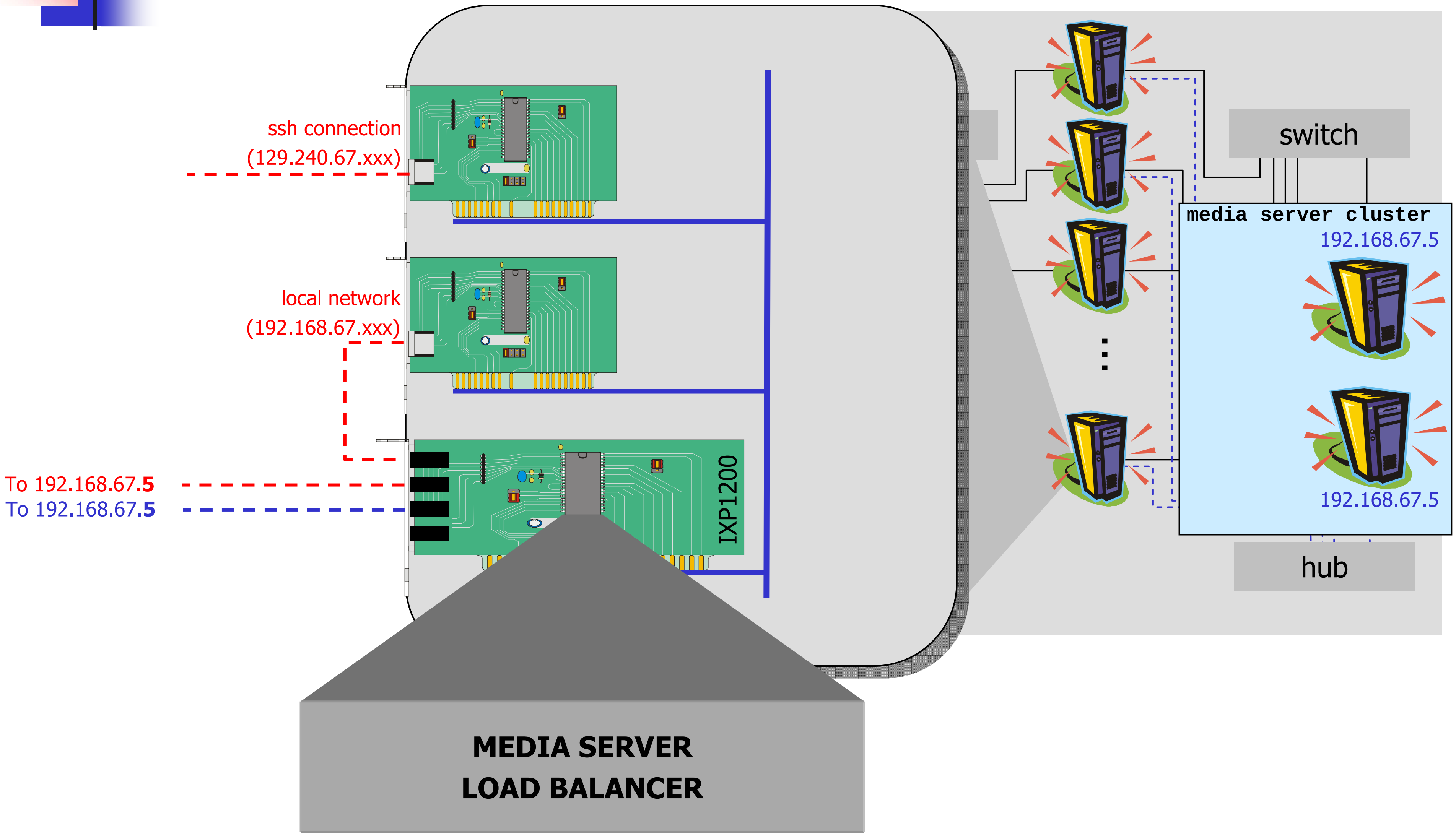
INF5060:
Multimedia data communication using network processors

Lab Assignment

15/10 - 2004

Assignment 6 – Lab setup



Assignment 6 – Scenario

- Media server–client scenario:
  - the two "server" machines
    - each run a streaming server
    - have their network interfaces configured equally, i.e., IP address 192.168.67.5
    - the servers should answer to requests and start a media stream
  - the clients
    - send requests to the server
  - the IXP card should implement a transparent load balancer selecting one of the "equal" servers to serve a request according to a simple policy

Assignment 6 – Client/Server Applications

- User inf5060 (same password as root)

- Server (**192.168.67.5** (also installed on the other machines)):
  - **komssys**: /home/inf5060/komssys
    - rtsp server: `/home/inf5060/komssys/LINUX/rtsp/server/`**`rstp_server`**
  - video file: /home/inf5060/video/full_1.mpg

- Client (**192.186.67.111-118**)
  - **mplayer**: `/home/inf5060/MPlayer-1.0pre2/`**`mplayer`**
  - starting playback:
    - retrieving data from server: `mplayer rtsp://192.168.67.5:9070/full_1.mpg`
    - retrieving data local client:
      - `mplayer /home/inf5060/video/full_1.mpg`
      - `mplayer rtsp://localhost:9070/full_1.mpg`
  - useful options:
    - no audio: `-ao null`
    - ascii art video output: `–vo aa`
    - ...

