Adaptive Run-time Resource Management on Heterogeneous Devices

Roel Wuyts
imec, Leuven, Belgium

ArtistDesign Meeting, July 6 2010, Leuven, Belgium

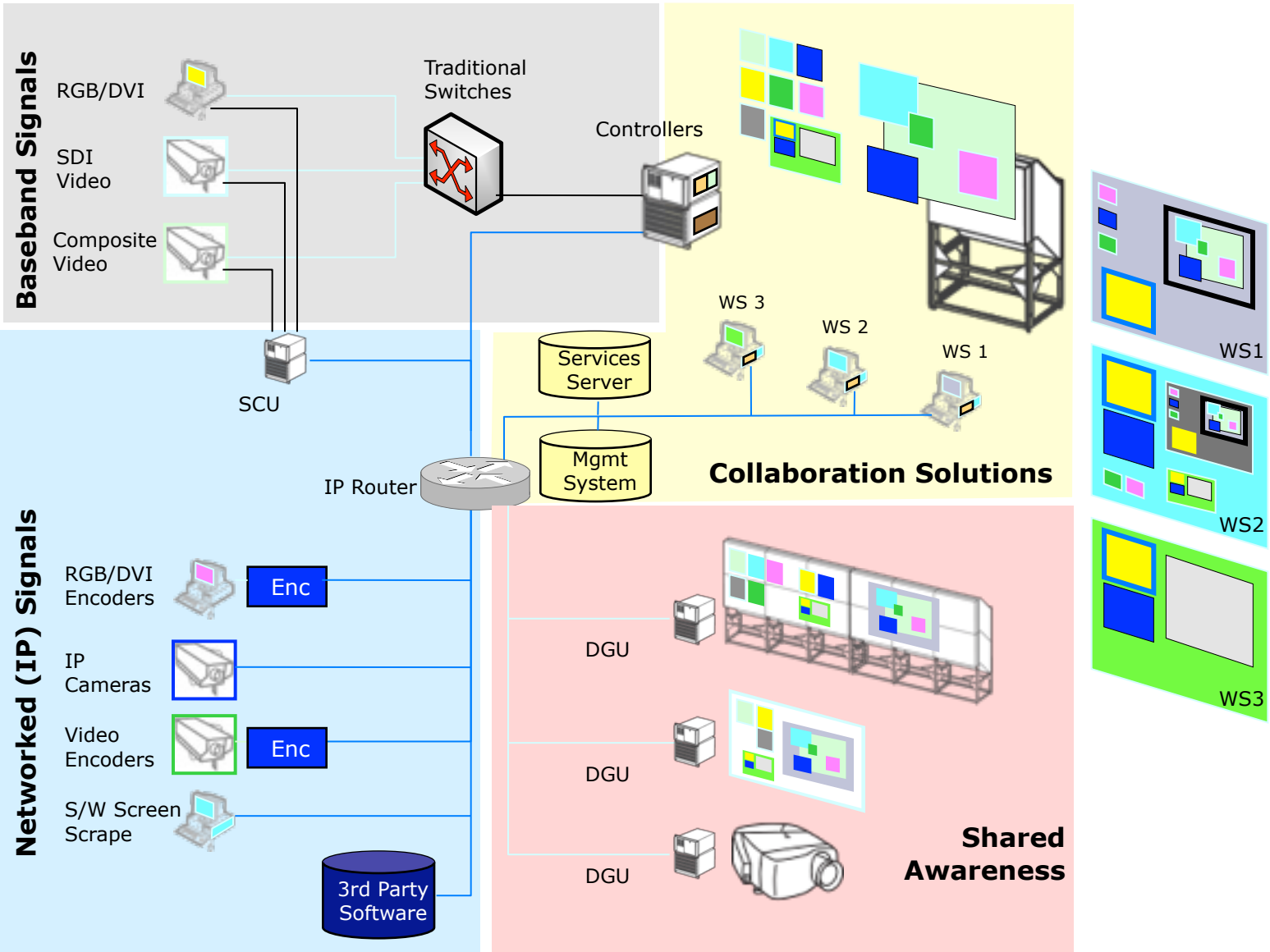
Agenda

- Background and motivations
- Network level resource manager + demo video
- Conclusions

Background info

- Collaborating with Barco in the Artemis project *Scalopes*
- Barco's products and vision
  - Vision: Networked Visualisation
    - Intelligent networked visualisation nodes (encoders, decoders, renderers).
    - Processing as a service 'in the cloud'
  - Products: Network Generation
    - NBMS – Networked Broadcast Monitoring System, including Broadcast Monitoring Suite (BMS) software.
    - TransformN – Network Visualisation platform for the wide controlroom market, including Controlroom Management Suite (CMS) software.
    - Common hardware platform with optimized usability of market dependent feature by means of dedicated applications.

