

Using Panlab Federation Mechanisms and Infrastructure for Cloud Experiments

sebastian.wahle@fokus.fraunhofer.de



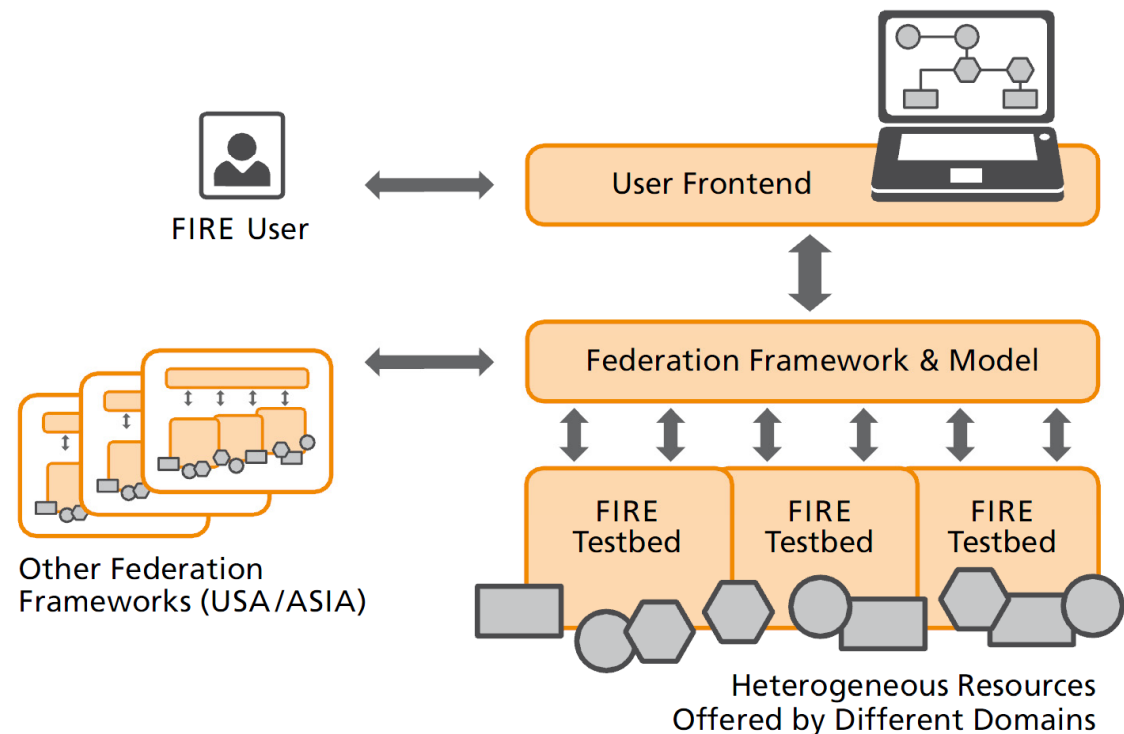
Panlab overview

- Initial federation and Panlab concepts started in 2006/2007 with the Panlab SSA (FP6)
- The FP7 IP Pan-European Laboratory Infrastructure Implementation (PII) started in 2008 and will run until Q4 2010.
- PII implements the Panlab concept of cross-domain, cross-layer resource federation
- Target is to create a large scale testing and experimentation facility by integrating existing and emerging testbeds
- Achieve scale and maintain independence through federation
 - that enables end-to-end interoperability testing of platforms, networks and services
 - that helps reducing the risks and costs of large-scale network infrastructure testing

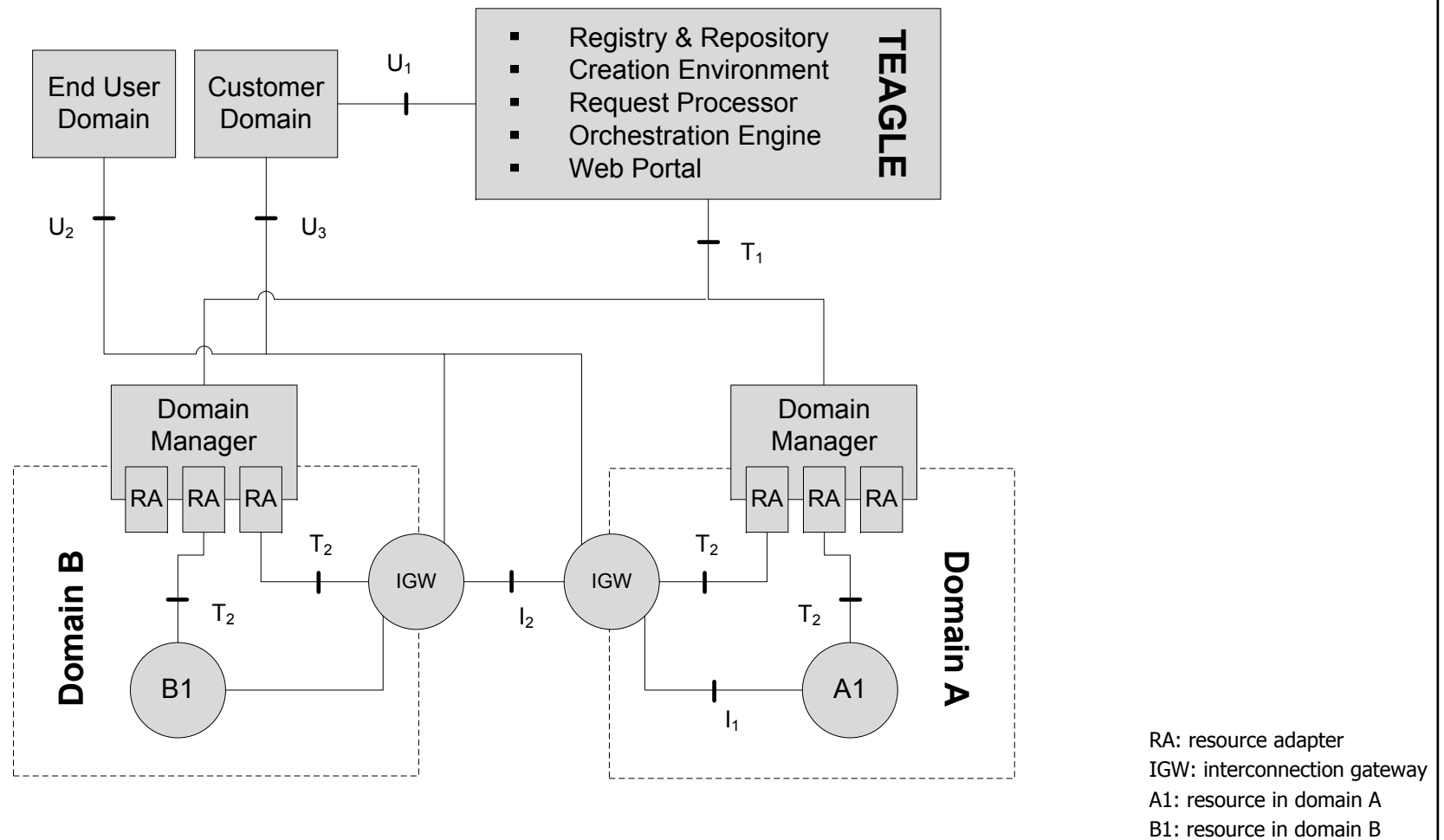


The high level Panlab architecture

- Various testbeds provide resources that are controlled via a federation framework and user frontend (Teagle)
- With the help of Teagle resources can be combined and configured for specific setups
- Panlab aims at satisfying broad R&D requirements by providing a generic resource control framework for distributed heterogeneous resources
- Federation with other frameworks is currently under investigation



