A Standard for Multimedia Middleware (The PREMO Standard)

David Duke

University of York

Ivan Herman

CWI, Amsterdam

PREMO and MM today

- MM programming environments exist, but
  - diverse features
  - fragmented specifications
  - emphasis on simple media (e.g., audio, video)
- MM and graphics
  - little integration
  - separate communities

PREMO and MM tomorrow

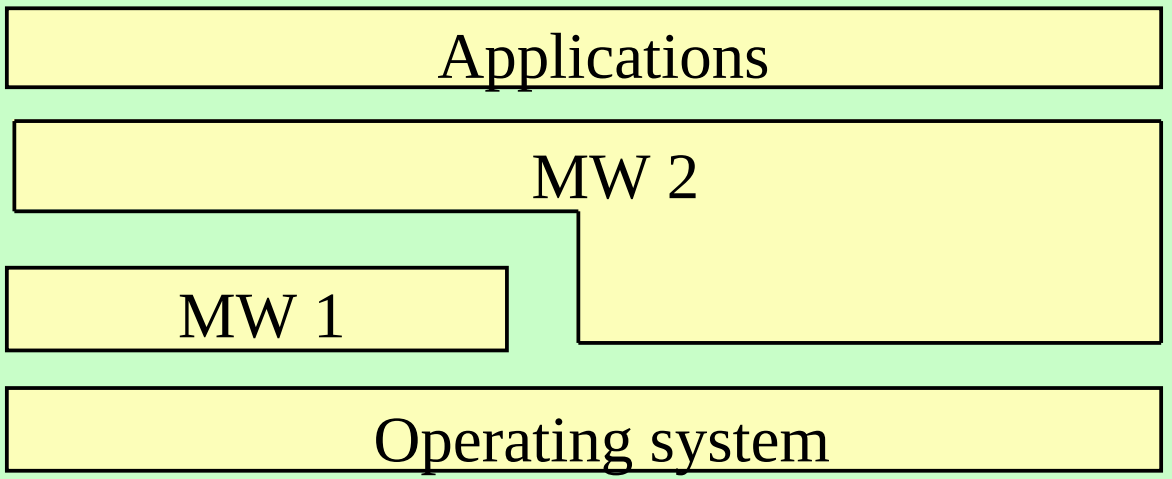
- Control over configurations
  - various media formats
  - adaptation of media to available resources
- Interoperability
  - cross–platform
  - cross–product
- Distribution

What is PREMO?

- "Programming Environments for Multimedia Objects" is a new ISO/IEC Standard
- Published as IS in spring 1998 (ISO 14478)
- Created by ISO/IEC JTC1/SC24 (Computer Graphics and Image Processing)
- Developed in cooperation with the IMA (International Multimedia Association)

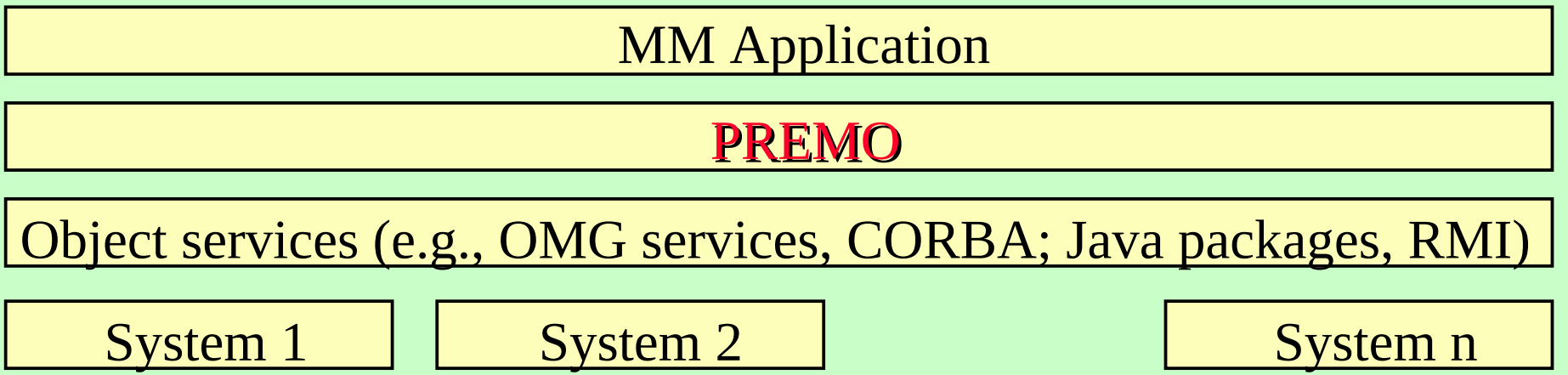
Concept of “middleware”

- Middleware: a layer between the operating system and the applications. Its role is to:
  - ensure interoperability of systems
  - ensure interoperability of programmers (...)
  - relieve application developers from local specificities
  - allow for an optimal and dedicated software development



PREMO as middleware

- Control over distributed multimedia objects
- Tools for synchronisation, configuration, adaptability
- Common platform for MM programming tools



PREMO as a framework

- PREMO does not specify
  - new media formats
  - new explicit rendering algorithms
  - explicit media/graphics primitive hierarchy
- PREMO offers ways to “plug–in” existing approaches
- PREMO facilitates application level co–operation

PREMO as reference model

- PREMO provides unifying concepts for MM programming (“portability of programmers”)
- PREMO organizes significant concepts into one coherent framework
- PREMO deliberately spells out details to make the general concepts clear

PREMO Document

- Cca. 300 page document in 4 Parts:
  - Part 1: Object model, fundamentals
  - Part 2: Foundation objects
  - Part 3: Multimedia System Services
  - Part 4: Modelling, Rendering, and Interaction
- Specifications for cca. 50-60 objects (interface, behaviour, properties)

Part 1: Object Model

- Traditional object model
  - objects, object types, object references
  - multiple inheritance
  - non–object datatypes
- Specialized features
  - activity of objects
  - operation request modes
    - synchronous;
    - asynchronous; or
    - sampled

Part 1: Environment

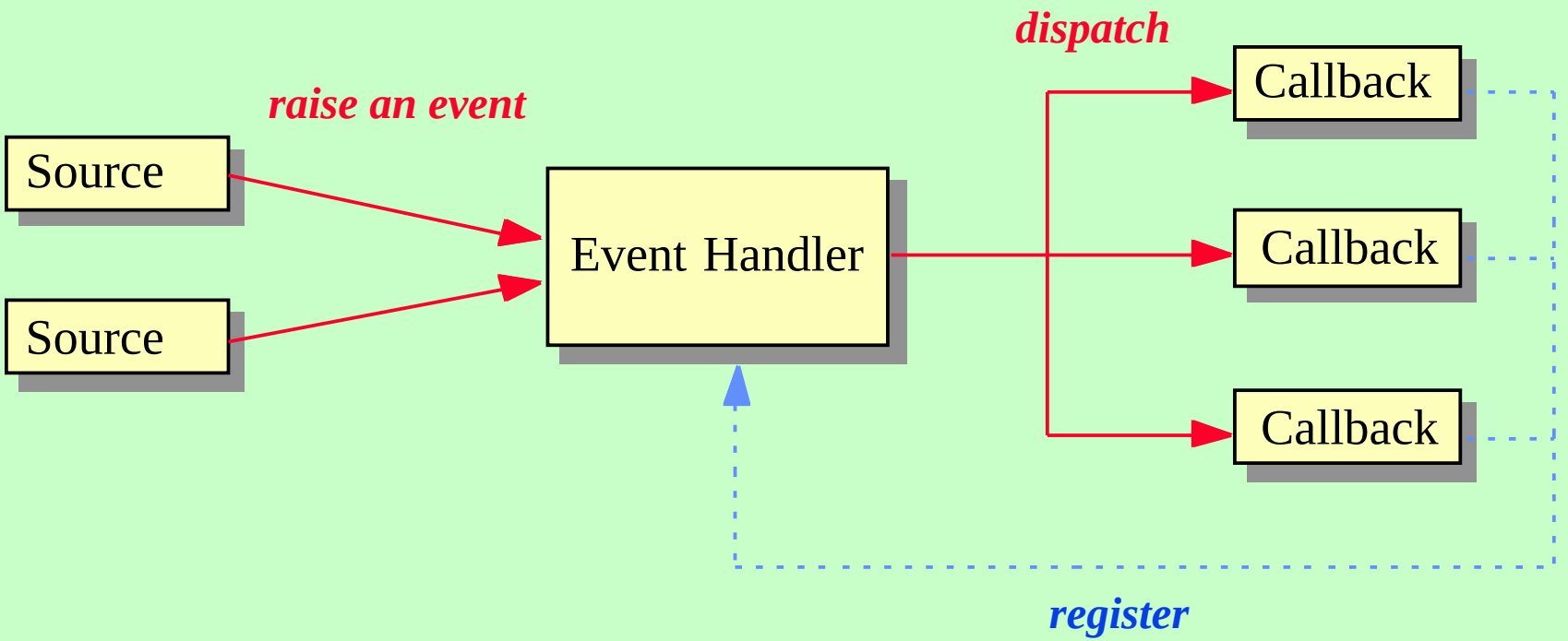
- External constraints
  - available programming languages (C++, Ada95, Java, etc.)
  - available distribution tools (OMG/CORBA, Java RMI, DCOM)
- “Environment” requirements:
  - object creation and destruction
  - object life–cycle
  - object references and garbage collection
  - casting

Part 2: Foundation objects

"Top" of the PREMO object hierarchy; defines interfaces for

- data structures
- general finite state machine objects
- event management
- clock/timer access
- general synchronisation facilities
- property control
- object factories

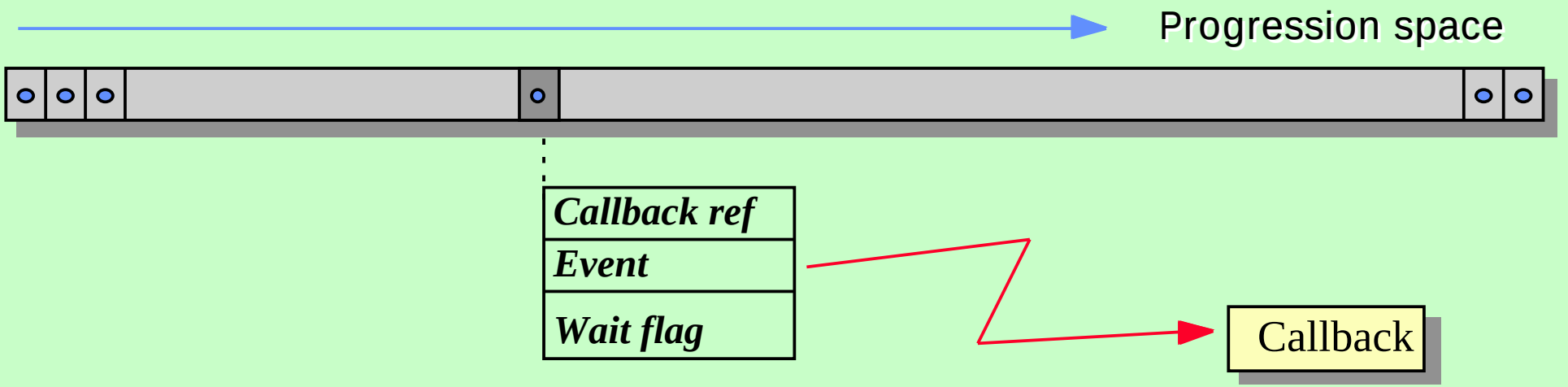
Event management



- Event handler is also a callback, i.e., chains can be constructed!

