

# Session Sharing as Middleware Service for Distributed Multimedia Applications

## MIPS 2003

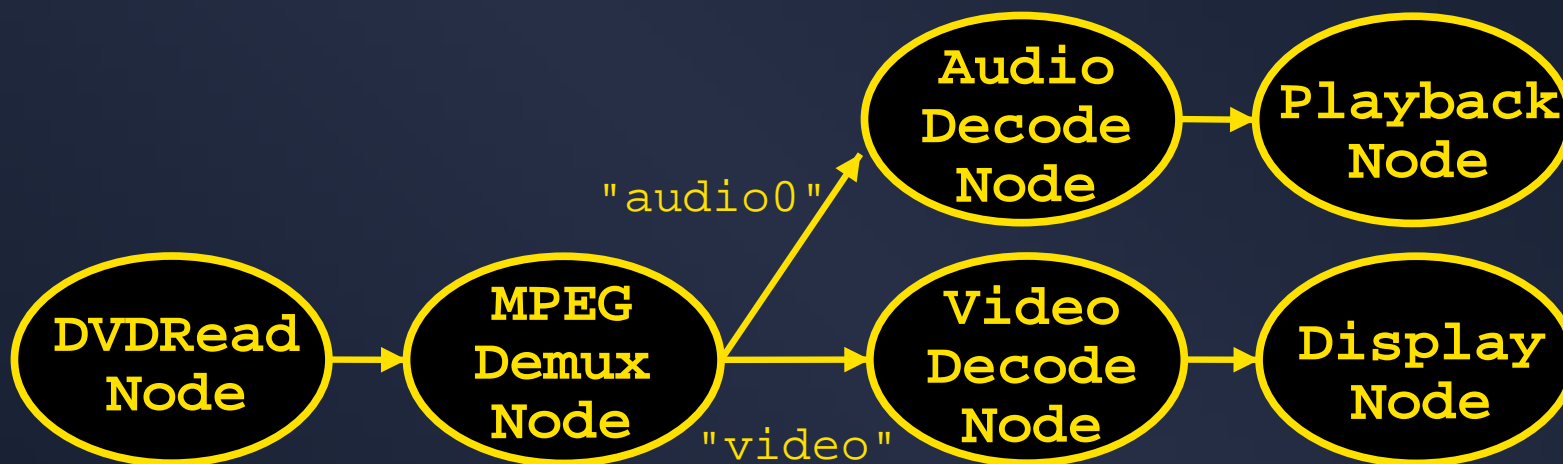
Marco Lohse, Michael Replinger, Philipp Slusallek

Computer Graphics Lab, Saarland University, Germany

- Increased number of mobile and stationary multimedia devices
  - Web pads, PDAs, mobile phones, PCs
- Middleware approaches for networked multimedia
  - Network-transparent access and cooperation
- Collaborative application scenarios
  - Number of users simultaneously enjoys the same (or similar) content using different devices
  - TV program being watched on stationary and mobile device
  - DVD playback with different audio tracks on different mobile devices

