Grid Scheduling Architecture RG

Ramin Yahyapour, Oliver Wäldrich, Philipp Wieder
OGF 19, January 29, 2007

OGF IPR Policies Apply

- "I acknowledge that participation in this meeting is subject to the OGF Intellectual Property Policy."
- Intellectual Property Notices Note Well: All statements related to the activities of the OGF and addressed to the OGF are subject to all provisions of Appendix B of GFD-C.1, which grants to the OGF and its participants certain licenses and rights in such statements. Such statements include verbal statements in OGF meetings, as well as written and electronic communications made at any time or place, which are addressed to:
  - the OGF plenary session,
  - any OGF working group or portion thereof,
  - the OGF Board of Directors, the GFSG, or any member thereof on behalf of the OGF,
  - the ADCOM, or any member thereof on behalf of the ADCOM,
  - any OGF mailing list, including any group list, or any other list functioning under OGF auspices,
  - the OGF Editor or the document authoring and review process
- Statements made outside of a OGF meeting, mailing list or other function, that are clearly not intended to be input to an OGF activity, group or function, are not subject to these provisions.
- Excerpt from Appendix B of GFD-C.1: "Where the OGF knows of rights, or claimed rights, the OGF secretariat shall attempt to obtain from the claimant of such rights, a written assurance that upon approval by the GFSG of the relevant OGF document(s), any party will be able to obtain the right to implement, use and distribute the technology or works when implementing, using or distributing technology based upon the specific specification(s) under openly specified, reasonable, non-discriminatory terms. The working group or research group proposing the use of the technology with respect to which the proprietary rights are claimed may assist the OGF secretariat in this effort. The results of this procedure shall not affect advancement of document, except that the GFSG may defer approval where a delay may facilitate the obtaining of such assurances. The results will, however, be recorded by the OGF Secretariat, and made available. The GFSG may also direct that a summary of the results be included in any GFD published containing the specification."

- OGF Intellectual Property Policies are adapted from the IETF Intellectual Property Policies that support the Internet Standards Process.

Agenda

- Session #1 and #2
  - Progress of the research group
  - Scheduler interoperation
    - Scenario
    - Required services & protocols
    - Basic set of JSDL parameters
    - Negotiation & agreement management
    - Execution management
  - Scheduling description language
  - Open issues & discussion

Milestones

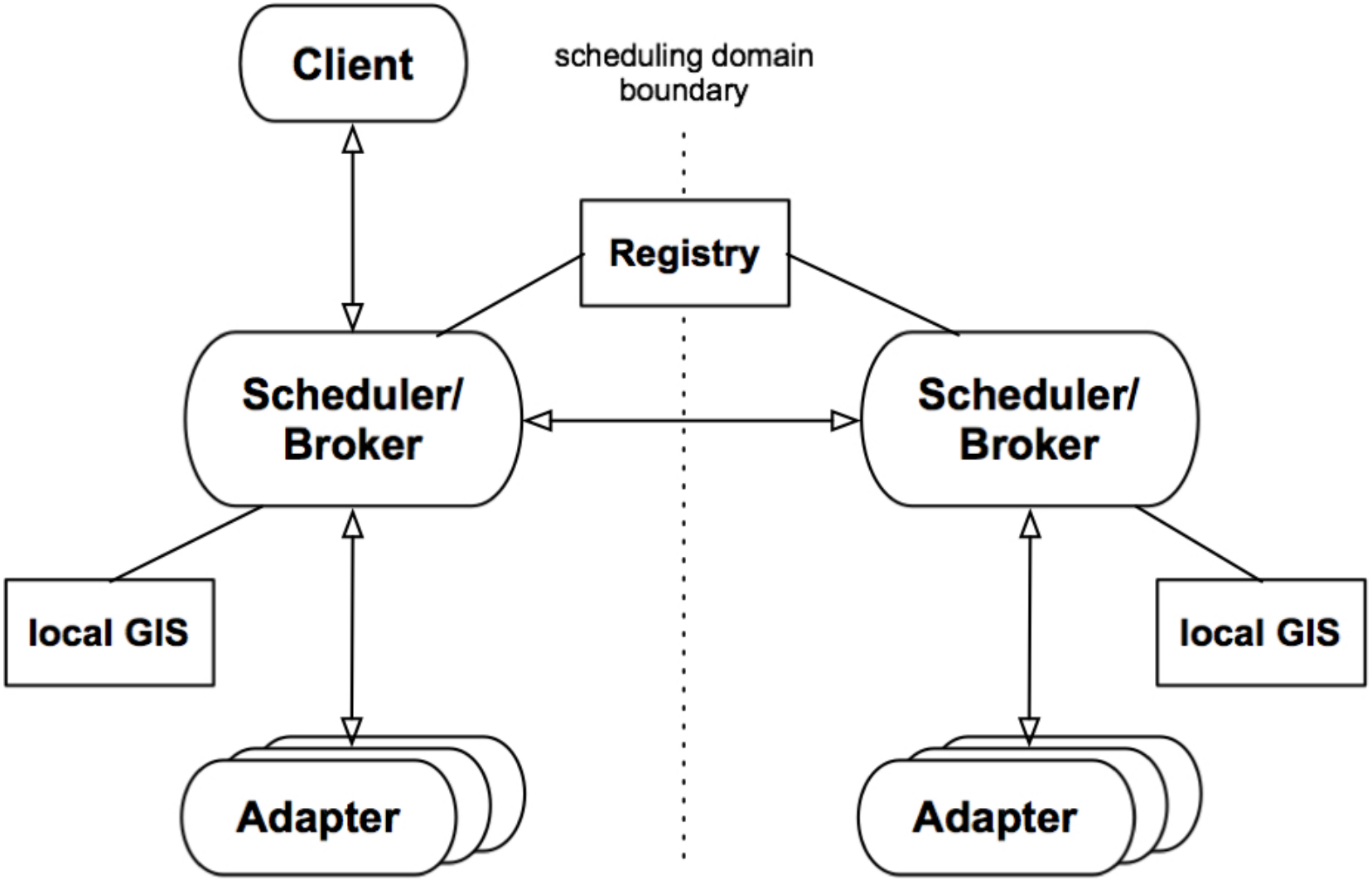
- Short term
  - Scheduler interoperation feasibility study
  - Practical show-case
  - Within 12 months from now
- Long term
  - Definition of a generic Grid scheduling architecture

