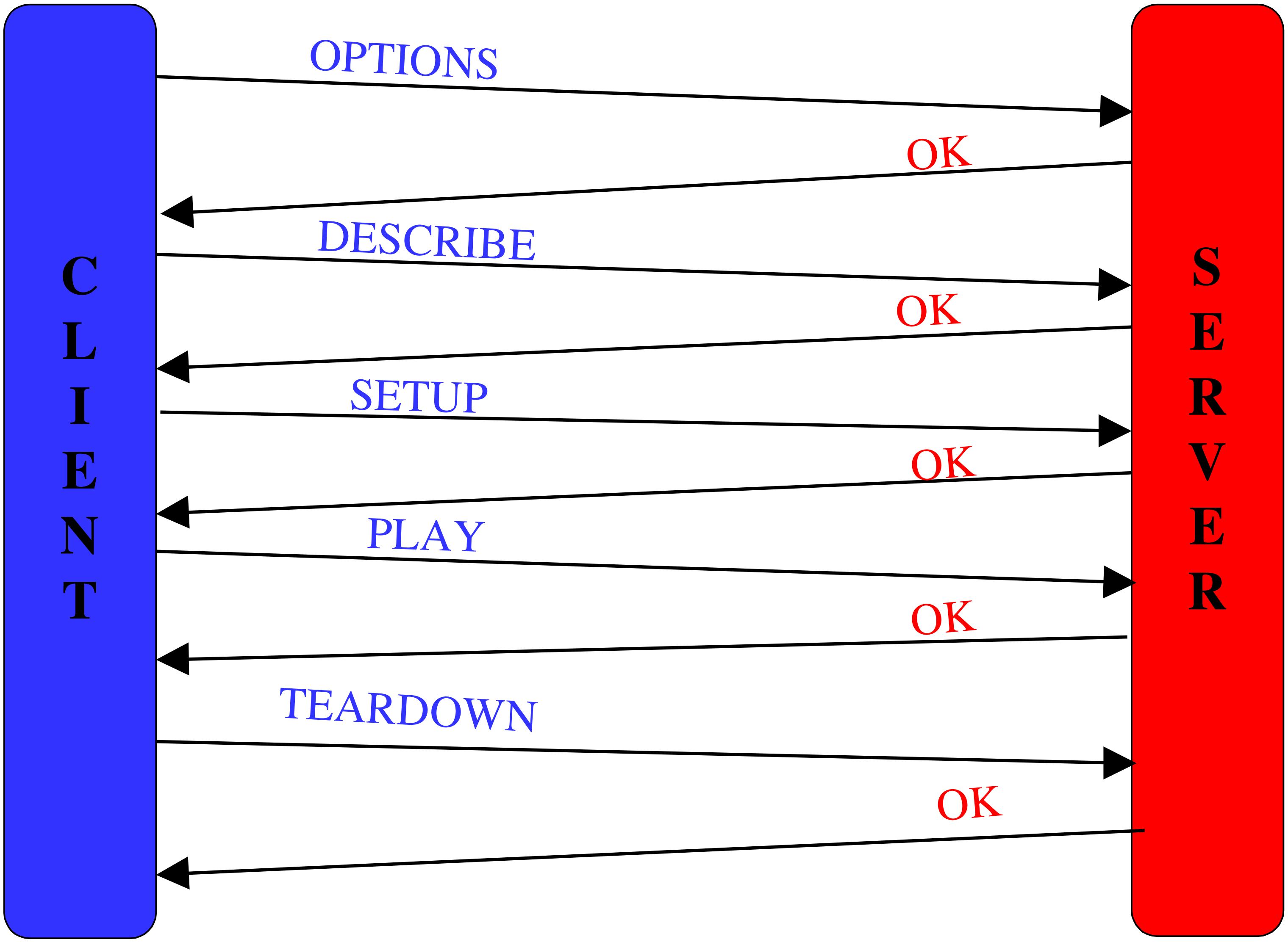
Prefix Caching

- Prefix caching
  - cache the initial portion (prefix) of the stream at proxy
  - transmit the prefix from the proxy to the client
  - fetch the suffix from the server to the proxy
- Advantages of prefix caching
  - hide start-up latency by sending data from the proxy
  - avoids the storage overhead of caching the whole stream
  - hides poor delay and throughput on server-proxy path
  - reduce server load and network traffic
  - can be realized in the context of existing protocols

