University of Duisburg-Essen, Institute for Experimental Mathematics

An Introduction to Reliable Server Pooling and the *RSPLIB* Implementation

Dr. Thomas Dreibholz

Institute for Experimental Mathematics

University of Duisburg-Essen

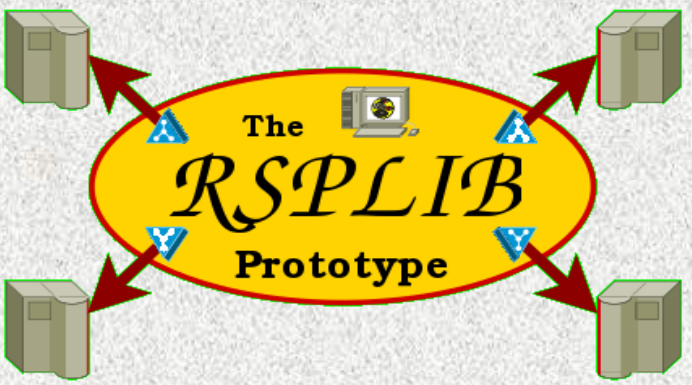
dreibh@iem.uni-due.de

http://www.exp-math.uni-essen.de/~dreibh

UNIVERSITÄT DUISBURG ESSEN

Table of Contents

UNIVERSITÄT DUISBURG ESSEN

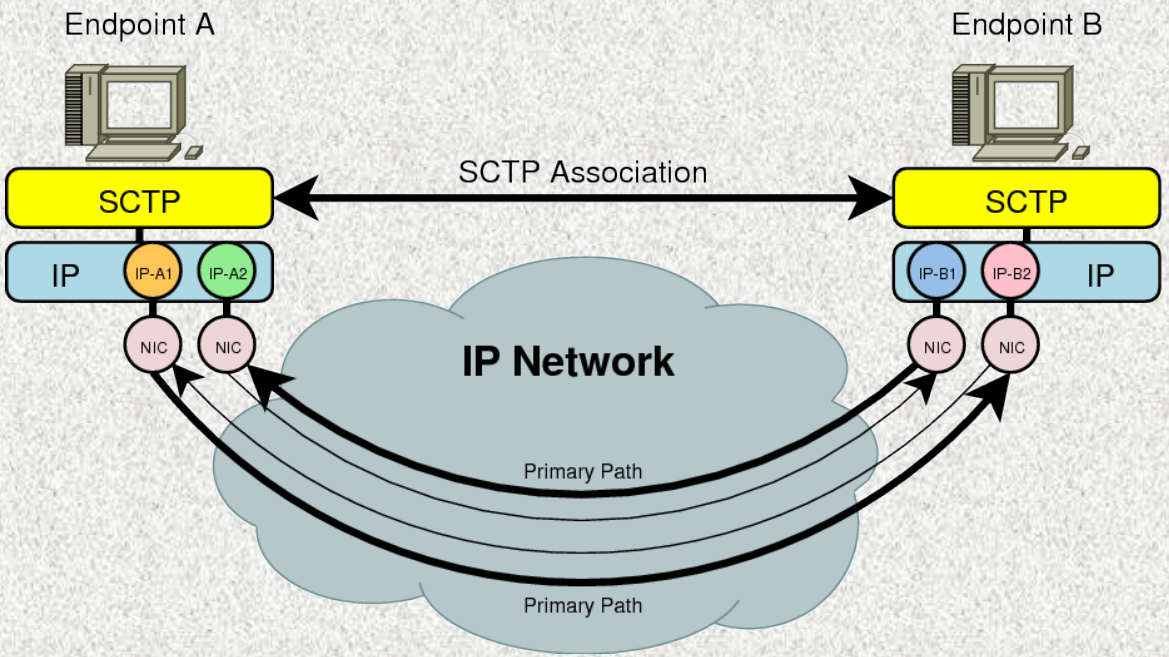


Thomas Dreibholz's Reliable Server Pooling Page
http://tdrwww.exp-math.uni-essen.de/dreibholz/rserpool/

Motivation (I)

UNIVERSITÄT DUISBURG ESSEN

- Original Motivation:
  - Telephone signalling (SS7 protocol) over IP networks
  - Strict requirements on availability
- The Stream Control Transmission Protocol (SCTP) [RFC 2960]
  - „TCP Next Generation“
  - **Multi-Homing**
  - Add-IP: dynamic address reconfiguration
  - Multi-Streaming
  - Message-Framing
  - Protection against DoS
    - 4-way handshake
    - „Verification Tag“



- SCTP protects against various network problems, but ...
- ... not against a **server failure**