Progress of the Research Group

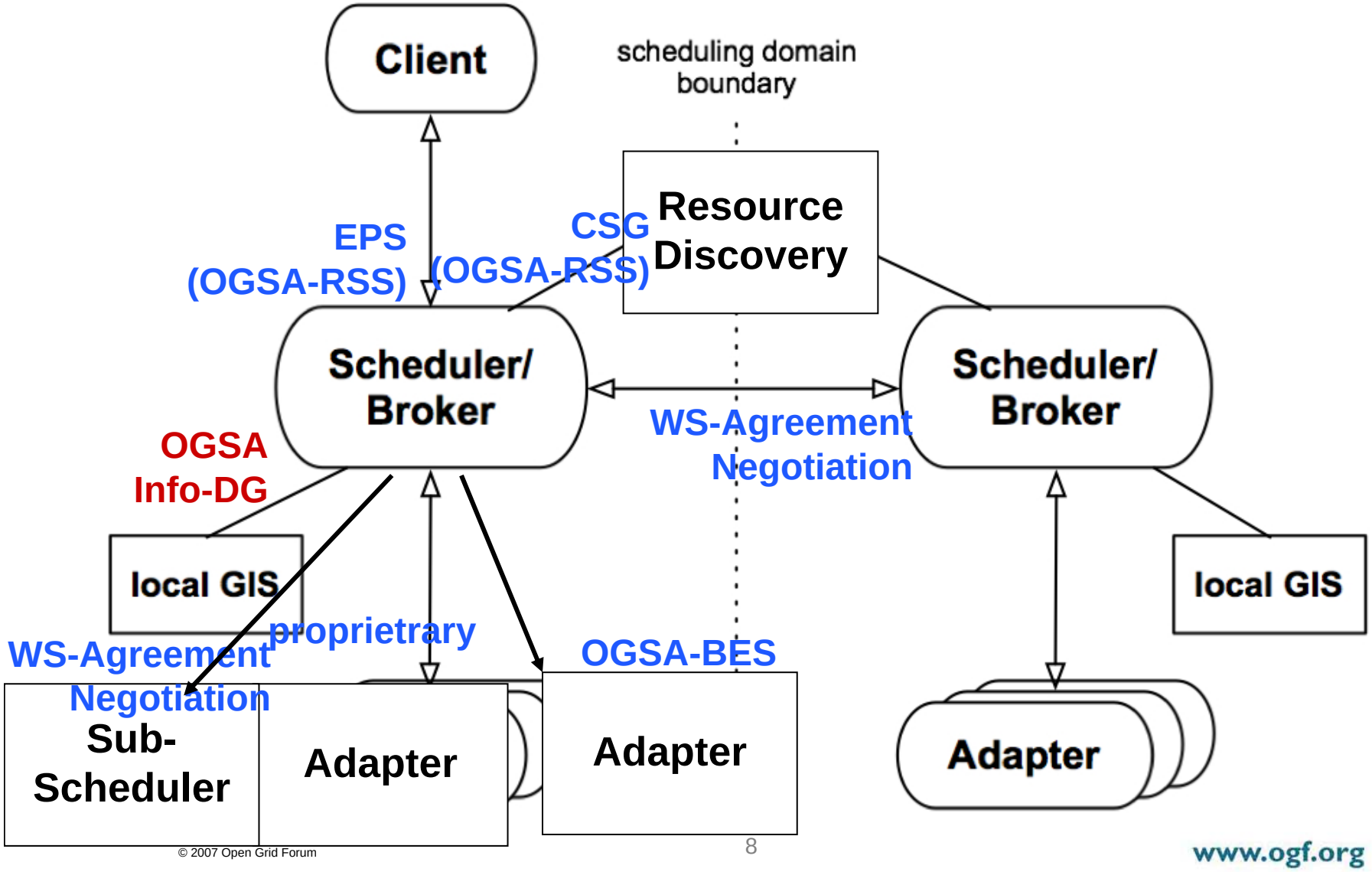
- Between OGF 18 and OGF 19
  - F2F discussion about scheduling interop in October
  - First outline of process at GSA-RG Wiki
  - First version of JSDL parameter document
  - First version of scheduler attributes document
- Grid Scheduling Architecture – Requirements document
  - Ongoing discussions about function and form
  - Status: still immature

Scheduler interoperation

Scenario – picture it!