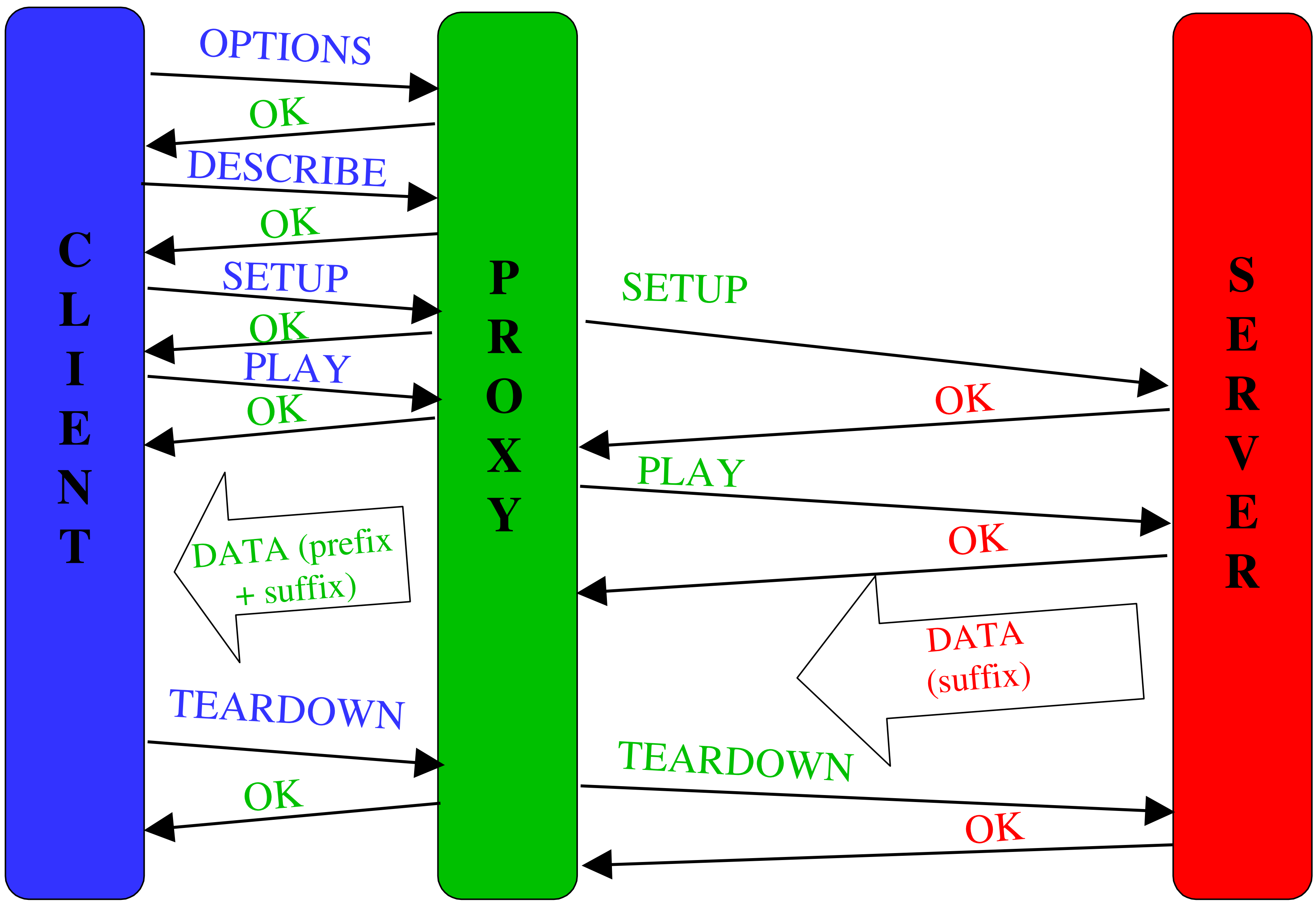
Protocols for Multimedia-on-Demand

- Real Time Protocol (RTP)
  - transport of multimedia stream with timing requirements
  - timestamp, seq number, source identifier, payload type
- Real Time Control Protocol (RTCP)
  - feedback on quality of data distribution (e.g., loss)
  - wall-clock time for synchronization across streams
- Real Time Streaming Protocol (RTSP)
  - coordinate delivery of multimedia stream from server
  - VCR functions (play, pause, rewind, fast-forward, jump)
  - request-response protocol similar to HTTP (but stateful!)