⇒ Concept for **server redundancy** is **required**

Motivation

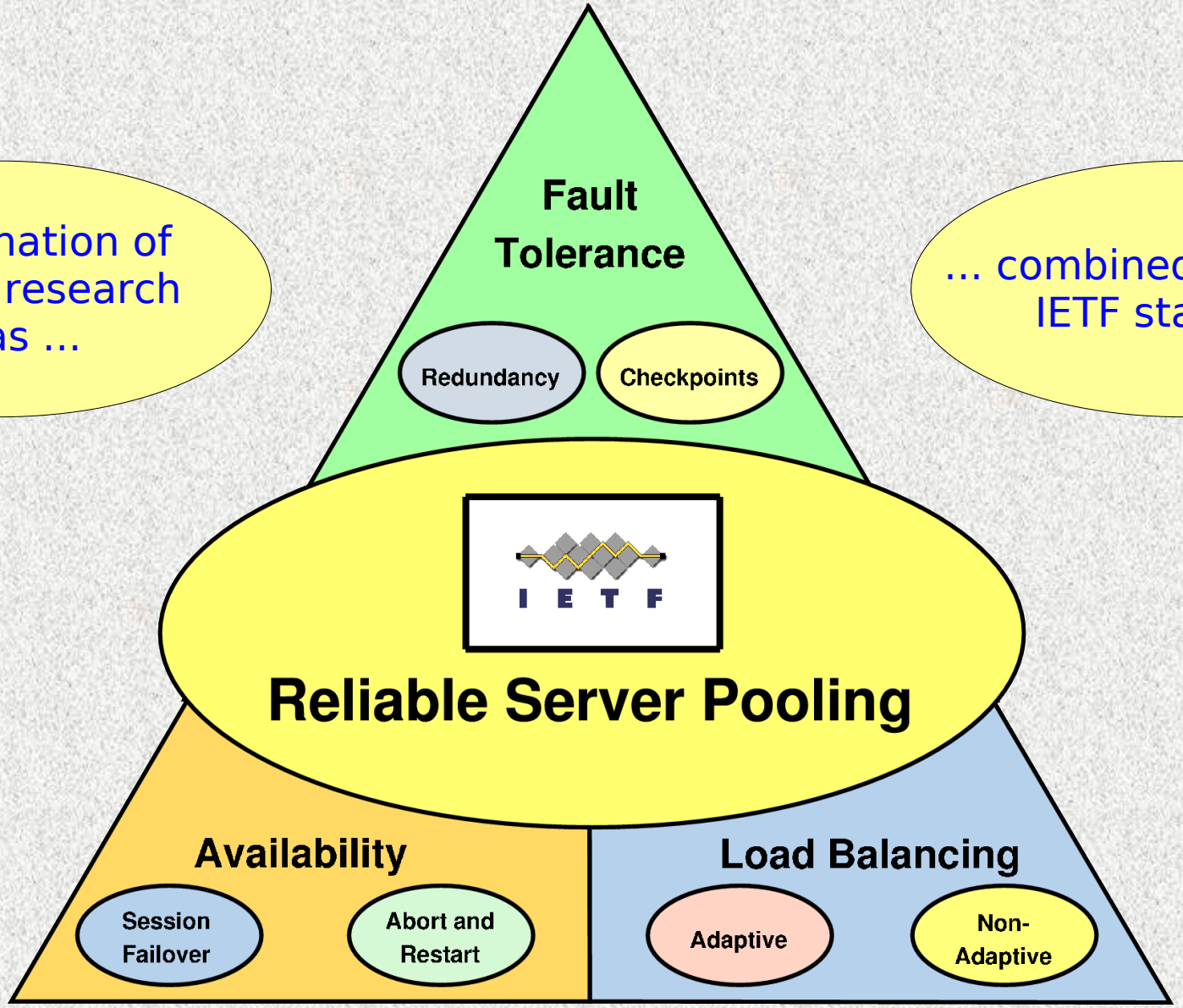
UNIVERSITÄT DUISBURG ESSEN

- Motivation of Reliable Server Pooling (RSerPool):
  - Unified, application-independent solution for service availability
  - Not available before => Foundation of the IETF RSerPool Working Group
- Application Scenarios for RSerPool:
  - Main motivation: **Telephone Signalling (SS7) over IP**
  - Under discussion by the IETF:
    - **Load Balancing**
    - Voice over IP (VoIP) with SIP
    - IP Flow Information Export (IPFIX)
  - ... and many more!
- Requirements for RSerPool:
  - **“Lightweight”** (low resource requirements, e.g. embedded devices!)
  - **Real-Time** (quick failover)
  - **Scalability** (e.g. to large (corporate) networks)
  - **Extensibility** (e.g. by new server selection rules)
  - **Simple** (automatic configuration: “just turn on, and it works!”)

Related Research Areas

UNIVERSITÄT DUISBURG ESSEN

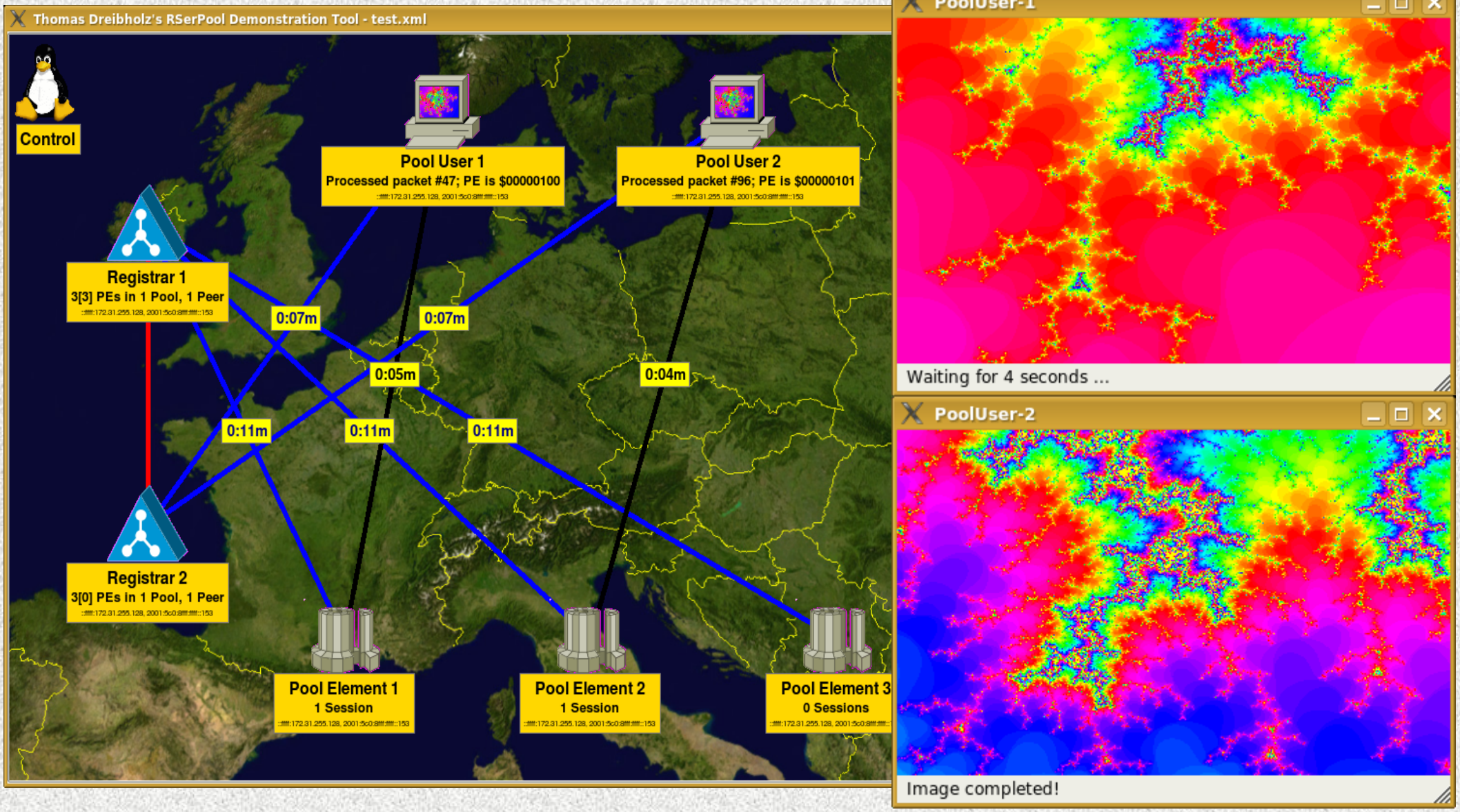
A combination of different research areas ...



... combined into a new IETF standard!

