A Timing Service for Policy-Based Management Systems

Christoph Angerer – christoph.angerer@inf.ethz.ch
Thomas Gross – thomas.gross@inf.ethz.ch

Computer Systems Institute
ETH Zurich

Adaptive Policy-Based Management

- PBM Approach: Specifying Networks in terms of high-level (business) entities instead of low-level tech. features
- Adaptive PBM: Policies adapt to changes and new situations

Adaptive if:
Policy-activation not only based on fixed dates but takes *environmental and internal events* into account

Overview

1. Basic Idea: Separating Events from Times
2. Related Work: Multi-Media Time Models
3. A Model for Timing Specifications:
   * Clocks,
   * Scalar/Indefinite/Dependent Time,
   * Timing Specifications
4. Architecture and Integration of the Timing Service
5. Concluding Remarks

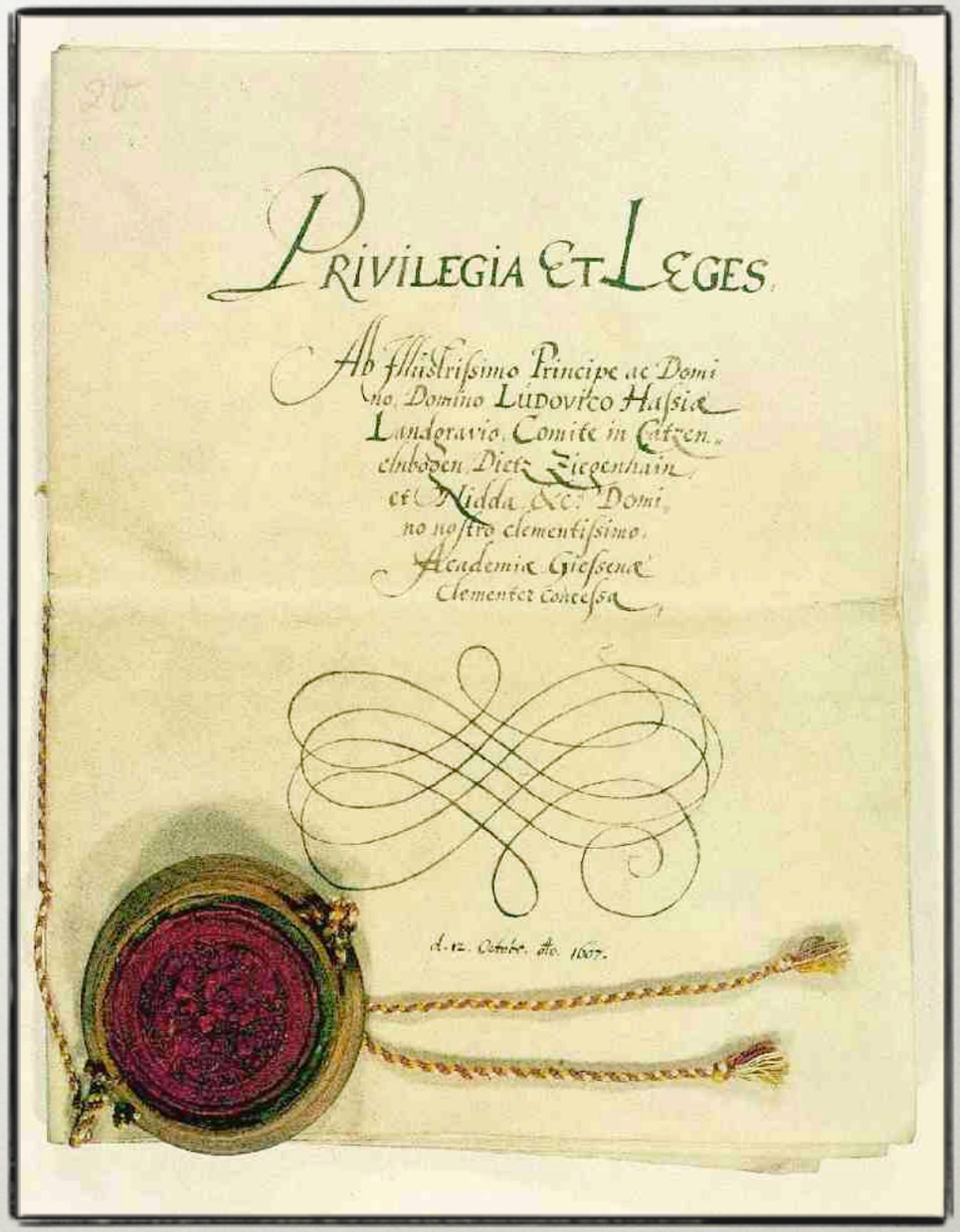
Basic Idea (1): Simple Policy

If: User is CEO

What: Application is "Streaming Video"

When: Time is between 9 a.m. and 11 a.m.