Functions & Interfaces



Demo video

- A screen cast demonstration video was shown during the presentation. The video can soon be found here: <http://www.fire-teagle.org/tutorials.jsp>



The Demo

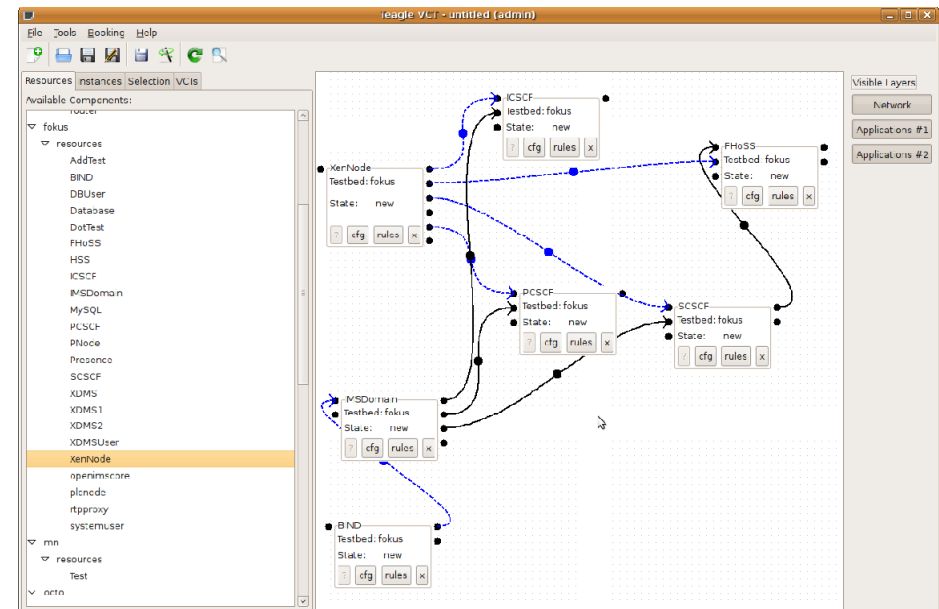
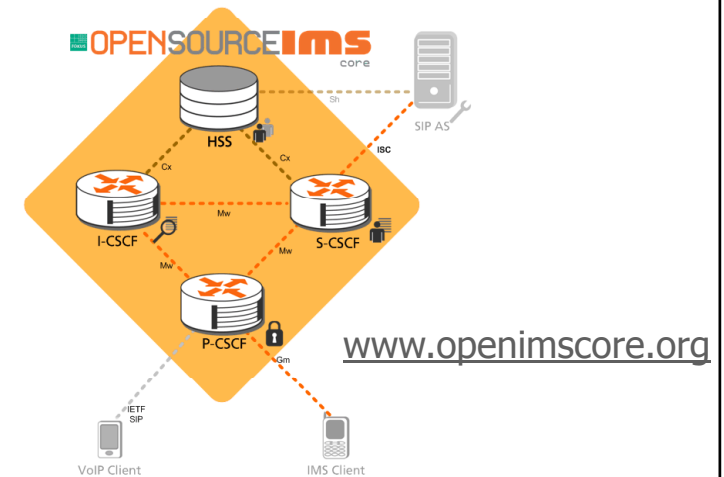


www.fire-teagle.org

configure

design custom
testbed

A custom testbed is designed with Teagle making use of infrastructure resources (hardware, virtual machines) and service layer resources (Open Source IMS core)



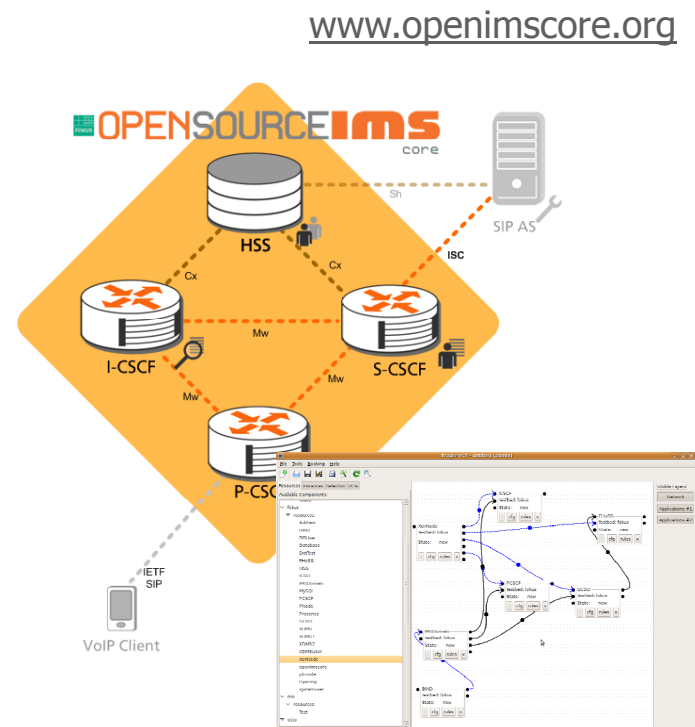
The Experiment

SIPNuke

Load generator
www.sipnuke.org

stress the core ...

... and break it



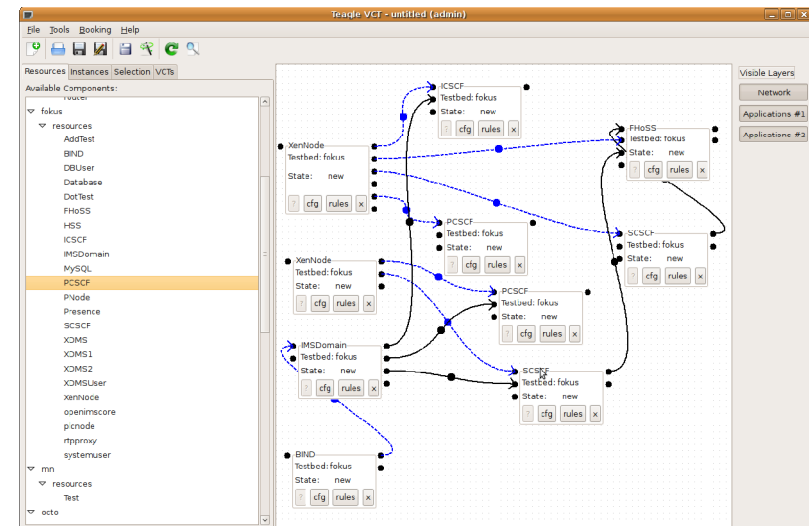
- * 50 IMS USER IDS
- * 5 REGISTER per second
- * increasing from 50 to 400 CALLS per second, looping through all users
- * UNREGISTER all users