What is „Reliable Server Pooling“? Prototype Demonstration

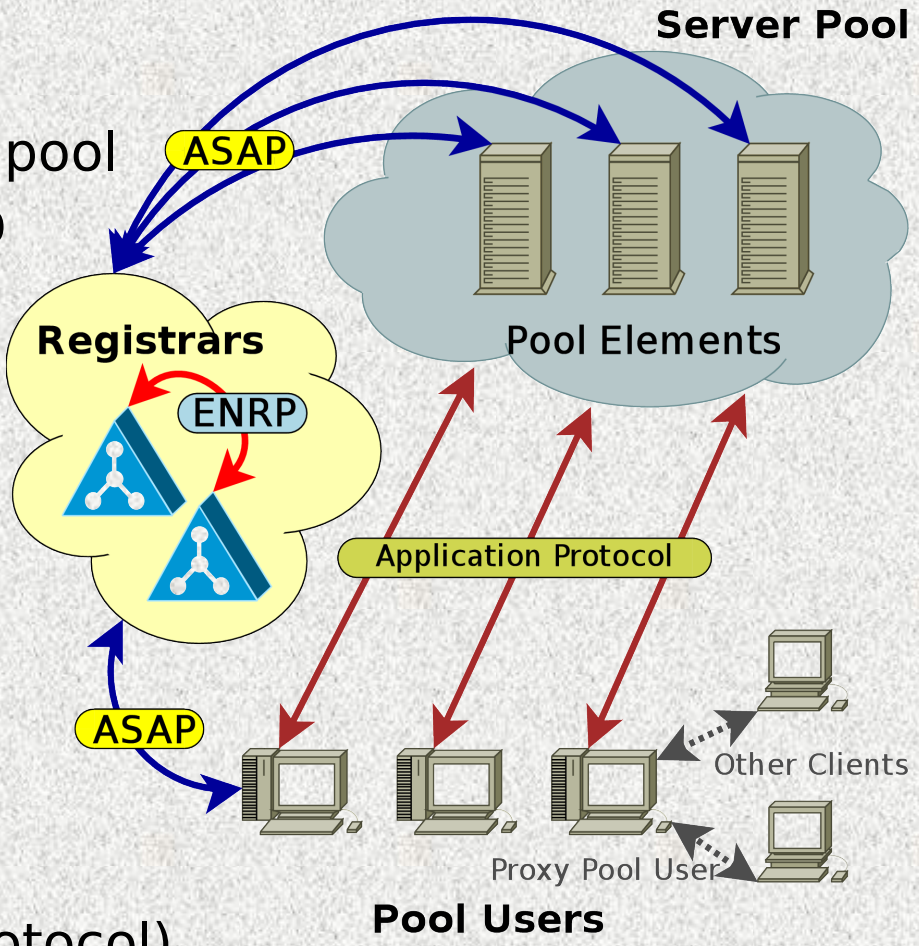
UNIVERSITÄT DUISBURG ESSEN



Reliable Server Pooling (RSerPool)

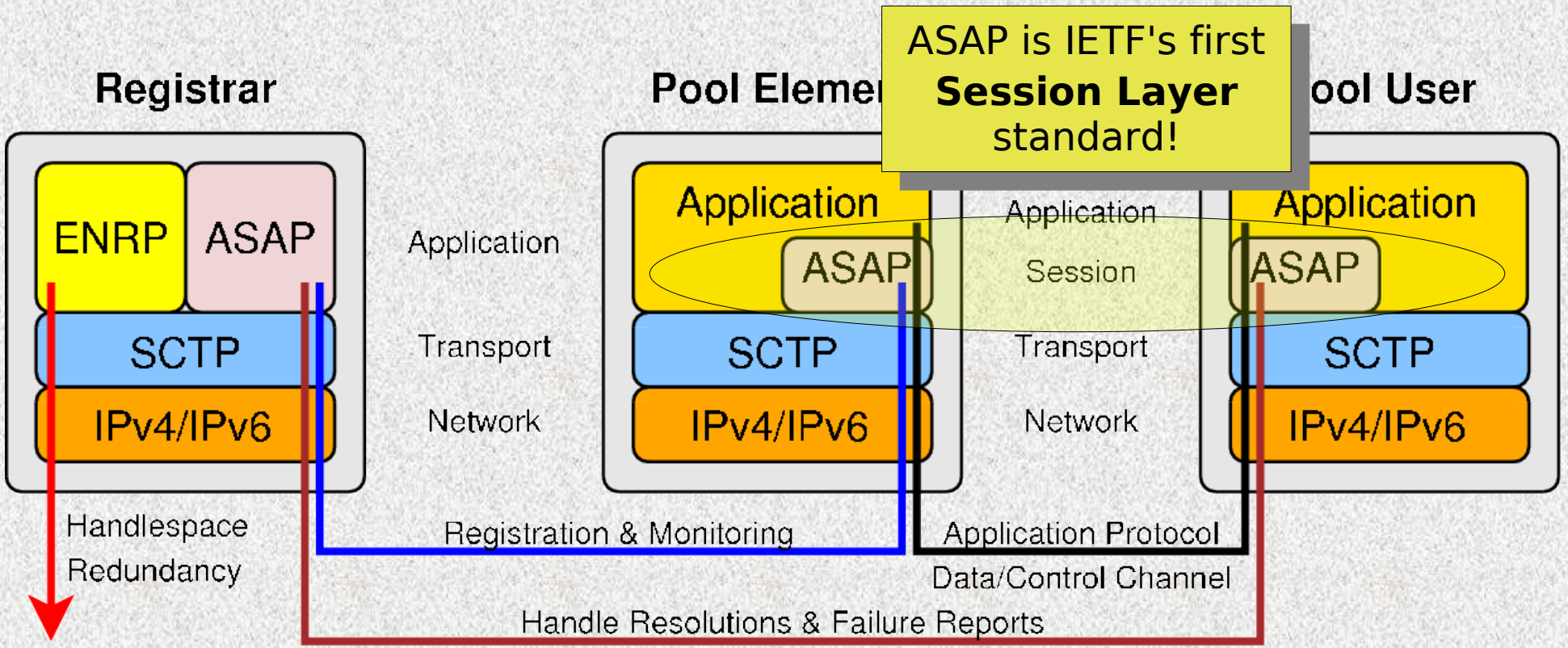
UNIVERSITÄT DUISBURG ESSEN

- Terminology:
  - **Pool Element (PE)**: Server
  - Pool: Set of PEs
  - PE ID: ID of a PE in a pool
  - Pool Handle: Unique pool ID
  - Handlespace: Set of pools
  - Pool **Registrar (PR)**
  - **Pool User (PU)**: Client
  - Support for Existing Applications
    - Proxy Pool User (PPU)
    - Proxy Pool Element (PPE)



- Protocols:
  - **ASAP** (Aggregate Server Access Protocol)
  - **ENRP** (Endpoint Handlespace Redundancy Protocol)