Then: User is entitled to a service level "Premium" with guaranteed throughput & latency



Basic Idea (2): Explicit Start and end Events

If: User is CEO

What: Application is "Streaming Video"

When: **Start:** 9 a.m. each
End: 11 a.m. each
} Interval

Then: User is entitled to a service level "Premium" with guaranteed throughput & latency

Basic Idea (3): Separating Events from Times

If: User is CEO

What: Application is "Streaming Video"

When:

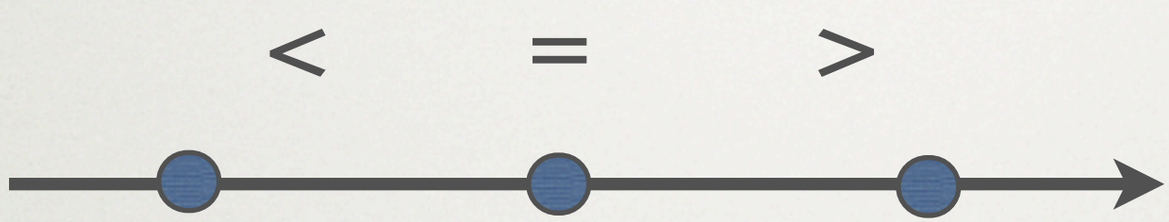
Event: Briefing starts → **Time:** 9 a.m. each day

Event: Briefing ends → **Time:** 11 a.m. each day

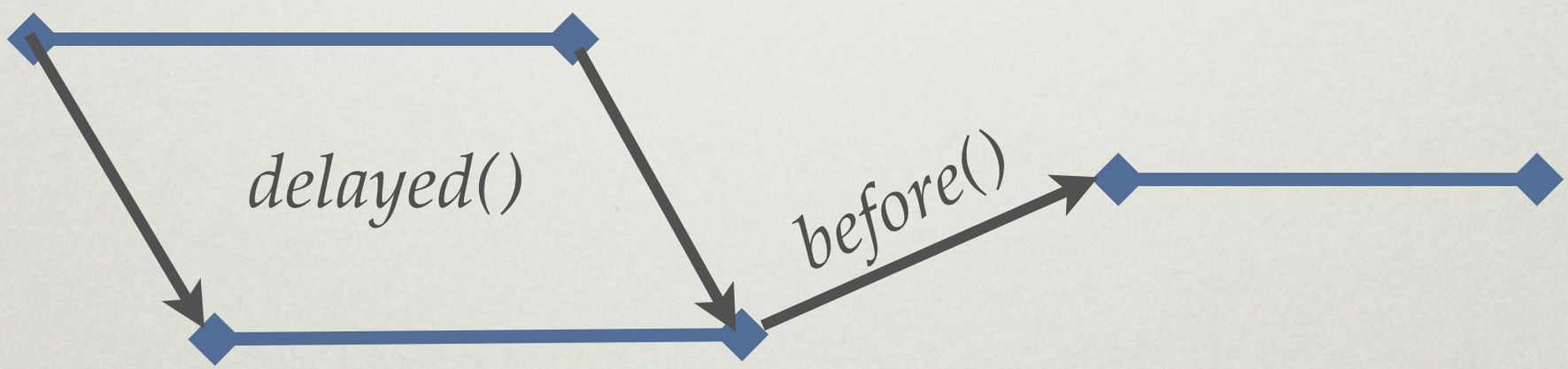
Then: User is entitled to a service level "Premium" with guaranteed throughput & latency