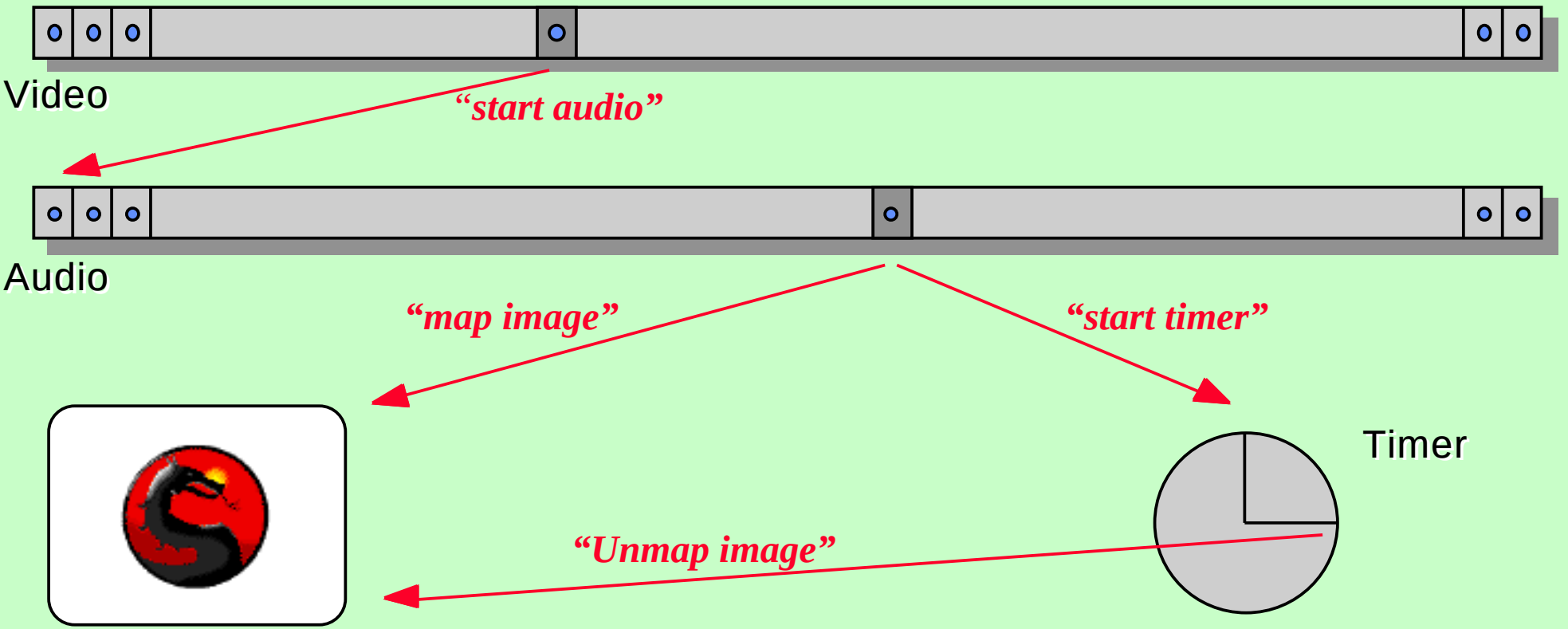
Event-based synchronisation

Synchronizable Object:

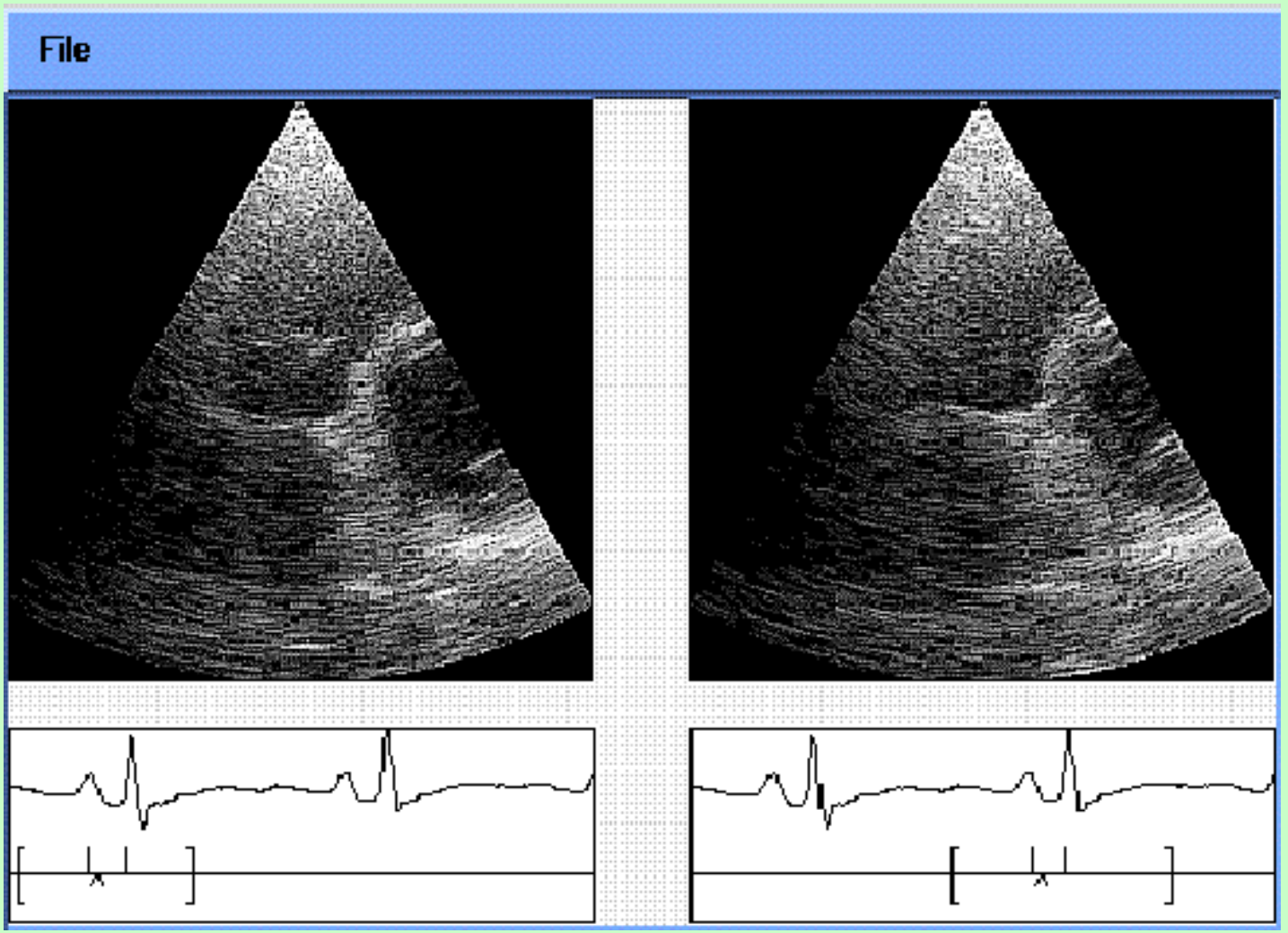


- Progression space can be integer, float, time
- Object is a finite state machine (STARTED, STOPPED, PAUSED, WAITING)
- “Presentation” and “progression” is abstract

(Very) simple example

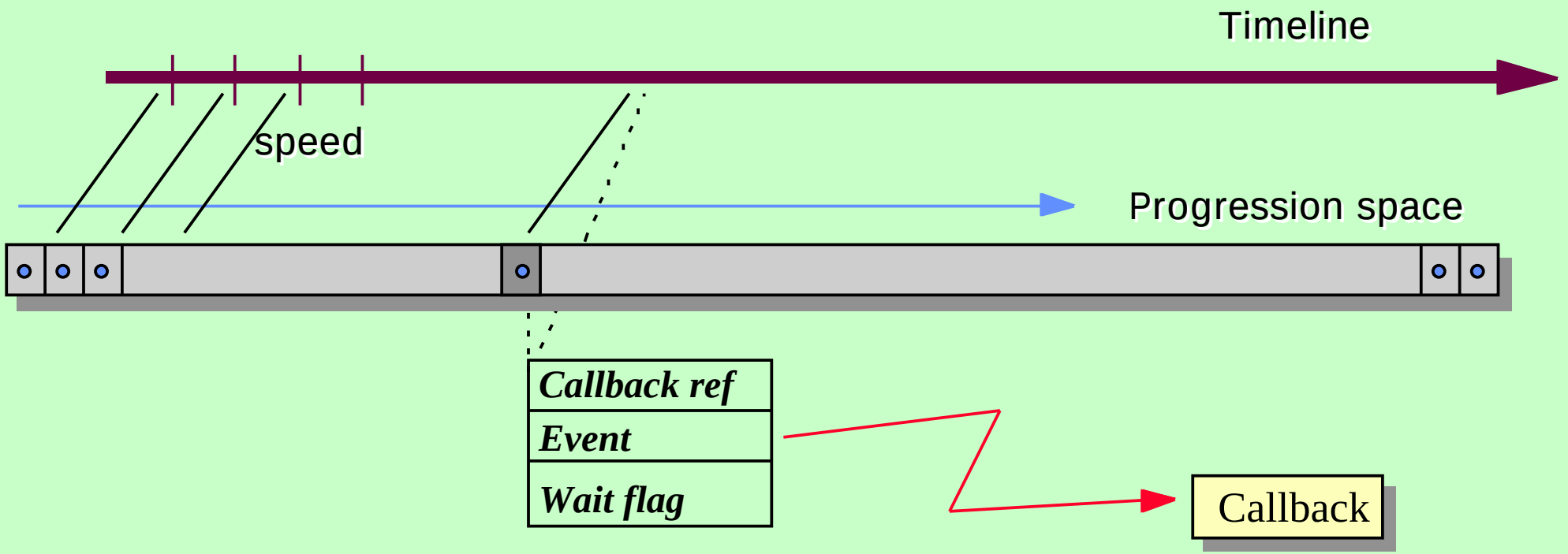


Why not time?