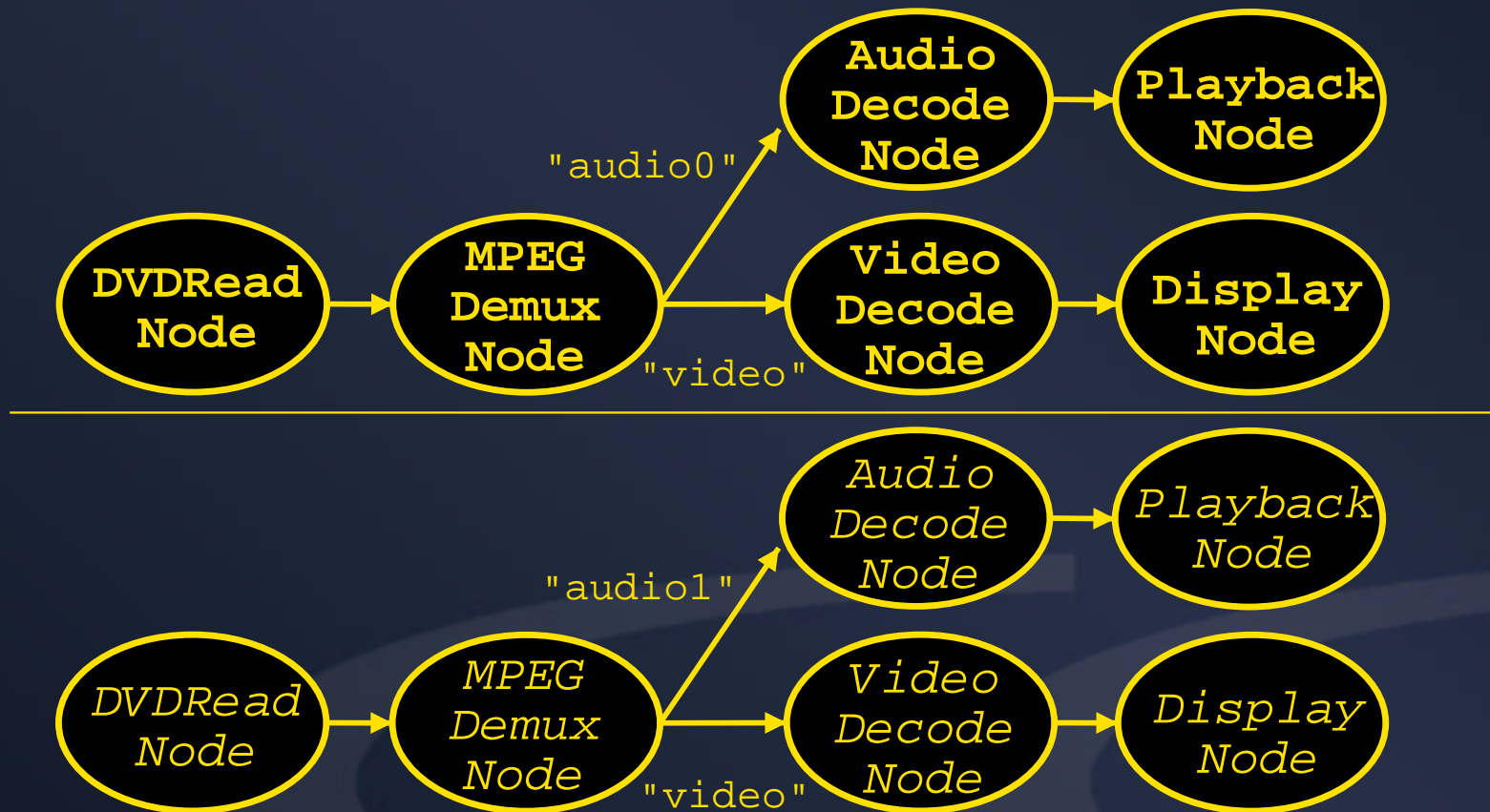
# Motivation

- DVD playback with different audio tracks on different mobile devices



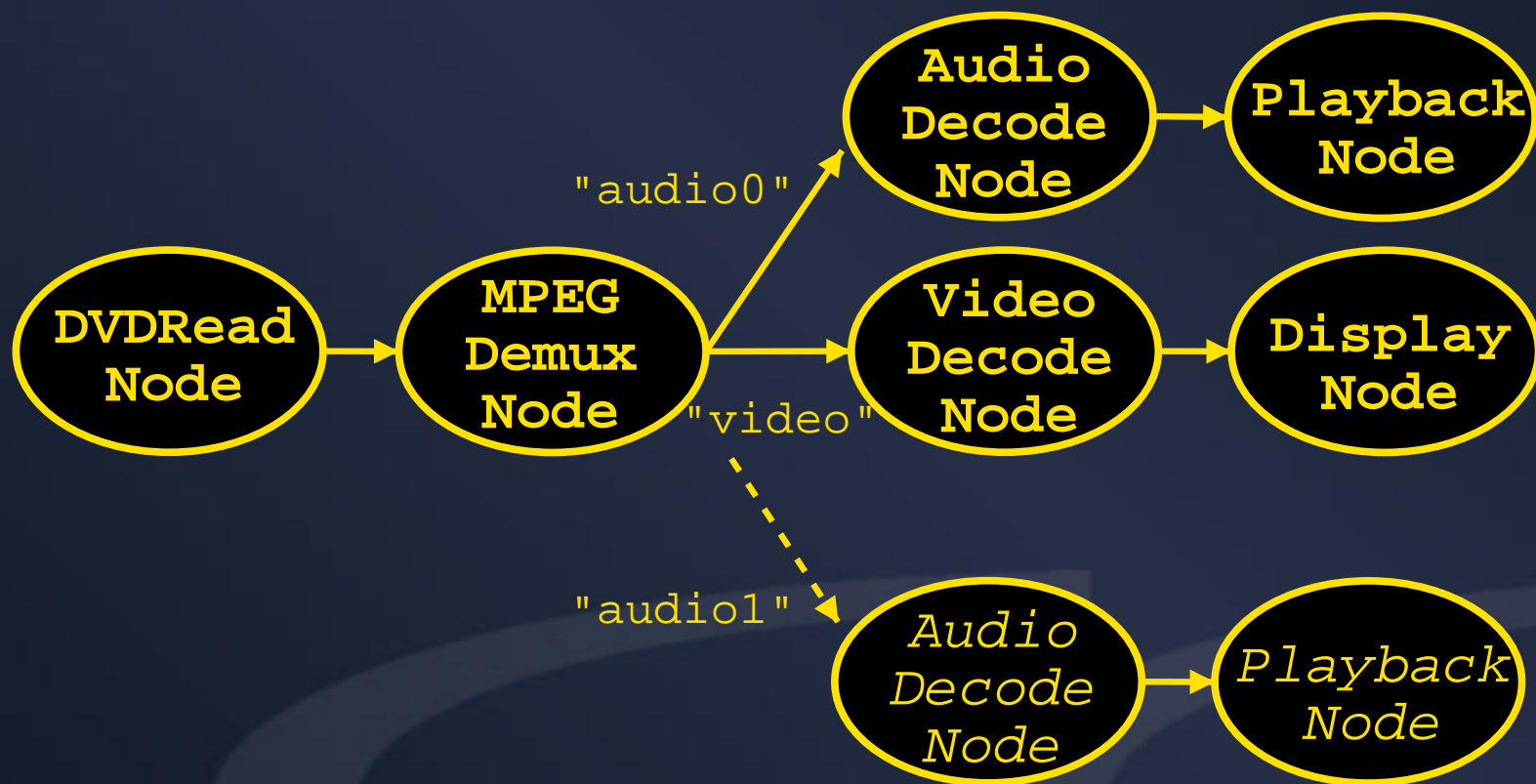
# Motivation

- DVD playback with different audio tracks on different mobile devices



# Motivation

- DVD playback with different audio tracks on different mobile devices

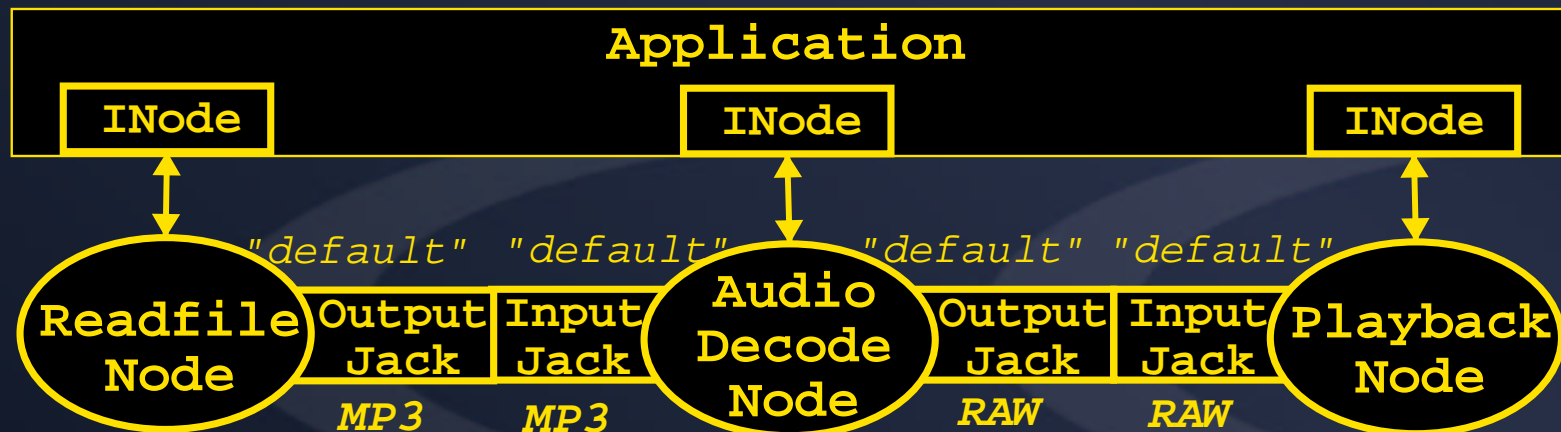


- Networked multimedia middleware
- Registry service
  - Realizes sharing policies
  - Administrates running flow graphs (sessions)
- Session sharing as middleware service
  - Automatic sharing of (parts of) active flow graphs
  - Different optimization criteria
  - Eliminates the problem of multiple access of hardware devices
  - Reduces the processing requirements through re-usage
- Distributed synchronization