Network Visualisation Concept



A broadcast example – seamless touring

- MGS handles seamless switching
  - Extraction of Digital TV service information
  - Extraction of thumbnails from Digital TV sources
  - Transition between sources
    - Video
    - Audio (level metering & listening)
    - Alarming
- Touring display element (displet) composition of smaller synchronised basic rendering elements.

Motivations: why dynamic resource management?

- High performance and low power requires parallel processors
- Some processors have domain-specific architectures to accelerate certain types of programs (e.g. GPU)
- Conventional task model uses static task off-loading which can lead to processors contentions/idling in a dynamic environment:
  - highly variable workloads
    - different video stream quality
    - different number of video streams
    - different processing, e.g. depending on video analysis
  - platform variations
    - video processing servers with variations in CPU and GPU have different load balancing schemes
- Therefore a dynamic task assignment of tasks can maximize throughput by using processors efficiently

Hierarchical resource management

- Low-level management
  - device level resource manager: the NodeRM
  - not the focus of this presentation
- High-level management
  - network level resource manager: the network RM

Network level resource manager

- continuous resource usage monitoring of all slave nodes
  - workload
  - memory usage
  - and network bandwidth
- Workload balance among all connected slave nodes
  - avoid workload over threshold
  - maximize overall throughput

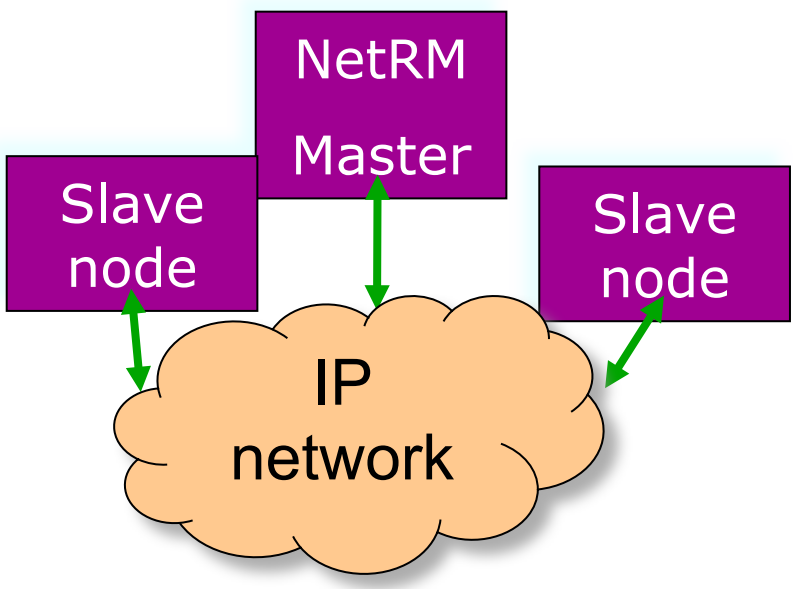
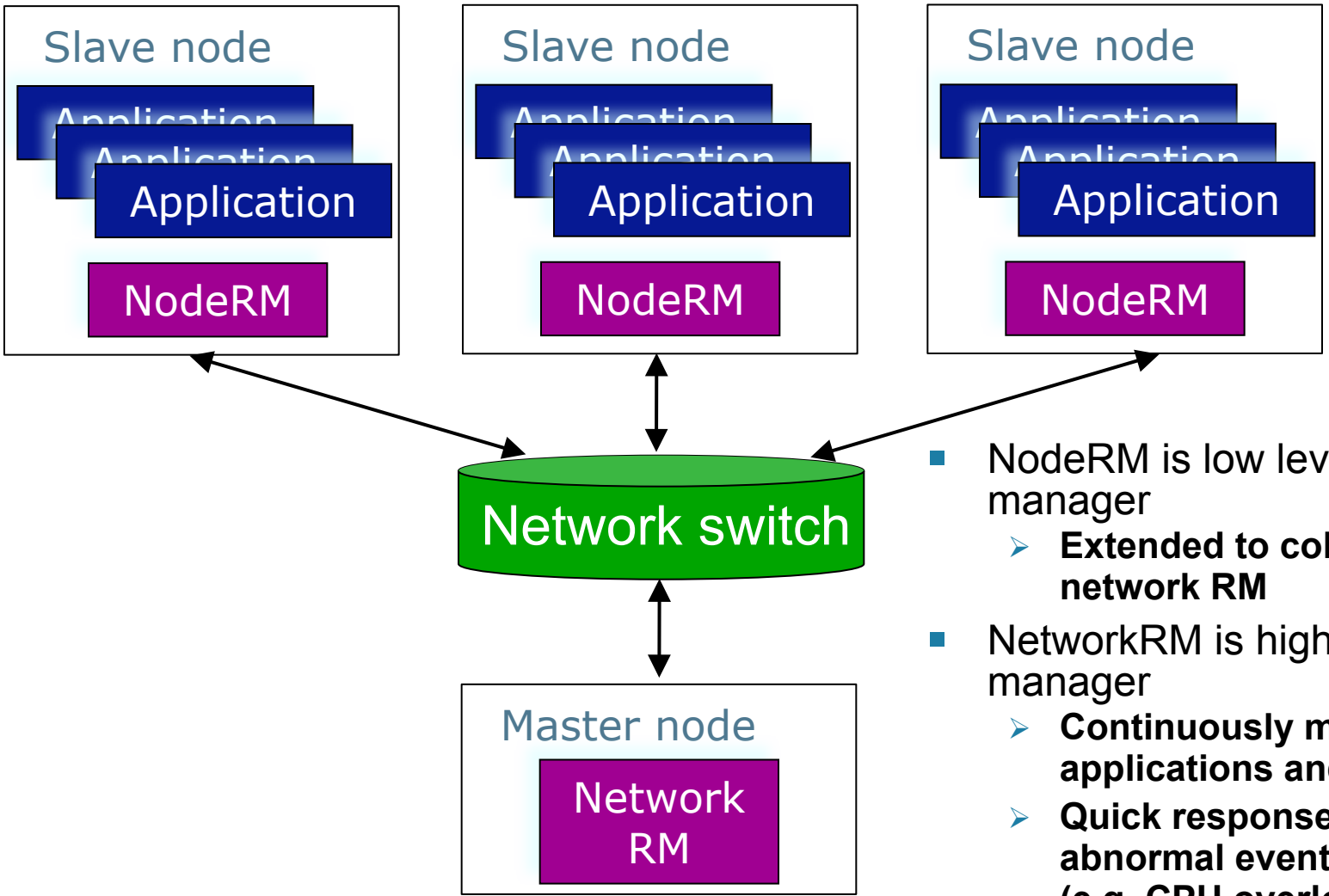