Second experiment execution



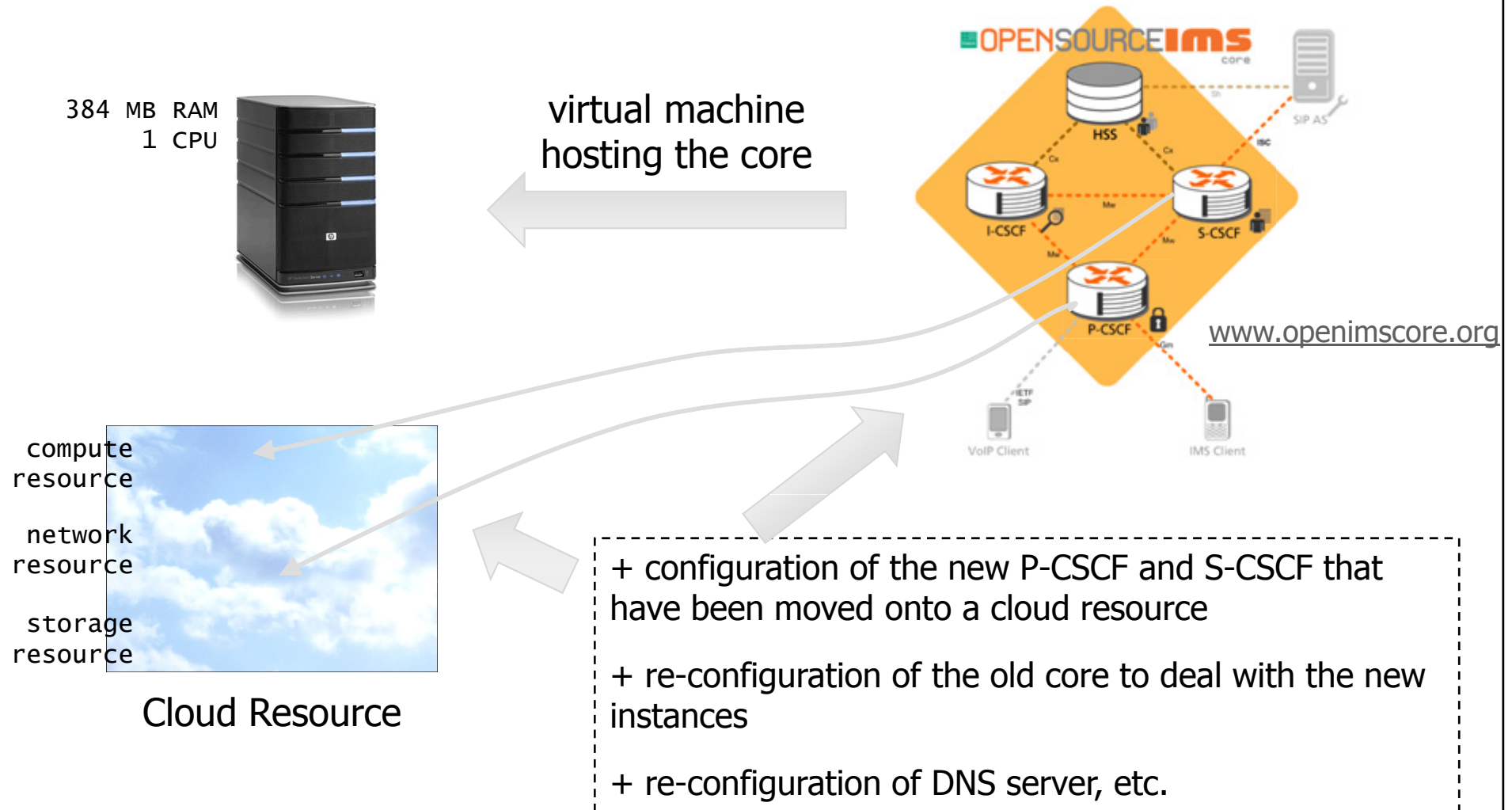
modify testbed



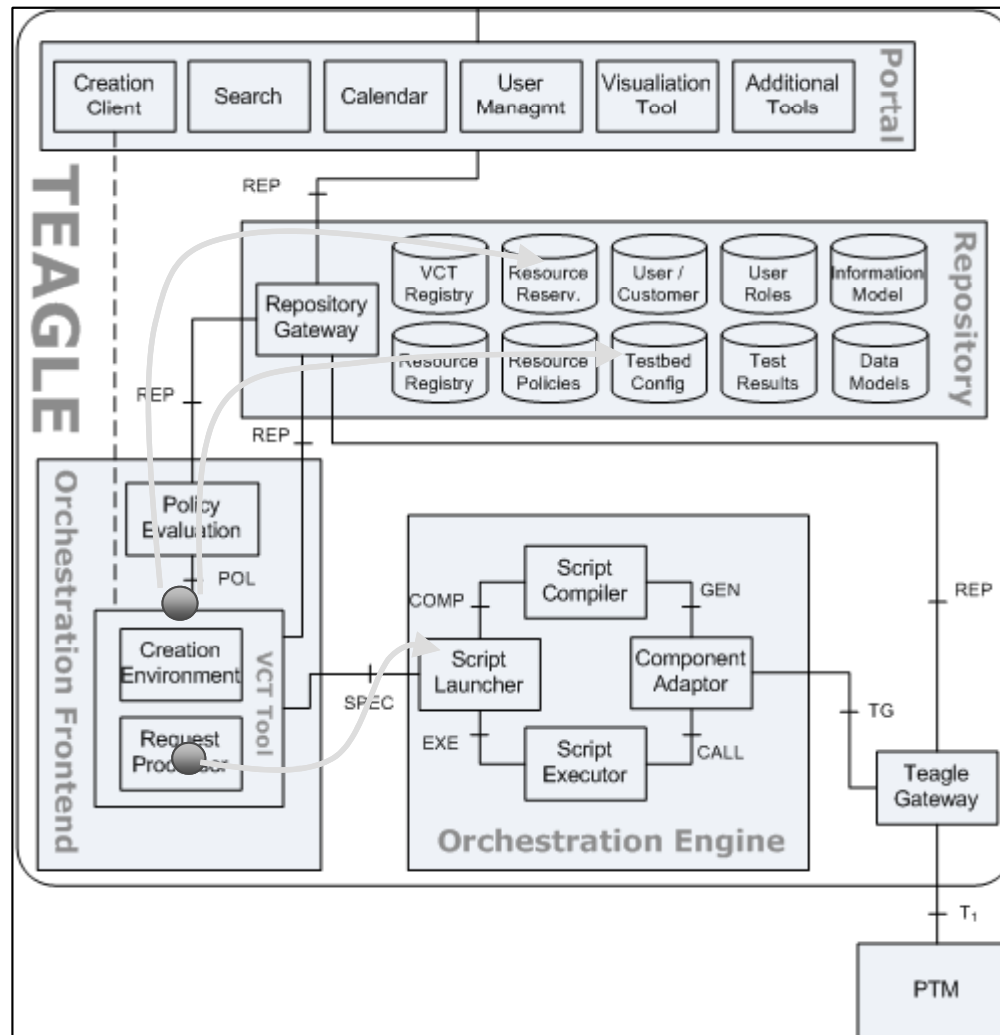
... re-execute the experiment



Modify the custom testbed



Behind the scenes



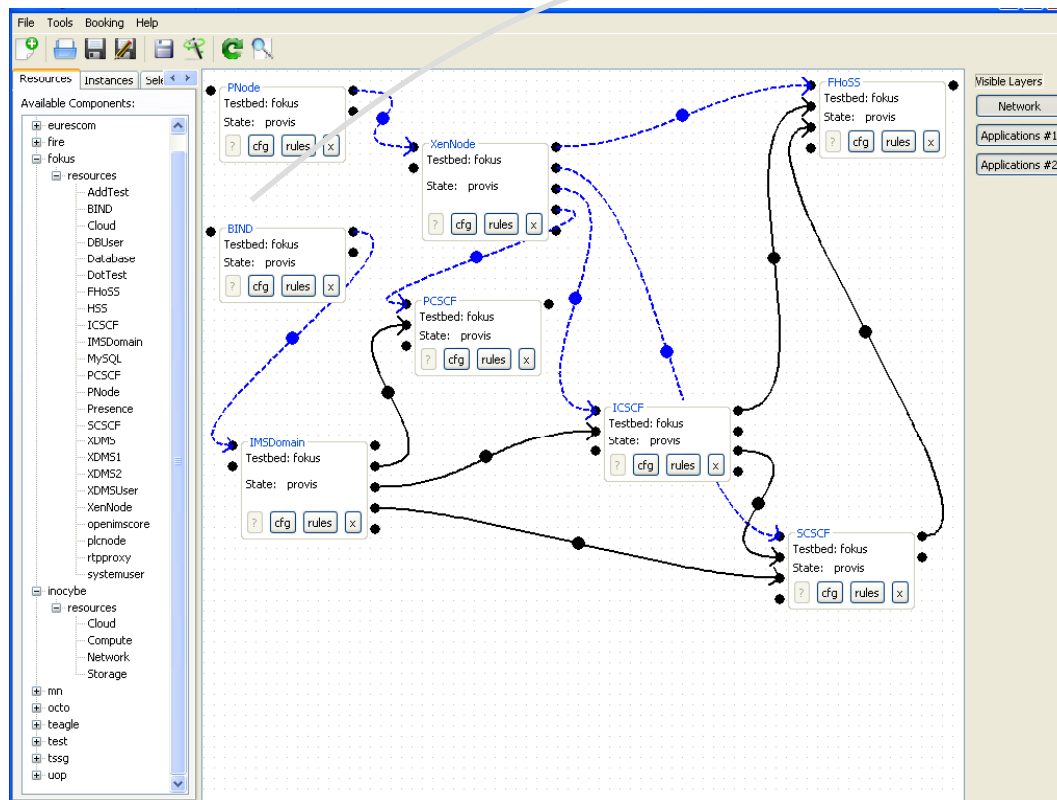
The Teagle tool helps reconfiguring the testbed and deploying new resource instances.