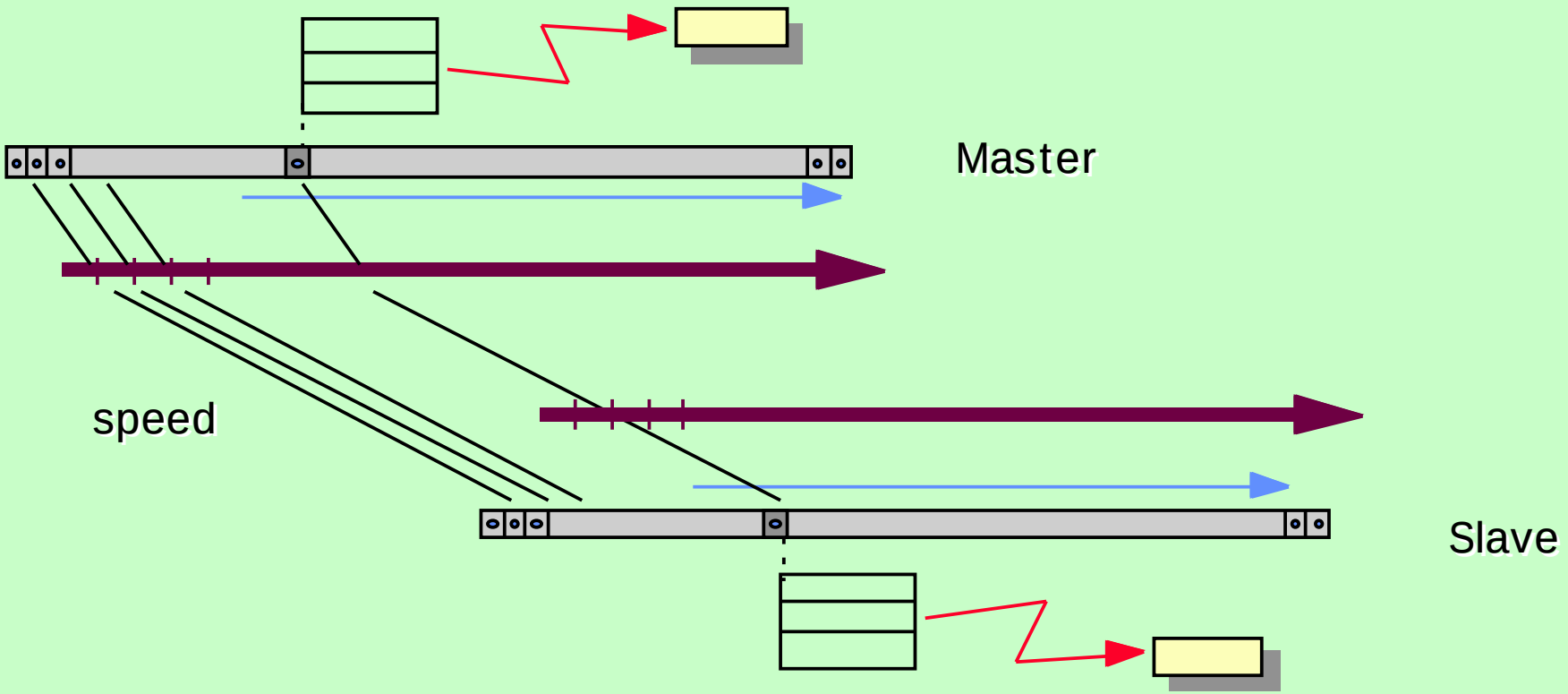
Synchronisation with time

Time Synchronizable Object:



Reference points can be set both in “time” and “space”

Time slaves and masters

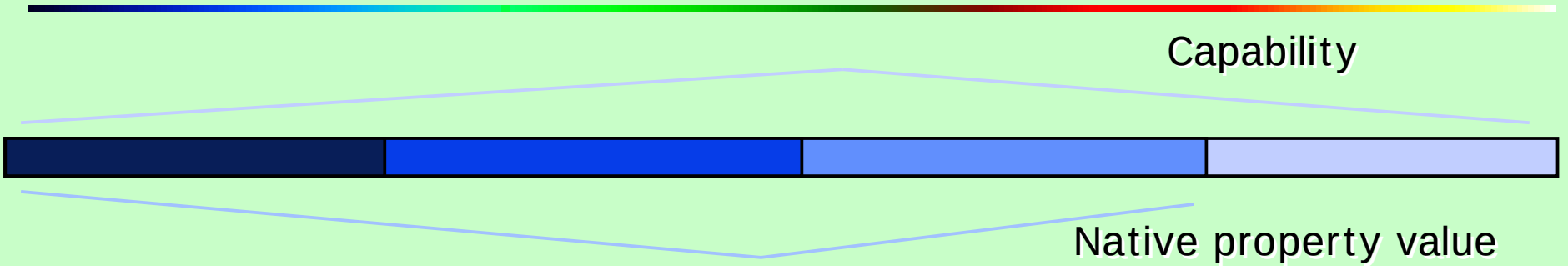


Slave measures the discrepancy between its own clock and the master’s

Property management

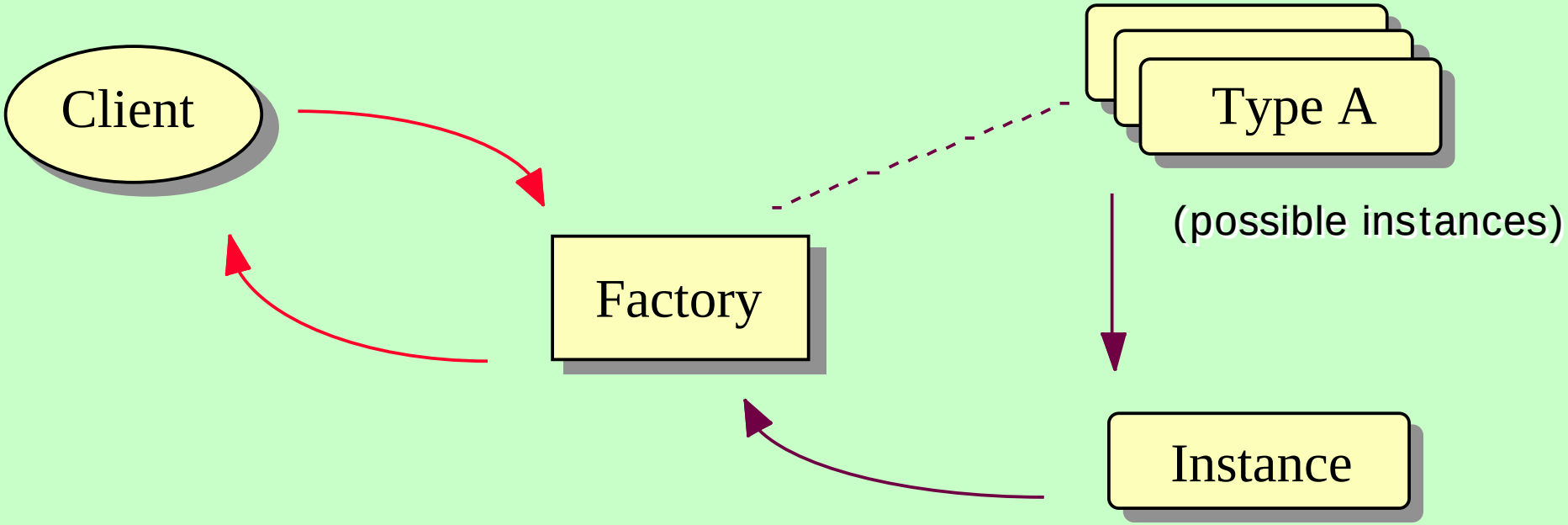
- Property: a key–value pair dynamically attached to an object
  - “dynamic attribute”
  - bypasses typing constraints
- Some objects have pre–defined attributes
- Possible values of a property may be inquired
- Properties can be constrained
- Properties management is a major tool for dynamic configuration

Property constraining



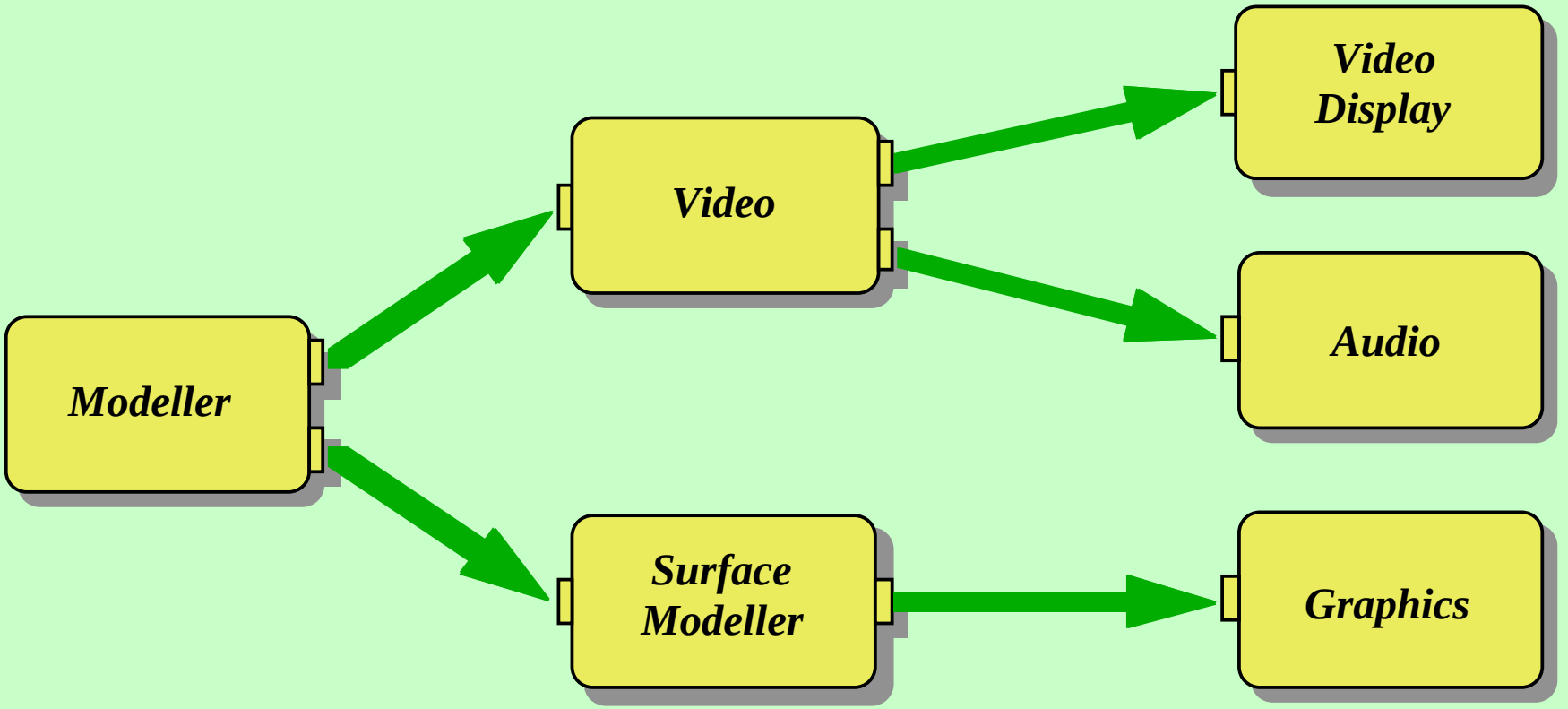
- Capability: possible values for a type
  (all possible audio formats for this type)
- Native property value: possible values for an instance
  (all possible audio formats for this instance)
- Constrained value: client selects among possible values
  (I want only these and these audio formats)
- Selected value: object selects optimal value
  (object selects optimal audio formats)