- Motivation
- Requirements
- Network-Integrated Multimedia Middleware (NMM)
- Registry service
- Session sharing
- Distributed synchronization
- Application scenarios
- Conclusions and future work

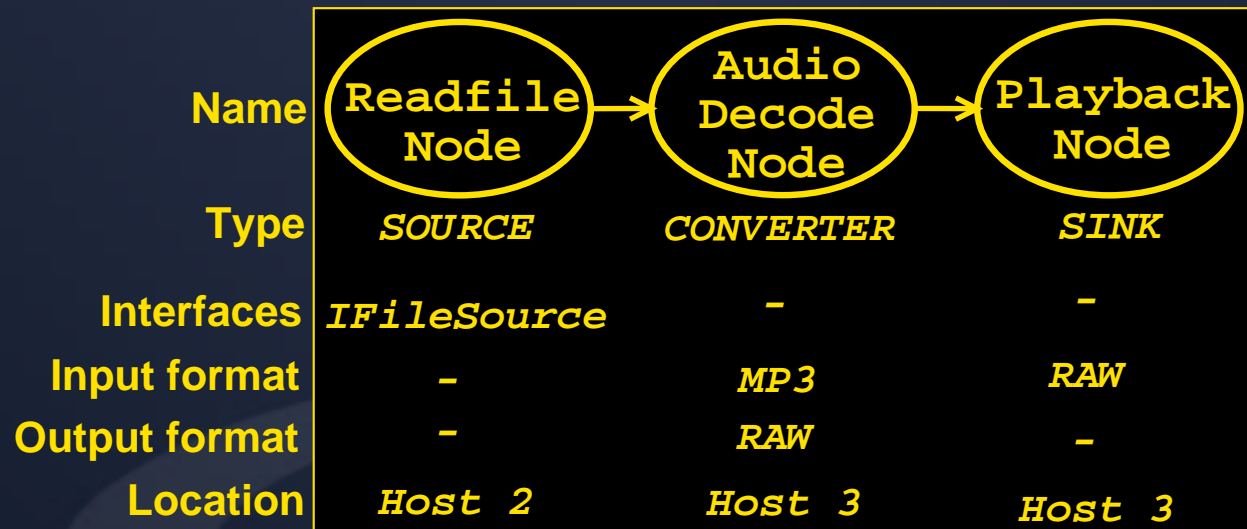
# NMM Flow Graphs

- Nodes as smallest processing unit
- Jacks to connect nodes
- Jack tags to identify inputs and outputs
- Formats to type connections
- Interfaces to access objects
- Multimedia data buffer and events



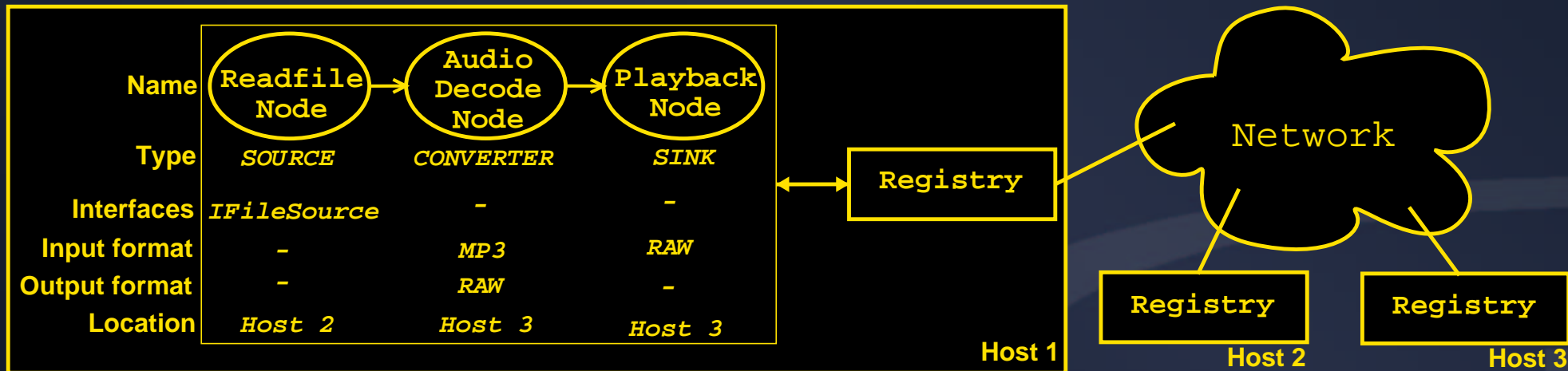
# Registry Service

- Administrates locally available nodes
- Queries
  - Single (subset) of a node description, or
  - Graph description



# Registry Service

- Administrates locally available nodes
- Queries
  - Single (subset) of a node description, or
  - Graph description

