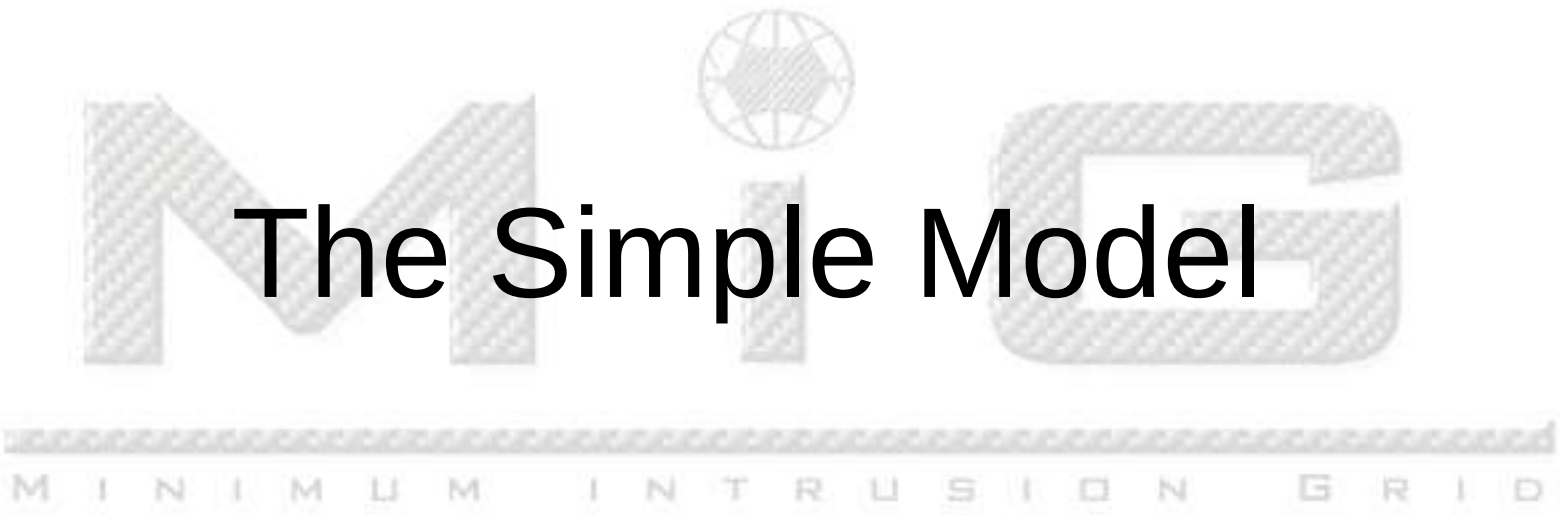
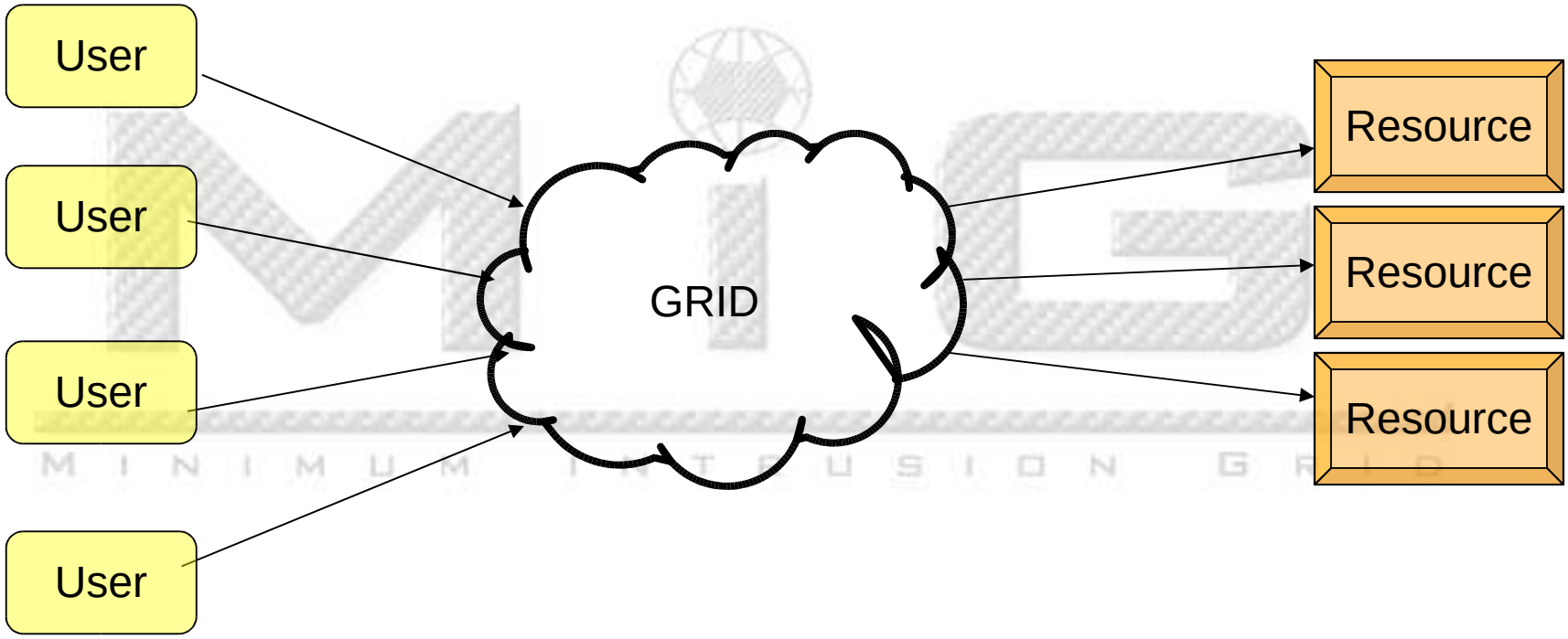
Minimum intrusion Grid