The RSerPool Protocol Stack

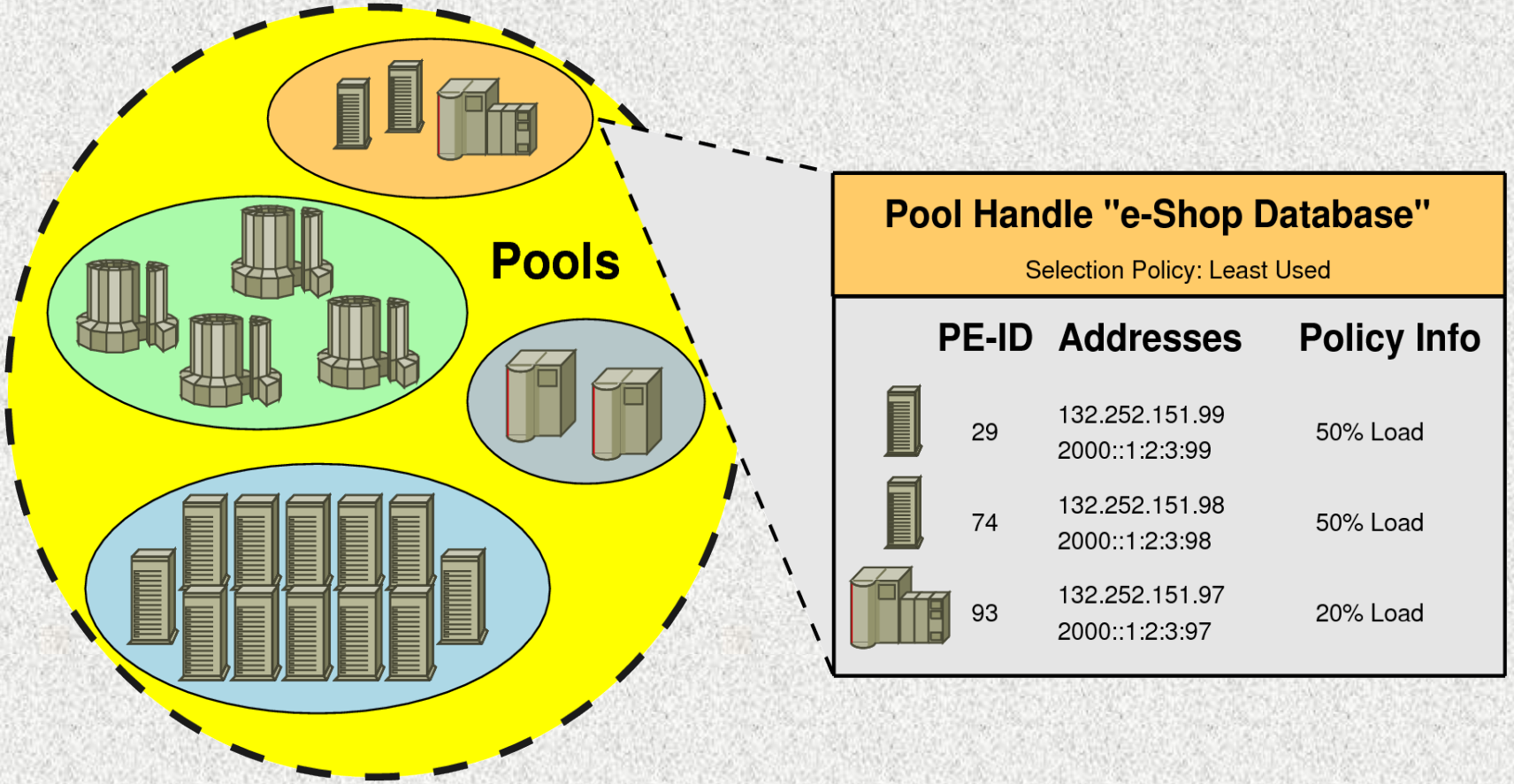


- Aggregate Server Access Protocol (ASAP)
  - PR ⇔ PE: Registration, Deregistration and Monitoring by Home-PR (PR-H)
  - PR ⇔ PU: Server Selection, Failure Reports
- Endpoint Handlespace Redundancy Protocol (ENRP)
  - PR ⇔ PR: Handlespace Synchronisation

The Handlespace and its Contents

UNIVERSITÄT DUISBURG ESSEN

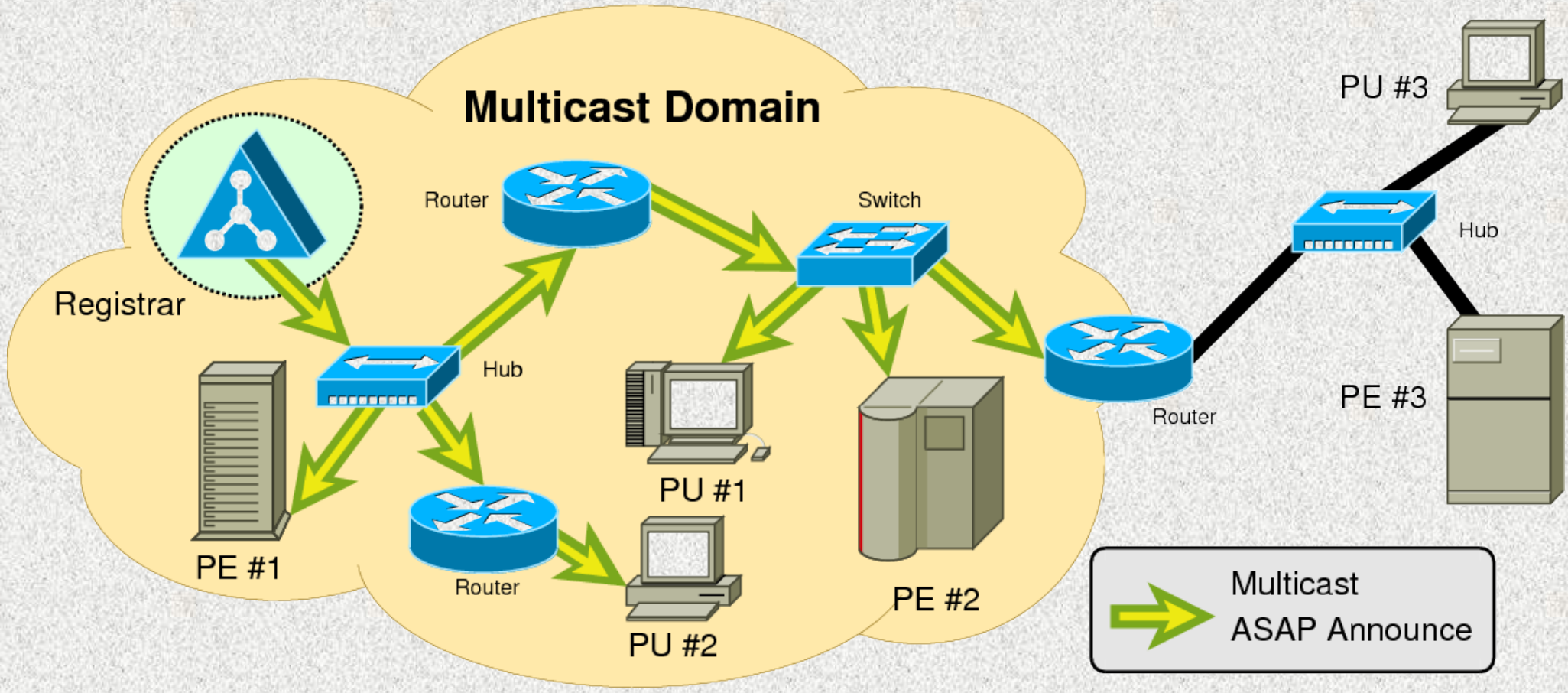
- Handlespace = set of non-empty pools
- “flat”, i.e. no hierarchy for PHs
  - very efficiently realizable (see [FGCN2007-HsMgt][Contel2005])