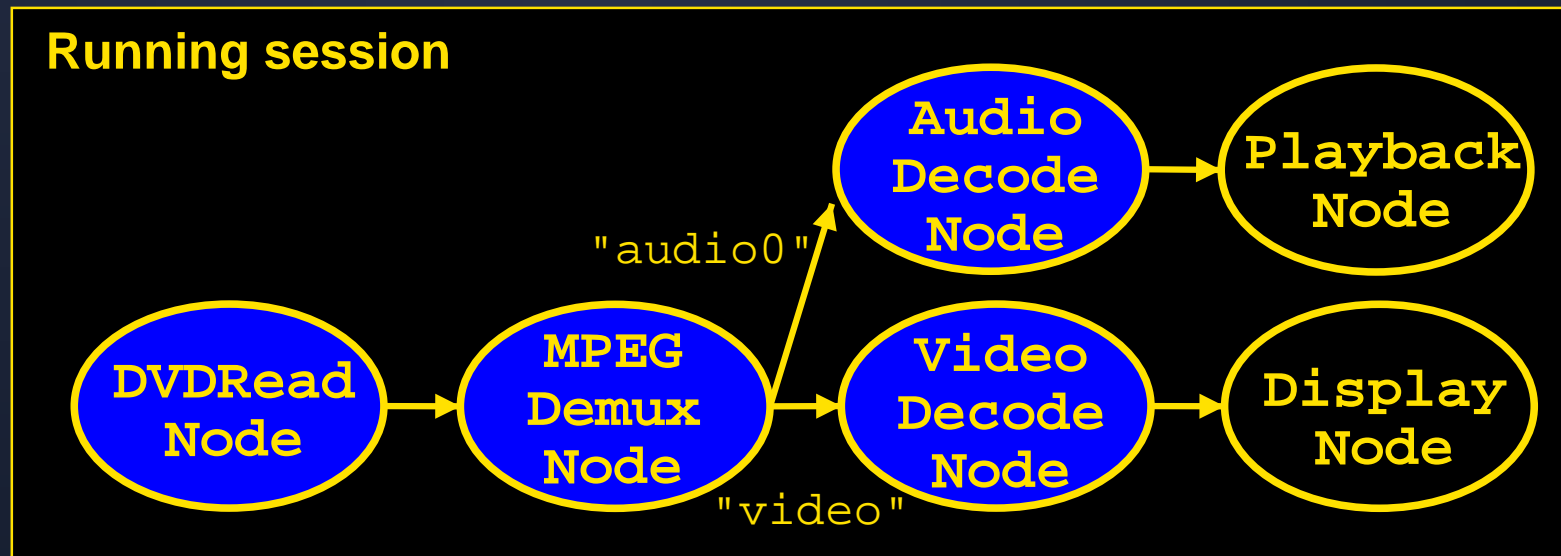


- Peer-to-peer registry for distributed flow graphs

- Sharing policy for each node description
  - Shared: explicitly request a shared node
  - Exclusive
  - Exclusive or shared: try exclusive first, then shared
  - Shared or exclusive
  - Exclusive, then shared: try exclusive, if successful share the node
- Registry service administrates running sessions

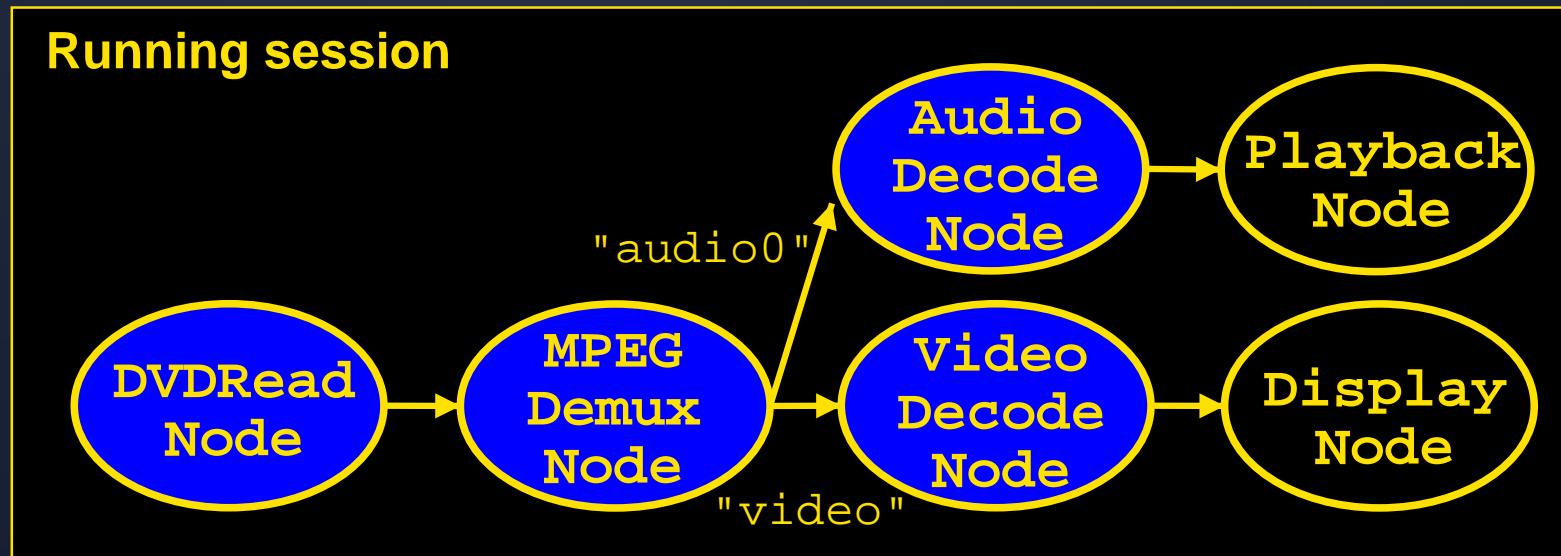
# Example

- Running session with shared nodes



# Example

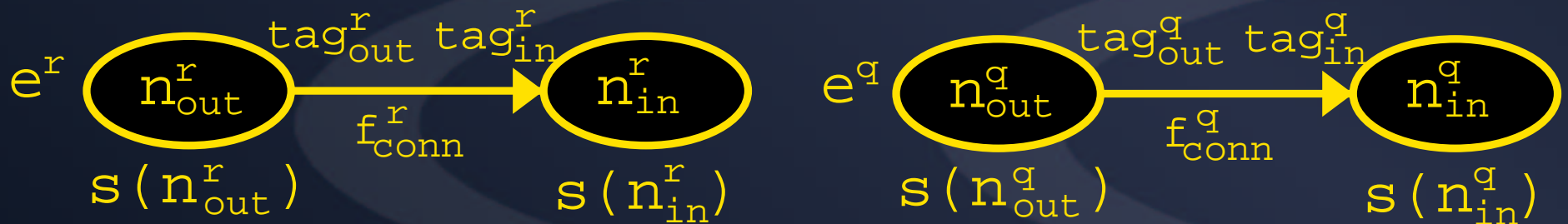
- Running session with shared nodes
- Query with policy *exclusive or shared*