Related Work: Multi-Media Time Models

- Point-based

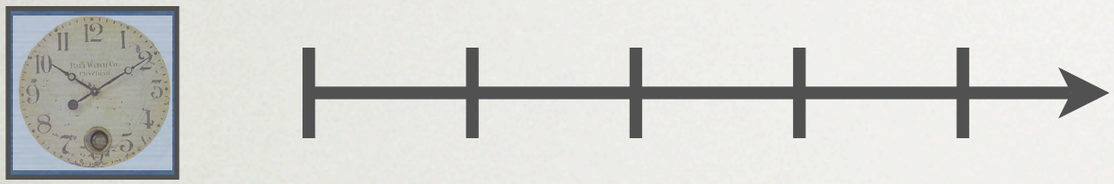


- Interval-based: 10 basic timing patterns, e.g.:

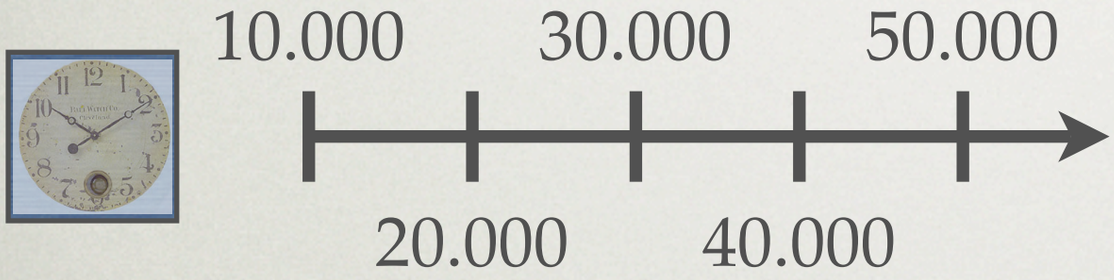


See: Thomas Wahl and Kurt Rothermel. *Representing time in multimedia systems*

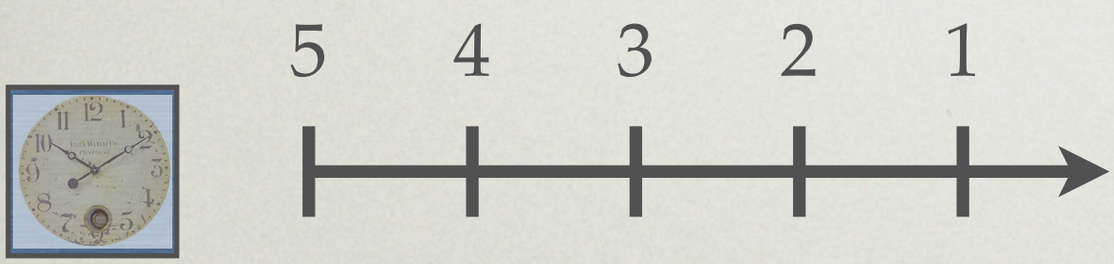
Clocks



- Functions that compute a scalar value at any time



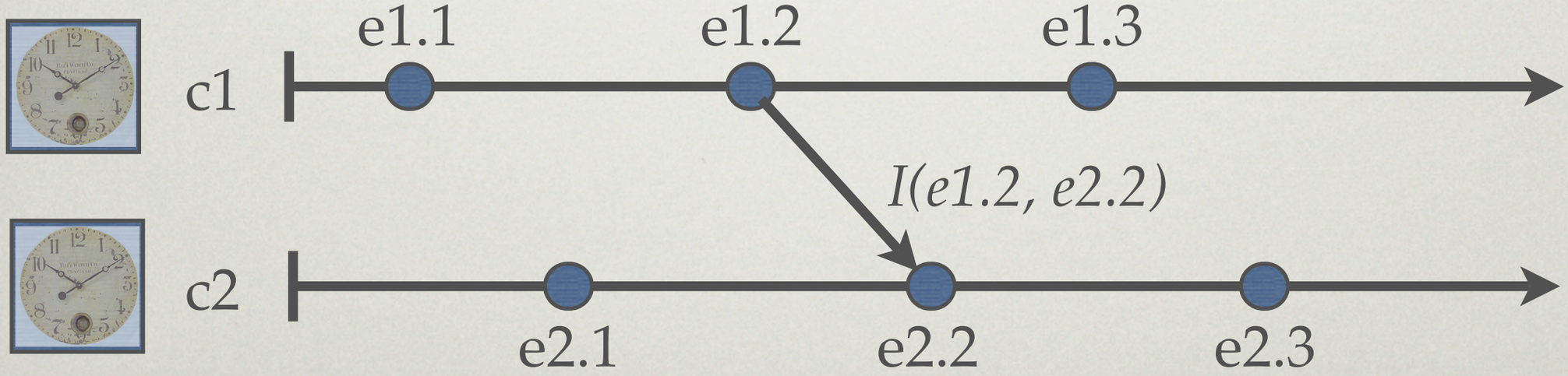
- Can count anything, e.g., # sold units