From the design environment requests are send to the repository to update resource configurations and reservations.

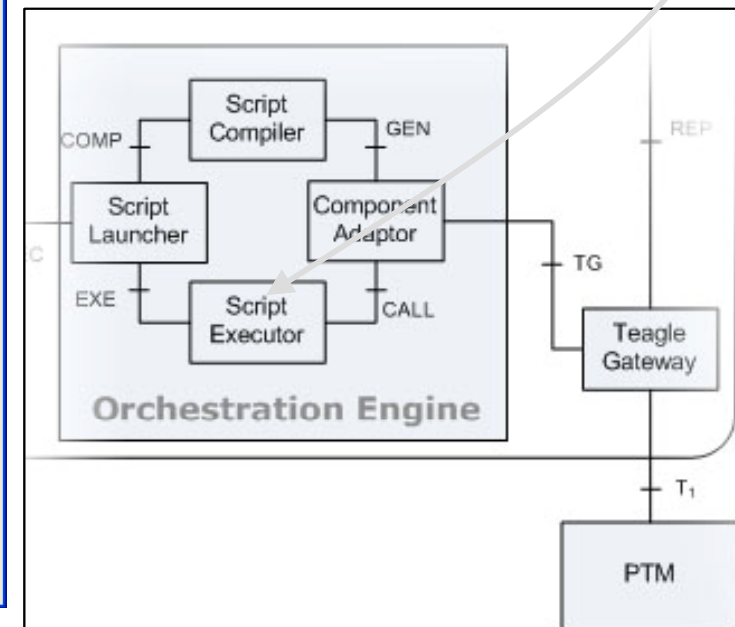
The topology-oriented testbed design is send to the Orchestration Engine by means of a XML document.

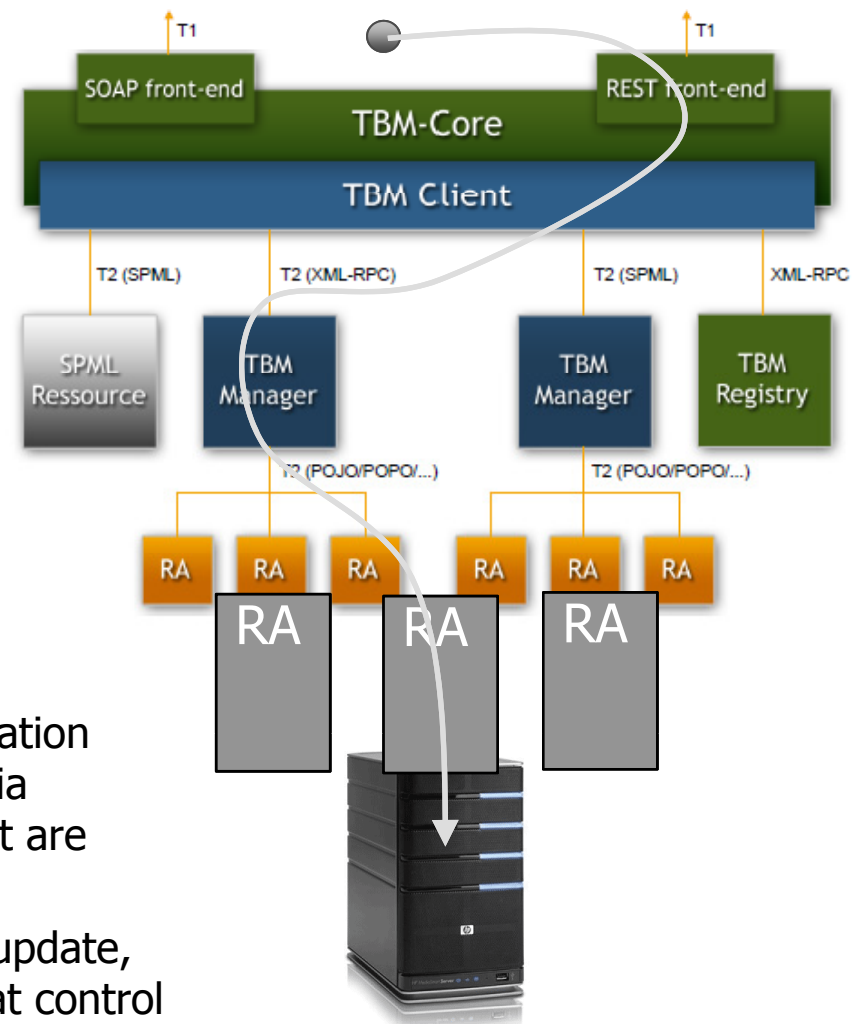
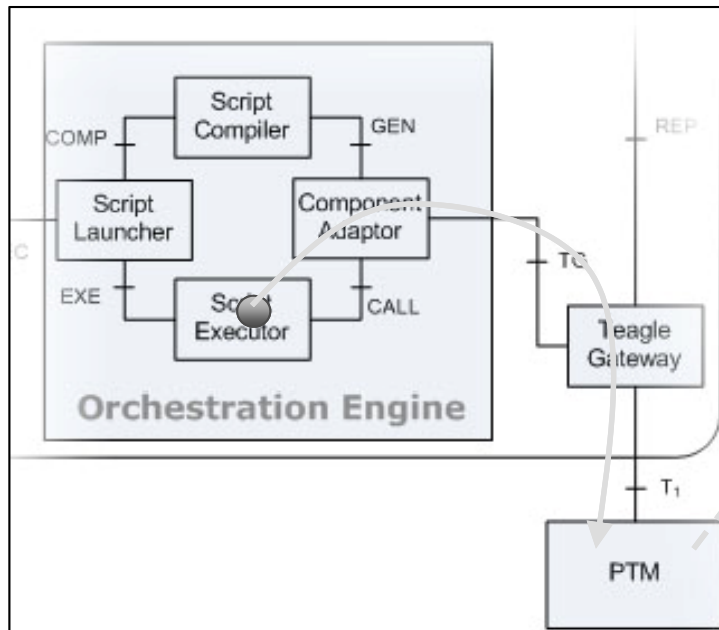