**Query**

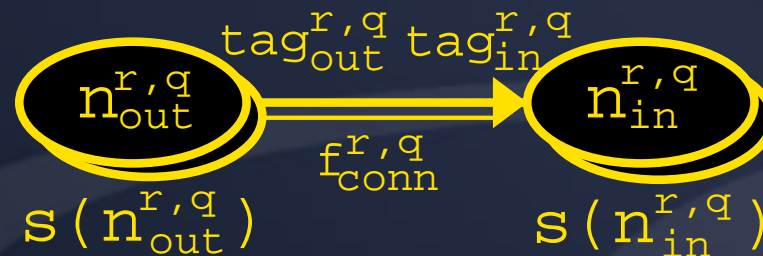


- Running sessions  $R_1, \dots, R_n$ , query  $Q$
- Idea
  - Start with node in  $Q$  with no incoming edges (e.g. source)
  - Try to find 'matching' node in  $R_i$
  - Try to expand overlapping sub-graphs
  - Individual test for edges  $e^r$  and  $e^q$
  - Value different overlappings



# Complete Overlap

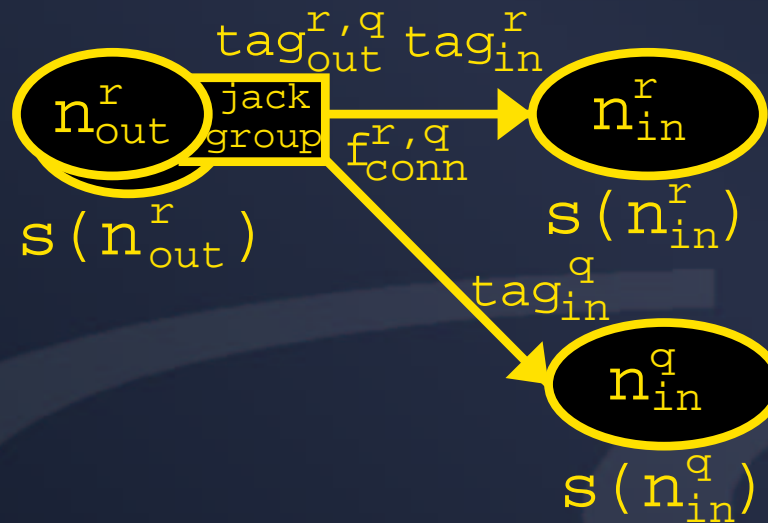
- Sharing policy:  $s(n_{out}^q) = s(n_{out}^r) = true$  and  $s(n_{in}^q) = s(n_{in}^r) = true$
- Node descriptions:  $n_{out}^q \subseteq n_{out}^r$  and  $n_{in}^q \subseteq n_{in}^r$ 
  - Includes test for location
- Tags:  $tag_{out}^q = tag_{out}^r$  and  $tag_{in}^q = tag_{in}^r$
- Connection format:  $f_{conn}^q \subseteq f_{conn}^r$ .



⇒ Re-use of complete sub-graph

# Copy Overlap

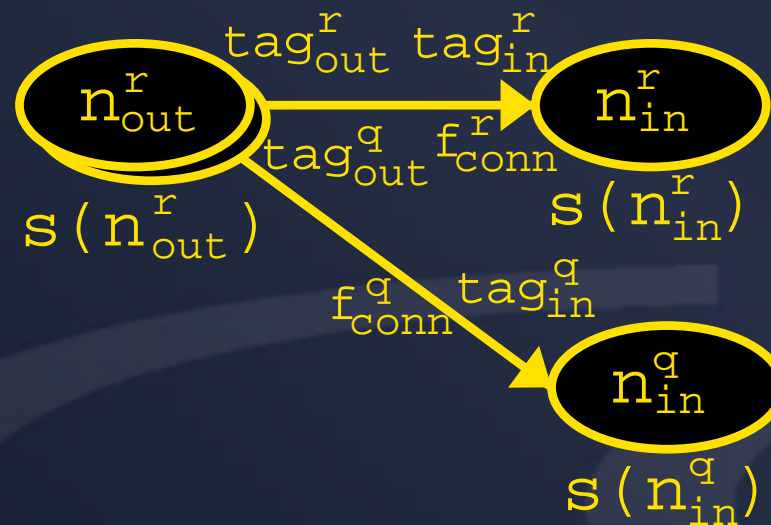
- Sharing policy:  $s(n_{out}^q) = s(n_{out}^r) = true$
- Node descriptions:  $n_{out}^q \subseteq n_{out}^r$
- Tags:  $tag_{out}^q = tag_{out}^r$
- Connection format:  $f_{conn}^q \subseteq f_{conn}^r$



⇒ Additional output jack and node for  $n_{in}^q$

# Output Overlap

- Sharing policy:  $s(n_{out}^q) = s(n_{out}^r) = true$
- Node descriptions:  $n_{out}^q \subseteq n_{out}^r$
- Tags:  $tag_{out}^q \neq tag_{out}^r$  and jack  $tag_{out}^q$  not connected
- Connection format:  $f_{conn}^q$  valid for  $tag_{out}^q$



⇒ Additional connection and node for  $n_{in}^q$



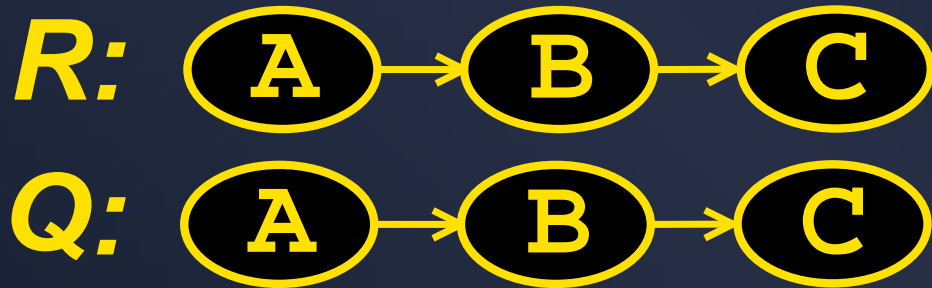
# Session Sharing Algorithm

```
Find nodes with no incoming edges in R
Find nodes with no incoming edges in Q
For each node q of Q and each node r of R
  For each outgoing edge of q and r
    if(copy overlap)
      Start recursion with sub-graphs(R'',Q'')
    if(complete overlap)
      Start recursion with sub-graphs(R',Q')
```

```
if(output overlap)
  Start recursion with sub-graphs(R'',Q'')
```