- No need to be monotonically increasing, e.g., countdown clocks

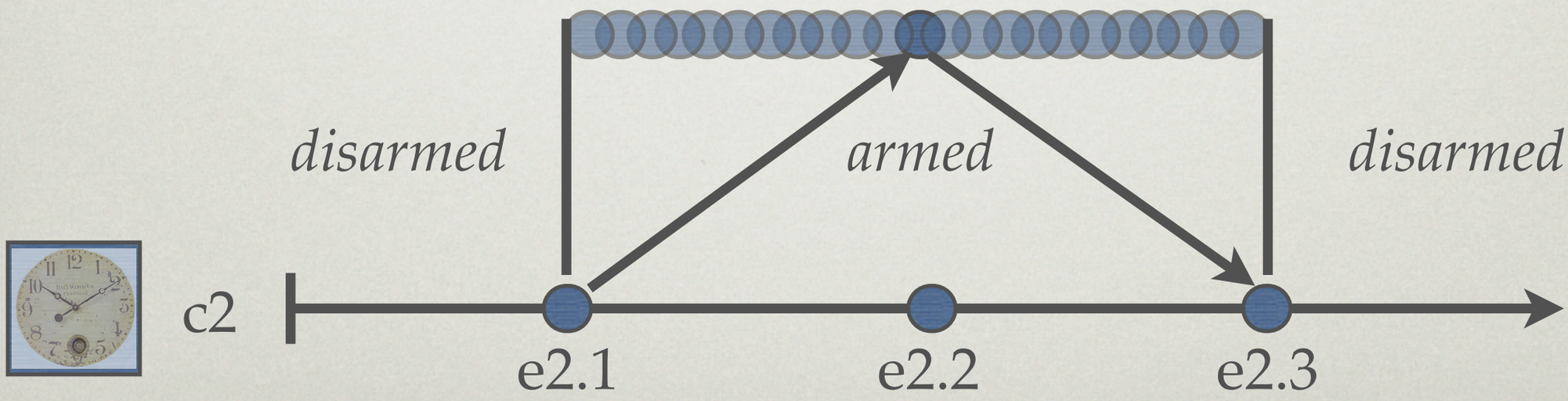
Scalar Times

- Defined in respect to a clock
- Within a clock: total ordering
- Between clocks: additional constraints (intervals) or *potentially concurrent*
- Operators: <, =, >, and ||



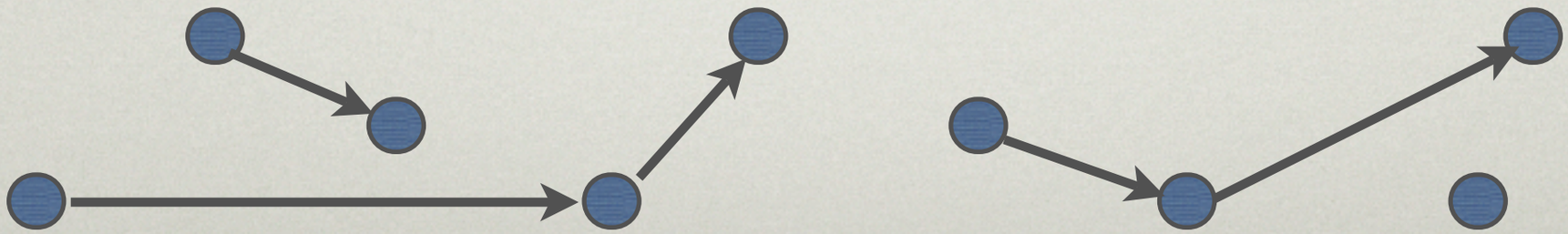
Indefinite Times

- Not associated with a clock (i.e., no scalar timestamp exists)
- Compares || to all other times by default
- Additional constraints may define a period in which the time is *armed*