The Orchestration Engine transforms the design environment output into an executable script and executes it (resolving dependencies, etc.)



```
class HSSProvisioning01 (VBEEntity):  
    ## meta information  
    META = {  
        'orchestrate': {'args': ('userid',)},  
    }  
  
    def __init__(self, SESSION):  
        self.SESSION = SESSION  
        self.appld = PLUGINCONF.PLUGINID  
        self.spatelsystem = SpatelSystem(  
            SESSION, self.appld, 'HSSProvisioning01')  
    ## *** operation orchestrate ***  
    def orchestrate(self, userid):  
        ## in userid:String -> String  
        from voicebench.com.VariantManager import invokeVariant  
        return invokeVariant(self, 'orchestrate', userid)  
    def orchestrate_v0(self, userid):  
        ## in userid:String -> String  
        ## use this for fake implementation  
        result = "" ## default result  
        return result
```

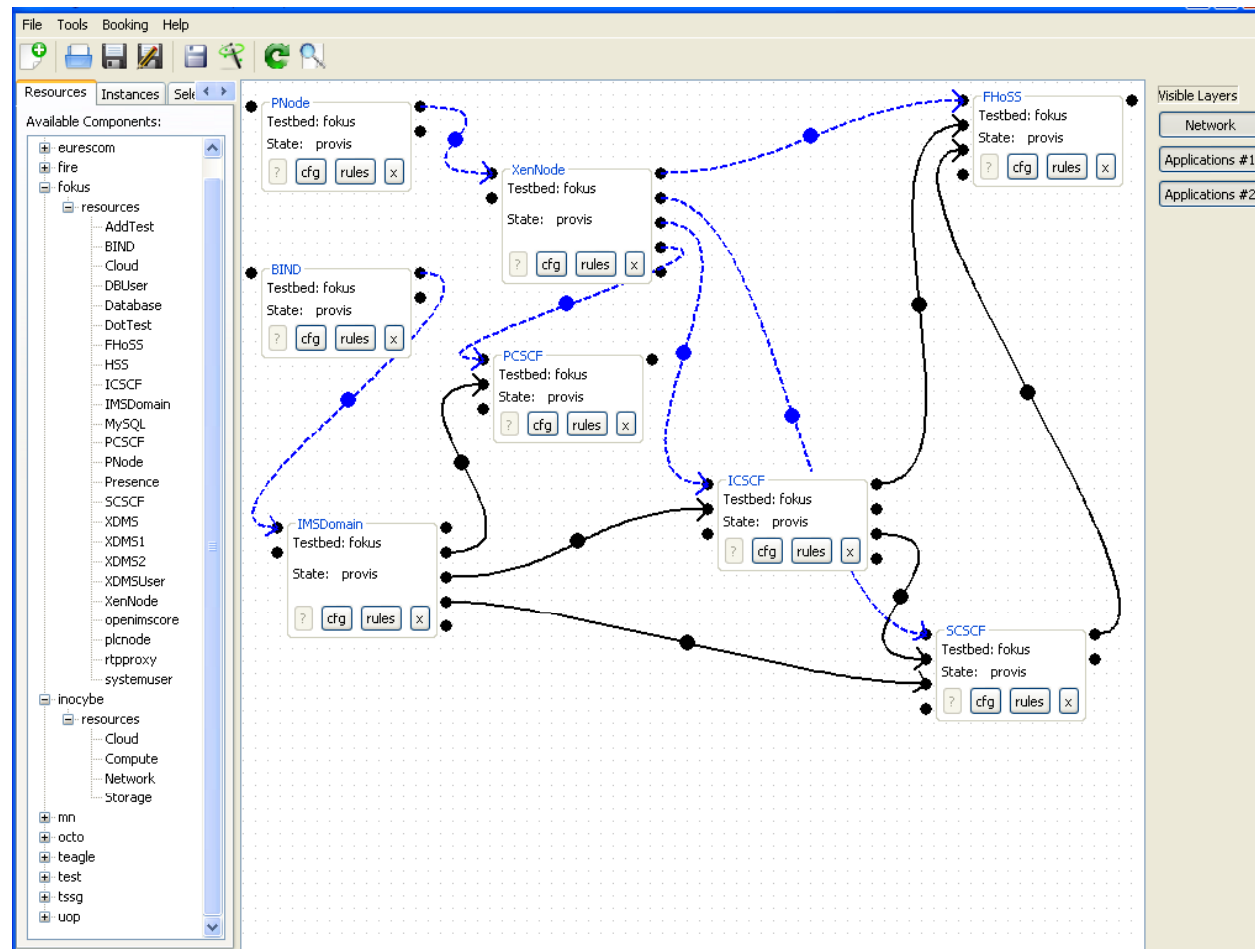




From the script executor provided by the Orchestration Engine, generic REST requests (CRUD) are sent via interface T1 to those domain managers (PTM) that are responsible for involved resources.

The PTM passes the generic CRUD (create, read, update, delete) requests to the resource adaptors (RA) that control the involved resources. Here, the generic commands are translated into resource specific requests and actions.

