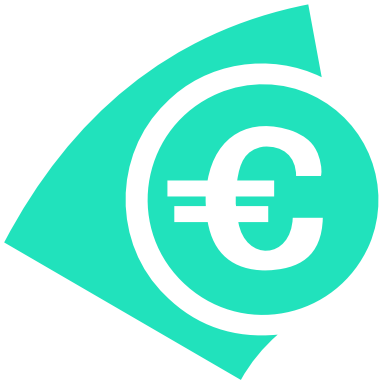
A MetaScheduling Service for Co-allocating Arbitrary Types of Resources

Oliver Wäldrich, Wolfgang Ziegler, FhG, Philipp Wieder, FzJ

Acknowledgements

Some of the work reported in this presentation is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #123456. This presentation also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST programme under grant #004265.



Fraunhofer SCAI Institut Algorithmen und Wissenschaftliches Rechnen

Forschungszentrum Jülich in der Helmholtz-Gemeinschaft

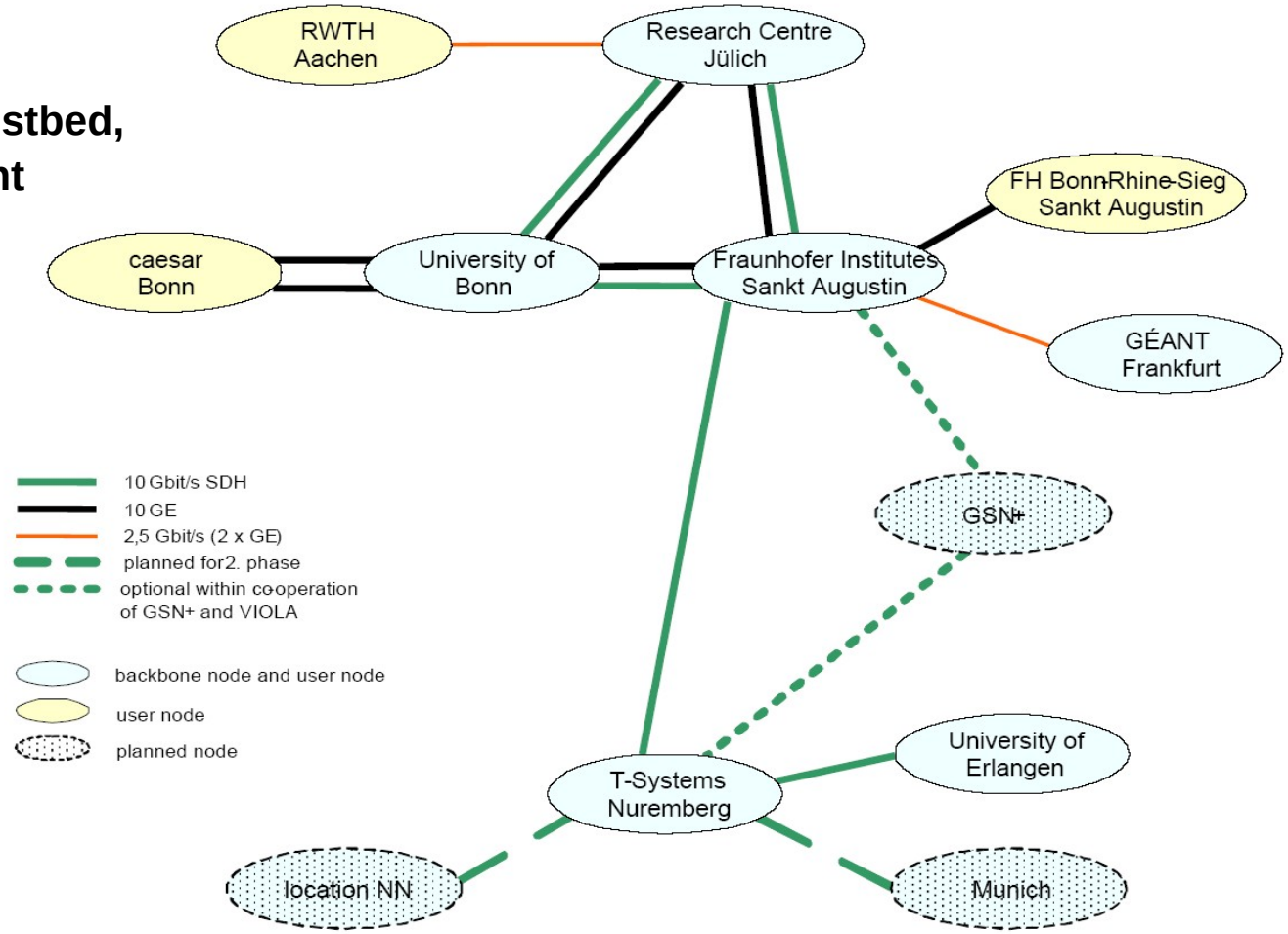
CoreGRID

VIOLA

VIOLA Network

VIOLA-Networking

- **Deployment and operation of the testbed, test of advanced network equipment**
- **Signaling and reservation**
  - **bandwidth- and QoS-reservations in the network**
  - **interfaces for user-driven reservation: immediate and in advance**