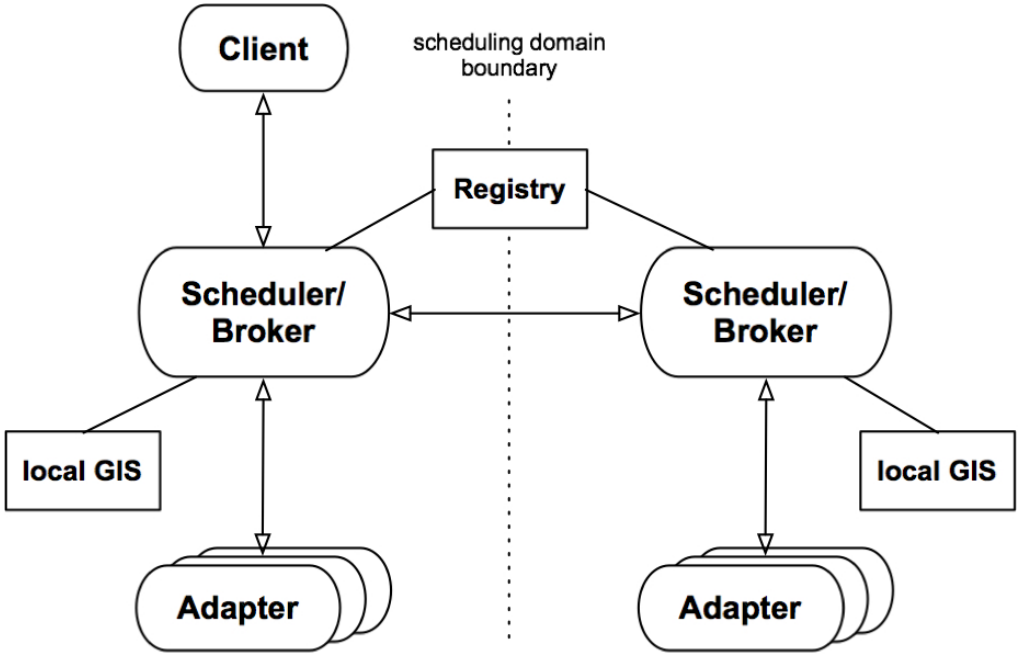


Scenario – picture it!

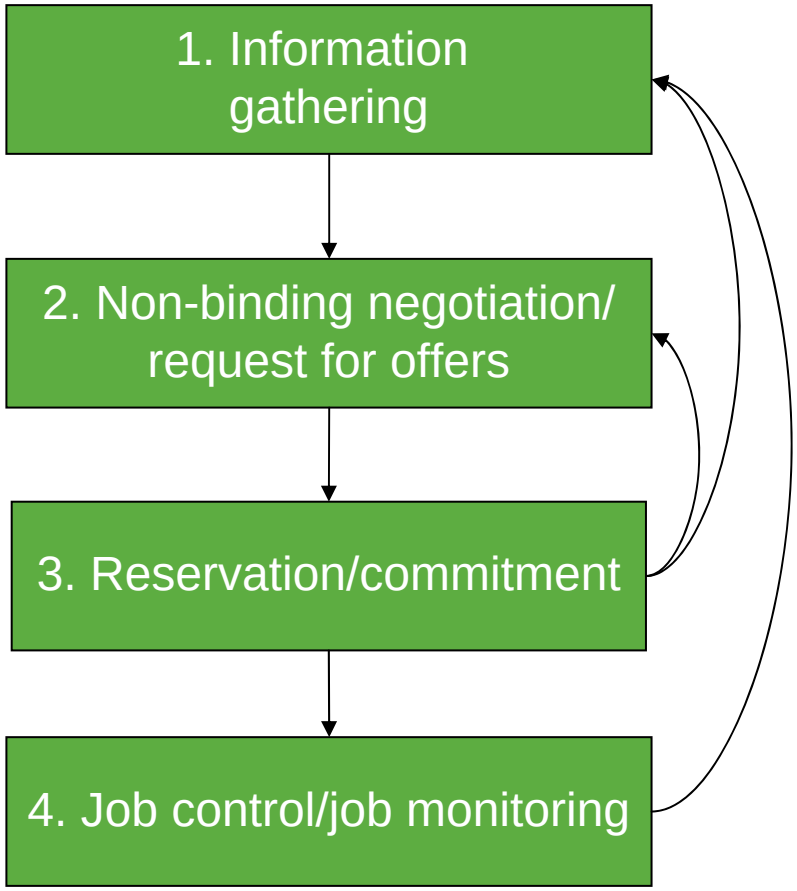


Entities involved

- Main entities
  - Client
  - Scheduler
  - Adapter
- Utility entities
  - Registry
  - Local GIS (Grid Information Service)



Communication Stages



1. Information gathering about available remote Grid schedulers
2. Non-binding negotiation may end up with several possible agreement alternatives (possibly in parallel)
3. Agreement creation and commitment; may fail and require return to previous stages
4. Handing over job control to remote Grid scheduler (responsibility remains at initiator)

Feasibility study

Goal & Interactions

- Simple use case: Two schedulers interoperate with each other
- Goal: Show feasibility through implementation
- Concrete interactions to reach agreement on delegation of scheduling decision:
  1. Scheduler A cannot fulfil a scheduling request
  2. Request is passed to Scheduler B
  3. Scheduler B checks its capabilities
  4. Scheduler A & B agree/disagree on conditions to fulfil the request
  5. [Potentially it is possible to re-negotiate the conditions]
  6. Scheduler B fulfils the scheduling request

Candidate “standards”

“Standard” descriptions:

- Common job description: JSDL
- Common resource model: OGSA Info model?
  - OR semantic translation services between different models
- Add. scheduling parameters: JSDL extensions?