Creation through Object factories



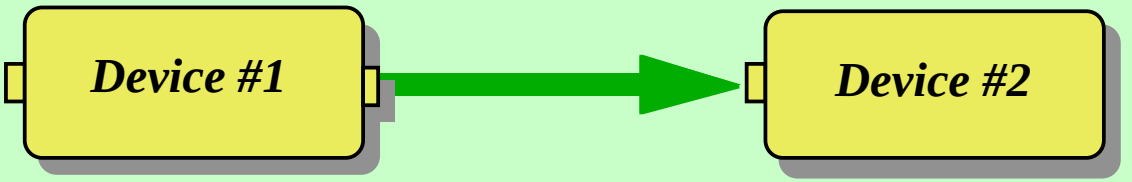
Example: “create an object which can manage AVI and MPEG, and runs on this and this IP address”

Part 3: Multimedia Systems Services



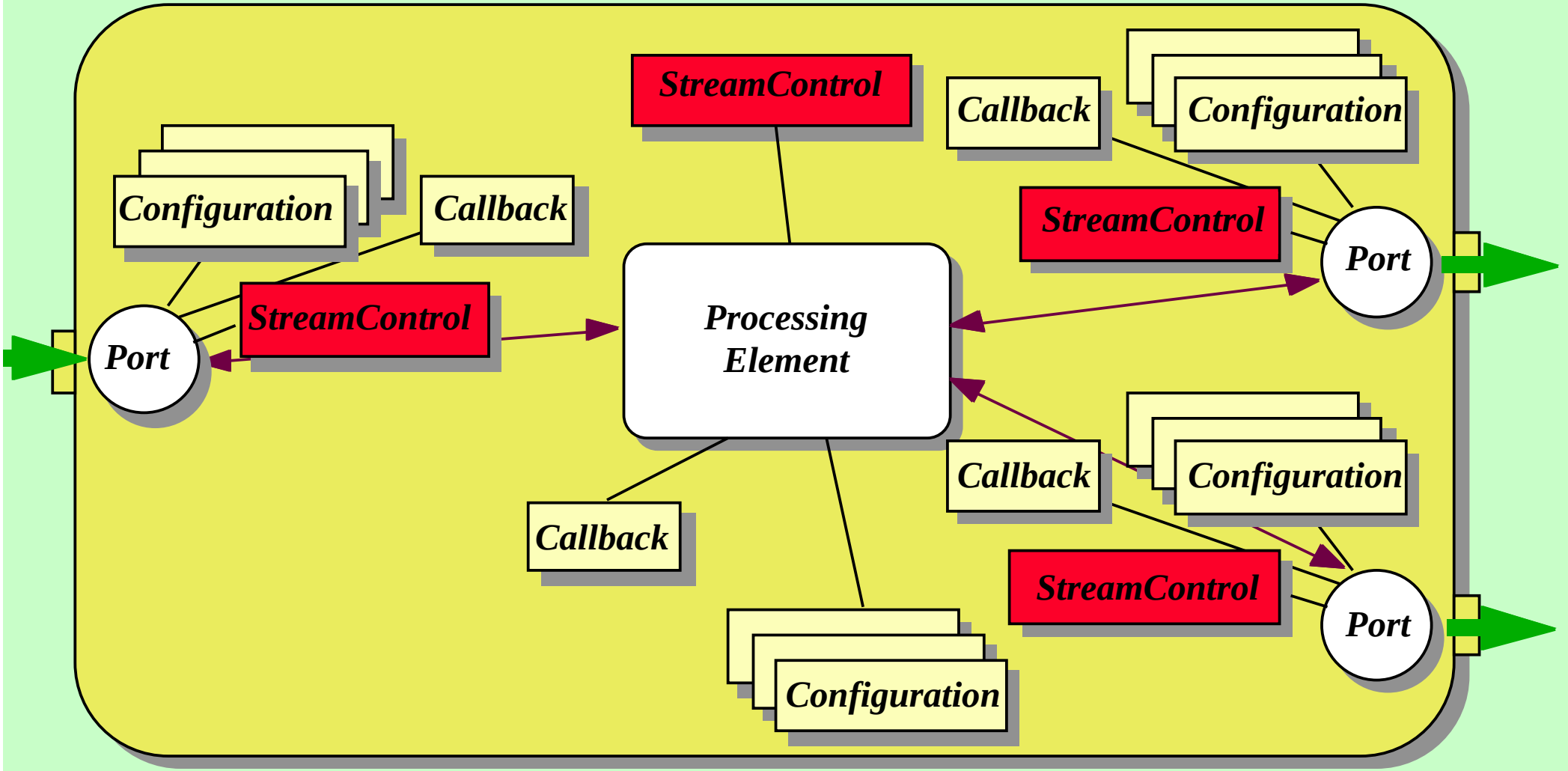
Multimedia System ≡ dataflow network of media devices connected through media streams

Characterisation of devices and streams

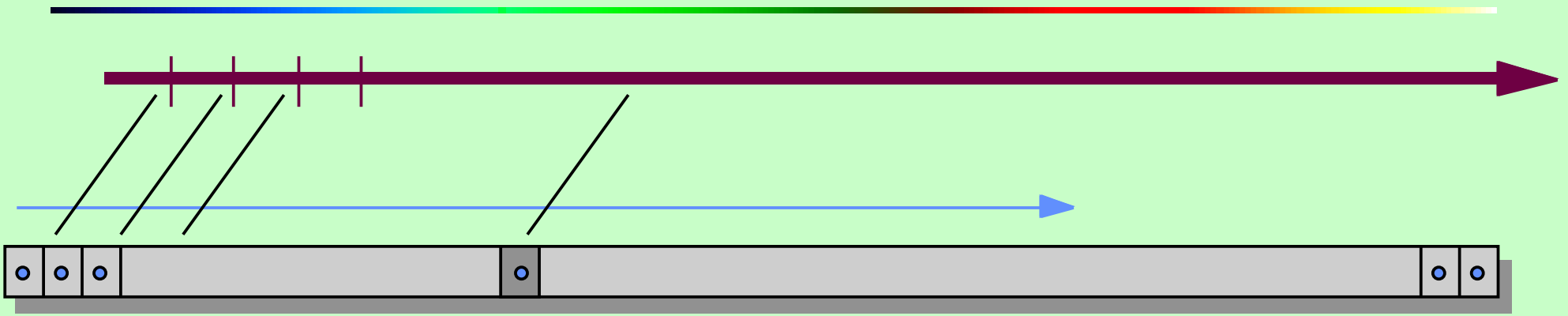


- Devices are (conceptually) distributed
- Devices are configurable
- Devices receive/send data through ports
- Devices are oblivious to whom they are connected
- Media data are synchronised on ports
- Media streams are opaque

Virtual device



StreamControl object



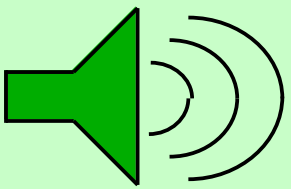
presentation may be:

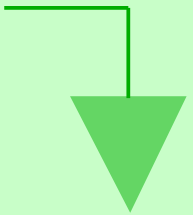
play

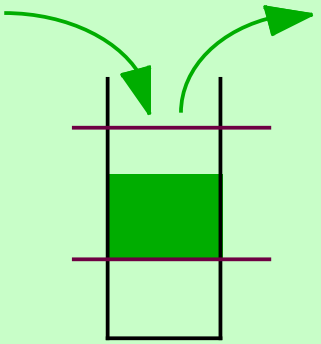
mute

prime

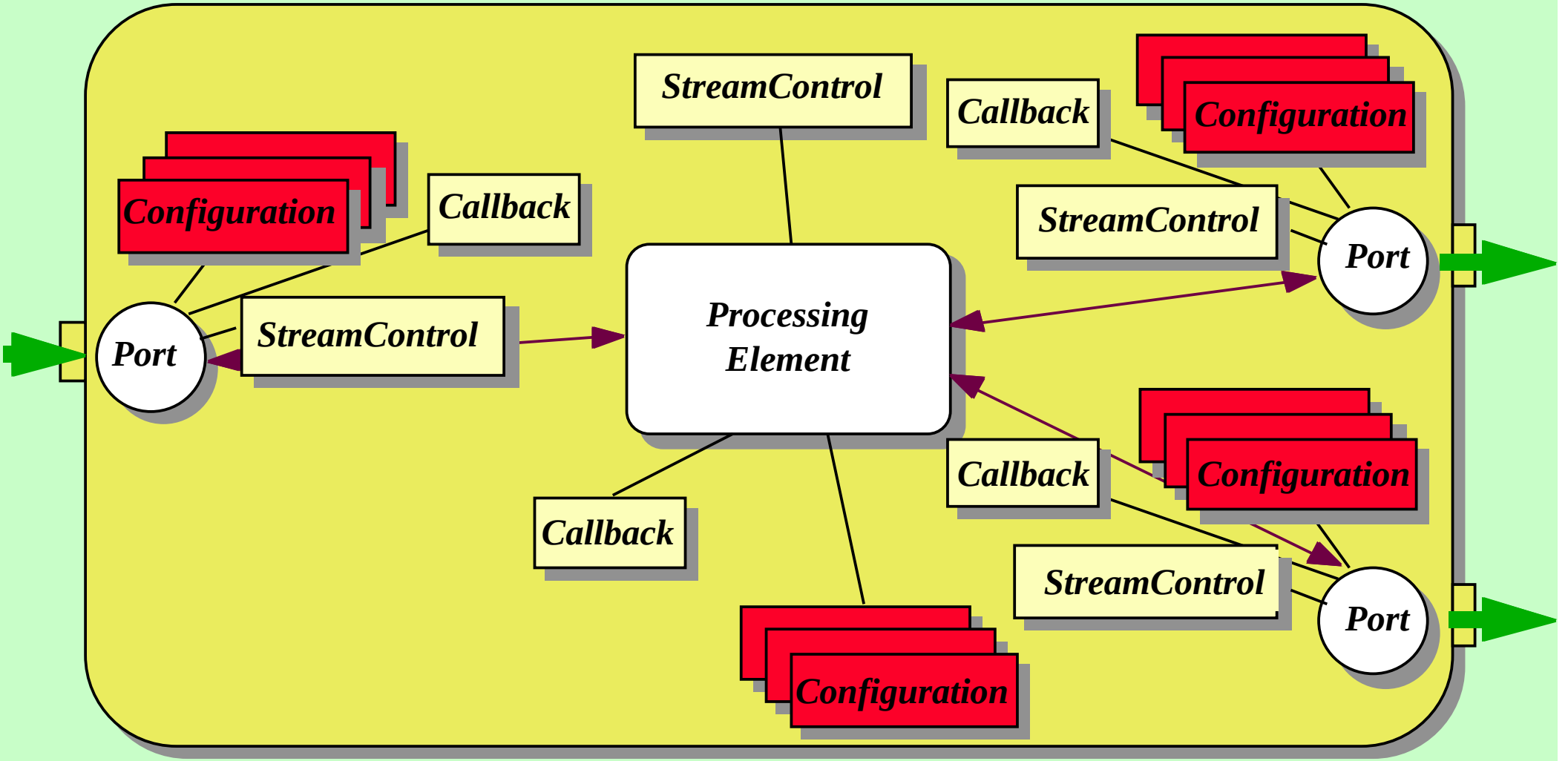
drain







Configuration objects



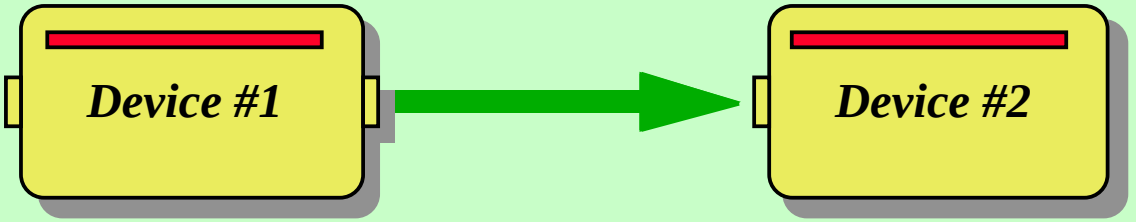
Configuration objects

- **Format objects:** describe media formats
  - **Examples:** MPEGVideoFormat, CATVFormat
  - **Properties:** IntraQMatrix, SampleRate, etc.
- **Multimedia Transport Protocol:** describe media independent communication protocol
  - **Examples:** IntraNodeConnection, InterNodeConnection; TCP, ATM, NETBIOS
  - **Properties:** ByteOrder
- **Quality of Service Descriptor:** describe QoS requirements
  - **Properties:** GuaranteedLevel, Jitter, BandwidthBounds

Configuration example

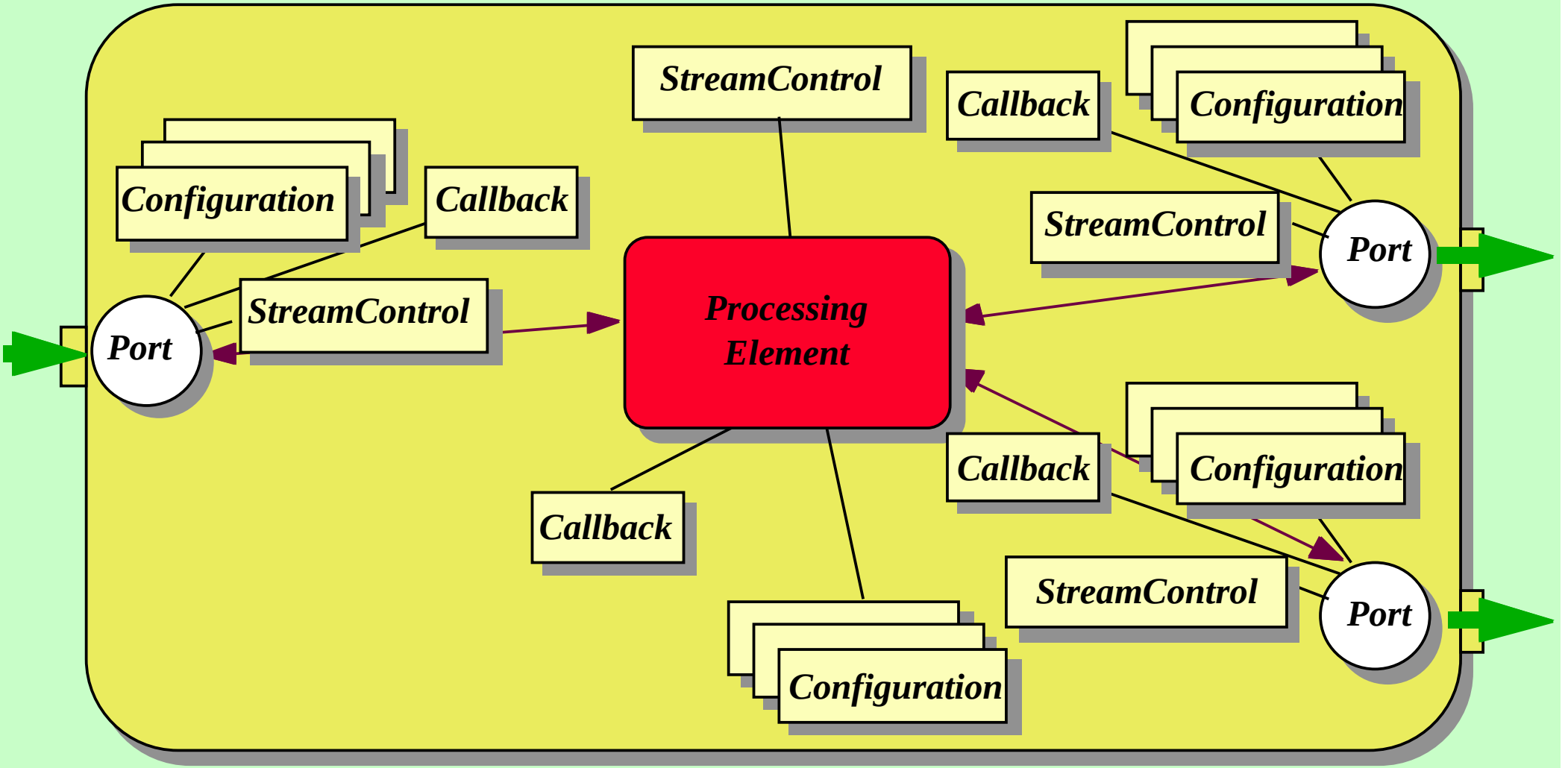
- Get an image device with PNG, GIF, JPEG, TIFF, or XPM image formats
  - done by object factory
- Retrieve lists of available formats on device instance
  - e.g., PNGFormat and GIFFormat are returned
- Set client's chosen format on a port
  - set GIFFormat as configuration object on a port
- More fine–grained configuration on the Format instance

Configuration example (cont.)

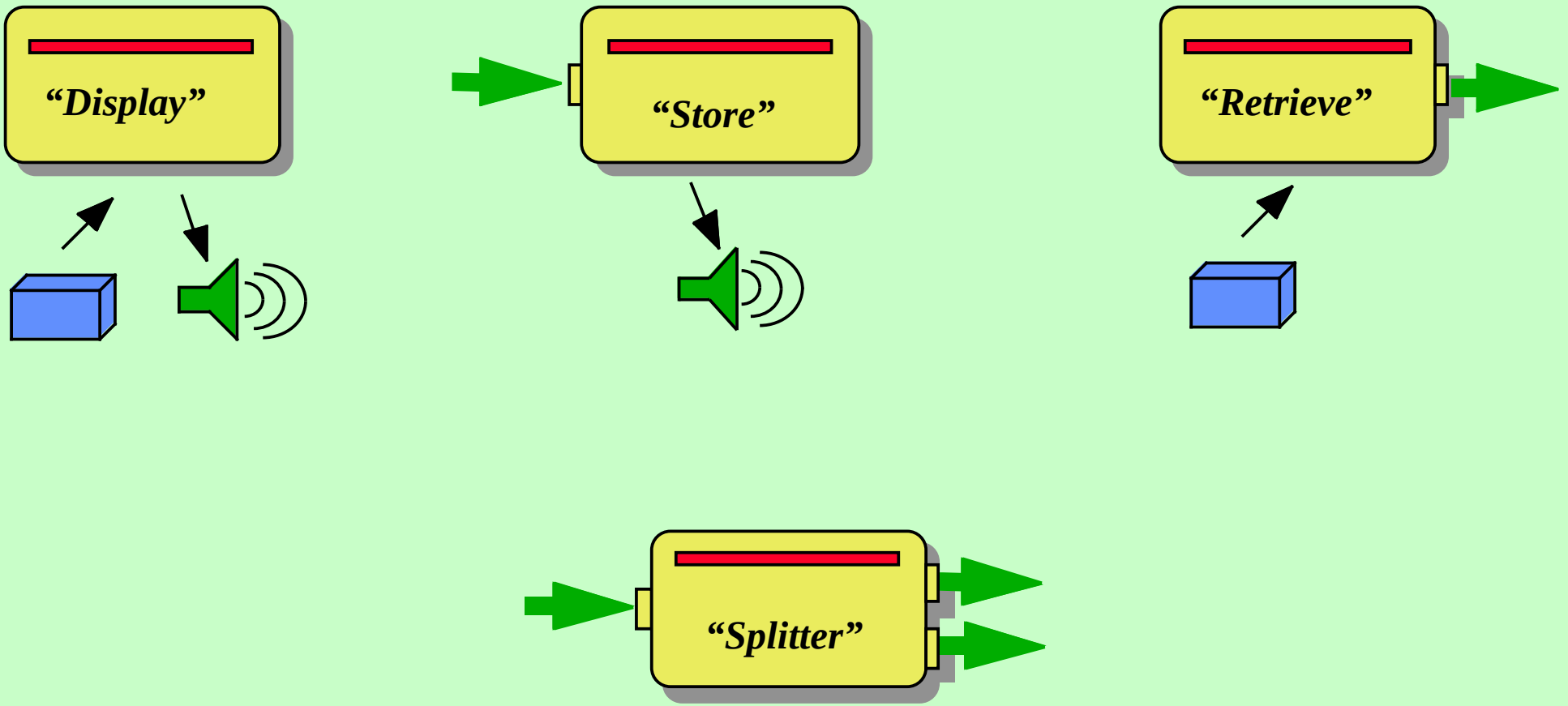


- Look at properties of the Format on port #1
  GIF version property might be “87” and “89”
- Make Format object select optimal value
  sets, say, 87
- Assign property for the Format on port #2
- Do the same with other properties, and with QoSDescriptor and MSP objects