Assignment 6 – Implementation

- The **assignment**:
  - implement the transparent load balancer
    - use the packet bridge with ARP support to forward packets (see assignment 5)
    - when a request is received, use a policy (random, RR) to choose which of the "equal" machines that should serve the request (to which port to forward packets)
    - the load balancer must remember which clients use which port
  - implement a load balancer monitor (see assignment 3):
    - let the load balancer keep a statistic of how many packets are sent to each server
    - implement a crosscall that can display the load statistics for the media servers (*total number of packets* and *percentage of total* for each server machine/port)

Assignment 6 – Deliverables

- The **assignment**:
  - write and deliver a short (2-3 pages) **report** describing your system
    - report
    - source code

    - deadline: Monday 29/11 - 2004

  - **present** your system to the class

    - give a 15-minutes *overview* over your implementation
      - design
      - what is running where (StrongARM vs. Microengine)
      - what is stored where (SDRAM vs. SRAM vs. Scratch)

    - give a *demo* of the system

    - presentation day: Friday 3/12 - 2004

Multimedia Signaling Protocols

- Signaling protocols
  - used for connection setup
  - distribution information about data protocols
- Primary Internet signaling protocols
  - RTSP – Real-Time Streaming Protocol
    - HTTP-like
    - mainly for on-demand audio and video streaming
  - SIP – Session Initiation Protocol
    - SMTP-like
    - mainly for IP telephony
  - SDP – Session Description Protocol
    - not really a protocol
    - carried inside RTSP and SIP for description of data stream
  - H.323
    - descriptions of data streams carried in ASN.1 encoding
    - mainly for IP telephony
- Data protocols
  - RTP/RTCP – Real-Time Transfer Protocol/RTP Control Protocol

Real-Time Streaming Protocol (RTSP)

- Internet media-on-demand
  - select and playback streaming media from server
  - similar to VCR, but
    - potentially new functionality
    - integration with Web
    - security
    - varying quality
  - need for control protocol
    - start, stop, pause, …
- RTSP is also usable for
  - Near video-on-demand (multicast)
  - Live broadcasts (multicast, restricted control functionality)
  - …

RTSP Approach

- In line with established Internet protocols
  - Similar to HTTP 1.1 in style
  - Uses URLs for addressing: rtsp://video.server.com:8765/videos/themovie.mpg
  - Range definitions
  - Proxy usage
  - Expiration dates for RTSP DESCRIBE responses
  - Other referenced protocols from Internet (RTP, SDP)
- Functional differences from HTTP
  - Data transfer is separate from RTSP connection
    - typically via RTP
    - unlike "HTTP streaming"
  - Server maintains state – setup and teardown messages
  - Server as well as clients can send requests