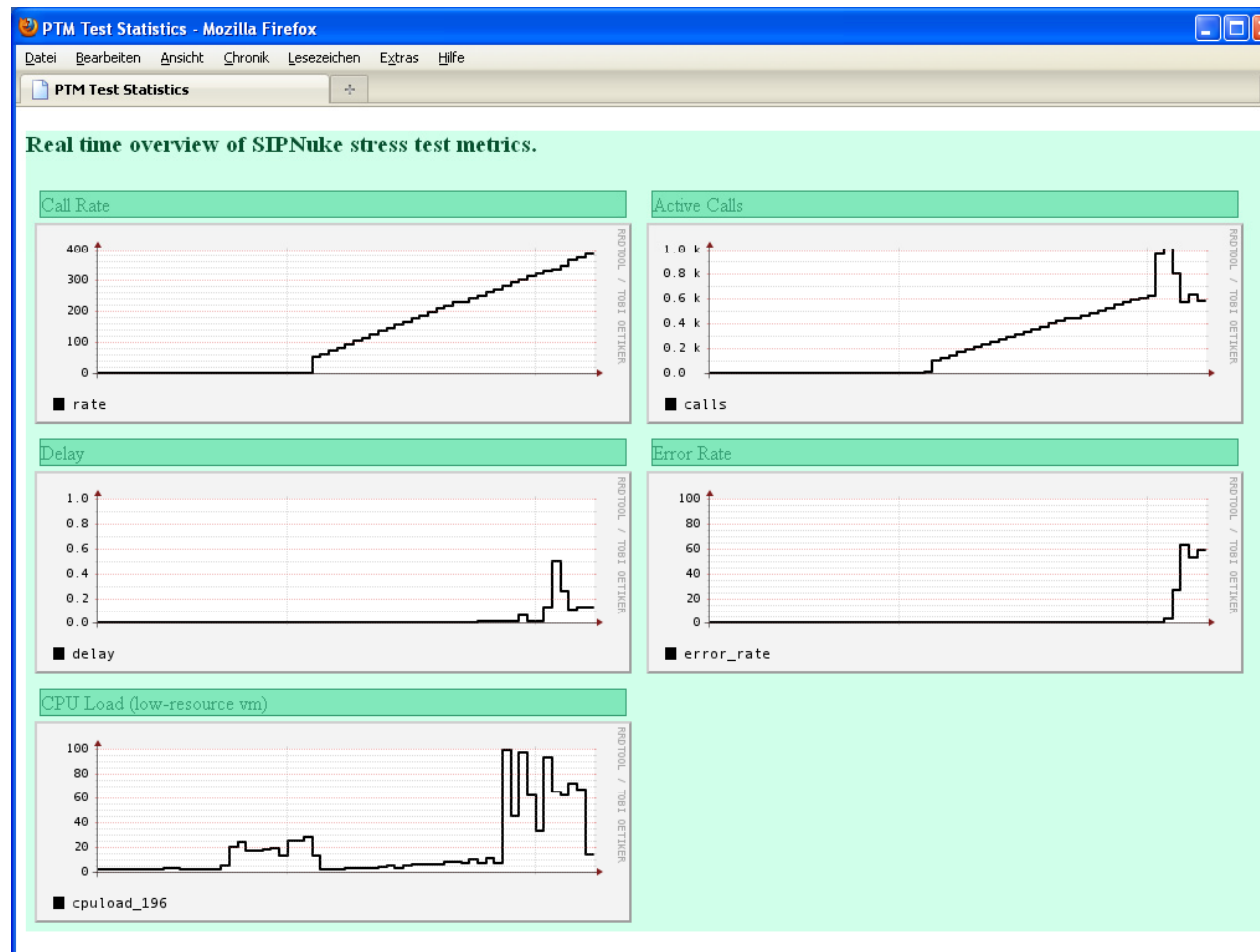
Some screenshots from the demo screen cast



Initial testbed design using the Teagle creation environment. The dotted lines reflect a containment relationship (e.g. the ICSCF is hosted by the XENnode). The solid lines represent a configuration reference.



Some screenshots from the demo screen cast

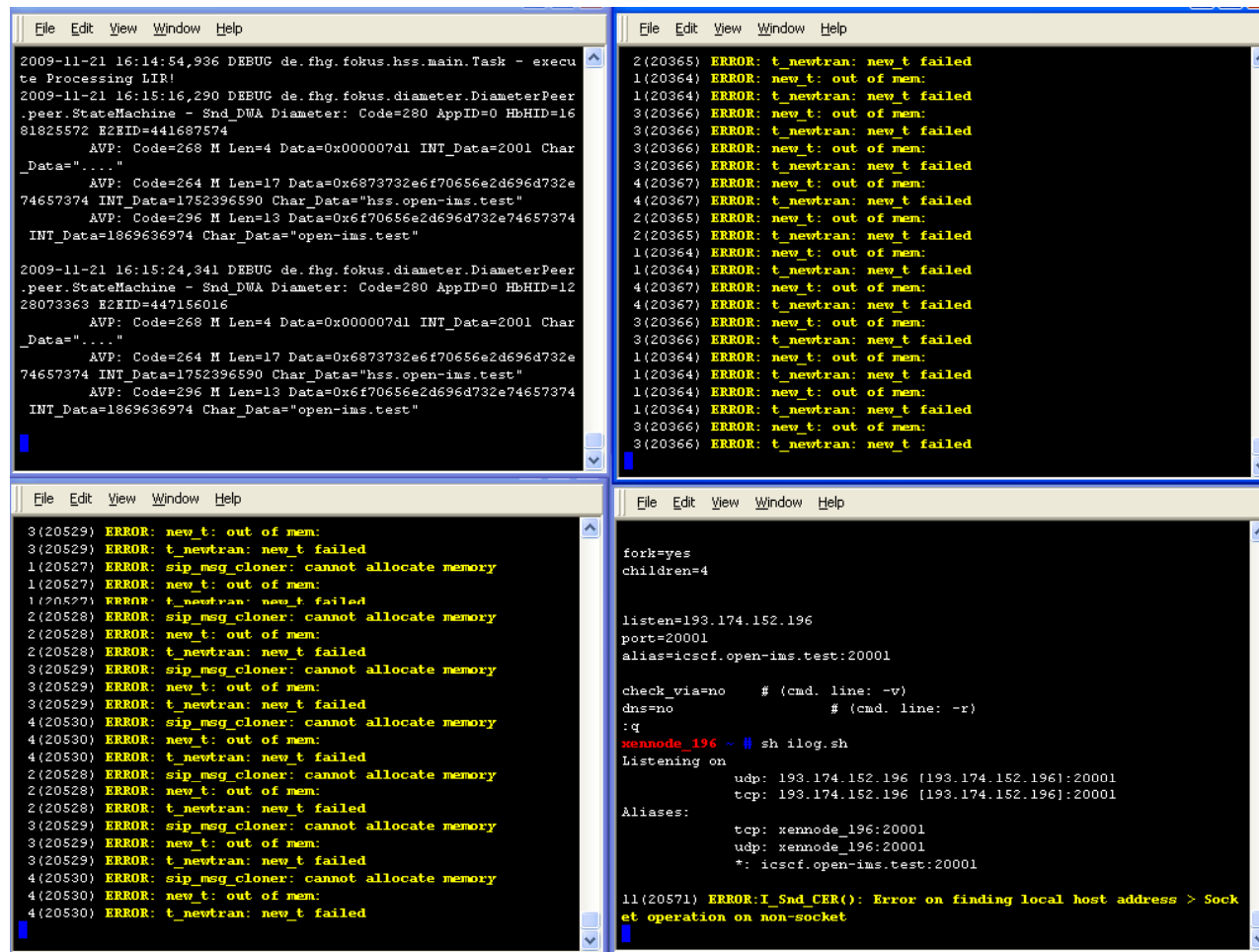


Execution of the first experiment. The CPU load of the small XEN reaches 100%, additionally the machine runs out of memory.

Therefore, delays and high call error rates are observed.