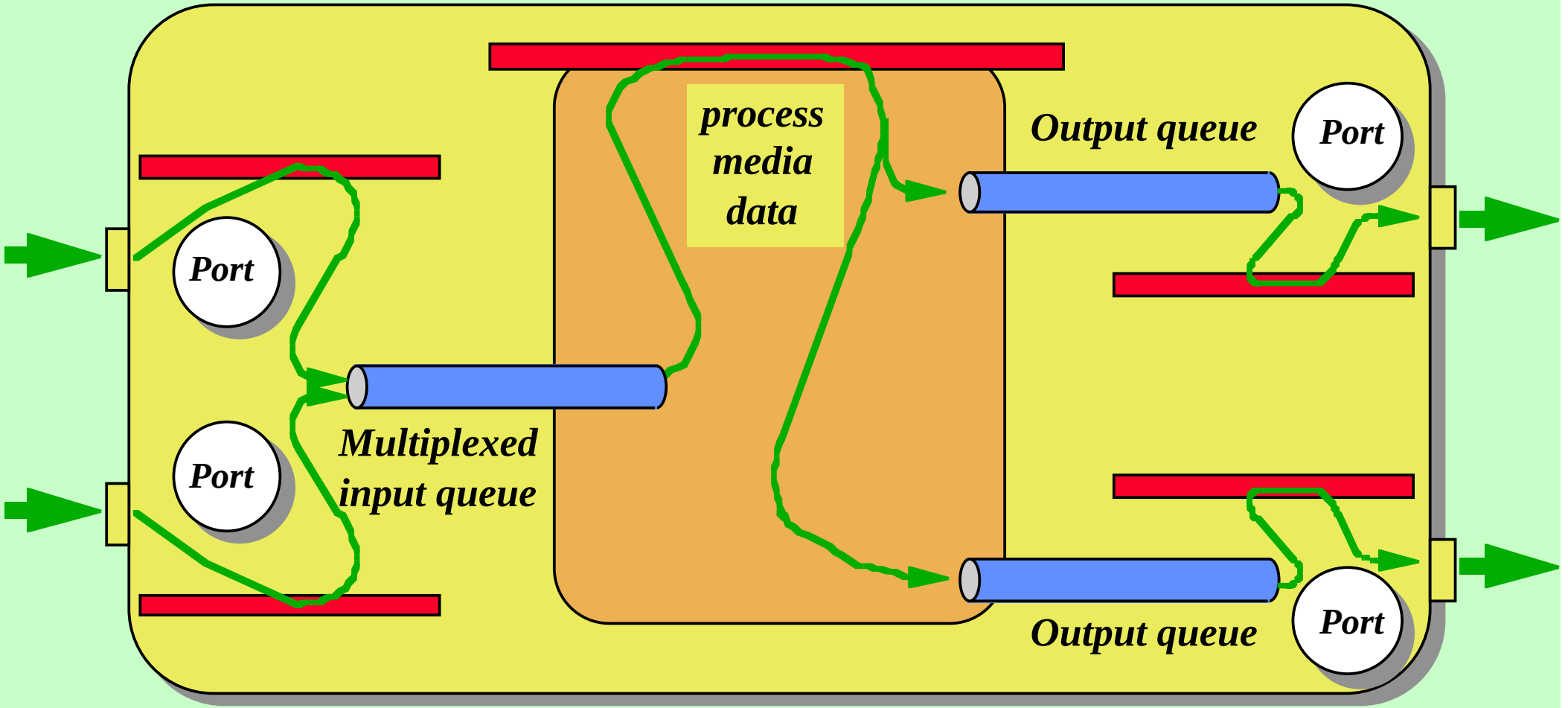
Processing Element



Device Examples



“Transformer” device

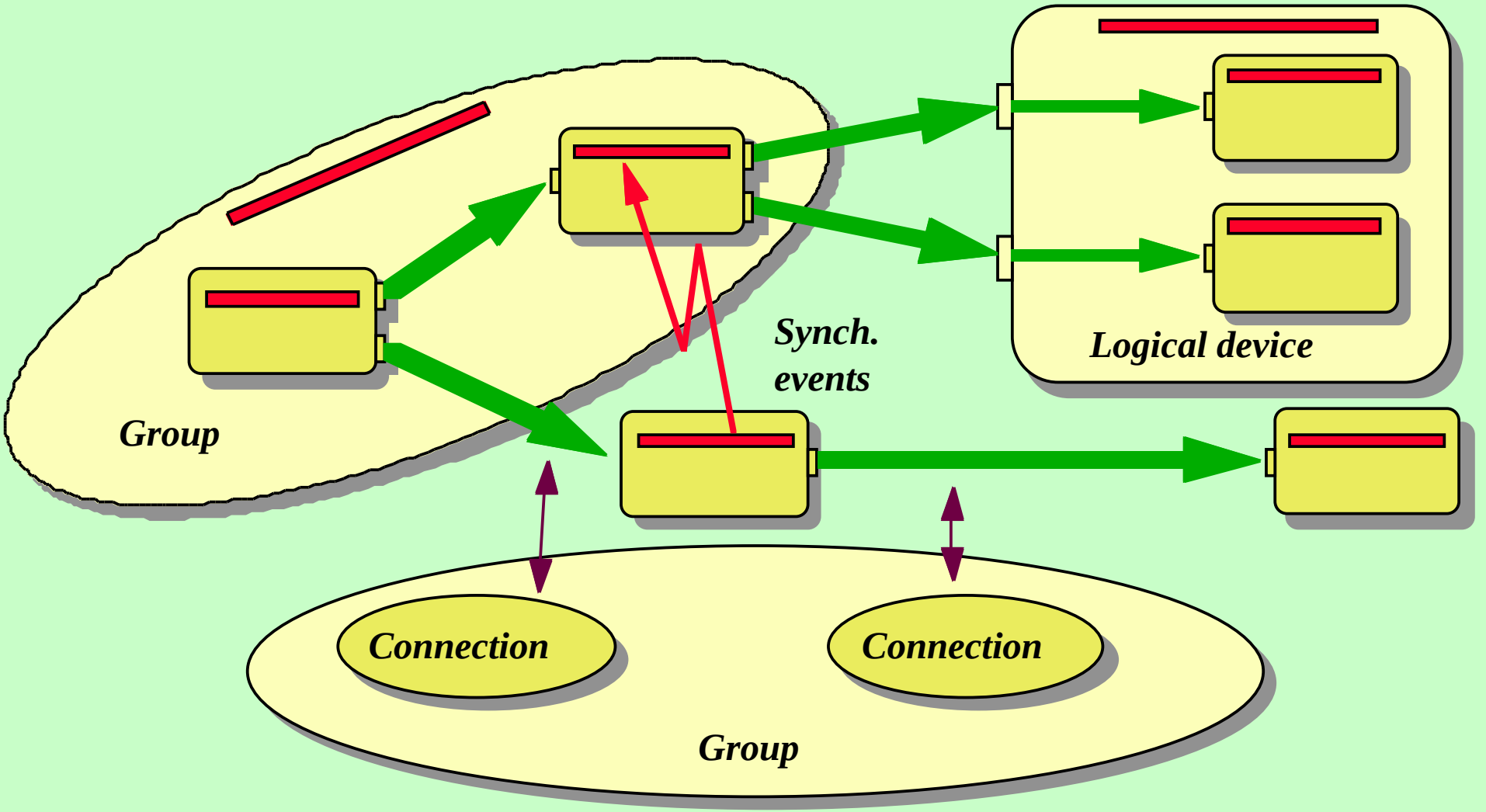


processing element ≡ global stream control thread

Network management objects

- **Connection objects:** set up an manage connection among devices
  - e.g., sets up a socket pair or a pipe among devices
- **Groups:** combines devices, connections, or other groups; controls start, stop, resource management for all constituents
  - e.g., controls a set of connections as one entry
- **Logical device:** connects a group of virtual devices and behaves like a device
  - hierarchies of devices can be constructed

Full network example



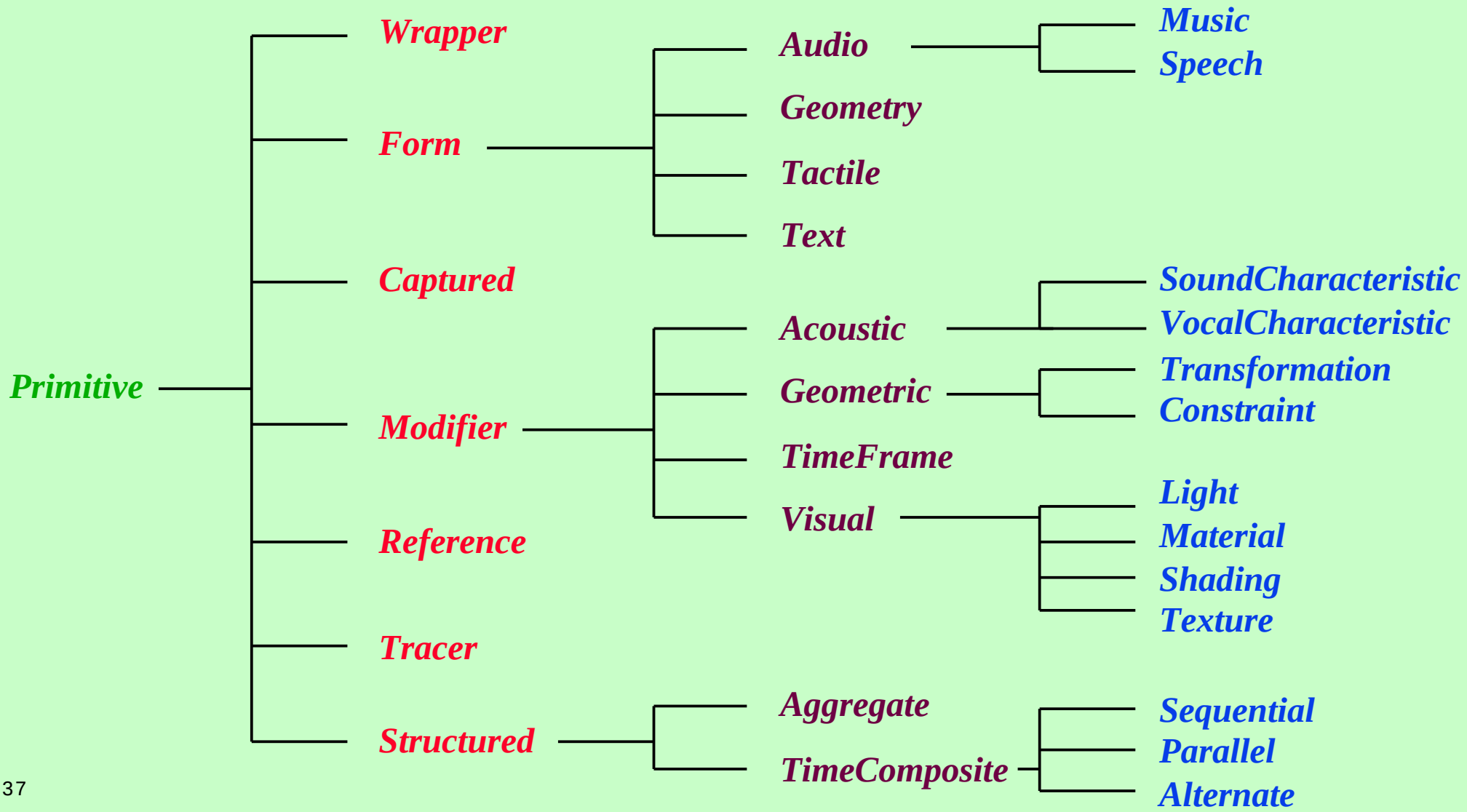
Part 4: Modelling, Rendering, and Interaction