RTSP Features

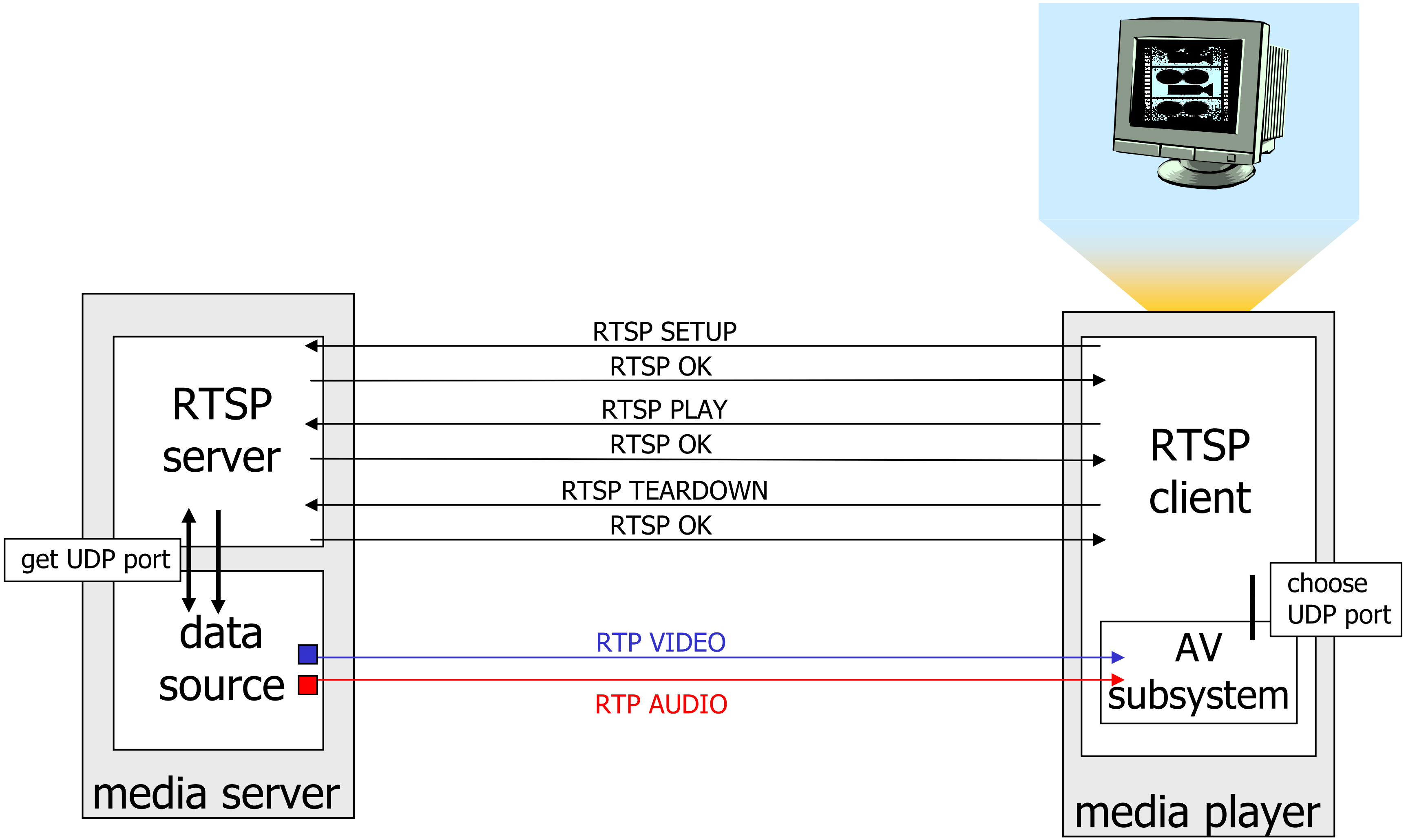
- Rough synchronization
  - Media description in DESCRIBE response
  - Timing description in SETUP response
  - Fine-grained through RTP sender reports
- Aggregate and separate control of streams possible
- Virtual presentations
  - Server controls timing for aggregate sessions
  - RTSP Server may control several data (RTP) servers
- Load balancing through redirect at connect time
  - Use REDIRECT at connect time
- Caching
  - Only RTSP caching (stream state) so far
  - Data stream caching is under discussion

RTSP Methods

OPTIONS	$C \rightarrow S$ $C \leftarrow S$	determine capabilities of server/client
DESCRIBE	$C \rightarrow S$	get description of media stream
ANNOUNCE	$C \leftrightarrow S$	announce new session description
SETUP	$C \rightarrow S$	create media session
RECORD	$C \rightarrow S$	start media recording
PLAY	$C \rightarrow S$	start media delivery
PAUSE	$C \rightarrow S$	pause media delivery
REDIRECT	$C \leftarrow S$	redirection to another server
TEARDOWN	$C \rightarrow S$	immediate teardown
SET_PARAMETER	$C \leftrightarrow S$	change server/client parameter
GET_PARAMETER	$C \leftrightarrow S$	read server/client parameter

RTSP Operation

- Integration with other real-time and multimedia protocols



Relevant RTSP Messages

- Complete protocol specified in RFC2326

- Client sends

```
SETUP rtsp://server.name.com:5540/path/leading/to/medium.typ RTSP/1.0
CSeq: 12
Transport: RTP/AVP;unicast;client_port=4588-4589
```

- Server answers

```
RTSP/1.0 200 OK
CSeq: 12
Date: 15 Oct 2004 10:15:00 GMT
Session: anything_even_with_newline_if_its_microsoft
Transport:  RTP/AVP;unicast;client_port=4588-4589;server_port=6256_6257
```

- RTSP works over TCP
  - no reliable message boundaries
  - RTSP headers end with double return, each return may be "`0xa`", "`0xd`", "`0xa 0xd`" or "`0xd 0xa`"
  - but RTSP can have a body as well
  - indicated by `Content-Length=`<bytes>, number of <bytes> after double return

- We assume that a session ends when the client closes the TCP connection
  - even though that is not standard compliant