Automatic Configuration by Registrar Announces

UNIVERSITÄT DUISBURG ESSEN

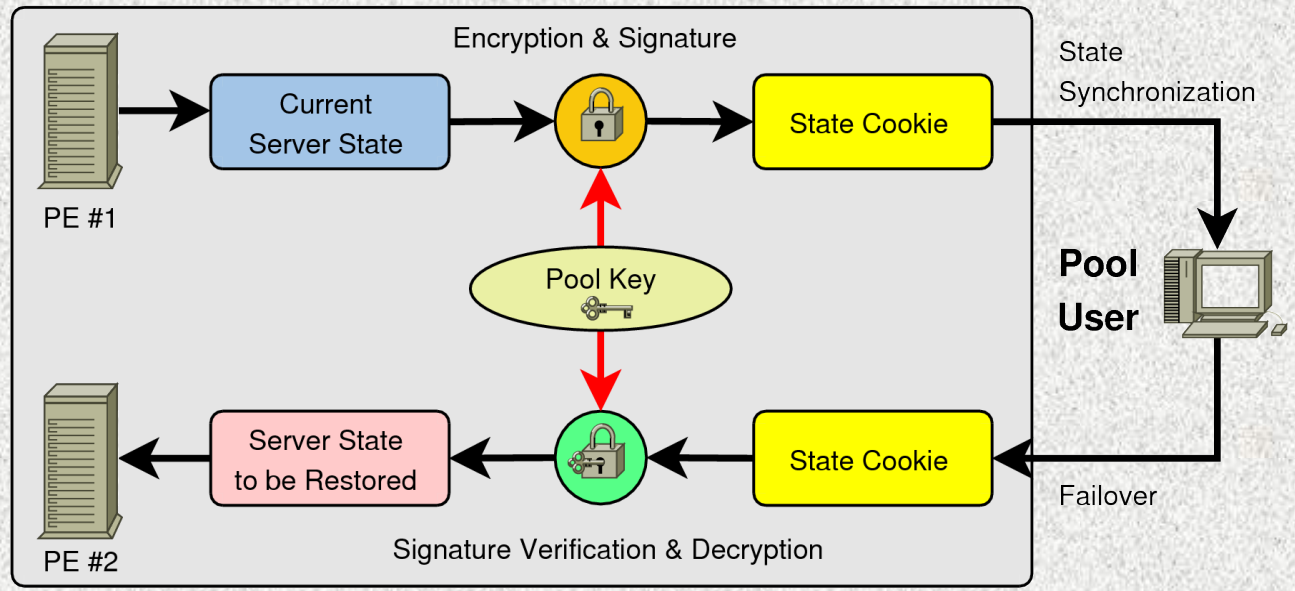
- Transport of the announces
  as UDP packets via IP multicast