Properties of local RMS required for a MetaScheduling Service

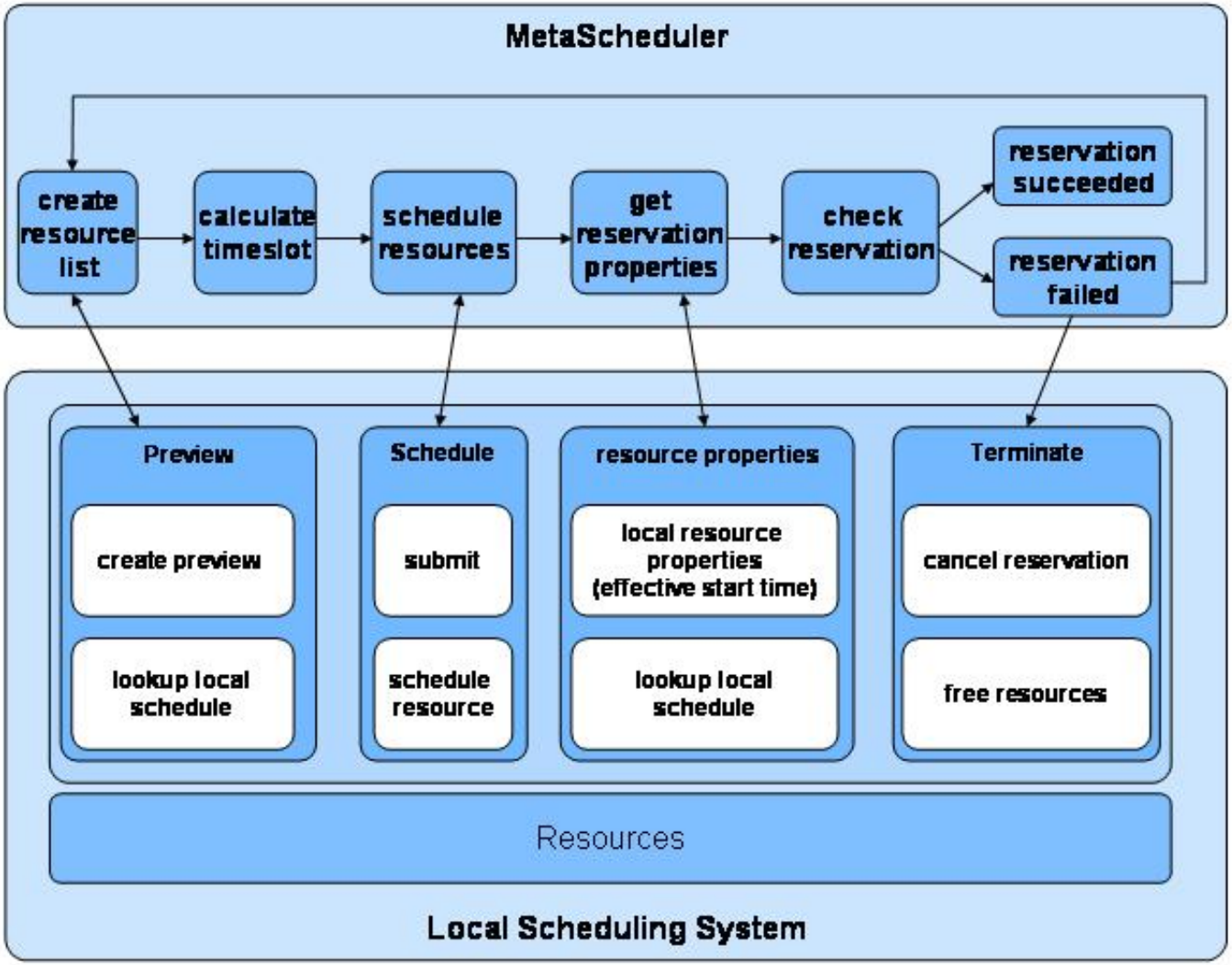
- Full backfill algorithm
- Estimation of worst case start/stop for each job (preview)
- Node range specification
- Start time specification (AT-jobs)
- Special resource requirement specification
- „very low priority“ jobs (Z-jobs)
- Communication friendly node allocation strategy
- Portable: available on different parallel machines
- Graphical user interface
- Status information available via WEB interface
- Priority scheme (project, resources, waited time)