Typical RTSP Timeline



A “Hit” in the Proxy Cache



Requesting the Suffix

- RTSP *Range* request
  - request header for indexing to arbitrary point in stream
  - indexing based on time, rather than number of bytes
- Potential accuracy problems
  - server may not stream from *exactly* the requested point
  - server would need efficient indexing to *every* frame
  - RTSP does not bound the potential inaccuracy
- Practical solution
  - proxy requests playback from a slightly earlier point
  - response header indicates the range that was chosen
  - proxy removes any duplicate data, and fills gaps

Linking Prefix and Suffix

- Dual roles for the proxy
  - prefix: proxy acts as a server
  - suffix: proxy acts as an application-level router
- Potential RTP inconsistencies
  - source id, base timestamp, and base sequence number
  - chosen by proxy for prefix, and by server for suffix
  - RTP packets have inconsistent header fields
- Practical solution
  - server's base timestamp and seq # in PLAY response
  - proxy changes the RTP header fields in the suffix

Cache Coherency

- Proxy caching
  - RTSP response to OPTIONS and DESCRIBE
  - prefix of RTP data sent in response to PLAY
- Potential cache coherency problem
  - change in meta-information or audio/video content
  - cached prefix may be inconsistent with suffix
- Practical solution
  - infrequent modifications (optimize for common case)
  - send DESCRIBE/SETUP while sending prefix to client
  - do not send the suffix if the stream has changed (error!)

Proxy Implementation

- High-level architecture
  - User-space implementation in C on Linux
  - Single process, event-driven server
  - Round-robin scheduling to read from busy sockets
  - Transmission of RTP messages based on timestamps
- Multimedia protocols
  - Mapping of server-proxy and proxy-client connections
  - Caching of prefix as an ordered list of RTP packets
  - Assigning of RTP header fields without extra data copy
  - Mandatory RTSP methods; caching of meta-information
  - RTSP over TCP, RTP over UDP, and RTCP over UDP

Multimedia Client and Server

- Evaluation testbed
  - RealServer G2 (trial version) on Windows NT 4.0 PC
  - RealPlayer G2 on Windows 95 PC
  - Point-to-point, private Ethernet segments
- Practical challenges
  - Server defaults to using proprietary *RDT* instead of RTP
    - » proxy modifies SETUP request to force use of RTP
  - *Encrypted* headers in RTSP messages
    - » proxy forwards requests to server to learn encrypted headers
  - Range header *missing* from server response to PLAY
    - » proxy assumes that the Range request was satisfied accurately

Modifying Client SETUP Request

- First request for the stream
  - Proxy sends SETUP message omitting the non-RTP options
  - Server streams data in RTP format (except for .ra/.rm files)
- Future requests for the stream
  - Proxy selects RTP for transport and streams the prefix
  - Proxy sends SETUP message for suffix omitting non-RTP options

```
SETUP rtsp://135.207.15.90:554/g2video.rm/streamid=0 RTSP/1.0
CSeq: 3
RealChallenge2: 7c7687849ca62b5a063fd037f9d13d7a01d0a8e3,
        sd=78920df3
Transport: x-real-rdt/udp;client_port=6970;mode=play,
        x-pn-tng/udp;client_port=6970;mode=play,
        rtp/avp;unicast;client_port=6970-6971;mode=play
```

Conclusions

- Prefix caching improves performance
  - hides network and protocol delays; reduces network load
- Uses standard multimedia protocols
  - RTSP range request and mapping of RTP header fields
- Prototype implementation
  - Linux-based, event-driven caching proxy
- Experimentation with RealNetworks client/server
  - circumvent use of proprietary streaming protocol
- Testing of prototype implementation
  - successful streaming of partially-cached content

Future Work

- RTCP feedback on quality of data distribution
  - proxy as router: blindly forward RTCP reports
  - proxy as server: generate and receive RTCP reports
  - proxy as both: delayed feedback to the server?????
- Separation of server-proxy and proxy-client paths
  - exploit prefix cache to decouple the two control loops
  - server-proxy: retransmit lost packets; use TCP; etc.
    - » prefix hides transient performance problems from client
  - proxy-client: transcoding, smoothing, patching, …
    - » proxy services to improve performance and/or efficiency