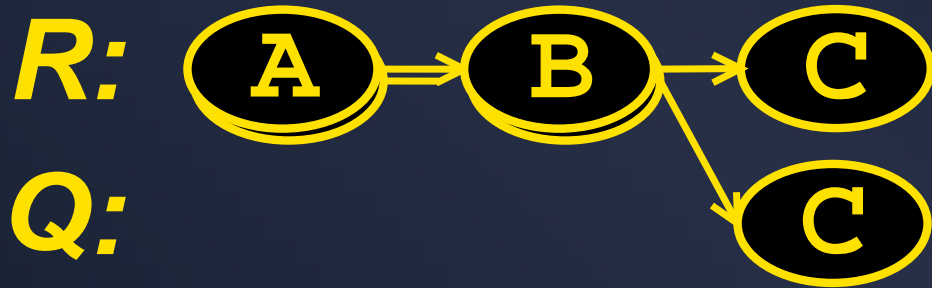
# Session Sharing Algorithm

- Sub-graph for complete overlap

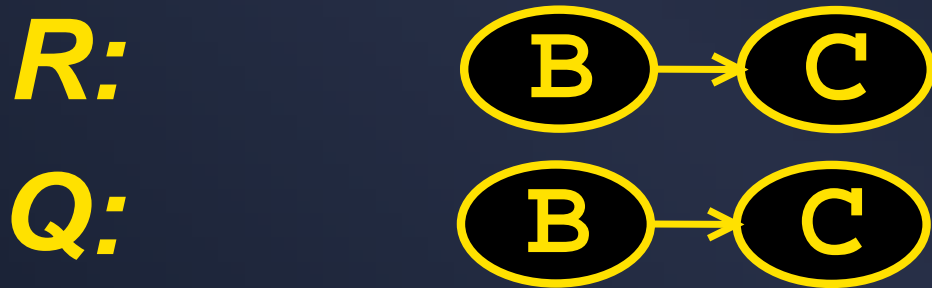


# Session Sharing Algorithm

- Sub-graph for complete overlap



- Sub-graph for complete overlap



# Session Sharing Algorithm

- Sub-graph for complete overlap



- Sub-graph for copy or output overlap

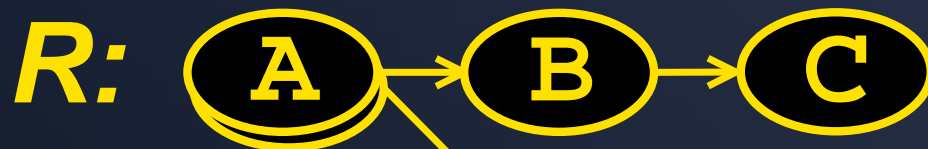


# Session Sharing Algorithm

- Sub-graph for complete overlap



- Sub-graph for copy or output overlap



# Session Sharing Algorithm

- Sub-graph for complete overlap



- Sub-graph for copy or output overlap



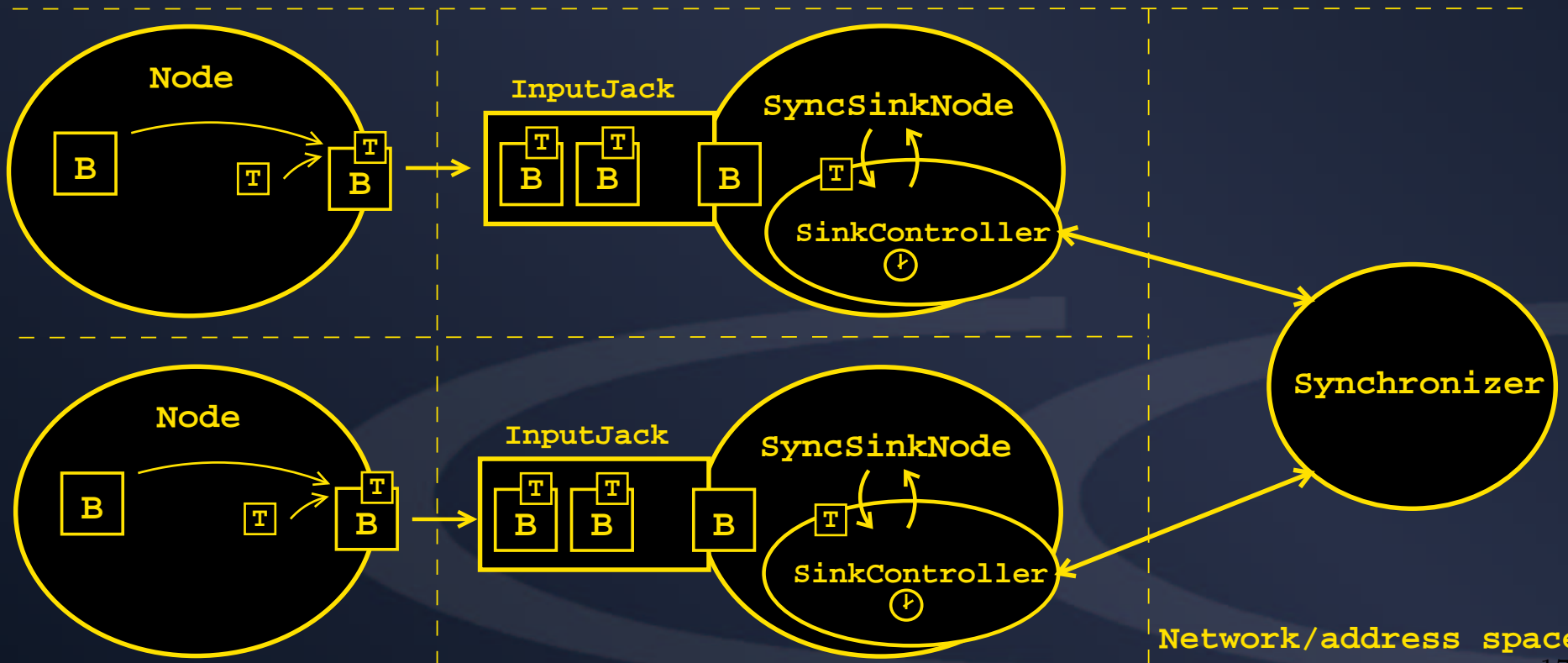


# Session Sharing Algorithm

- Why exhaustive depth-first search ?
  - Run-time
$$\mathcal{O}((|n_{max}^q| * |n_{max}^r| * |e_{max}^q| * |e_{max}^r|)^{depth_{max}})$$
  - 'Best' solution demands completeness
  - Possibility for 'dead end' after complete overlap
  - Number of recursions is very small due to strict criteria and cutting of search space
    - E.g. the jack tag is unique and has to match
  - Allows to apply different value functions later on
- Runtime around 100 ms for typical setups
- Value functions counts overlapped nodes and edges