Pseudo-code of the co-allocation algorithm

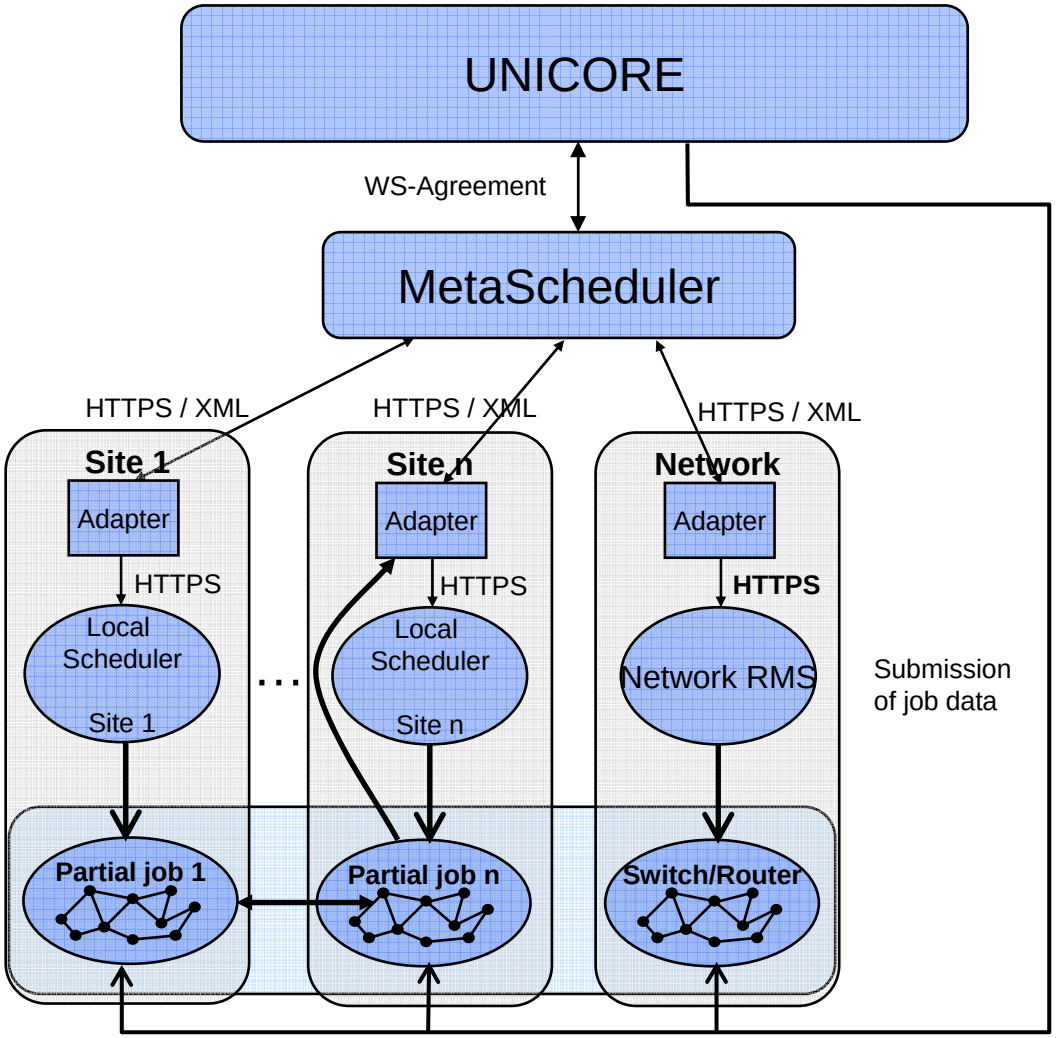
```
set  n                      =  number of requested resources
set  resources[1..n]        =  requested resources
set  properties[1..n]       =  requested property per resource   # number of nodes, bandwidth, time, ..
set  freeSlots[1..n]        =  null                                                  # start time of free slots
set  endOfPreviewWindow             =  false
set  nextStartupTime        =  currentTime+someMinutes          # the starting point when looking for free slots
while (endOfPreviewWindow = false) do {
            for 1..n do in parallel {
                        freeSlots[i] = ResourceAvailableAt( resources[i], properties[i], nextStartupTime)
            }
            for 1..n do {
                        set needNext = false
                        if ( nextStartupTime != freeSlots[i]) then {
                                    if ( freeSlots[i] != null) then {
                                                if( nextStartupTime < freeSlots[i]) then {
                                                            set nextStartupTime =  freeSlots[i]
                                                            set needNext        =  true
                                                }
                                    } else {
                                                set endOfPreviewWindow = true}
                        }
            }
}

if ( ( needNext = false ) & ( endOfPreviewWindow = false) ) then return
            freeSlots[1] else return “no common slot found”
```