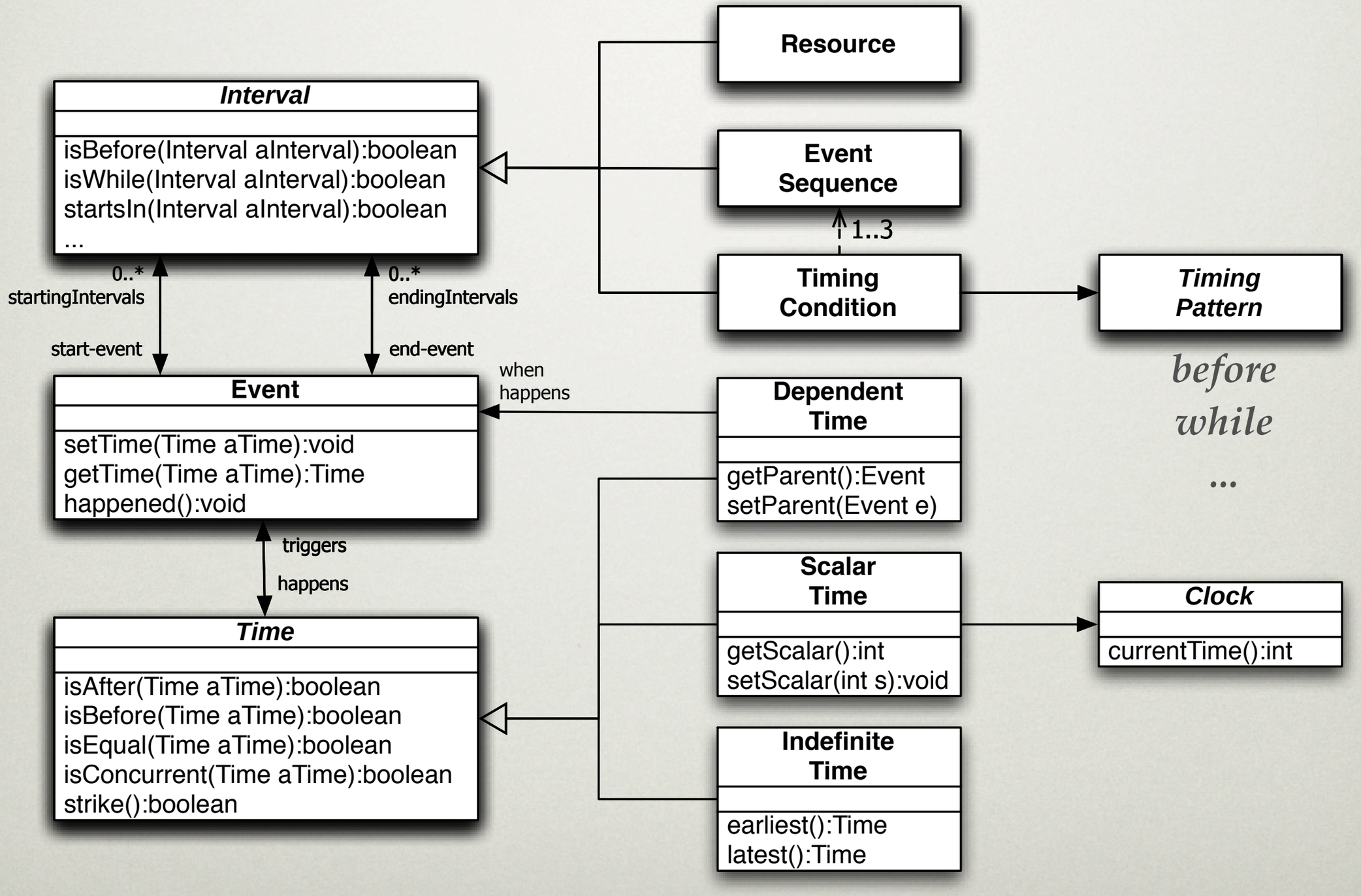


Timing Specifications

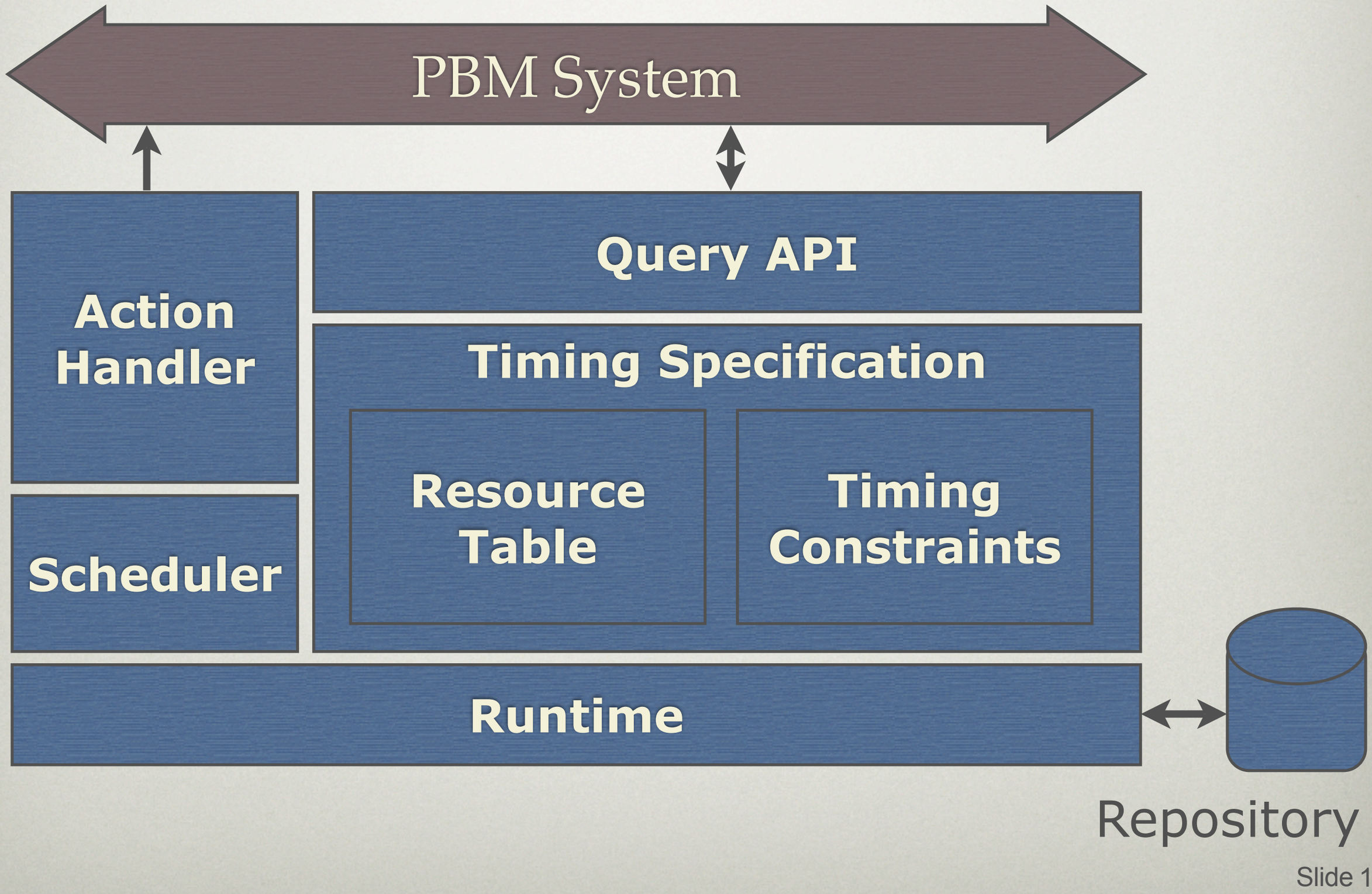
- Timing constraints (“arrows”) introduce “<” relations between times
- Constraints increase partial ordering of the time (event) space
- Constraints are mostly created implicitly:
  - defining the lifetime of a resource
  - creating a higher-level temporal relationship (e.g., *“A while B”)*