“Standard” protocols

- Agreement creation: WS-Agreement
- Negotiation: WS-Negotiation?

Participating projects

Confirmed

- Grid Resource Management System (GRMS)
- MetaScheduling Service (MSS)
- D-Grid

Interested

- GridWay
- … your project?

JSDL Profile

Goal

- Agree on a basic set of attributes to be referenced in an SLAs during negotiation between Grid schedulers
- JSDL profile attributes …
  - … needed for scheduling (e.g. info about data, required number of CPUs, bandwidth etc.),
  - … referenced in service description terms in SLA negotiations (e.g. amount of memory etc.),
  - … essential for proper job execution by LRMS (e.g. job arguments, environmental variables etc.)
  - … are defined in GFD.56
- Question: What is the minimal set of attributes a scheduler MUST support?

Minimal attribute set

- Intersection of attributes supported by different systems is small
  - Comparison of GRMS and MSS resulted in only a hand of common attributes
- Full JSDL support not in focus
- Start with traditional resources: compute & files
- … Discussion needed

JSDL Profile Candidate Attributes

- JobIdentification
  - JobName
  - JobProject
- Resources
  - CandidateHosts (HostName)
  - FileSystem (name)
  - OperatingSystem
  - CPUArchitecture
  - IndividualCPUSpeed
  - IndividualCPUTime
  - IndividualCPUCount
  - IndividualNetworkBandwidth
  - IndividualPhysicalMemory
  - IndividualVirtualMemory
  - IndividualDiskSpace
- DataStaging
  - FileName
  - Source (URI)
  - Target (URI)
  - Executable
- Application
  - ApplicationName
  - ApplicationVersion
- POSIX Application
  - Executable
  - Argument
  - Input
  - Output
  - Error
  - Environment

Attributes Supported

- JobIdentification
  - **JobName**
  - JobProject
- Resources
  - **CandidateHosts (HostName)**
  - FileSystem (name)
  - OperatingSystem
  - CPUArchitecture
  - IndividualCPUSpeed
  - IndividualCPUTime
  - IndividualCPUCount
  - IndividualNetworkBandwidth
  - IndividualPhysicalMemory
  - IndividualVirtualMemory
  - IndividualDiskSpace
- DataStaging
  - FileName
  - Source (URI)
  - Target (URI)
  - Executable
- Application
  - ApplicationName
  - ApplicationVersion
- POSIX Application
  - **Executable**
  - **Argument**
  - Input
  - Output
  - Error
  - Environment

Summary of JSDL Profile

- So far only GRMS and VIOLA examined
  - Other schedulers should be included to have a broader view
- Only 4 attributes common for Grid schedulers!
- However some of attributes needed only by LRMS so they can be just passed if LRMS supports JSDL
- Some of attributes are essential but not available in VIOLA. Will they be implemented?

Scheduling Description language

Scheduling Attributes

- The goal: information needed by Grid schedulers to schedule a job(s) according to user's requirements and preferences
- In general these requirements and preferences may concern time constraints, job dependencies, job priorities, scheduling objectives, data-dependent scheduling information, and other
- An initial set of attributes proposed
- From this wide set of attributes those supported by available Grid schedulers selected

Argumentation

- Attributes exist which are
  - … out of scope of JSDL (sic!), but …
  - … within the scheduling scope
- Approach
  - a) Extend JSDL
  - b) Create a Grid Scheduling Language
  - c) Steel something out there
- Note: b) could be a showstopper aiming for fast progress