Allocation Agreement Protocol

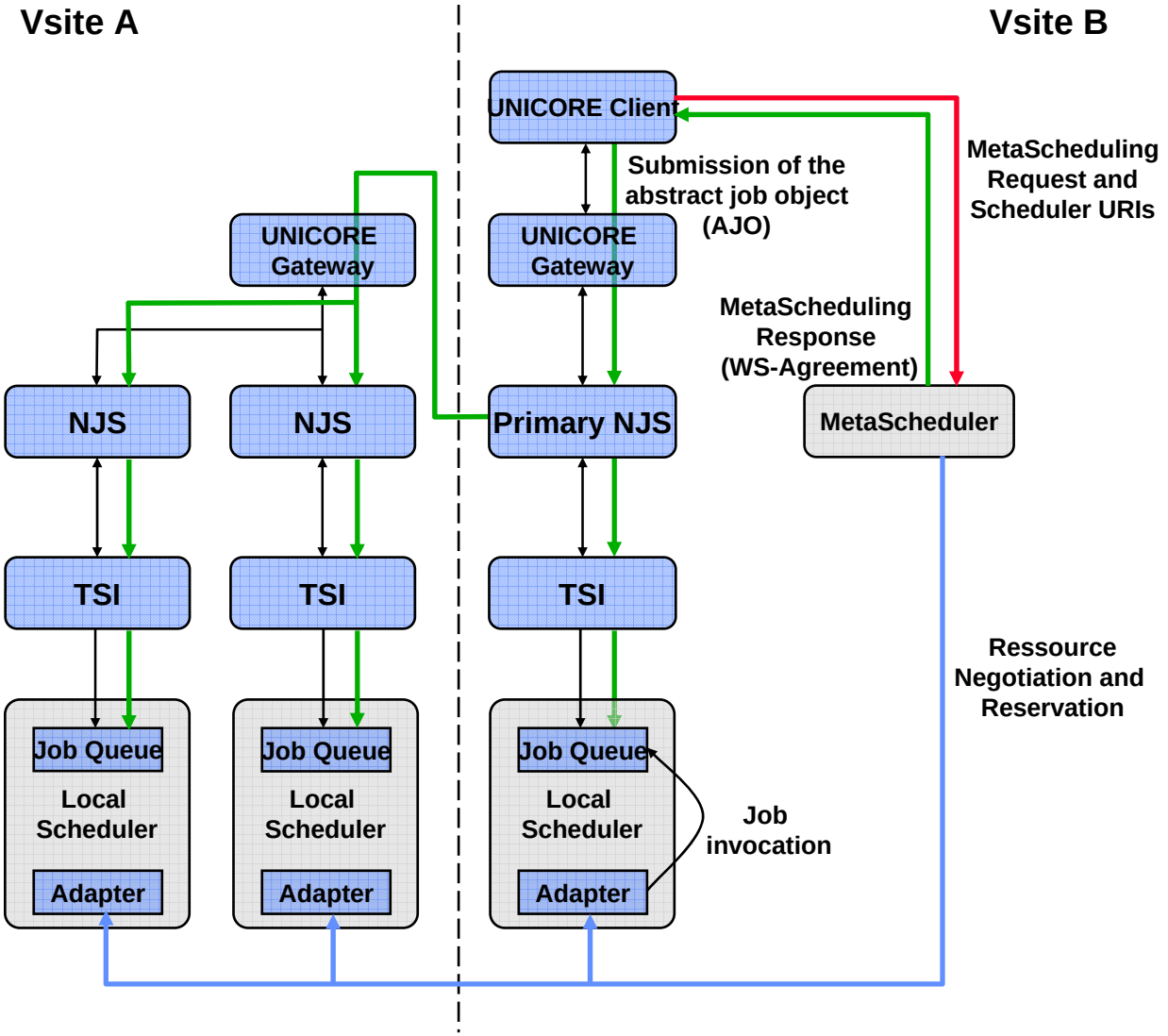


MetaScheduler - Integration of local Schedulers



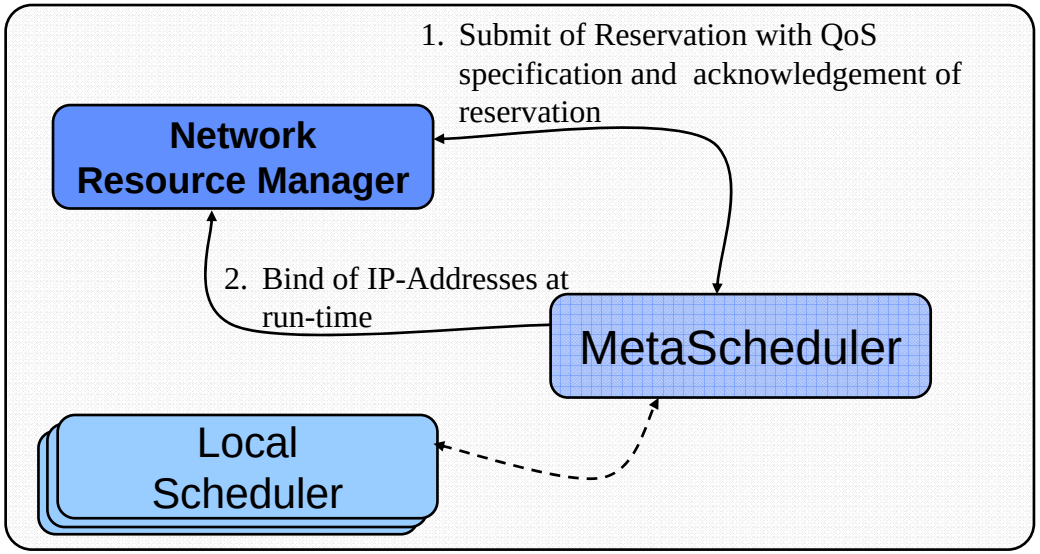
- Negotiation of timeslot & nodes with local schedulers for each job
- UNICORE initiates the reservation and submits the job-data
- UNICORE Client / MetaScheduler Service interface using WS-Agreement protocol
- Interface MetaScheduler / Adapters based on HTTPS/XML (SOAP)
- Interface between MetaScheduler Service and local RMS implemented with adapter pattern
- Communication between Adapter and local Scheduler with HTPPS

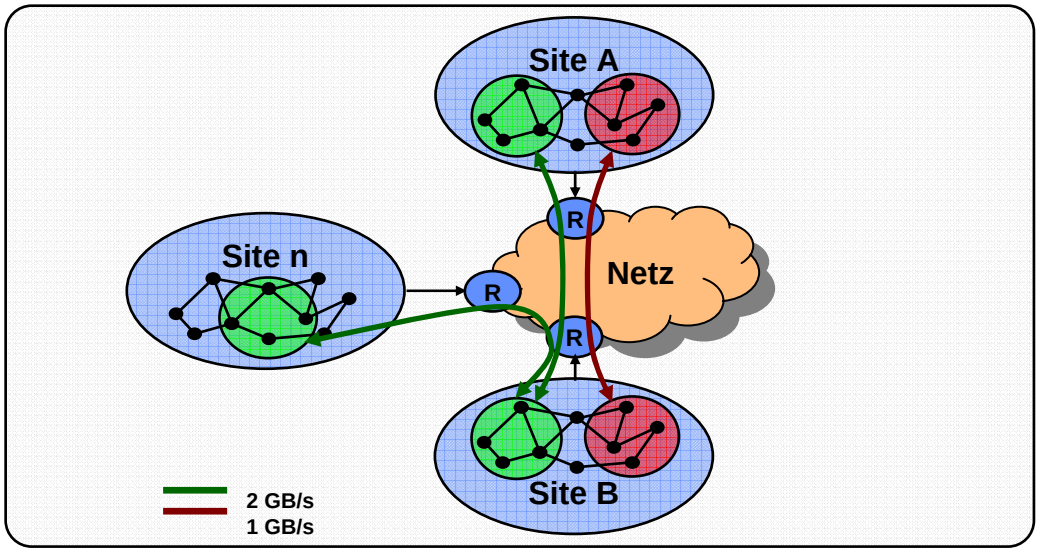
MetaScheduler - Integration in UNICORE



- UNICORE Client sends request to MetaScheduler (WS-Agreement template)
- MetaScheduler negotiates earliest time to run this job, requests the reservation of the requested resources and returns the WS-Agreement with additional Status, ID
- UNICORE Client creates Abstract Job Object (AJO) and sends it to the Primary Network Job Supervisor (NJS)
- NJS incarnates the AJO according to the information in the AJO and the UIDB, forwards it to the local Target System Interface (TSI) and sends the AJO to all other NJSs
- TSI creates the entry for the Meta-Job in the UNICORE Job Queue, and creates the UNICORE wrapper in the User-directory
- Scheduler triggers job at start time