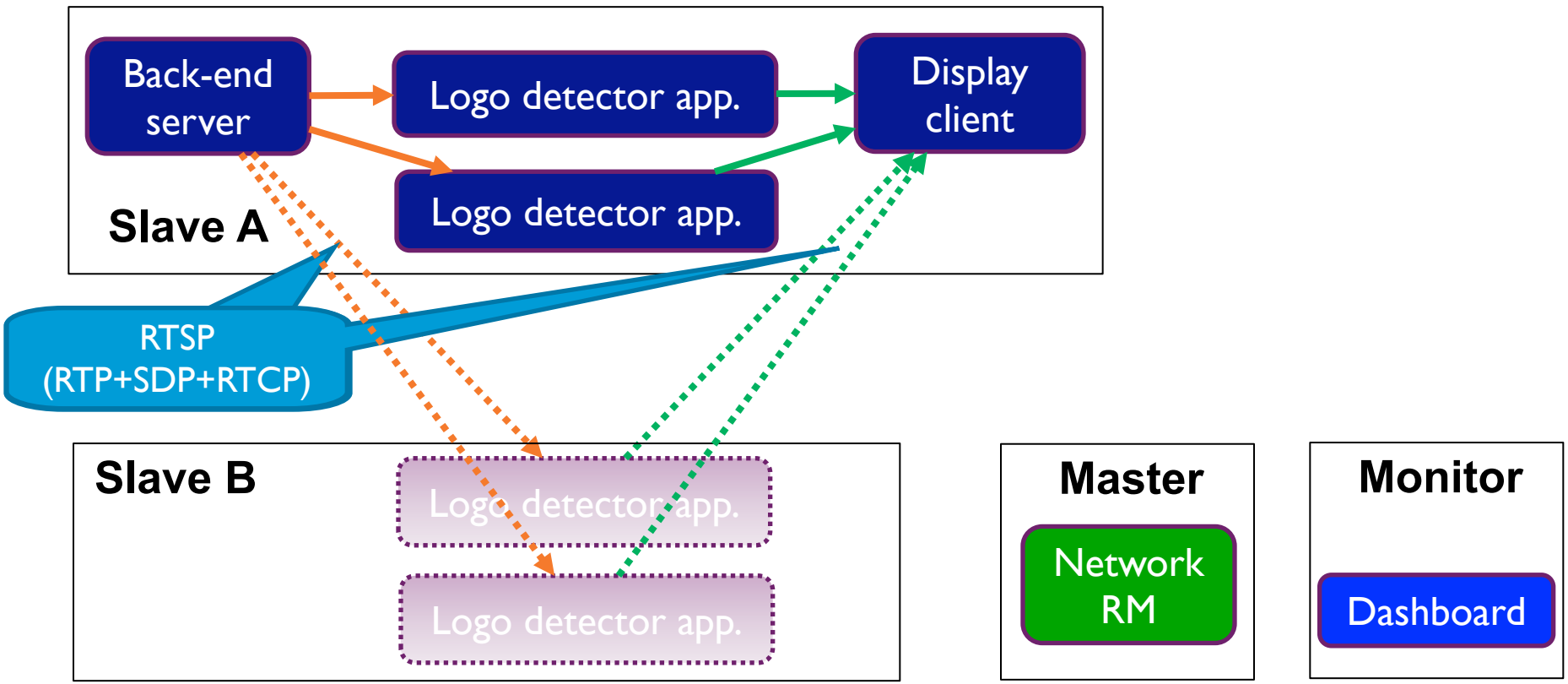
Illustration of network level resource management