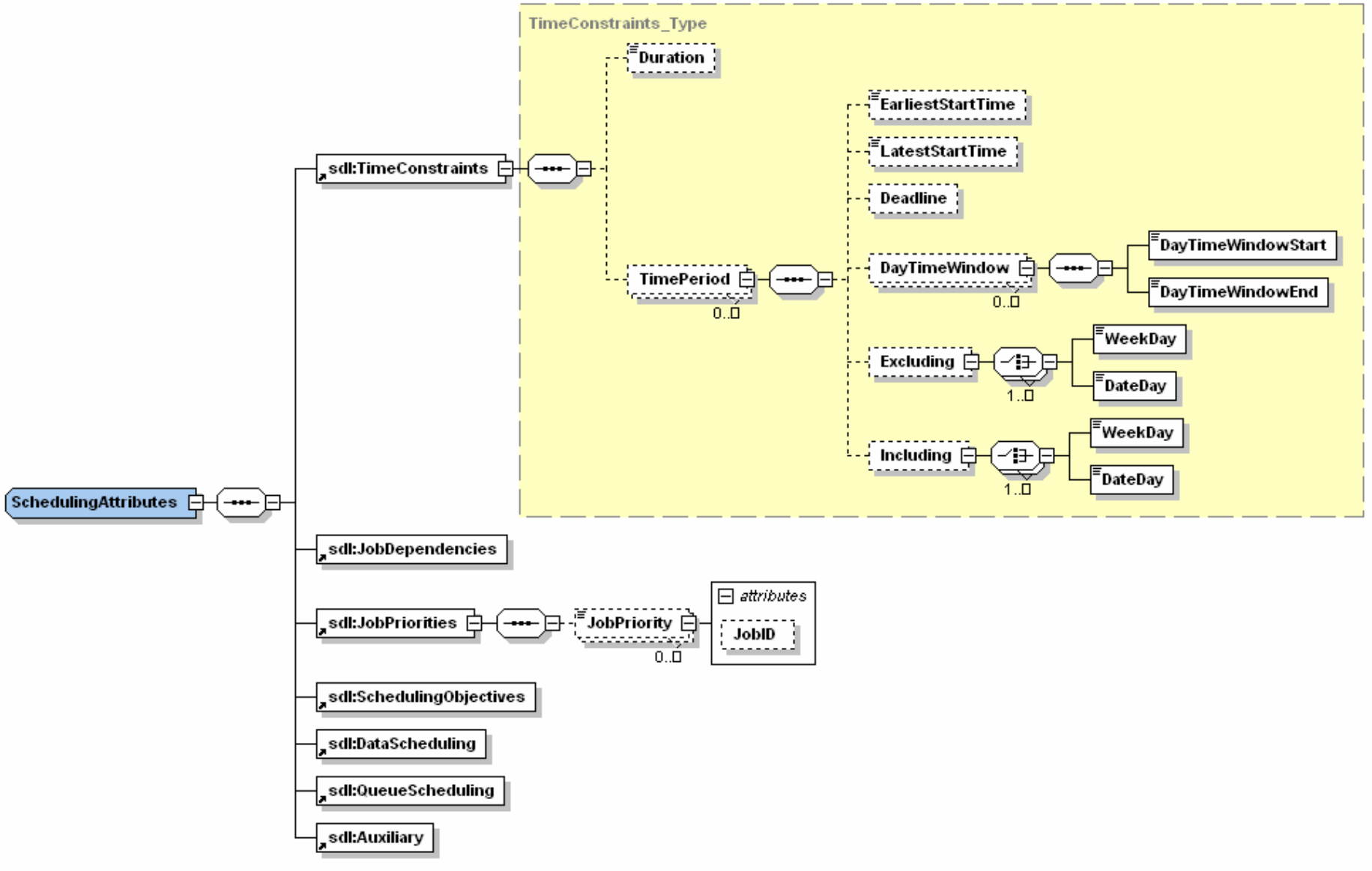
Attributes Supported

- **Duration** (required)
  - Maximal job execution time
- **EarliestStartTime**
  - Specifies the earliest start time of a job (e.g. 31st January 2007 10:00AM)
- **LatestStartTime**
  - Specifies the latest start time of job (e.g. 12th February 2007 16:00PM)
- **NoCoallocation**
  - Defines whether resources required for a given job can be allocated at more that one site

Scheduling Description Language (SDL)

- In general, specification of scheduling attributes is missing
- Not available even for local resource management systems (e.g. to specify requirements concerning advance reservation)
- Categories of scheduling attributes considered
  - time constraints,
  - job dependencies,
  - job priorities,
  - scheduling objectives/preferences,
  - data-dependent scheduling information,
  - queue-based scheduling information,
  - miscellaneous.

Preliminary Proposal of SDL



SDL Open Issues

- How to define SDL?
  - JSDL extension ?
    - But in the JSDL spec is said that SDL is out of scope
  - WS-Agreement ?
    - But concrete terms are needed
  - Separate specification?
    - Design group within GSA?
    - Another group ?
- Should be the same for LRMS?
- Which attributes should be selected?
- What else is out there?

Argumentation

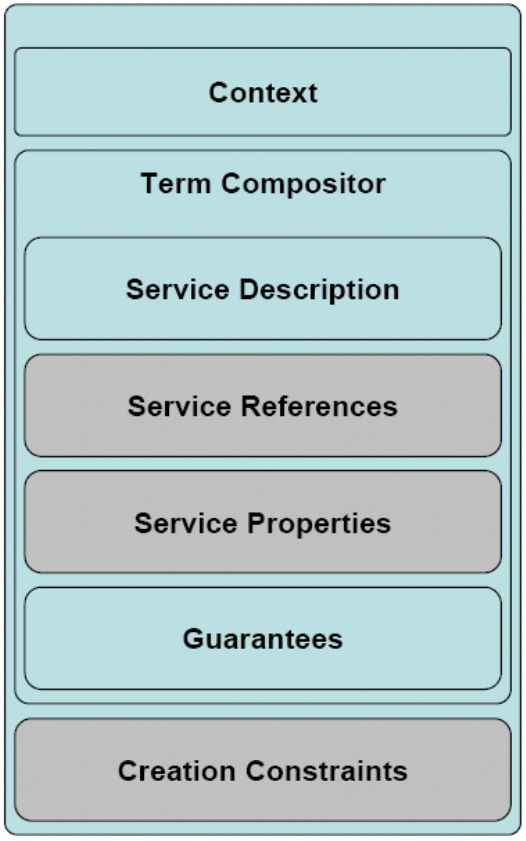
- Attributes exist which are
  - … out of scope of JSDL (sic!), but …
  - … within the scheduling scope
- Approach
  - a) Extend JSDL
  - b) Create a Grid Scheduling Language
  - c) Steel something out there
- Note: b) could be a showstopper aiming for fast progress