Netzwork Resource Management System





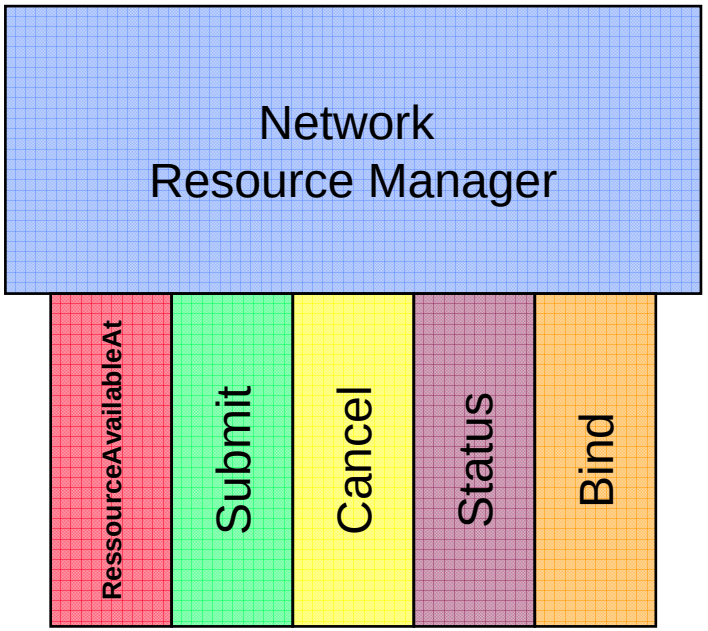
1.) Reservation of required Resources

- Submit of a Reservation to the Network Resource Manager
- Acknowledgement of Reservation

2.) Bind of IP-Addresses at Run-time

- IP-Addresses are published at run-time of the job through the local Adapter
- Bind of the IP-Addresses by the Network Resource Manager
- Without explicit Bind the QoS Parameters for the Site-to-Site Interconnection are used

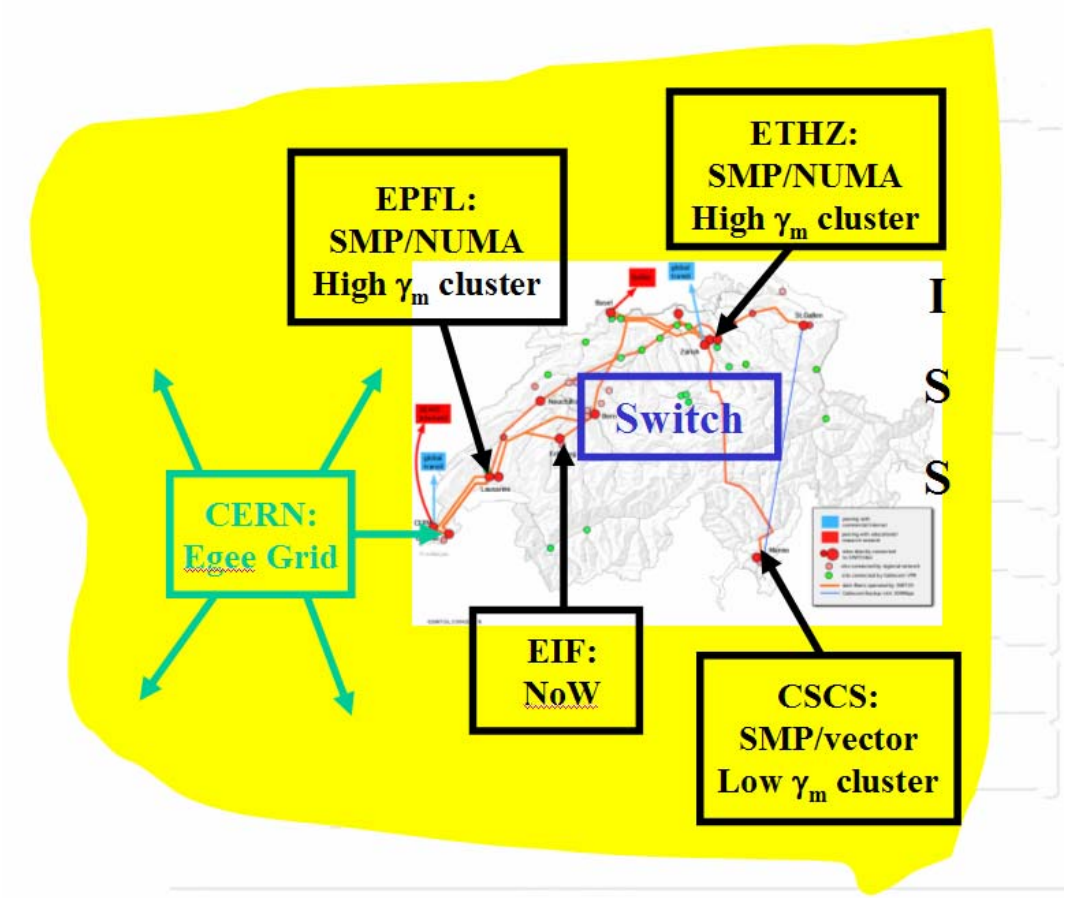
Applikation Interface of the Network Resource Manager