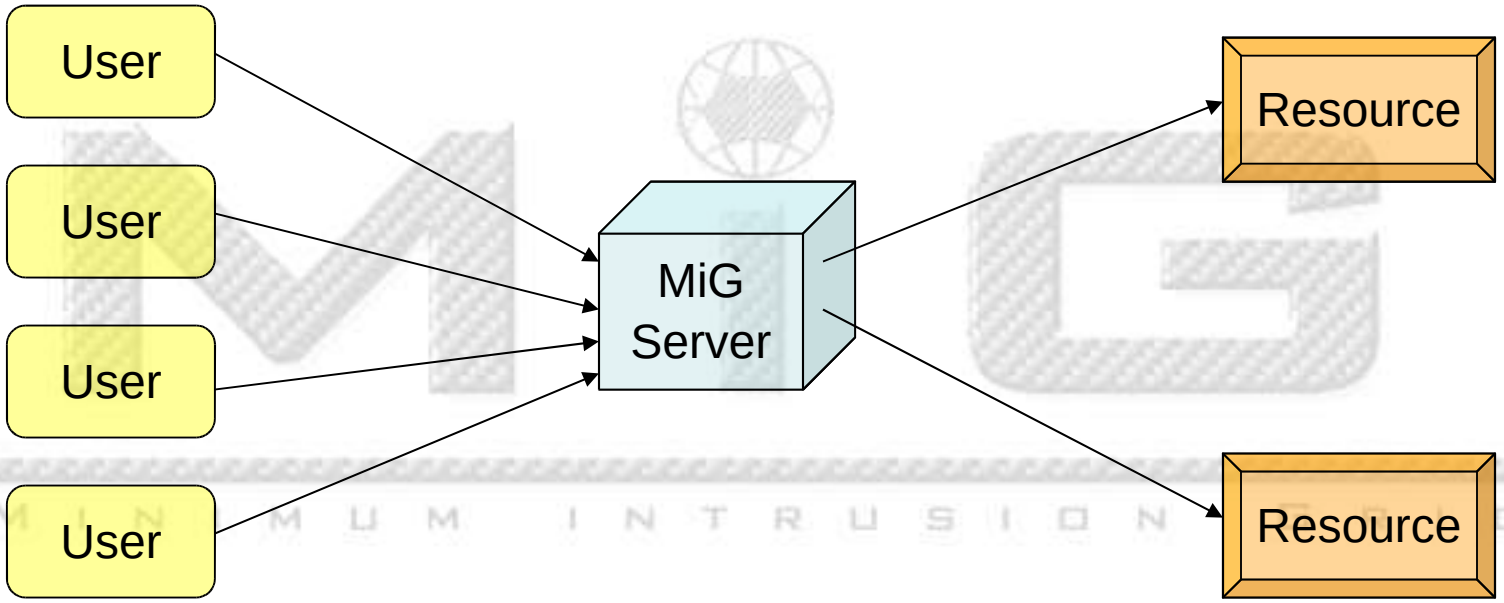
The Simple Model

Design and implementation

The Abstract MiG Model

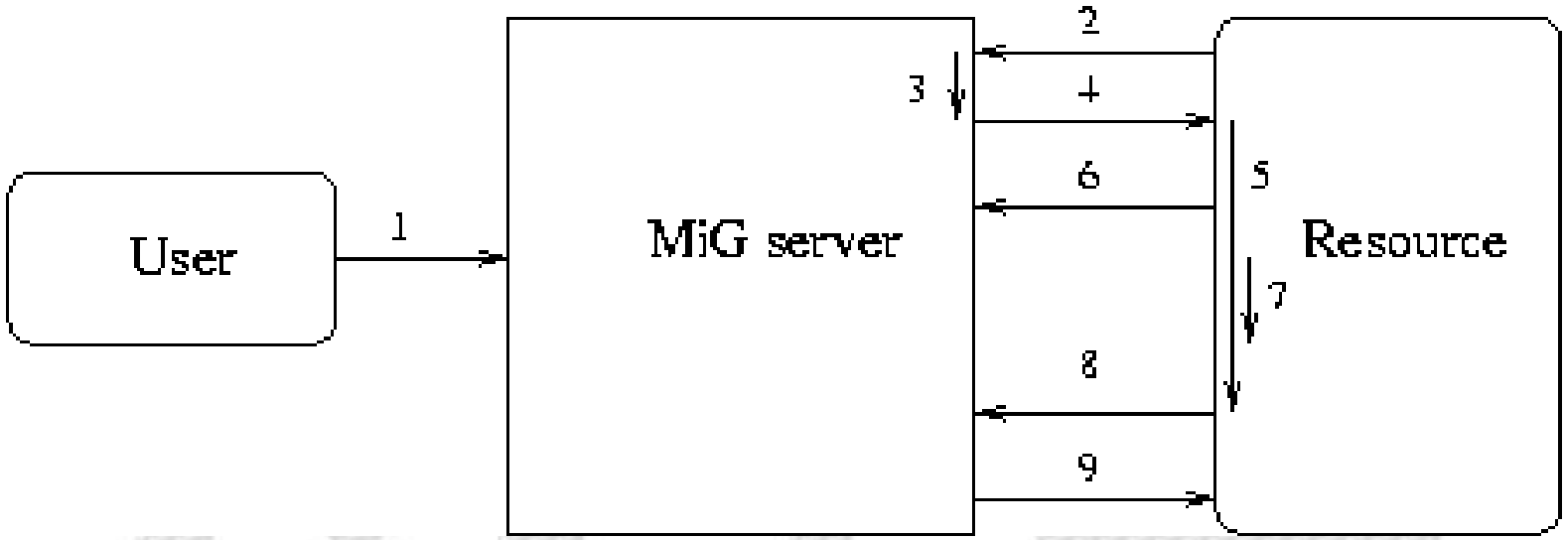


The Simple MiG Model



- Grid cloud is now a single MiG server
- Full featured Grid solution
- Obvious problems: single pt of failure, performance

Simple model - design



1. User communicates with the MiG server using HTTPS and certificates.
2. Resource requests a new job to execute (HTTPS+SID).
3. The MiG server creates the job script
4. MiG server sends the job to the resource using SCP.
5. The resource starts the job script.
6. Resource requests the needed input files from the MiG server (HTTPS+SID).
7. The actual job is executed.
8. Resource sends output files to the MiG server (HTTPS+SID).
9. MiG server cleans up the resource using SSH (files and processes).

