Approaches for a Web-based Initiation of Quality-based Communication

Mark Doll

Institute of Telematics
Prof. Dr. Martina Zitterbart
University of Karlsruhe
Germany

1. Introduction
2. QoS-aware Applications
3. Legacy Applications
4. Conclusion

Introduction

Objectives

- Simple QoS support for legacy (non QoS-aware) applications
- User should be able to influence QoS selection

Assumptions about QoS signaling

- Supports end-to-end reservations
- On a per-flow basis
- Sender-initiated
- Feasible: scalable, low setup latency

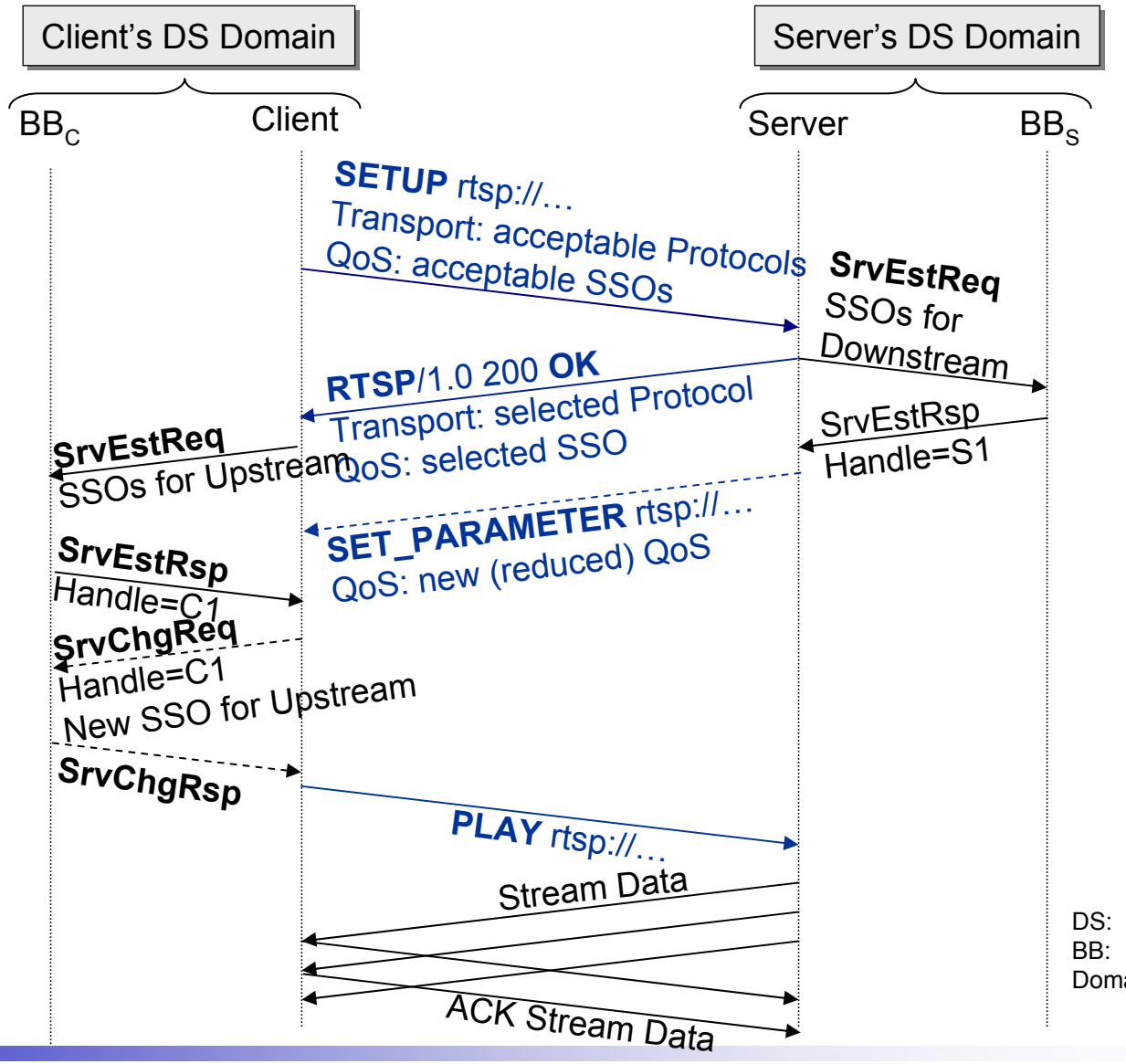
Focus

- Derive new requirements for signaling
- Determine limitations (keep signaling simple)

Based on own QoS signaling

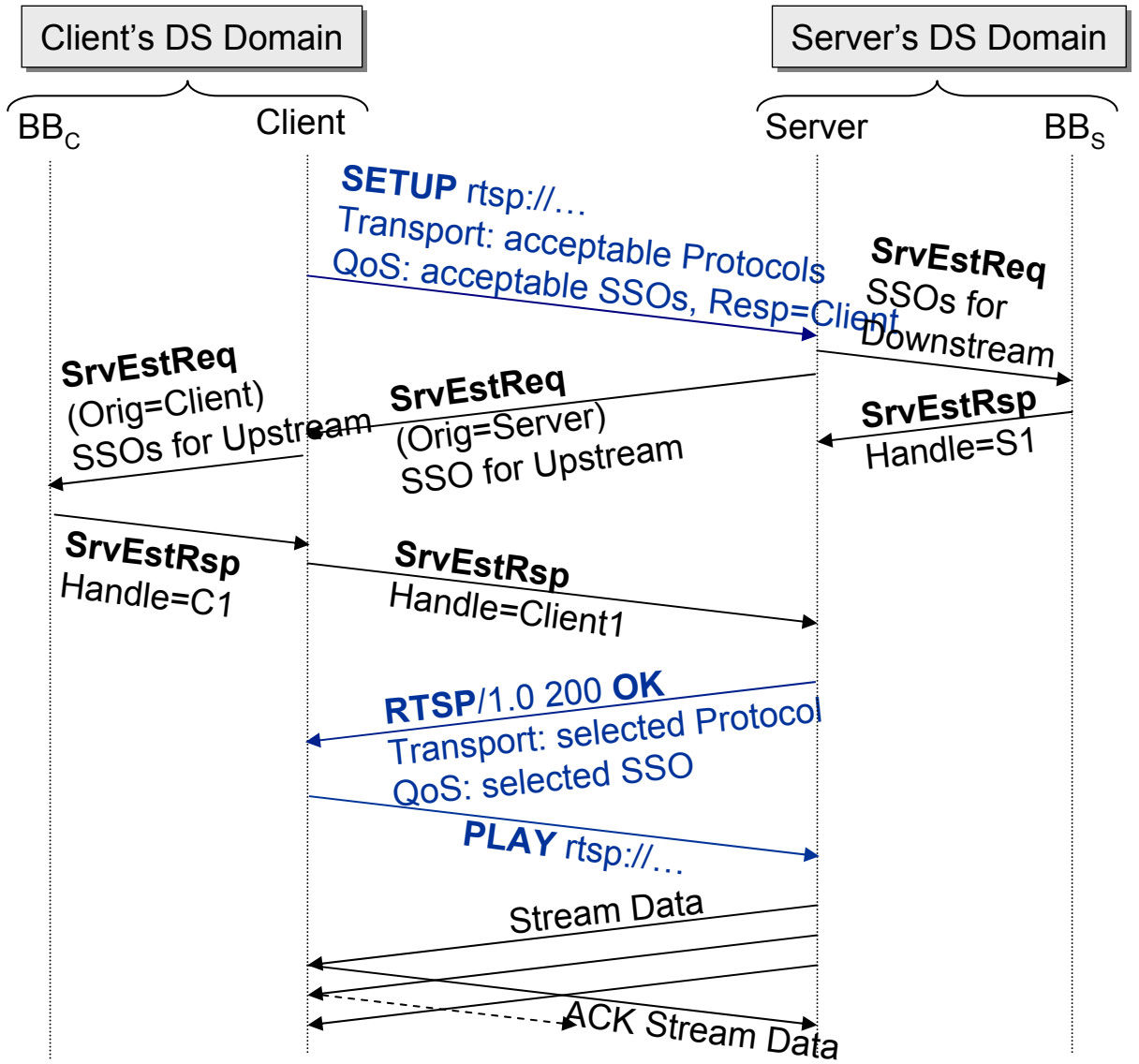
- **DMSP** – Domain Manager Signaling Protocol