- Defines framework for media stream content
  - MSS is indeendent of media stream content
  - declarative model of media primitives
- Defines collection of media-oriented devices
  - ‘interface’ between the graphics and MM worlds
  - generalised notions of modeller, renderer, etc.
- Provides for coordination of concurrent media
  - primitives and tools for hypermedia contents

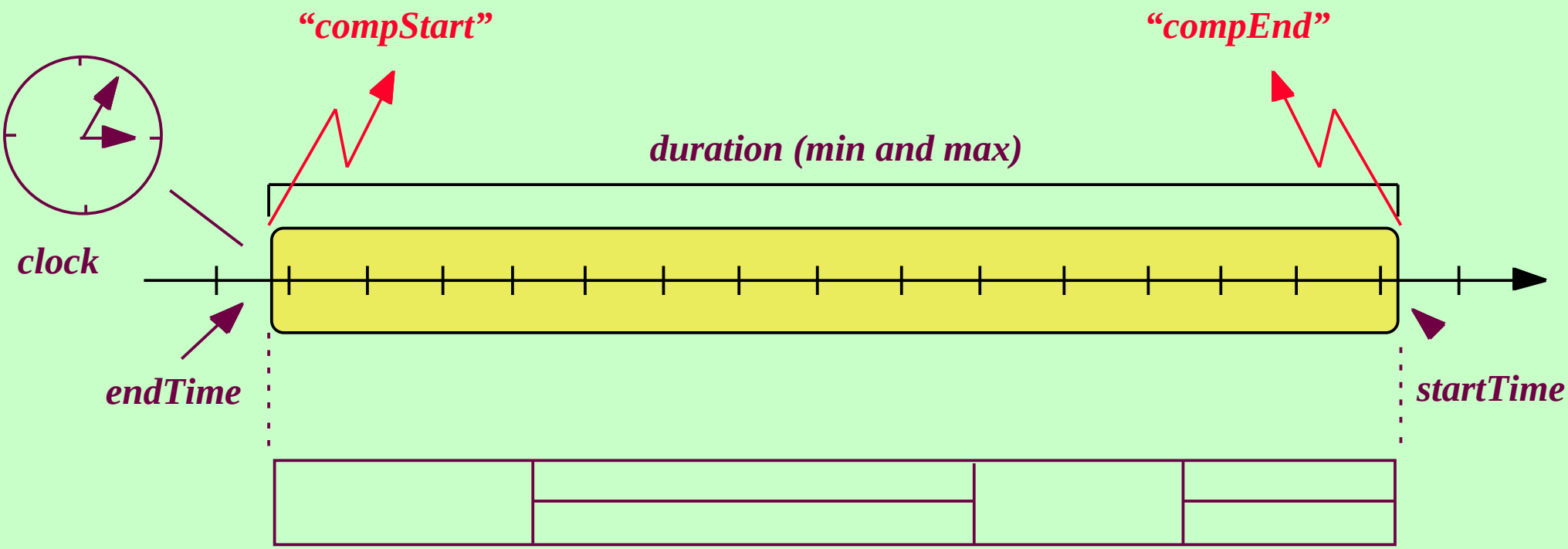
Primitives

- Design question: which primitive set is best?
  - None!
  - PREMO concerned with interoperation
  - abstract from renderer-specific details
- Specifies the content of the media streams
  - top level of a (rendering) primitive hierarchy
  - possibilities to describe the composition in time of other primitives
  - possibilities to combine with “captured” media

MRI Primitives



Time composite



- components: sequential, parallel, or alternate
- finishes between min and max. duration

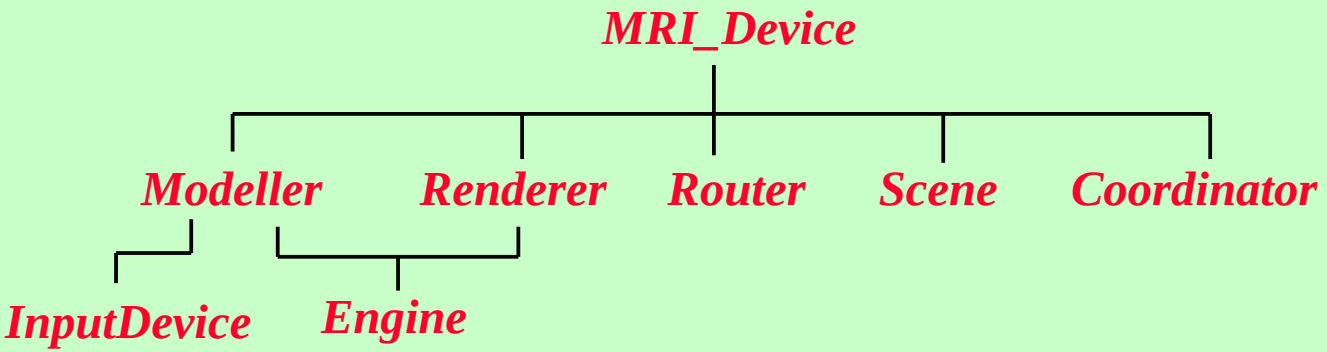
Time components

- **Sequential:** constituents are displayed sequentially
  - **attributes:** startDelta, endDelta
- **Parallel:** constituents are displayed in parallel
  - **attributes:** startSync, endSync
- **Alternate:** choice of constituent based on the state of an FSM
  - **attributes:** references to an FSM, state–to–primitive table

MRI Devices

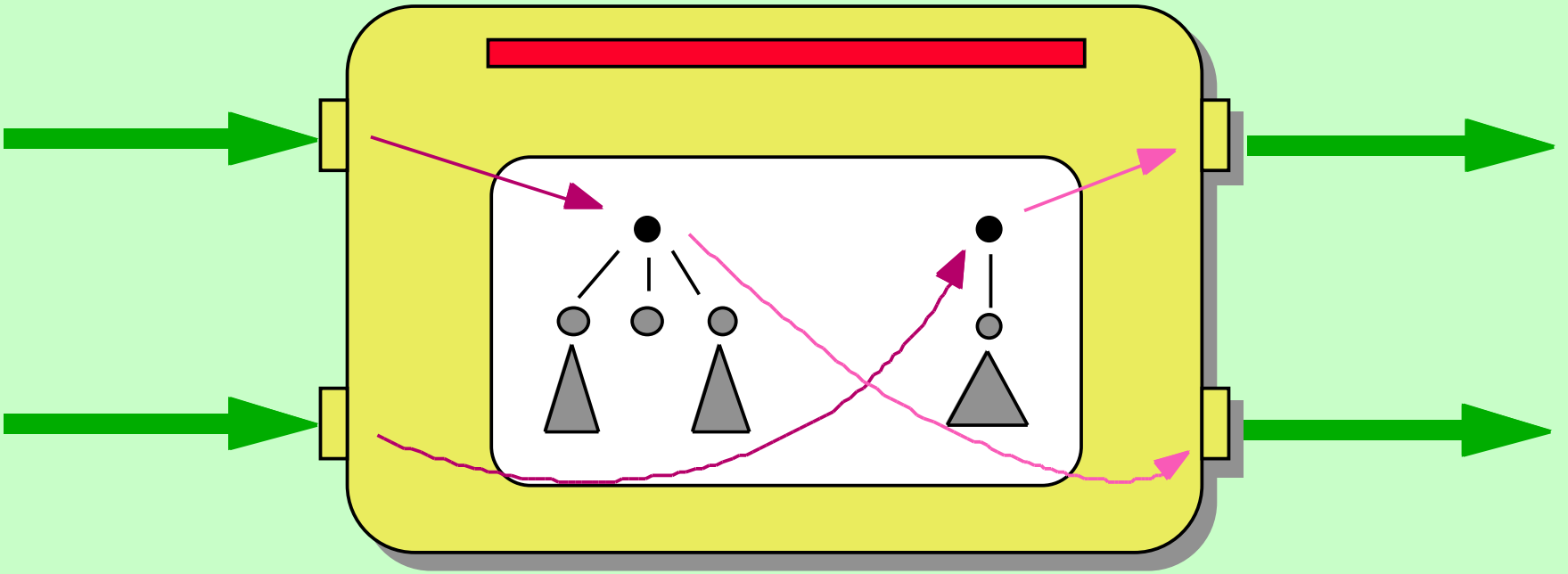
- Specialised virtual devices
  - understand MRI primitives
  - specify and negotiate processing capabilities for primitives
  - devices to build up complex scenes
  - devices to interpret the time composition of primitives
- Input devices
  - operation in sampled, event and request modes

MRI Devices



- **Modeller/Renderer:** output/input with MRI Primitives; **Engine** has both
- **InputDevice:** turn primitive data into objects
- **Router:** “switchboards” between input and output ports
- **Scene:** virtual database
- **Coordinator:** planner and scheduler for TimeComposite objects