New concepts

- A job also consists of getting the input files and sending the output files (Transparent Remote On-demand File Access: under development)
- Home directory on MiG server: Input files must be in the user's personal home directory and output files are sent to the home directory

MiG from the user POV

- Browser (signed x.509 certificate, HTTPS)
  - Manage files in grid home directory
  - Submit jobs, view job status etc.
- MiGscripts:
  - Wrappers around cURL (http://curl.haxx.se)
  - File handling:
    - MiGput, MiGget, MiGremove, MiGlist, MiGcat
  - Job management:
    - MiGsubmit, MiGstatus, MiGallstatus, (MiGkill)

MiG from the user POV

- http://mig-1.imada.sdu.dk
- Signed x.509 certificate in p.12 format
- Import certificate in browser.

MiGscripts

- Very simple shell scripts
  - As much code as possible is placed on the MiG server.
- Example (from MiGlist.sh):

```
curl --cert $certfile --key $key \
    $migserver/cgi-bin/myfiles.py?search=$pattern
```

```
karlsen@adina:~/mig/user> ./MiGlist.sh "*.txt"
README.txt
txtfile.txt
```

mRSL job specification

- Globus RSL and Nordugrid xRSL too complex.
- Most keywords are similar to those in RSL/xRSL:
  - EXECUTE,INPUTFILES,OUTPUTFILES,
  - EXECUTABLES,ARCHITECTURE,CPUCOUNT,
  - NODECOUNT,CPUTIME,MEMORY,DISK,
  - RUNTIMEENVIRONMENT,ENVIRONMENT,
  - MAXPRICE,JOBNAME,NOTIFY
- Hide grid mechanisms

mRSL example

::EXECUTE::
echo "Hello World"
uname -a
cat inputfile >> outfile

::NOTIFY::
jabber: karlsen@jabbernet.dk
karlsen@imada.sdu.dk

::INPUTFILES::
inputfile

::OUTPUTFILES::
outfile

::MEMORY::
128

::DISK::
10

::MAXPRICE::
30

::CPUTIME::
1000

::JOBNAME::
myjobname

MiG from the resource POV

- Resource configuration on MiG server
  - Updated using browser or script (HTTPS+cert)
  - Scheduling:
    - architecture, disk, memory, cpucount,
    - nodecount, runtimeenvironment
  - Pricing:
    - minprice
  - Other:
    - scriptlanguage (sh or python)
    - hosturl, miguser, hostkey (for scp/ssh)

MiG from the resource POV

- miniscript.sh requests and executes a single job:

```
newjob=`curl \
  $migserver/cgi-bin/requestnewjob?cputime=$cputime`
chmod +x newjob
./newjob
```

- no_queue.sh for resources without queue system:

```
while [ 1 ] ; do
  ./miniscript.sh
done
```

The central MiG server

- Apache server, HTTPS w. x.509 certificates
- cgi-scripts:
  - jobstatus.py, removefile.py,
  - requestnewjob.py, myfiles.py, etc.
- Main script:
  - When a new job is received from a user and it is parsed successfully the script is notified.
  - The same script is notified when a resource requests a new job to execute.

The central MiG server

- First Fit, Best Fit, ... Scheduler
- Job script generator (language in res. conf.)
  - createJobDirectory, cdToJobDirectory,
  - getInputFiles, getExecutables,
  - chmodExecutables, setEnvironments,
  - setRuntimeEnvironments, execute,
  - sendOutputFiles, sendStatusFiles
- Sends job to resource using SCP

Implementation status

- Most basic functionality implemented
- One user (genetic research)
- 3 resources, approx 128 P4's on 83 nodes
- Monitor:
  - http://mig-1.imada.sdu.dk/monitor.html

The simple model - future

- Continue the implementation phase
- Add features (e.g. Killjob)
- Have more test users and resources with different setups (large PBS clusters etc).
- This will without doubt create new feature requests
- Documentation