More information

- JSDL profile draft document
  - https://forge.gridforum.org/sf/go/doc14009?nav=1
- Scheduling attributes (SDL) draft document
  - https://forge.gridforum.org/sf/go/doc14026?nav=1

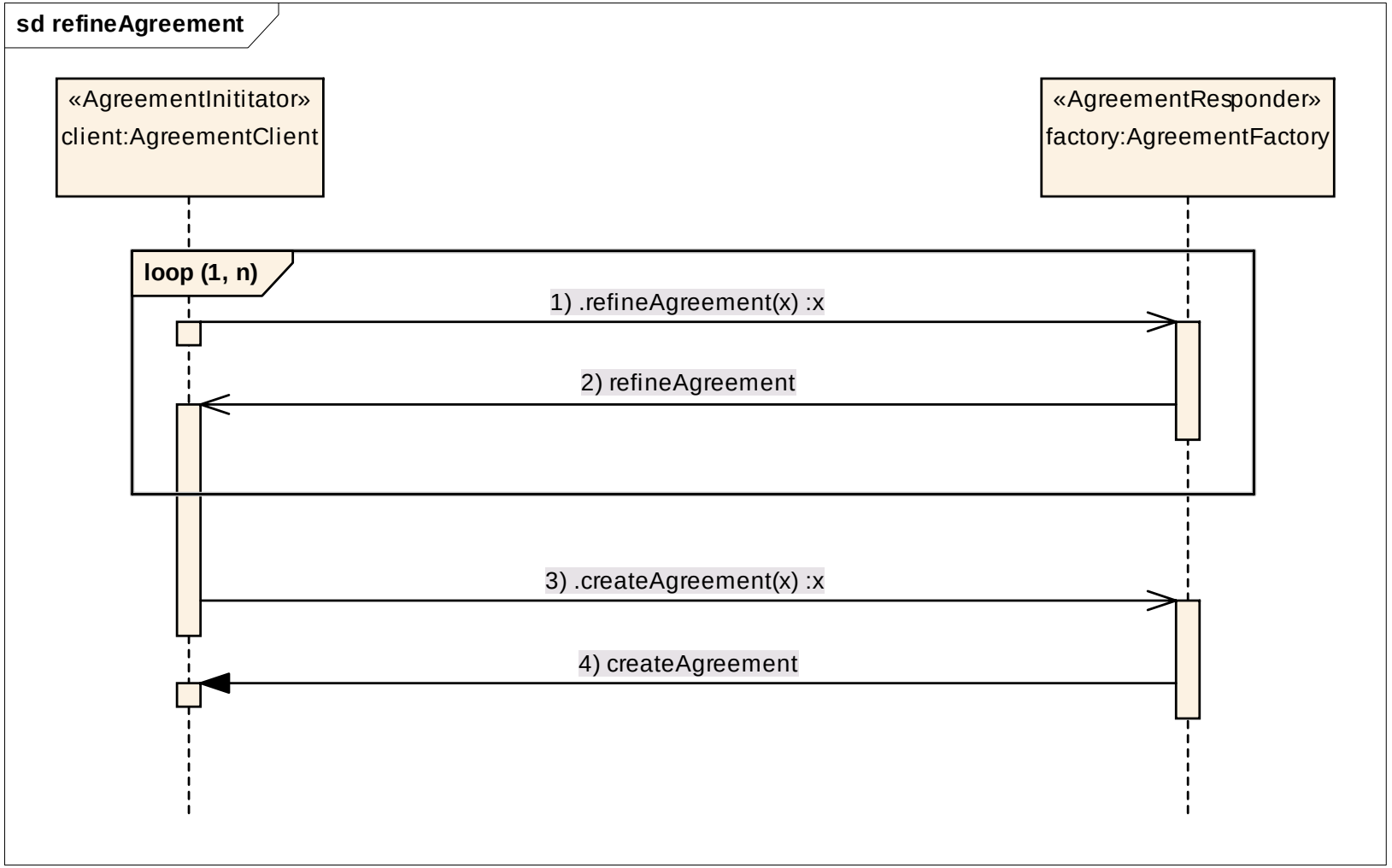
Negotiation Protocol

“Extend” WS-Agreement

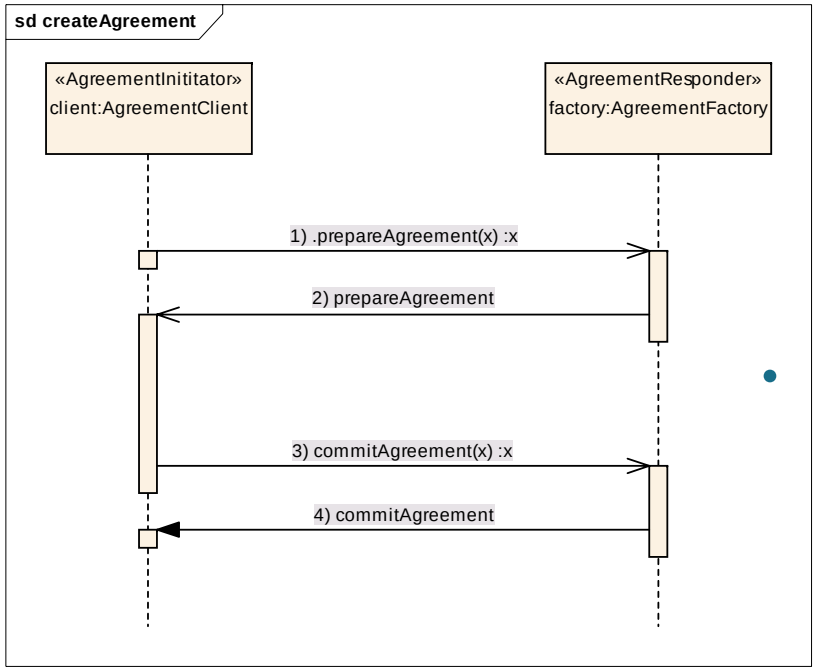


- New method refineAgreement()
- Input is a normal agreement offer
- No agreement created / no EPR sent back
- Response is (possibly refined) counter offer