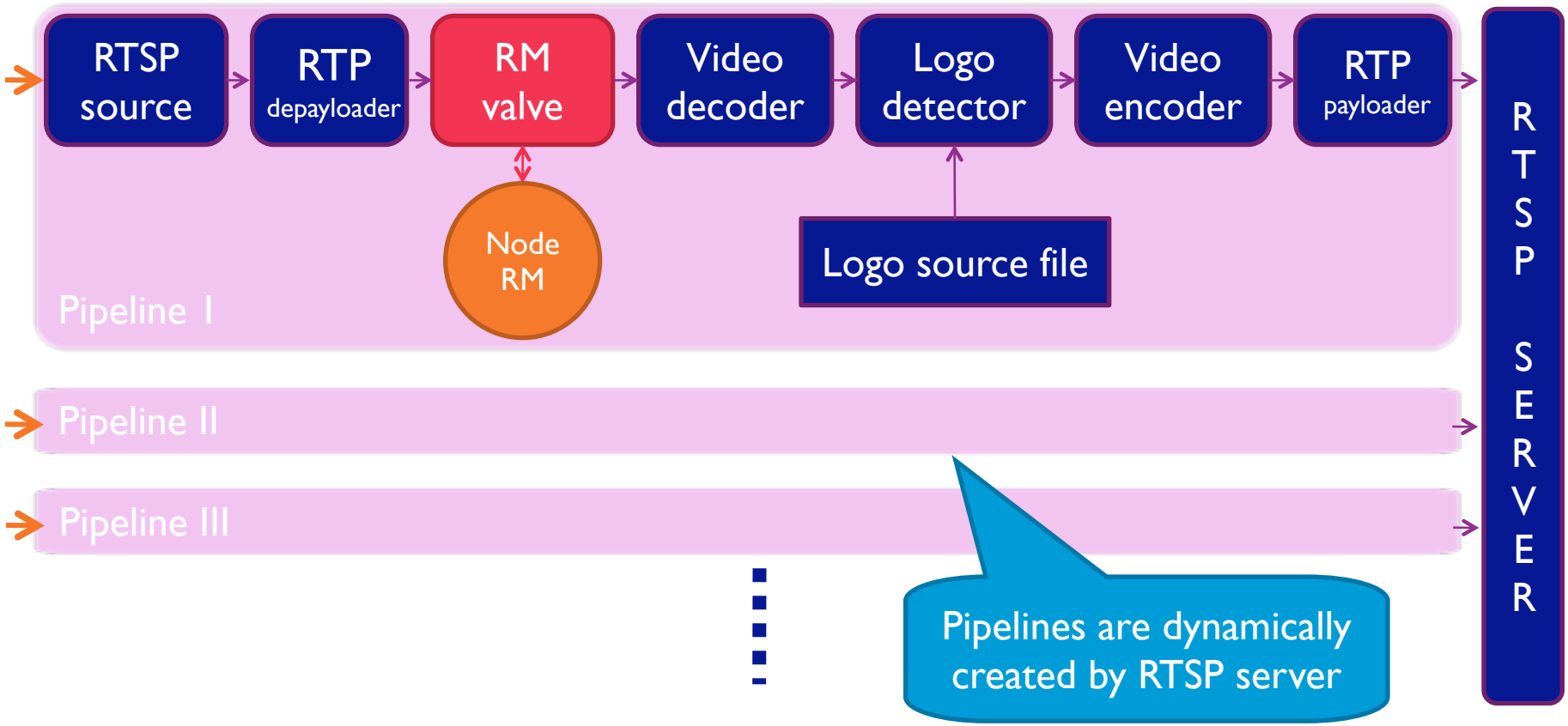


- NodeRM is low level resource manager
  - **Extended to collaborate with network RM**
- NetworkRM is high level resource manager
  - **Continuously monitor all applications and slave nodes**
  - **Quick response whenever abnormal events are detected (e.g. CPU overloaded)**

Network RM demo

- Objectives:
  - To demonstrate the collaboration of NodeRM and NetworkRM
  - To show the effectiveness of NetworkRM algorithm in a dynamic networked working environment
- Hardware setup:
  - 4 workstations: 2 slave nodes + 1 master node + monitoring node
  - Slave A has a dual-core CPU, Slave B has quad-core CPU
  - All nodes are connected via Ethernet 1000Base-T FD
- Software setup:
  - All workstations with vanilla Linux (Ubuntu 9.10 for slave nodes)
  - Multiple instances of video logo detector (based on Gstreamer framework)


Back-end server
Logo detector app.
Display client
Logo detector app.
Slave A
RTSP (RTP+SDP+RTCP)
Slave B
Logo detector app.
Logo detector app.
Master
Network RM
Monitor
Dashboard


RTSP source
RTP depayloader
RM valve
Video decoder
Logo detector
Video encoder
RTP payloader
Node RM
Logo source file
Pipeline I
Pipeline II
Pipeline III
RTSP SERVER
Pipelines are dynamically created by RTSP server

The demo setting…

- Left screen = performance monitoring
  - Top graphs = slave A, bottom = slave B
  - Left graphs = network load, right graphs = processor load.
- Middle screen = client screen
  - Runs viewers fed by slave A and/or slave B.