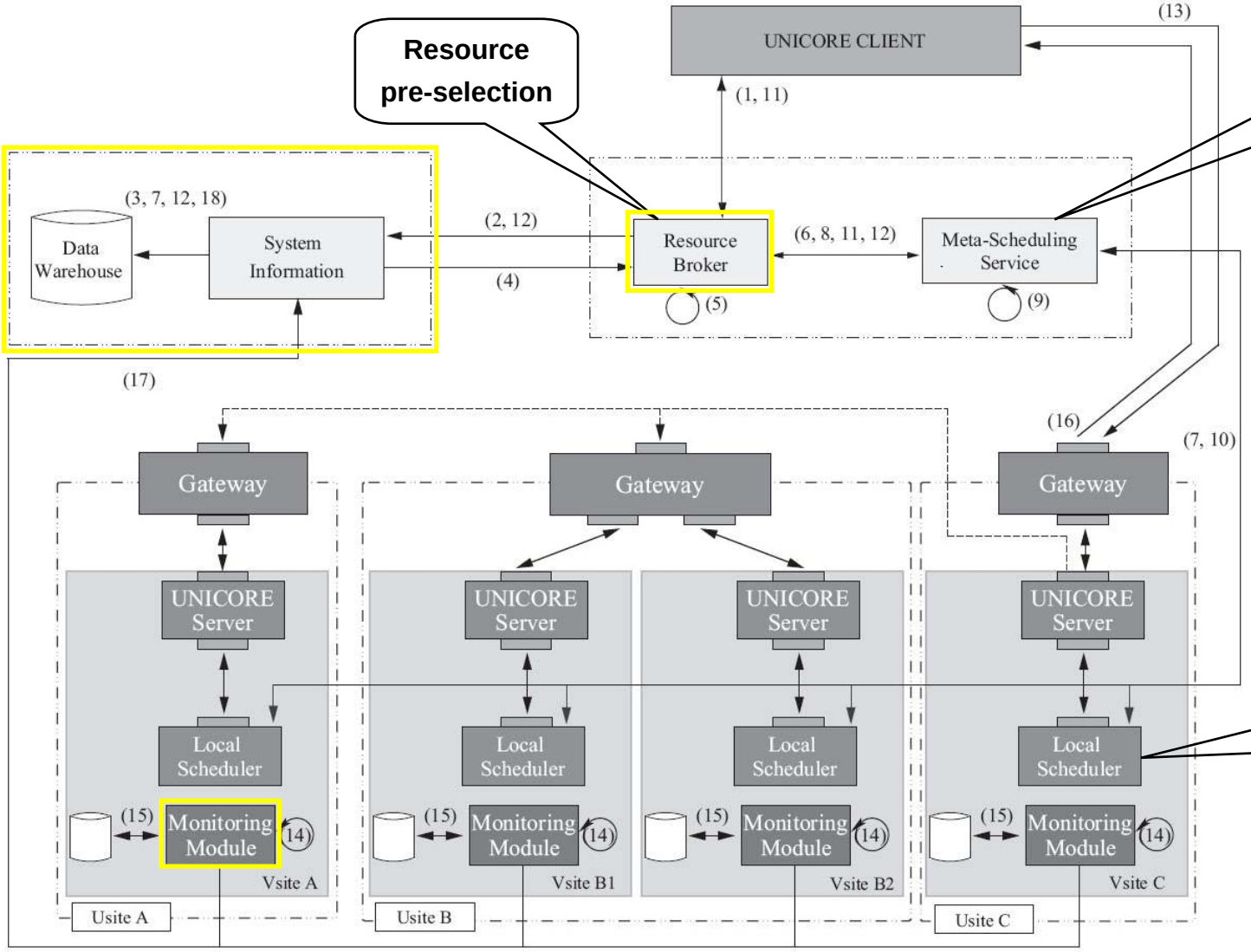
Necessary Functions:

- *ResourceAvailableAt (Preview)*
  - Returns time slots when a Resource (End-to-end connection with QoS Level) will be available
- *Submit*
  - Start-time, Duration, Class, Start-/End-pointt (Site), User,
  - Returns a Resource Identifier (RESID)
- *Cancel <RESID>*
  - Resource Manager frees the Resources attached to Resource Identifier (RESID)
- *Status <RESID>*
  - Returns state of a connection (submitted, active, released, Class, start-time, end-time, user, etc.)
- *Bind <RESID>*
  - Binding of IP-Addresses of nodes

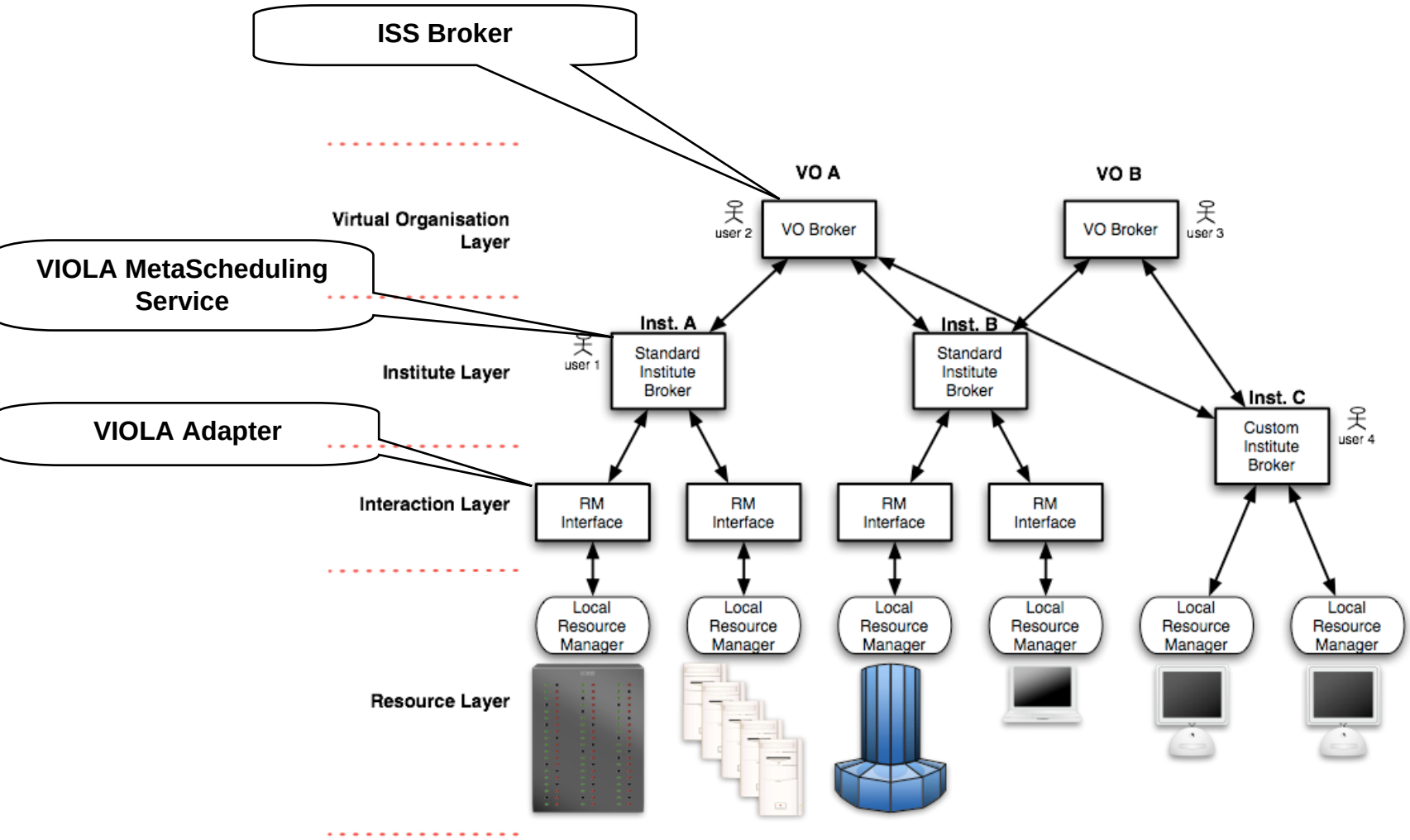
SwissGrid – Integration of UNICORE, MetaScheduling Service and ISS



Scheduling Architecture of the VIOLA-ISS Integration



Example of a scheduling infrastructure for HPC Grids (from GSA Requirements)



Mapping to the OGSA-EMS structure