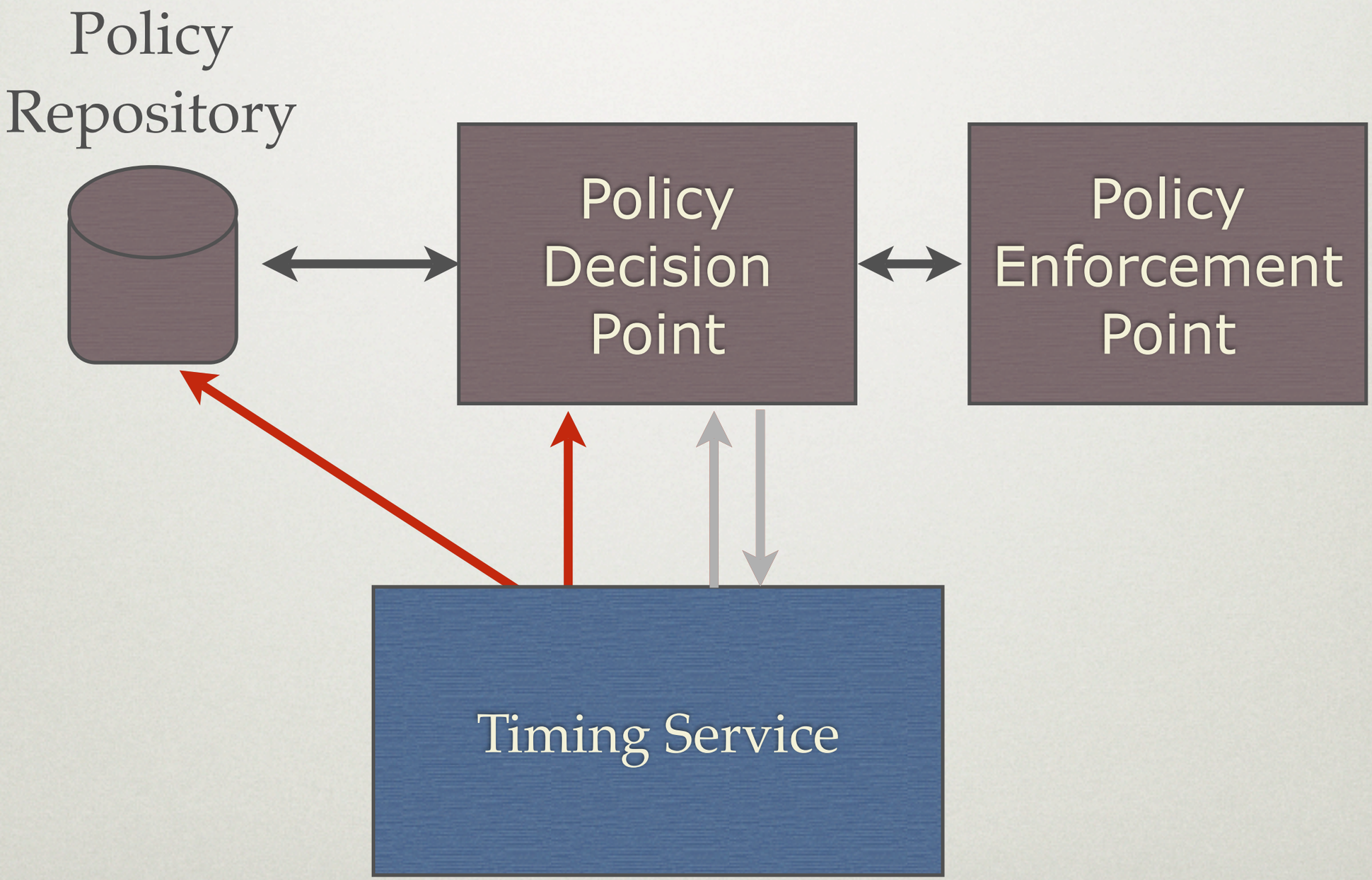
Implementation Model for Timing Specifications



Architecture of the Timing Service



Integration with existing Systems



Proposed Application Domain

- Authentication, Authorization, and Accounting of distributed resources in Peer-to-Peer networks
- No central authority: Users are responsible for own resources
  - Session management
  - Granting read/write access to files *during* a session
  - "*After* the session some participants still need read access *until* the project ends"

Concluding Remarks

- The separation of the events from the times they actually happen offers great flexibility for A-PBM
- Interval-based multi-media time models provide easy to use and easy to understand timing patterns
- But: Experience with larger (P2P) systems is still missing
  - Mutual dependencies of policies could lead to an explosion of complexity
  - Efficient user interfaces for entering and managing all the timing constraints have to be developed
  - Application in distributed and decentralized P2P networks where users/devices without global view join and leave the network arbitrarily has to be tested

Thank you for your attention...