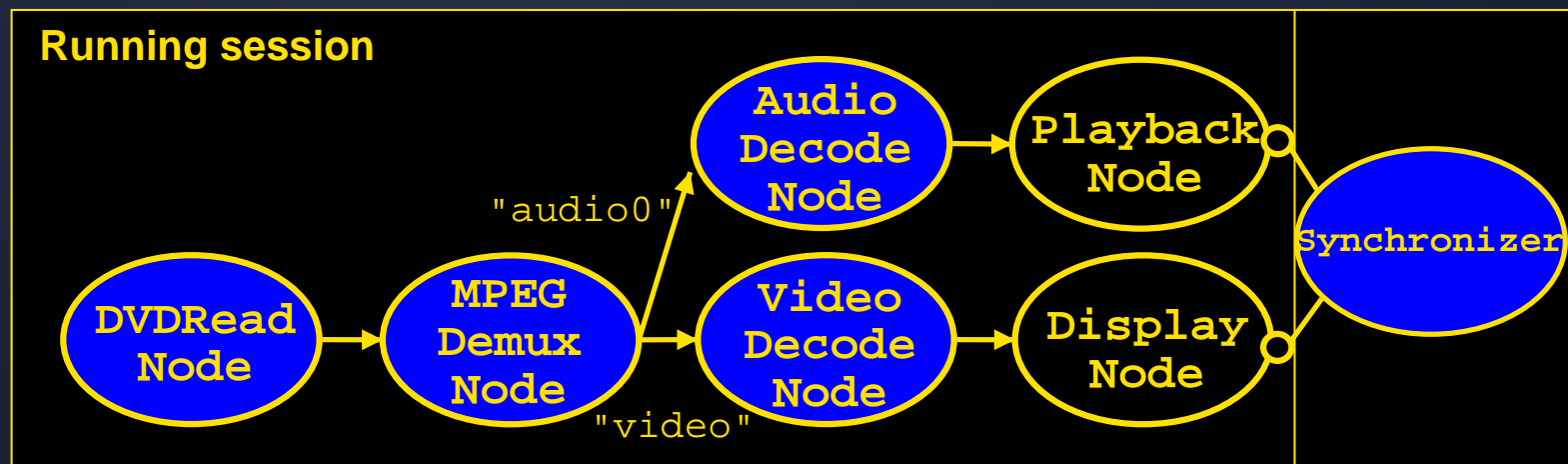
# Distributed Synchronization

- Strict separation of intra-stream and inter-stream synchronization
  - Locally running controllers
  - (Shared) Synchronizer adjusts latencies



# Application Scenario

- Shared access to DVD, different audio tracks for mobile devices (e.g. iPAQ H3870 with Linux)