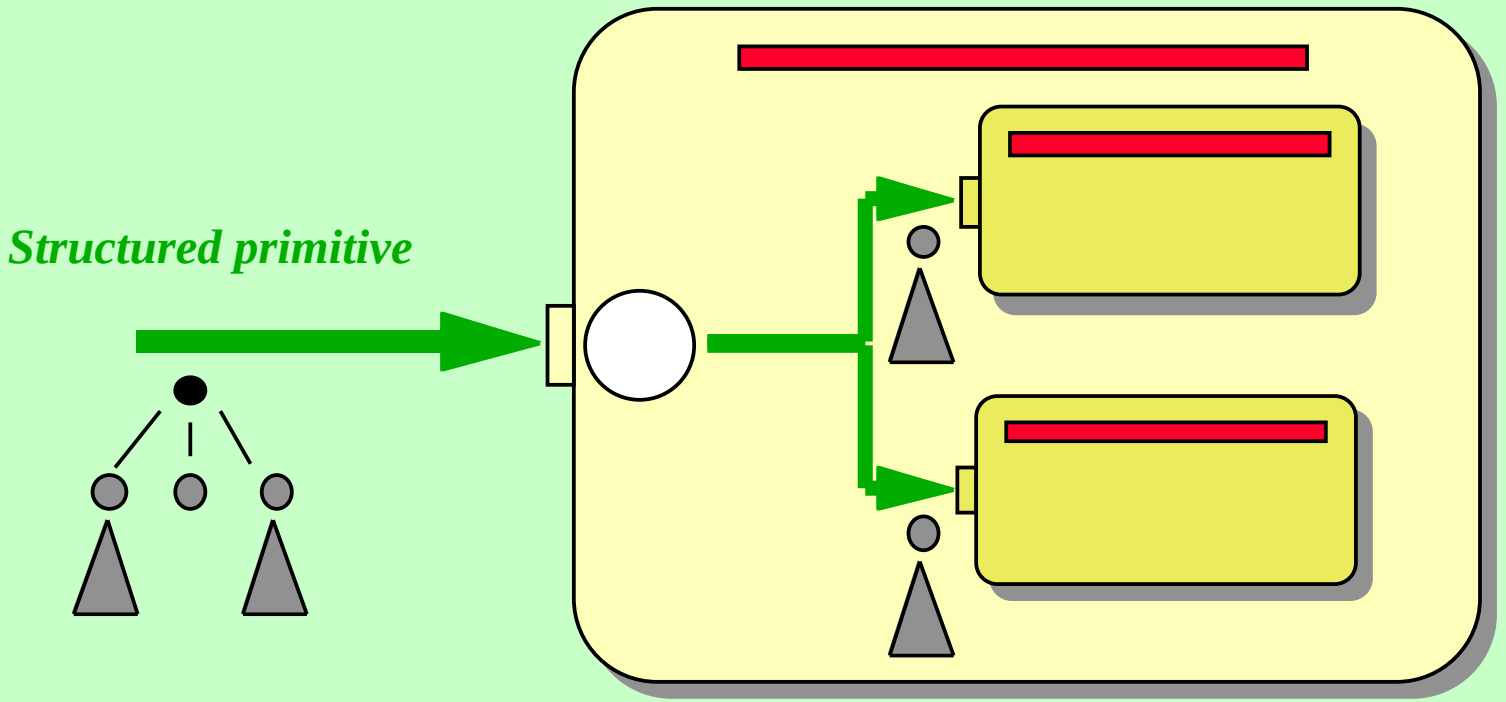
Scene



A “conceptual” database system

(e.g., element storage in PHIGS)

Coordinator

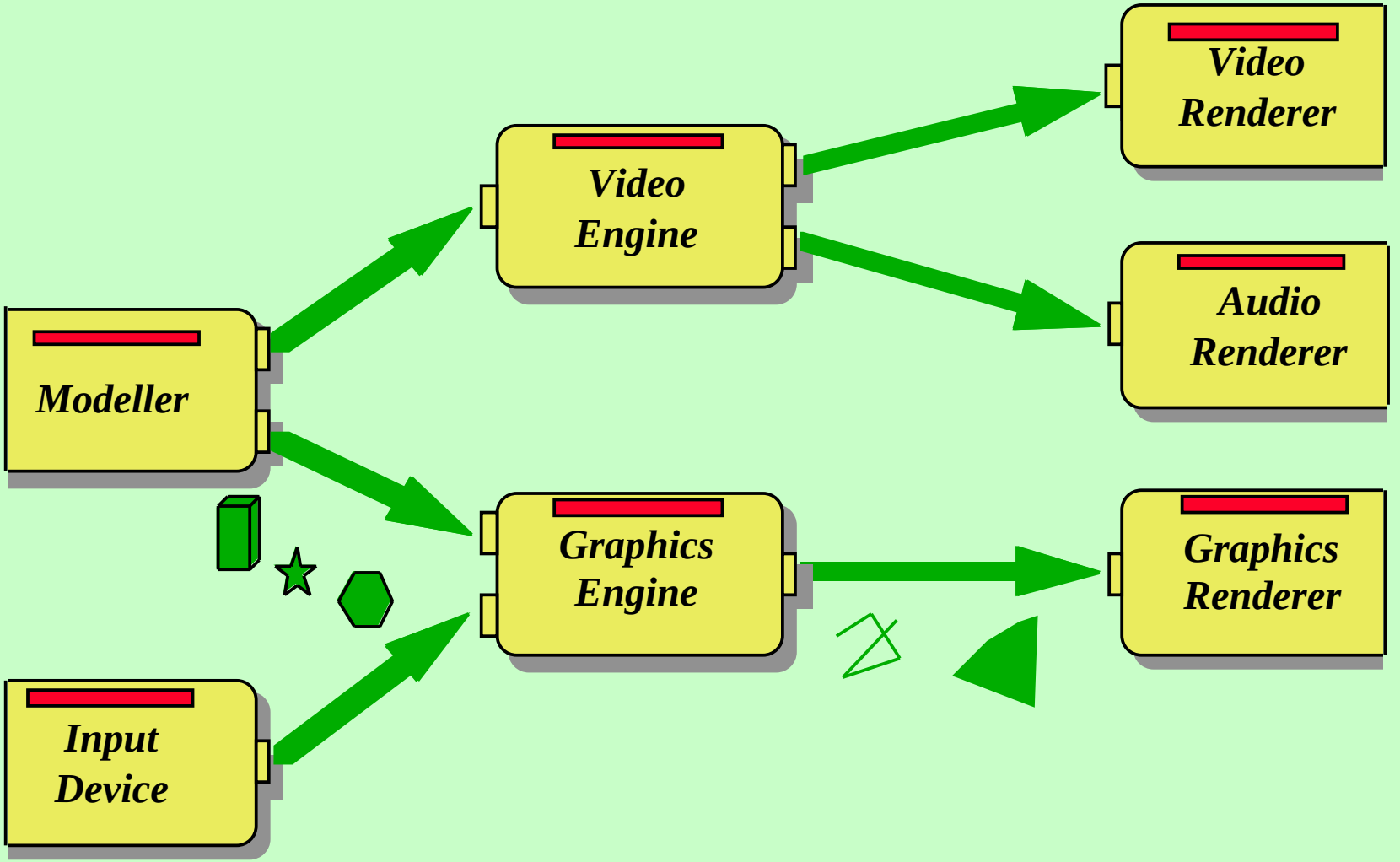


"Planner" and "Scheduler" of hypermedia presentations

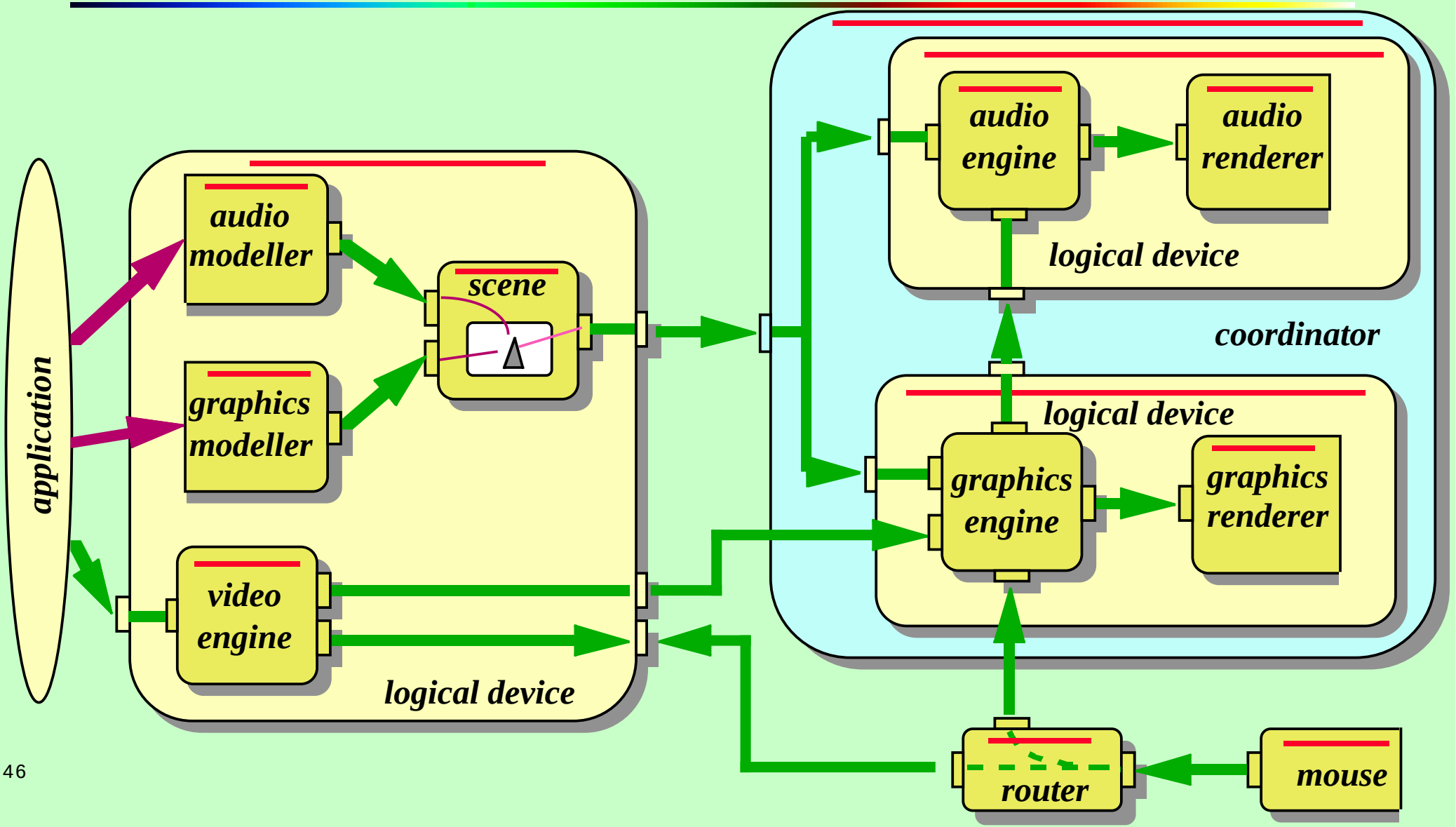
Coordination in Practice

- Allocate primitives to available processors
  - compare primitive type against port configuration
- Schedule primitives for presentation
  - layout primitives along “virtual tracks”
  - align primitive boundaries based on composite structure
- Monitor and adjust progess of presentation
  - periodic milestones on port StreamControl objects
  - inject “tracer” primitives
  - control over stream progression (stop, drain, etc.)

Simple MRI example



MRI example


application
audio modeller
graphics modeller
scene
video engine
logical device
audio engine
audio renderer
logical device
coordinator
logical device
graphics engine
graphics renderer
router
mouse

Miscellaneous

- Part of the Standard has been described through formal description tools (Object–Z, Lotos)
- A proof–of–concept implementation is under preparation in Java+RMI
- A request has been sent to ISO/IEC to put the document into public domain through the Internet