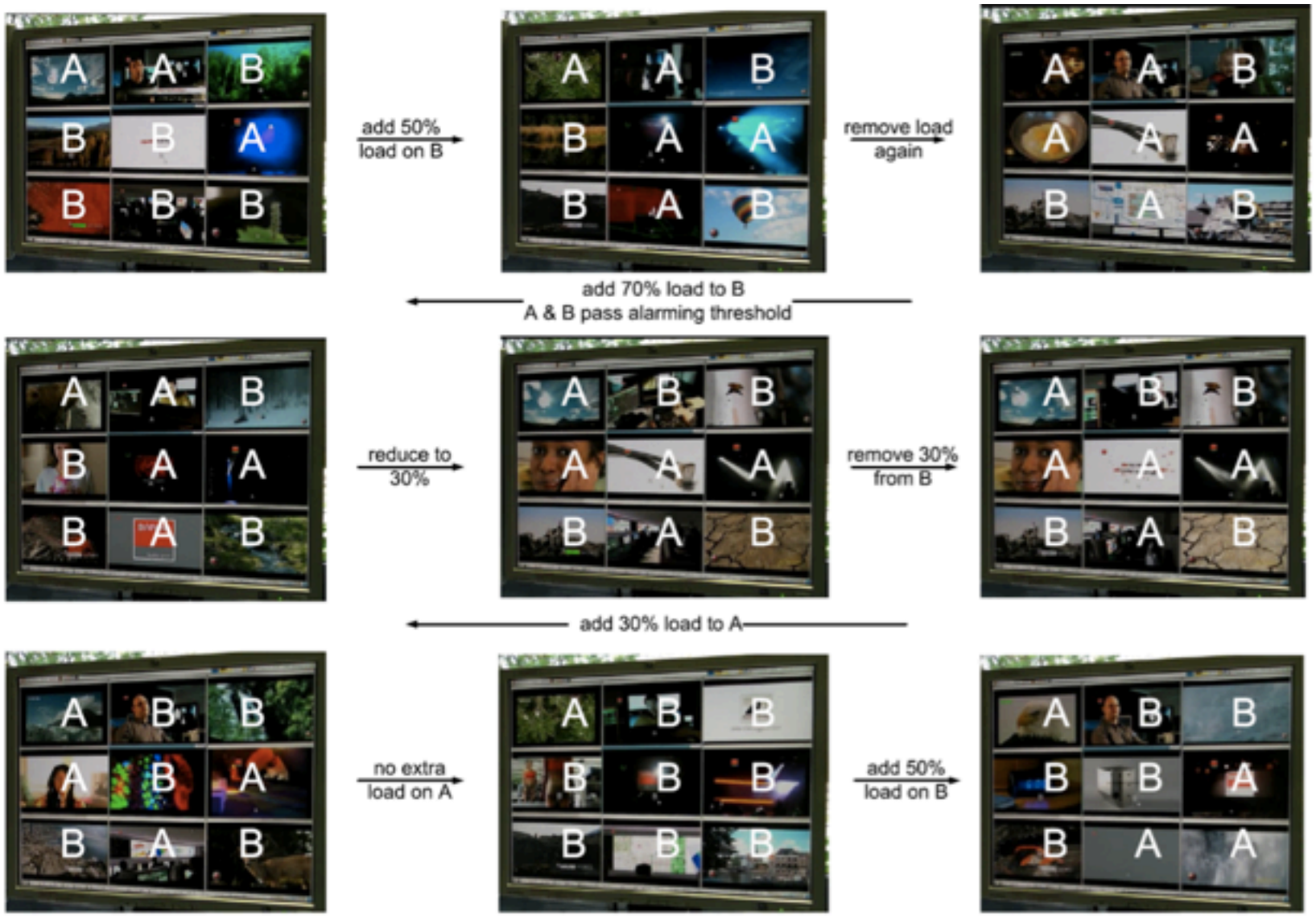
- Right screens = network resource manager
- Netbook = allows to add additional load on slave A or slave B

- Video streams have different characteristics… different loads required.

The demo startup…

- Some things to keep in mind…
  - Pipeline load is varying
    - At certain intervals the logo-detection demands more effort.
    - Video content is ‘dynamically changing content’.
  - Letters A and B on top of the video (client workstation)
    - The smaller the letter the more resource intensive the pipeline…
    - Letters A and B swap when windows first appear as resource manager is deciding to let slave A or B do the work…
    - Logos are flashing green (logo BBC, Barco)… indicating the logo-detection in the pipeline is functional and detected the logo..
- After startup we normally have approx. 3 windows fed by slave A and 6 by slave B.
- Load on slave A and B is manually changed to check resource management functionality.

Summary of the demo



Conclusions and future work

- Dynamic resource management is important to the efficient and robust running of a video processing system on networked heterogeneous platforms, we demonstrated
  - Device level RM can improve device throughput automatically
  - Network level RM can keep workstations workload on track
- In the future we will
  - Build a simulation environment to quickly evaluate various RM strategies
  - Consider other system constraints: real-time, reliability…
  - Construct RM strategies for different resources: power budget…

Thank you for your attention

We value your opinion and questions