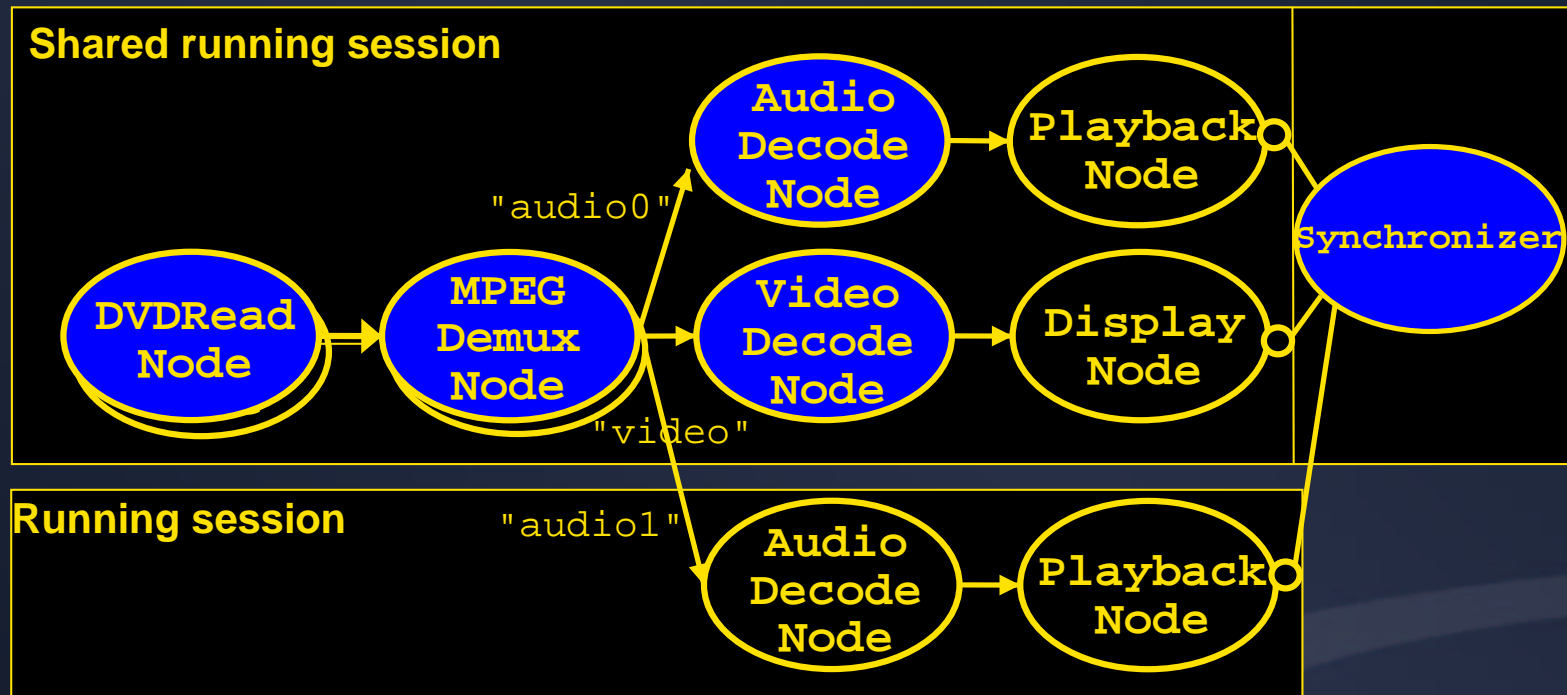


## Query



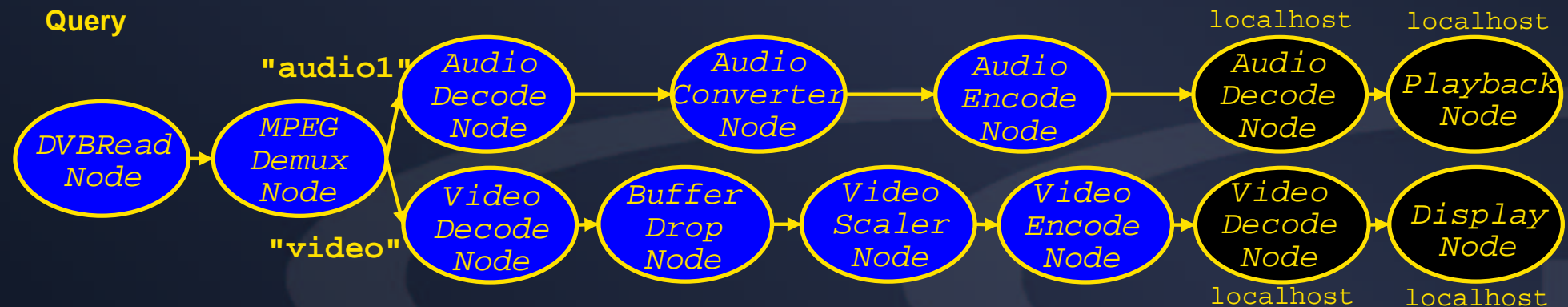
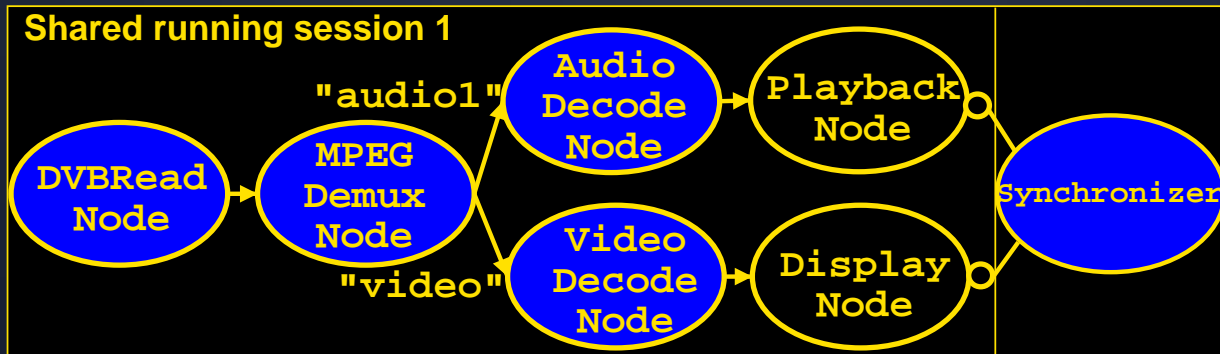
# Application Scenario

- Shared access to DVD, different audio tracks for mobile devices (e.g. iPAQ H3870 with Linux)



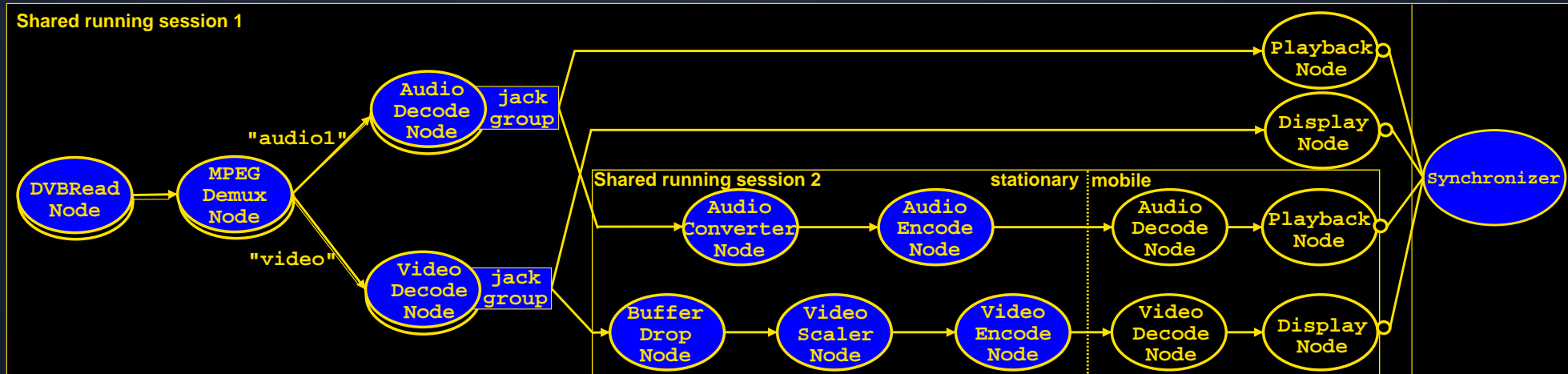
# Application Scenario

- Shared access to TV, stationary and mobile device
  - Transcoding needed for mobile device



# Application Scenario

- Shared access to TV, stationary and mobile device
  - Transcoding needed for mobile device



- iPAQ H3870: 11 kHz mono audio, 320x240 at 8-10fps
- Second mobile device can re-use large sub-graph

# Conclusions

- Extended registry service
  - Administrates running flow graphs (sessions)
- Session sharing as middleware service
  - Automatic sharing of (parts of) active flow graphs
  - Eliminates the problem of multiple access of hardware devices
  - Reduces the processing requirements through re-usage
- Distributed synchronization
- Application scenarios
  - Collaborative multimedia access

# Future Work

- Usage of multicast networking for shared sessions
- Migration of parts of sessions during runtime
- Quality of Service measurement to improve value function
- Evaluation of different application scenarios
- User interfaces to select shared sessions

# Questions?