- Content may change
  - e.g. ranges (CPU, memory, …) may be refined
- refineAgreement() can be called multiple times in the negotiation phase
  - Negotiating start times, resources, …
- Only valid counter offers are returned
  - Counter offer can be created in principle
- Counter offer is only a hint (no guarantee)
  - No pre-reservation implied here
- Grey terms may be skipped in a first step (needed?)

Refine & create agreement



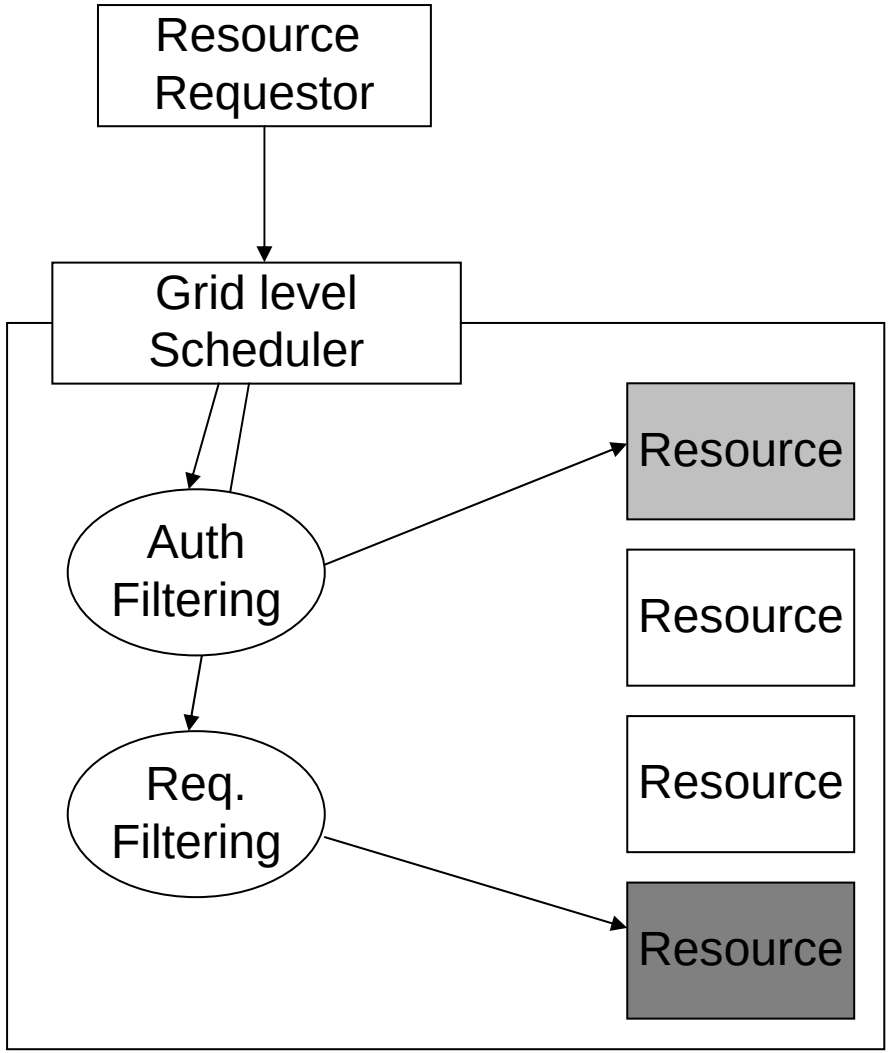
createAgreement (2PCP)