Some screenshots from the demo screen cast



```
2009-11-21 16:14:54,936 DEBUG de.fhg.fokus.hss.main.Task - execute Processing LIR!
2009-11-21 16:15:16,290 DEBUG de.fhg.fokus.diameter.DiameterPeer .peer.StateMachine - Snd_DWA Diameter: Code=280 AppID=0 HbHID=16 81825572 E2HID=441687574
AVP: Code=268 M Len=4 Data=0x000007d1 INT_Data=2001 Char_Data="...."
AVP: Code=264 M Len=17 Data=0x6873732e6f70656e2d696d732e74657374 INT_Data=1752396590 Char_Data="hss.open-ims.test"
AVP: Code=296 M Len=13 Data=0x6f70656e2d696d732e74657374 INT_Data=1869636974 Char_Data="open-ims.test"

2009-11-21 16:15:24,341 DEBUG de.fhg.fokus.diameter.DiameterPeer .peer.StateMachine - Snd_DWA Diameter: Code=280 AppID=0 HbHID=12 28073363 E2HID=447156016
AVP: Code=268 M Len=4 Data=0x000007d1 INT_Data=2001 Char_Data="...."
AVP: Code=264 M Len=17 Data=0x6873732e6f70656e2d696d732e74657374 INT_Data=1752396590 Char_Data="hss.open-ims.test"
AVP: Code=296 M Len=13 Data=0x6f70656e2d696d732e74657374 INT_Data=1869636974 Char_Data="open-ims.test"

3(20529) ERROR: new_t: out of mem:
3(20529) ERROR: t_newtran: new_t failed
1(20527) ERROR: sip_msg_cloner: cannot allocate memory
1(20527) ERROR: new_t: out of mem:
1(20527) ERROR: t_newtran: new_t failed
2(20528) ERROR: sip_msg_cloner: cannot allocate memory
2(20528) ERROR: new_t: out of mem:
2(20528) ERROR: t_newtran: new_t failed
3(20529) ERROR: sip_msg_cloner: cannot allocate memory
3(20529) ERROR: new_t: out of mem:
3(20529) ERROR: t_newtran: new_t failed
4(20530) ERROR: sip_msg_cloner: cannot allocate memory
4(20530) ERROR: new_t: out of mem:
4(20530) ERROR: t_newtran: new_t failed
2(20528) ERROR: sip_msg_cloner: cannot allocate memory
2(20528) ERROR: new_t: out of mem:
2(20528) ERROR: t_newtran: new_t failed
3(20529) ERROR: sip_msg_cloner: cannot allocate memory
3(20529) ERROR: new_t: out of mem:
3(20529) ERROR: t_newtran: new_t failed
4(20530) ERROR: sip_msg_cloner: cannot allocate memory
4(20530) ERROR: new_t: out of mem:
4(20530) ERROR: t_newtran: new_t failed

2(20365) ERROR: t_newtran: new_t failed
1(20364) ERROR: new_t: out of mem:
1(20364) ERROR: t_newtran: new_t failed
3(20366) ERROR: new_t: out of mem:
3(20366) ERROR: t_newtran: new_t failed
3(20366) ERROR: new_t: out of mem:
3(20366) ERROR: t_newtran: new_t failed
4(20367) ERROR: new_t: out of mem:
4(20367) ERROR: t_newtran: new_t failed
2(20365) ERROR: new_t: out of mem:
2(20365) ERROR: t_newtran: new_t failed
1(20364) ERROR: new_t: out of mem:
1(20364) ERROR: t_newtran: new_t failed
4(20367) ERROR: new_t: out of mem:
4(20367) ERROR: t_newtran: new_t failed
3(20366) ERROR: new_t: out of mem:
3(20366) ERROR: t_newtran: new_t failed
1(20364) ERROR: new_t: out of mem:
1(20364) ERROR: t_newtran: new_t failed
3(20366) ERROR: new_t: out of mem:
3(20366) ERROR: t_newtran: new_t failed

fork=yes
children=4

listen=193.174.152.196
port=20001
alias=icscf.open-ims.test:20001

check_via=no # (cmd. line: -v)
dns=no # (cmd. line: -r)
:q
xennode_196 ~ # sh ilog.sh
Listening on
udp: 193.174.152.196 [193.174.152.196]:20001
tcp: 193.174.152.196 [193.174.152.196]:20001

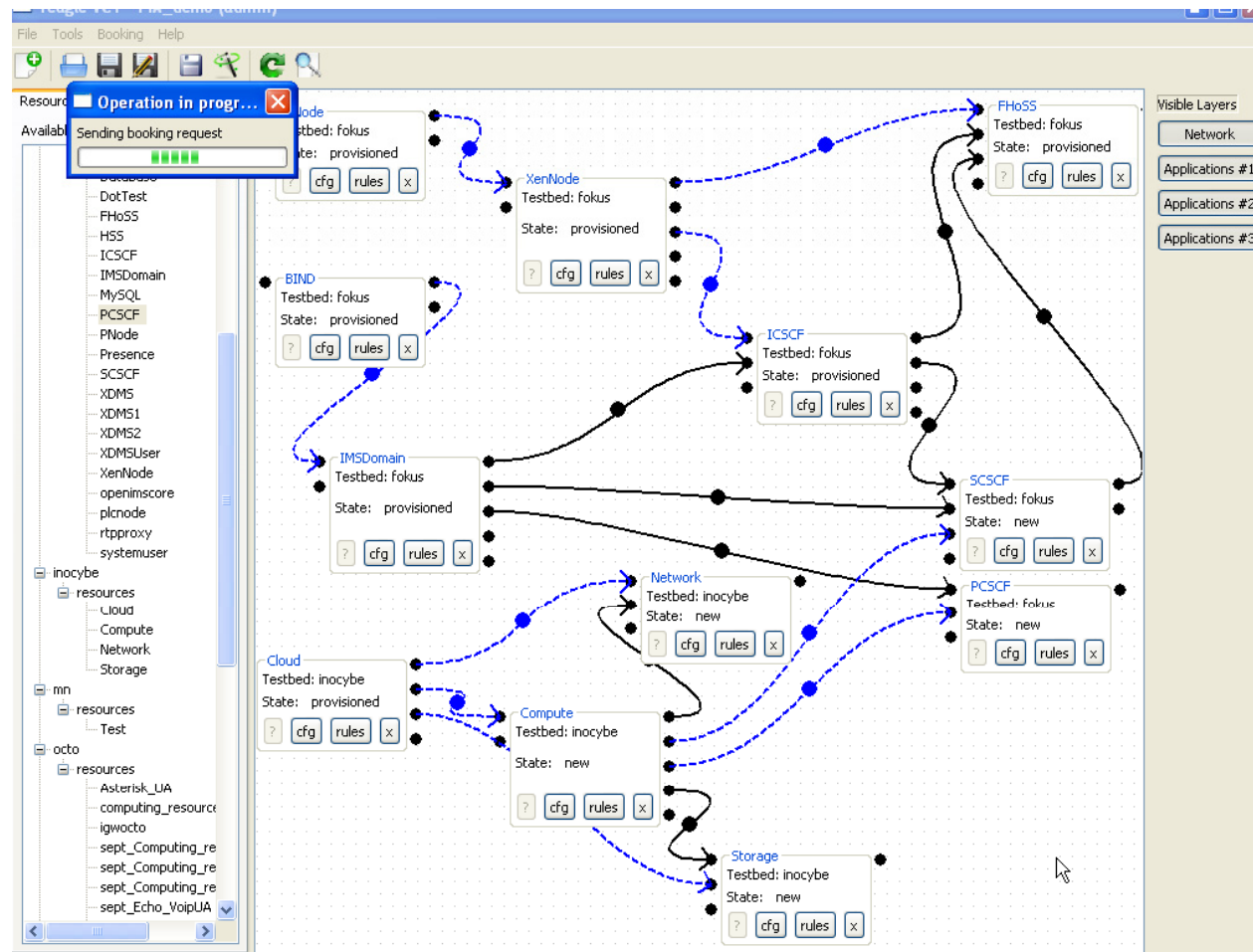
Aliases:
tcp: xennode_196:20001
udp: xennode_196:20001
*: icscf.open-ims.test:20001

11(20571) ERROR: I_Snd_CER(): Error on finding local host address > Socket operation on non-socket
```

The IMS Core logs show out of memory errors. At this stage the core is unstable and high delays and error rates are observed.