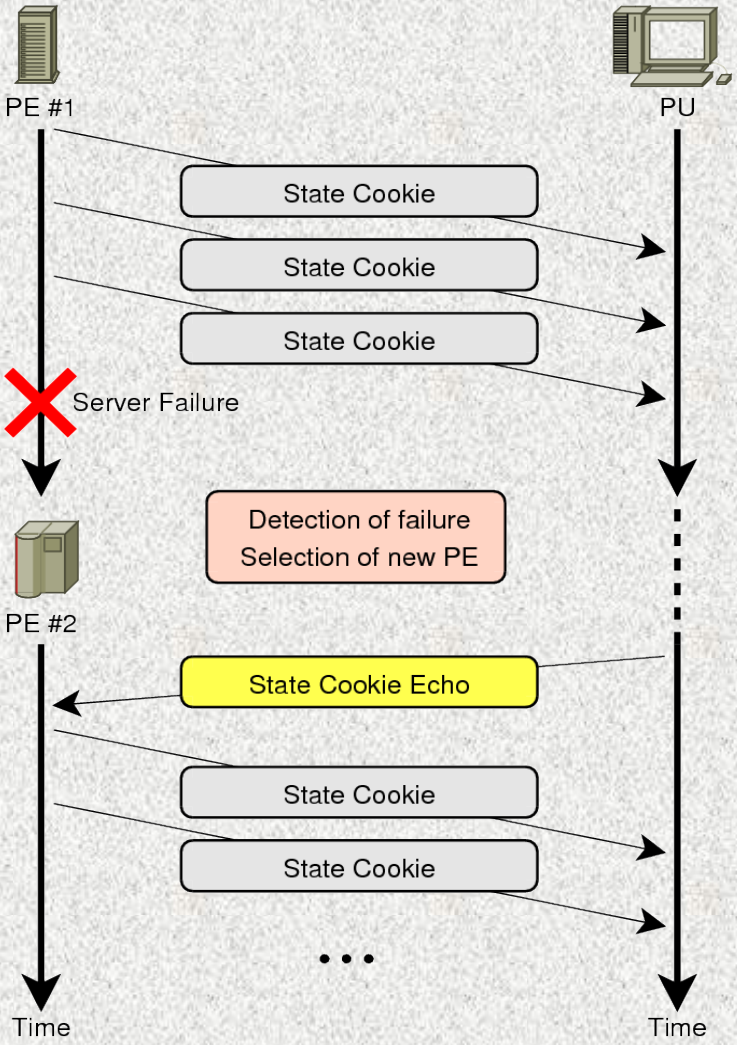


Session Failover using Client-Based State Sharing

UNIVERSITÄT DUISBURG ESSEN

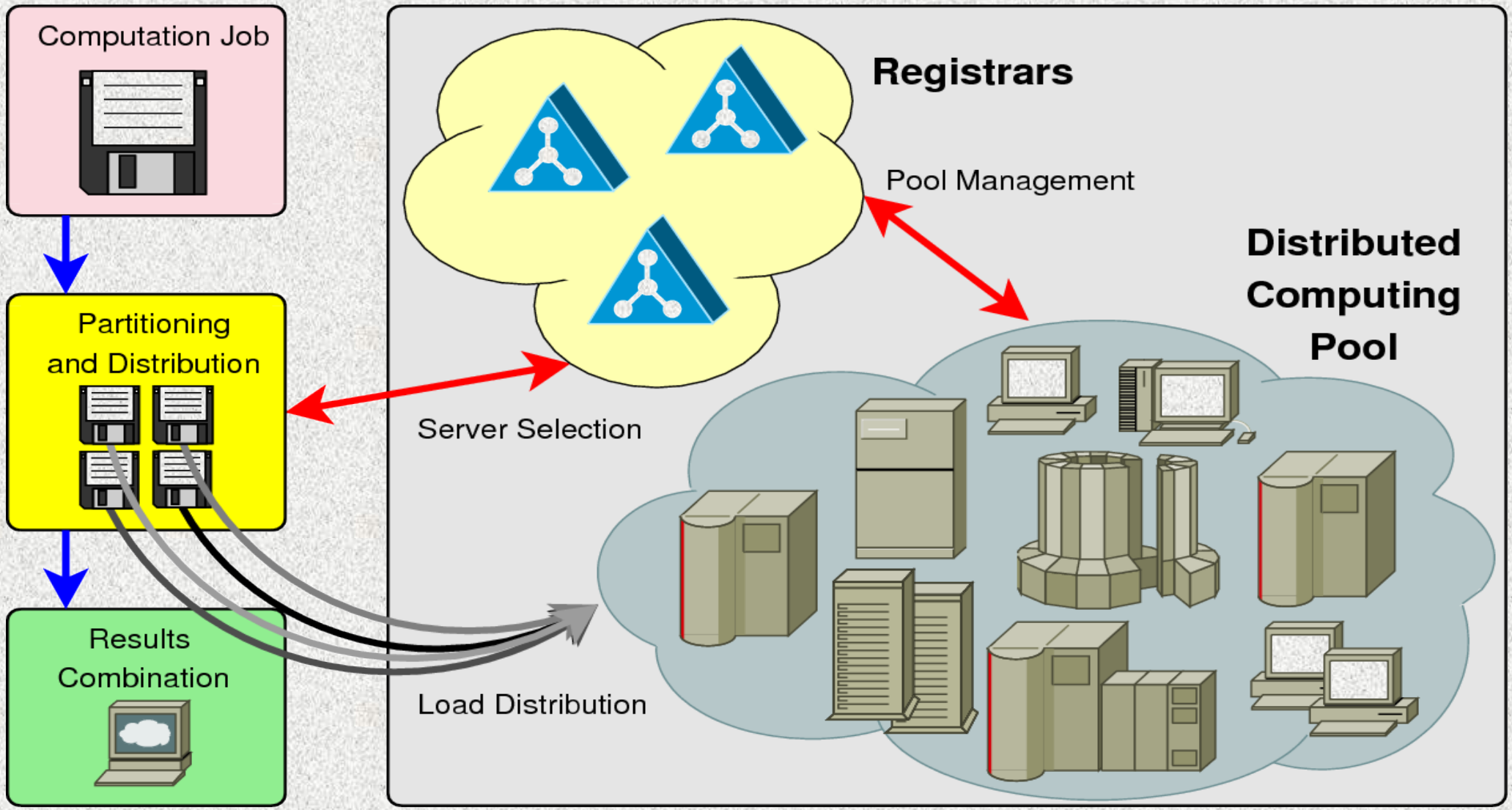
- Necessary to handle failover:
  A new PE must be able to recover the session state of the old PE
- Simple solution for many applications:
  Usage of „state cookies“ [LCN2002]
  Now part of the ASAP protocol!





RSerPool Application Scenario: Real-Time Distributed Computing

UNIVERSITÄT DUISBURG ESSEN

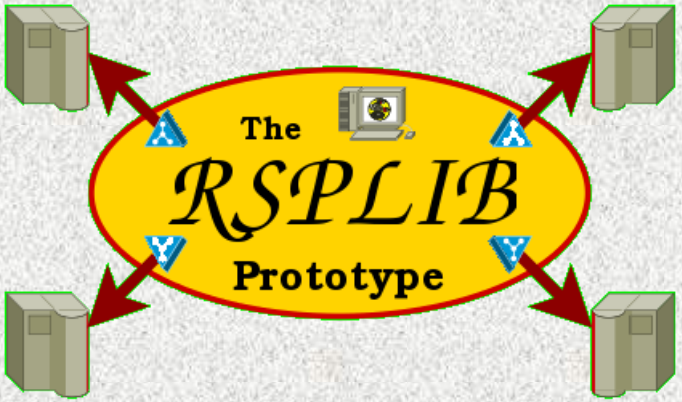


Described in [draft-dreibholz-rserpool-distcomp-03.txt]

The *RSPLIB* Implementation

UNIVERSITÄT DUISBURG ESSEN

- Design decisions:
  - Open Source, GPLv3 license
  - Platform independence
    - Systems: Linux, FreeBSD, MacOS X, Solaris
    - CPUs: x86, x86_64, PPC, MIPS
  - Implemented in ANSI-C
- Basic components:
  - *RSPLIB* library for PUs and PEs
    - ASAP protocol (PU/PE side)
  - Registrar
    - ASAP protocol (PR side)
    - ENRP protocol
  - Demo system and many examples



Developed in cooperation with Siemens AG, Munich
Supported by BMBF and DFG

Thomas Dreibholz's Reliable Server Pooling Page
http://tdrwww.iem.uni-due.de/dreibholz/rserpool/

Installation

UNIVERSITÄT DUISBURG ESSEN

- Download of the source archive:
  - http://tdrwww.iem.uni-due.de/dreibholz/rserpool/
- Dependencies:
  - *lksctp* package (Library for access to kernel SCTP features)
    (Alternative: our own userland SCTP implementation *sctplib/socketapi*)
  - *Qt3* developer files package (for fractal graphics demo, optional)
  - *BZip2* developer files package (for CalcApp test application)
  - C++ compiler (for the example applications)
  - Debian/Ubuntu:
    ```
    sudo apt-get install libsctp-dev libbz2-dev libqt3-mt-dev g++
    ```
- Compiling the sources:
  - `tar xzf rsplib-<Version>.tar.gz`
  - `cd rsplib-<Version>`
  - `export QTDIR=/usr/share/qt3` (Debian/Ubuntu, may be different for others)
  - `./configure --enable-kernel-sctp --enable-qt`
  - `make`
  - `sudo modprobe sctp` (if necessary: load kernel module for SCTP)