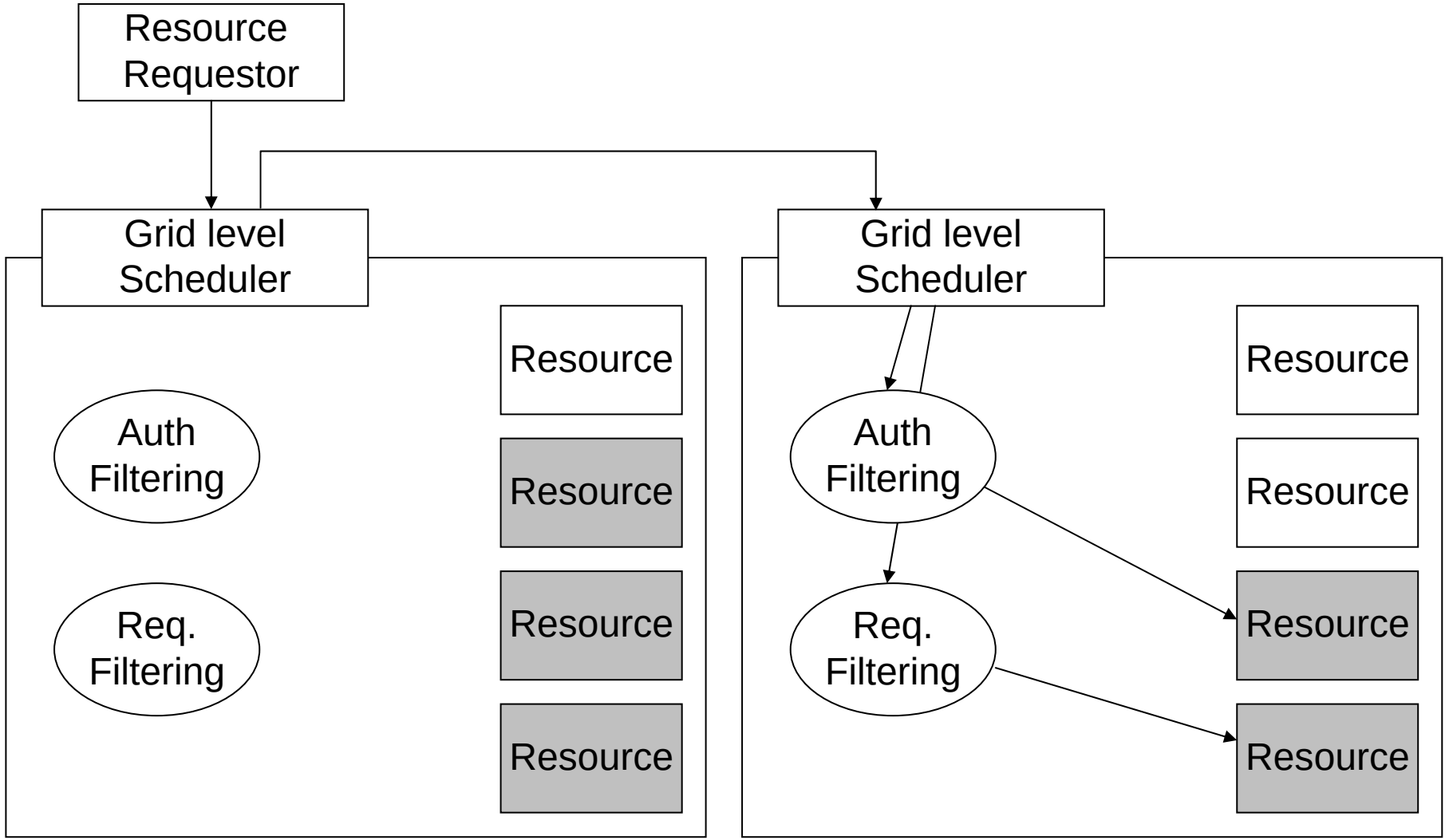
- EPR prepareAgreement(offer)
  - Same signature as createAgreement()
  - Creates a new Agreement with new state *prepared*
  - Short lifetime (e.g. one minute)
  - Includes resource pre-reservation
  - No or reduced Penalties?
- void commitAgreement(EPR)
  - Commits the prepared agreement referenced by the EPR
  - Changes the state to *observed*
  - Regular lifetime
  - Agreement costs/rewards/penalties become active
  - No errors are thrown, since resources are pre-reserved

Grid scheduling process

- Resource Discovery
  - Authorization Filtering
  - Minimal requirement filtering
- System Selection
  - Information gathering (caching)
- Reservation
  - Advance reservation
  - Task Submission
- Monitoring
- Cleanup



Multi domain negotiation



Execution Management

Execution Management after the Agreement Commitment

Prior to job execution	• Job-Transfer • Data-Transfer • Provisioning
During execution	• Monitoring/Reporting • Failure-Detection/Management
After execution	• Access to results • Clean-Up

Execution Management – Prior to Job Execution

- How is the job description transferred by whom?
- What about additional data?

Long-term:

- Data management as separate activity that is scheduled managed by initiating scheduling instance.

Short-term:

- JSDL job description is attached to agreement.
- Data management is not supported

Execution Management – Job Control

- Who initiates the job execution?
- Does it need to be triggered?
  - How is the agreement referenced to be considered?
- Or is it done automatically based on the agreement?

Long-term:

- OGSA-BES?
- OGSA-HPC-P?

Short-term:

- The job execution is initiated automatically by the agreement provider based on the JSDL description.

Execution Management – During Job Execution

- How is the job monitored?
- Who recognizes failures?
- Who manages failures?

Long-term:

- If possible, failures should be handled at the resource provider; if not possible to recover locally, it is reported to and handled by the initiator.

Short-term:

- Failures are automatically handled.
- Monitoring?

Execution Management – After Job Execution

- How are results accessed?
- How long are job information available?

Long-term:

Short-term:

Security Context – Authentication

- How do we manage access control and authentication?

Long-term:

- VOMS?

Short-term:

- ?

Issues, questions, …

Issues to discuss

- Is the current extension to WS-Agreement feasible?
- Schedulers remain automonous. All information needed has to be passed via one scheduler interoperation interface. Do we cover all aspects
- There is no common information model shared between the schedulers. Solution?
- ….

Full Copyright Notice