All QoS-aware – Reverse Path Signaling by Application



DS: Diffserv
BB: Bandwidth Broker
Domain: Diffserv Domain (Autonomous System)

All QoS-aware – Reverse Path Signaling via DMSP



Legacy Client

Client not QoS-aware

- Upstream reservation (for feedback) must be initiated by Server

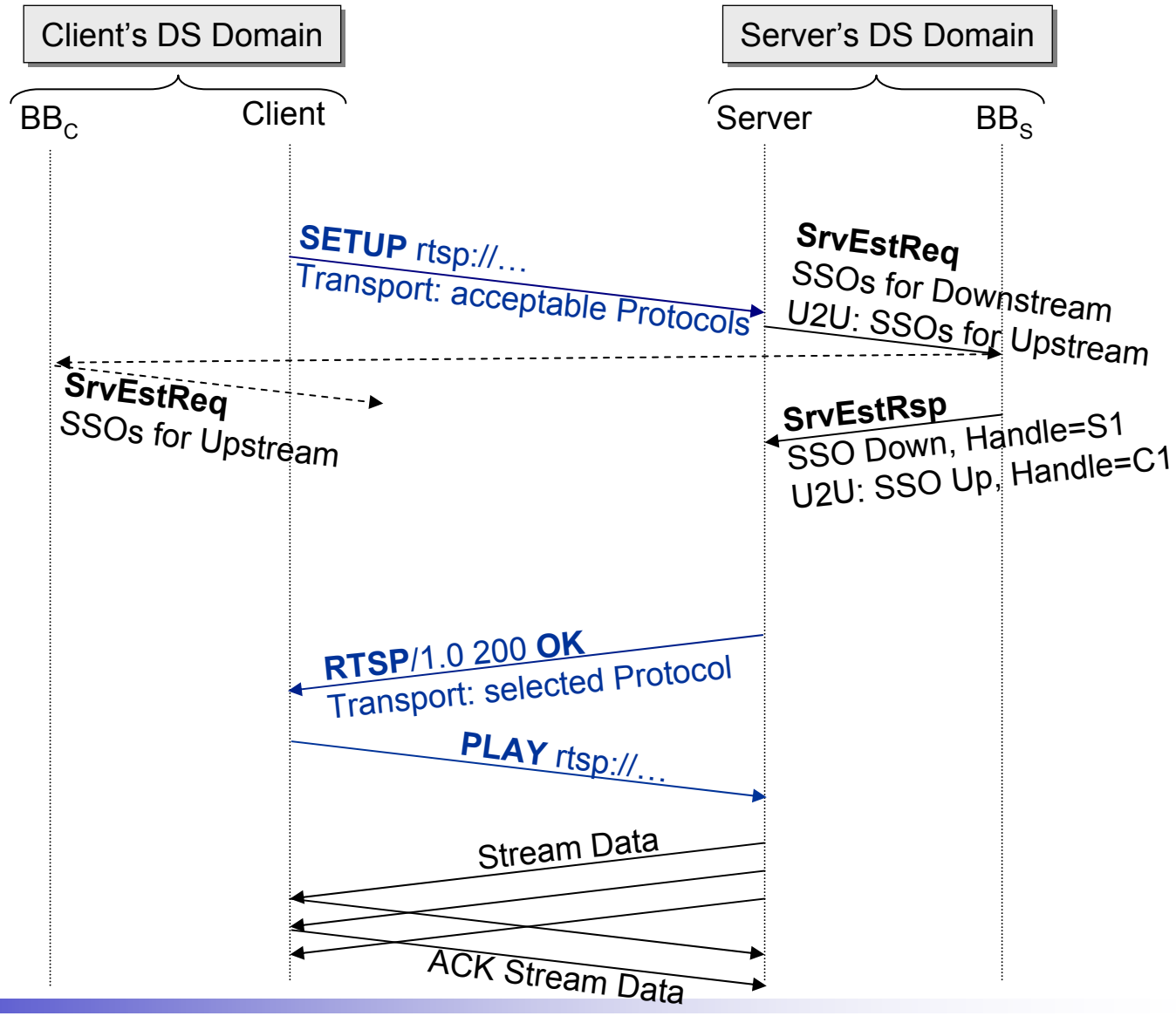
Establish reservation for reverse direction