A Small Test Setup

UNIVERSITÄT DUISBURG ESSEN

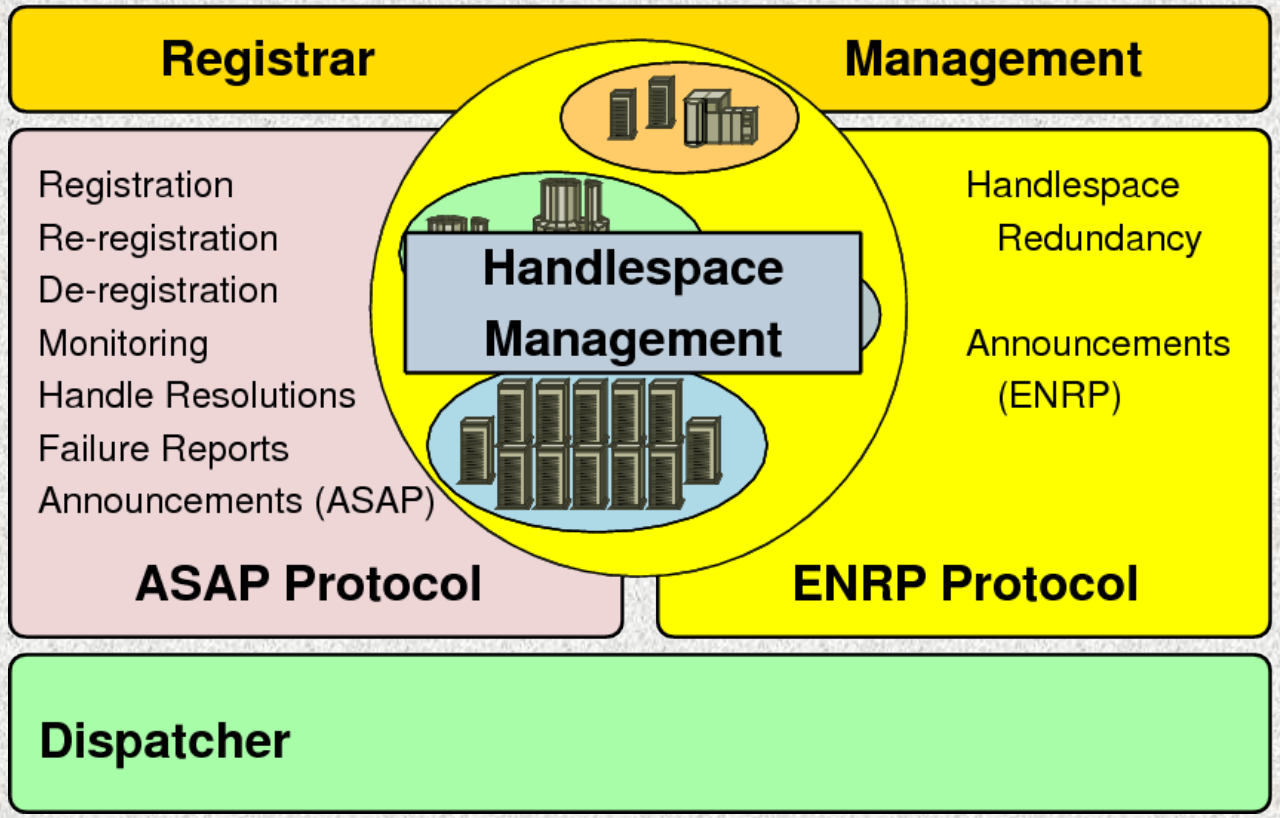
- Start of the PR:
  - `registrar`
  - For loopback usage (i.e. on computer without network connection):
    - Host requires at least a private-scoped IP address (e.g. 192.168.x.y)
    - Interface must have set the „multicast“ flag
    - `sudo ifconfig dummy0 192.168.100.200 netmask 255.255.255.0 up multicast`
- Start of the fractal graphics PE:
  - `server -fractal`
- Start of the fractal graphics PU:
  - `fractalpooluser`

Further information can be found in the
RSPLIB Handbook
on our project website
http://tdrwww.iem.uni-due.de/dreibholz/rserpool/

The Building Blocks of the Registrar

UNIVERSITÄT DUISBURG ESSEN

- Dispatcher:
  - Platform-specific functionalities:
    - Timers
    - Sockets
    - Threads
- Protocols:
  - ASAP
    - PR↔PE
    - PR↔PU
  - ENRP (PR↔PR)
- Registrar Mgt.:
  - Access control
  - Address verification and -filtering



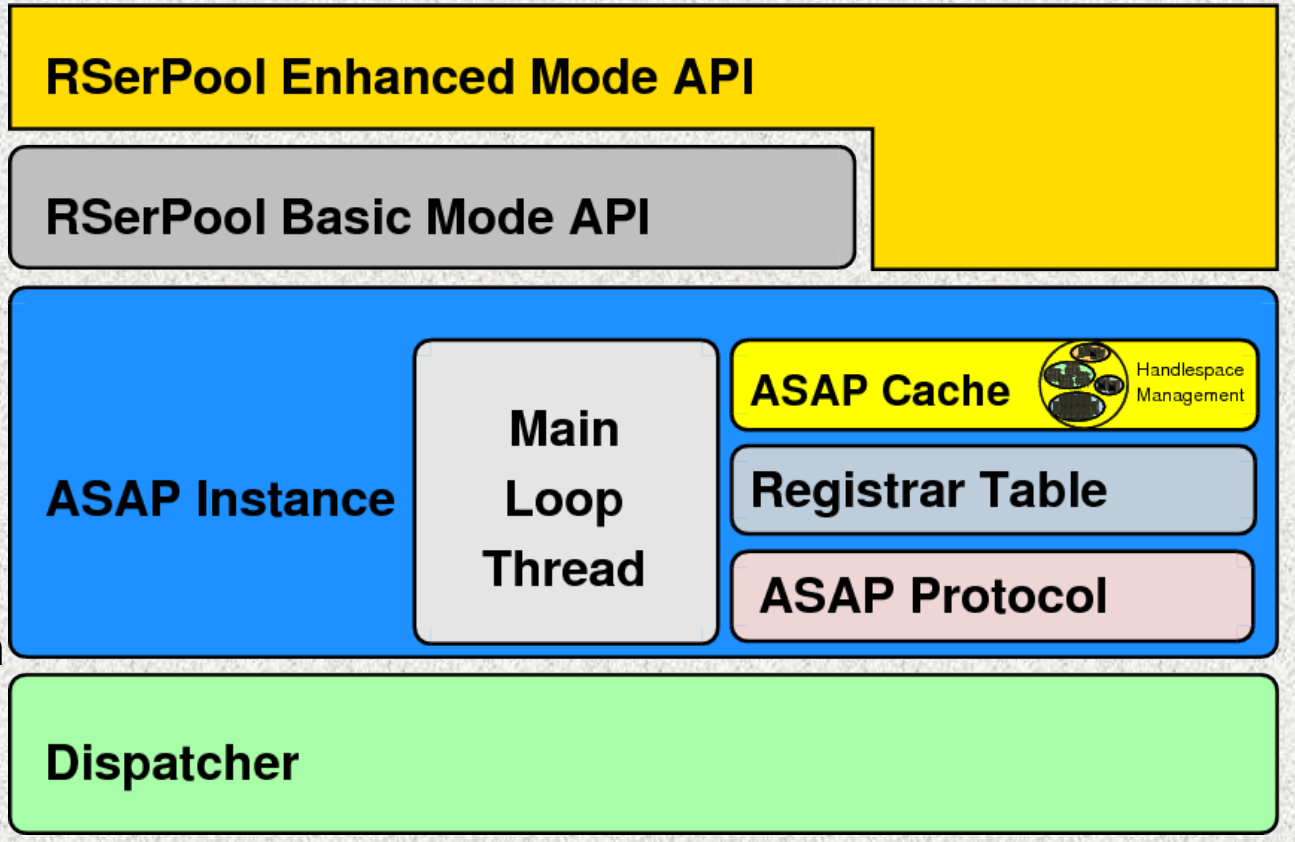
- Handlespace Management (see [FGCN2007-HsMgt][Contel2005])