Some screenshots from the demo screen cast



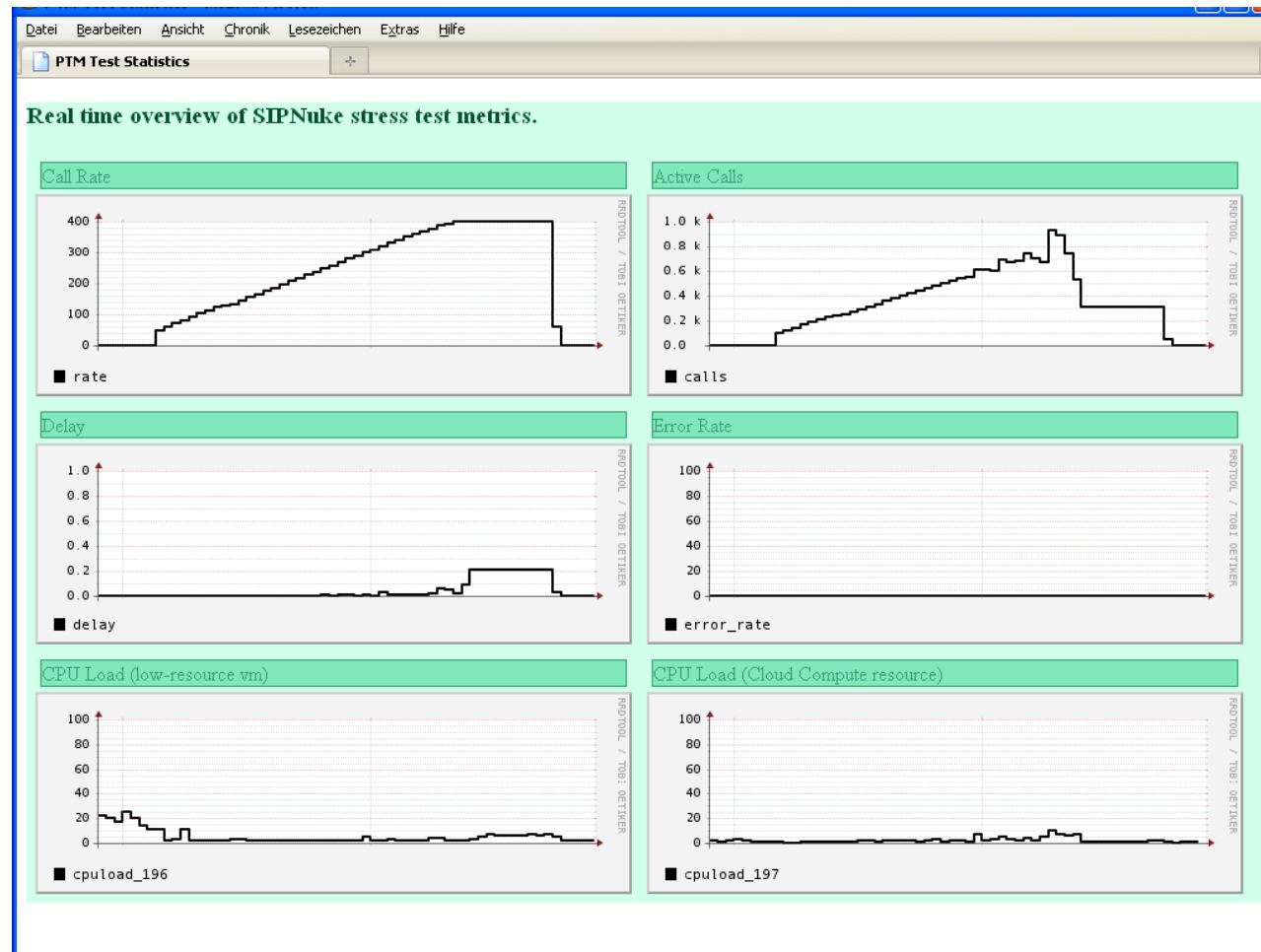
The testbed after the re-design.

The IMS Core services PCSCF and SCSCF have been moved onto the cloud computing resource and are now running "in the cloud".

The resources that are in *state: new* are additionally booked into the testbed by sending created and update commands to the involved domains.



Some screenshots from the demo screen cast



As expected the metrics show a much better performance now. 400 calls per second could be handled easily after modifying the testbed setup.

No errors and no higher delays than 0.2ms.

The CPU load on the computing resource (4 CPUs, 4GB mem) offered by the cloud stays below 20%.



Thank You & Contact



Sebastian Wahle

Dipl.-Ing.
Next Generation Network Infrastructures

Fraunhofer Institute for Open Communication Systems
Kaiserin-Augusta-Allee 31 | 10589 Berlin | Germany

Phone +49 30 3463 -7365 | Fax -8000
sebastian.wahle@fokus.fraunhofer.de
www.fokus.fraunhofer.de



Further Information

Some Links & Publications

- Panlab main website: <http://panlab.org/>
- Teagle website: <http://www.fire-teagle.org/>
- VCT Tool information & tutorials: <http://www.fire-teagle.org/tutorials.jsp>

- Last year's FIA book:

Anastasius Gavras, Halid Hrasnica, Sebastian Wahle, David Lozano, Denis Mischler, and Spyros Denazis. Towards the Future Internet - A European Research Perspective, chapter Control of Resources in Pan-European Testbed Federation, pages 67 - 78. IOS Press, May 2009. ISBN 978-1-60750-007-0.

- Forthcoming:

Konrad Campowsky, Anastasius Gavras, Bogdan Harjoc, Thomas Magedanz, and Sebastian Wahle. Pan-European Testbed and Experimental Facility Federation – Architecture Refinement and Implementation. Inderscience International Journal of Communication Networks and Distributed Systems (IJCNDs), Special Issue: Recent Advances in Test-bed Driven Networking Research. Accepted for publication in 2010.

Konrad Campowsky, Thomas Magedanz, and Sebastian Wahle. Resource Management in Large Scale Experimental Facilities: Technical Approach to Federate Panlab and PlanetLab. In 12th IEEE/IFIP Network Operations and Management Symposium (NOMS 2010). IEEE/IFIP, 2010. Accepted for publication in 2010.

- More Panlab publications: <http://www.panlab.net/publications.html>