- Domain-hop by domain-hop
  - Probably wrong path due to asymmetric routes
- End-to-end reservation for reverse direction
  - Trigger some node in receiver domain

Find/contact appropriate originator (aka BB of client domain)

- Well-known anycast address (like subnet-router anycast address)
- New DNS resource record
  - Need authorization
  - Exposed to DoS attacks
  - Might not pass firewalls
  - In case of DNS additional messages/delay for resolution process
- DMSP as transport layer (use DMSP's *User-to-User Information*)
  - Slower due to hop-by-hop communication
  - Increased signaling load on intermediate nodes

Only Server QoS-aware – DMSP as Transport Layer



Legacy Client & Server – Web-based initiation

Neither Client nor Server are QoS-aware

- Reservations must be initiated by a third party
- User should be able to control QoS

Web-based service initiation

- Many sessions are initiated via the ubiquitous Web interface
  - „Misuse“ this start phase for QoS
- Extend Content Negotiation to cover QoS
  - Web Server provides Information about document
  - Browser provides user preferences
- Agent-driven
  - Hyperlinks accompanied by HTML/XML Tags describing their QoS requirements
  - Selection algorithms to be executed by browser
  - Browser selects QoS and triggers signaling
- Server-driven
  - Browser generates HTTP extension headers according to user preferences
  - Web server selects QoS and triggers signaling

Problems

Ephemeral ports & transport protocol negotiation

- No knowledge of port number unless involved in data communication
- Ways to obtain knowledge
  - Application specific proxy