The Building Blocks of the *RSPLIB* Library

UNIVERSITÄT DUISBURG ESSEN

- Dispatcher

- ASAP Instance:
  - ASAP protocol
    - PE↔PR
    - PU↔PR
    - PU↔PE
  - ASAP thread
    - Request pipelining
  - List of PRs
    - from announces
    - static configuration
  - Cache for PE selection



- RSerPool APIs:
  - Basic Mode
  - Enhanced Mode

The Two RSerPool APIs

UNIVERSITÄT DUISBURG ESSEN

- Basic Mode API
  - Only core functionalities (registration, deregistration, handle resolution)
  - PU $\leftrightarrow$ PE-communication **realized by** the **application itself**!
- Enhanced Mode API
  - Complete **session layer**
  - For PEs:
    - Registration management
    - Management of incoming sessions
    - Client-based state sharing
  - For PUs:

    Sessions with pools, including:
    - Selection of PEs
    - Establishment, monitoring and **management** of a **transport connection**
    - **Failover** support
    - Cookie storage and failover using client-based state sharing

The API of the *RSPLIB* Library: Enhanced Mode for PUs

UNIVERSITÄT DUISBURG ESSEN

- API similar to TCP sockets client:
  - For TCP sockets: *socket*() -> *connect*() -> ... -> *close*()
  - Now: Session (RSerPool socket) instead of a simple transport connection!

```
/* Create session */
session = rsp_socket(0, SOCK_STREAM, IPPROTO_SCTP);
rsp_connect(session, "MyPool", ...);

/* Run application: file download */
rsp_send(session, "GET Linux-CD.iso HTTP/1.0\r\n\r\n");
while((length = rsp_recv(session, buffer, ...)) > 0) {
   doSomething(buffer, length, ...);
}

/* Close session */
rsp_close(session);
```

- Note:

  doSomething() may contain repetitions – depending on the cookie interval!

The API of the *RSPLIB* Library: Enhanced Mode for PEs

UNIVERSITÄT DUISBURG ESSEN

- API similar to TCP sockets server:
  - For TCP sockets: *socket*() -> *bind*() -> *listen*() -> *accept*()
  - Again: Session (RSerPool socket) instead of transport connection!

```
void serviceThread(session)
{
  rsp_recv(session, command, ...);
  if(command is a cookie) {
    /* Got a cookie -> restore session state */
    Restore state;
    rsp_recv(session, command, ...);
  }
  do {
    /* Handle commands from pool user */
    Handle command;
    rsp_send_cookie(session, current state);
    rsp_recv(session, command, ...);
  } while(session is active);
  rsp_close(session);
}


int main(...)
{
```

```
  /* Create and register pool element */
  poolElement = rsp_socket(0,SOCK_STREAM,IPPROTO_SCTP);
  rsp_register(poolElement, "MyPool", ...);

  /* Handle incoming session requests */
  while(server is active) {
    /* Wait for events */
    rsp_poll(poolElement, ...);

    if(incoming session) {
      /* Accept new session */
      session = rsp_accept(poolElement, ...);
      Create service thread to handle session;
    }
  }

  /* Deregister pool element */
  rsp_deregister(poolElement);
  rsp_close(poolElement);
}
```

The Scripting Service: Using RSerPool in Shell Scripts

UNIVERSITÄT DUISBURG ESSEN

- Another example application: **Scripting Service**
  - **Scripting PE**:
    - Gets Tar/GZip file from PU
    - Archive is extracted, a contained script is executed
    - Results will be Tar/GZip-archived and sent back to PU
  - **Scripting PU**:
    - Get (from user) a Tar/GZip archive with script (and input files)
    - Distributes archive to scripting PE in pool
    - Receives back the results

- Application example:
  - **Distribution** of **simulation runs**
  - Realized with only about 50 lines of *bash* shell code.

- Current work on the scripting service (as student projects):
  - Security! Idea: usage of **virtualization** (e.g. Xen)
  - Failover handling: application **checkpointing**

Our RSerPool Activities

UNIVERSITÄT DUISBURG ESSEN

- Research as part of a DFG-funded project since October 2004
  - Simulation model *RSPSIM*
  - Prototype implementation *RSPLIB*

Interested in our RSerPool research papers and presentations? Have a look at our website!

from simulation to reality

from research to application

- Standardization in the IETF
  - Contribution of 4 **Working-Group-Drafts** ...
    - draft-ietf-rserpool-overview-02.txt
    - draft-ietf-rserpool-policies-07.txt
    - draft-ietf-rserpool-mib-04.txt
    - draft-ietf-rserpool-api-00.txt
  - ... and multiple **Individual Submissions**
  - IETF standardization relies on „running code“ - we have it!
  - *RSPLIB* is the world's first complete RSerPool implementation
    - **Open Source** (GPLv3 license)
    - **Reference implementation** of the IETF RSerPool WG

Thank You for Your Attention!
Any Questions?

UNIVERSITÄT DUISBURG ESSEN

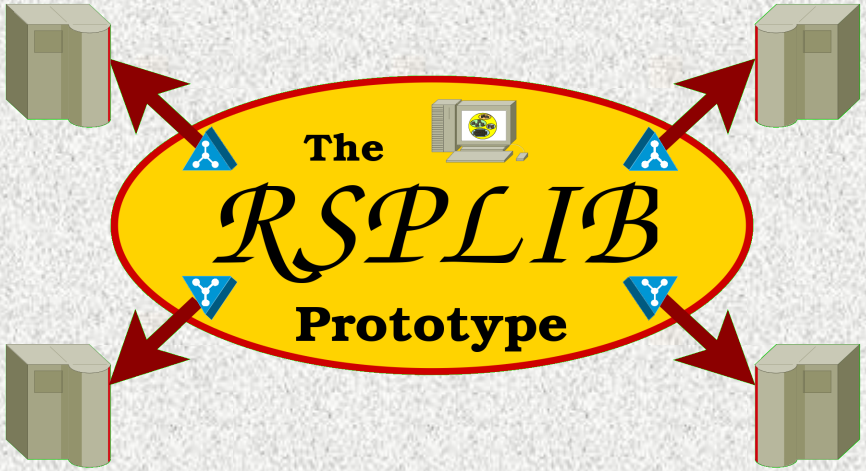
Technik der Rechnernetze

UNIVERSITÄT DUISBURG ESSEN

IETF

Deutsche Forschungsgemeinschaft DFG

IEEE



Fachhochschule Münster University of Applied Sciences

To be continued ...

Visit Our Project Homepage:

http://tdrwww.iem.uni-due.de/dreibholz/rserpool/

Thomas Dreibholz, dreibh@iem.uni-due.de