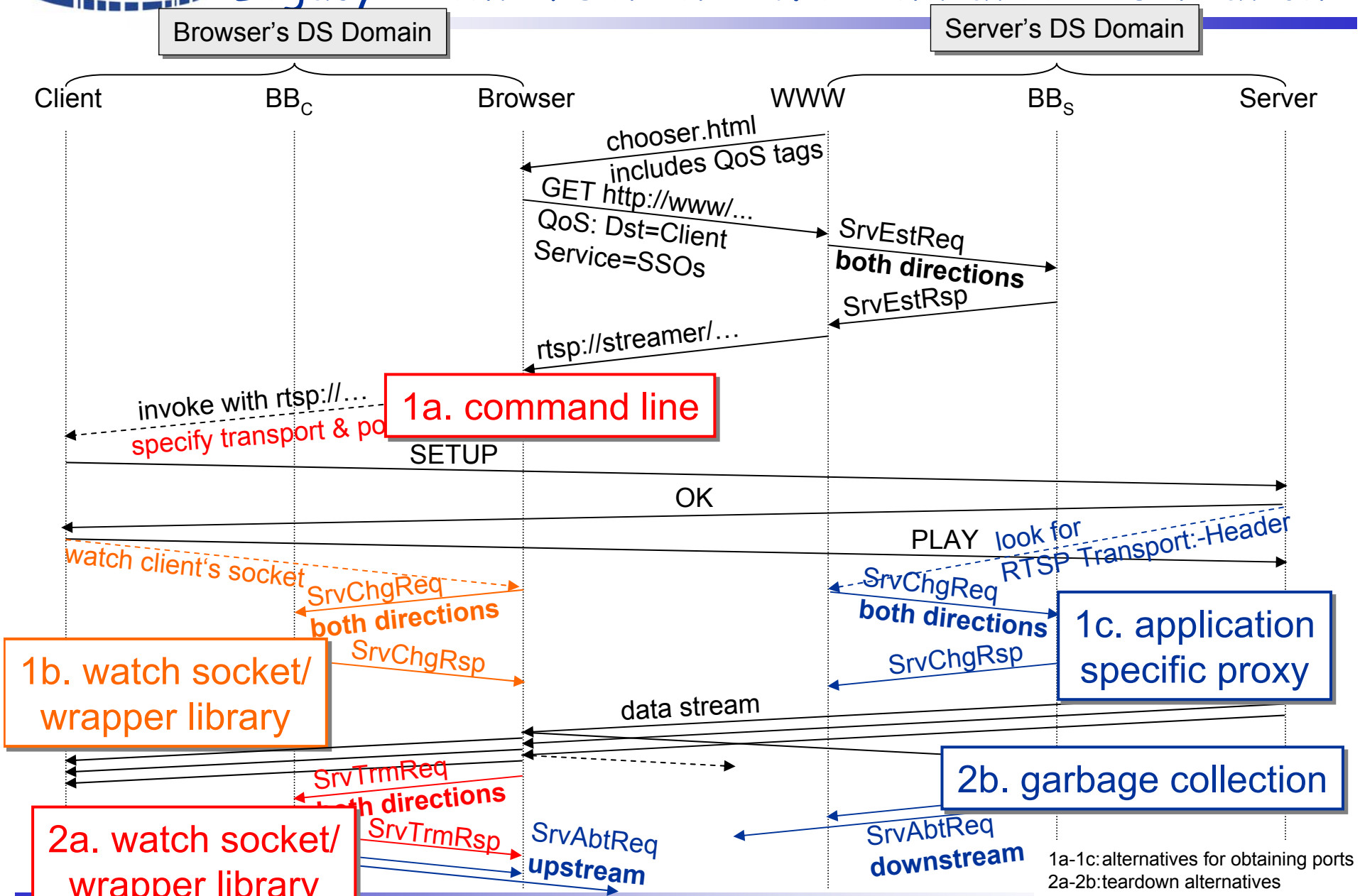
  → not general
  - Watch client's socket(s)
  - Wrapper library intercepting client's socket operations

  → nasty
- Control legacy application to use a given port, i. e., command line

End of Reservation

- Reservation termination (by originator/responder only)
  - Obtain information about end of communication between sender & receiver
    - Wait for process termination
    - Watch socket
- Reservation abort (by any node involved)
  - Garbage collection by first hop router
    - No data packets during a certain period of time
    - In case of soft-state originator/responder: when refreshes are missing
    - Both waste resources for some time

Legacy Client & Server – Web-initiated Reservation



Conclusions – Outlook

Benefits

- Only two modified applications needed: browser and web server
  - Alternatively modified proxy instead of modified browser
- QoS support independent of target application
- Possible migration strategy from no QoS to full QoS

Limitations

- General support for ephemeral ports is costly/requires ugly measures
- Transport not adopted to QoS
  - Application might wrongly reduce rate in case of packet loss

Requirements to QoS signaling

- Support third party initiated reservations (possibly limited to same domain)
- Work as “transport protocol” (for those third party initiations)

Future Work

- Implement QoS-aware browser and web server
  - Mozilla and Apache
- Support for ephemeral ports
  - Selected applications for A/V-streaming
- Testbed with DSDM prototype
  - Latency and user perception