University of Duisburg-Essen, Institute for Experimental Mathematics

SCTP and RSerPool: Architectures and Protocols for the Future Internet

Dr. Thomas Dreibholz
University of Duisburg-Essen, Germany
dreibh@iem.uni-due.de
http://www.iem.uni-due.de/~dreibh

Prof. Xing Zhou
Hainan University, China
zhouxing@hainu.edu.cn

University of Duisburg-Essen, Institute for Experimental Mathematics

SCTP and RSerPool: 下一代互联网架构标准及其协议

UNIVERSITÄT DUISBURG ESSEN

Dr. Thomas Dreibholz
University of Duisburg-Essen, Germany
dreibh@iem.uni-due.de
http://www.iem.uni-due.de/~dreibh

海南大学
Hainan University

Prof. Xing Zhou
Hainan University, China
zhouxing@hainu.edu.cn

Table of Contents

- IETF and Standardization Background
- Next Generation Internet Architecture
- Stream Control Transmission Protocol (SCTP)
- Reliable Server Pooling (RSerPool)
  - Overview
  - The RSPLIB Implementation
  - Demo Presentation
  - How to Apply RSerPool for Applications
    - Adapting existing applications and developing new applications
    - Examples: client and server
- Summary

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

IETF and Standardization Background (1)

Internet Engineering Task Force (IETF):

- International organization for the standardization of Internet protocols
- All standards are released by IETF as RFC (“Request for Comments”)
- Examples: TCP, UDP, IP, ...
- Organized into different Working Groups (WG), e.g.
  - Transport Services (TSVWG) (responsible for SCTP)
  - Signalling Transport (SigTran)
  - Reliable Server Pooling (RSerPool)
  - ...

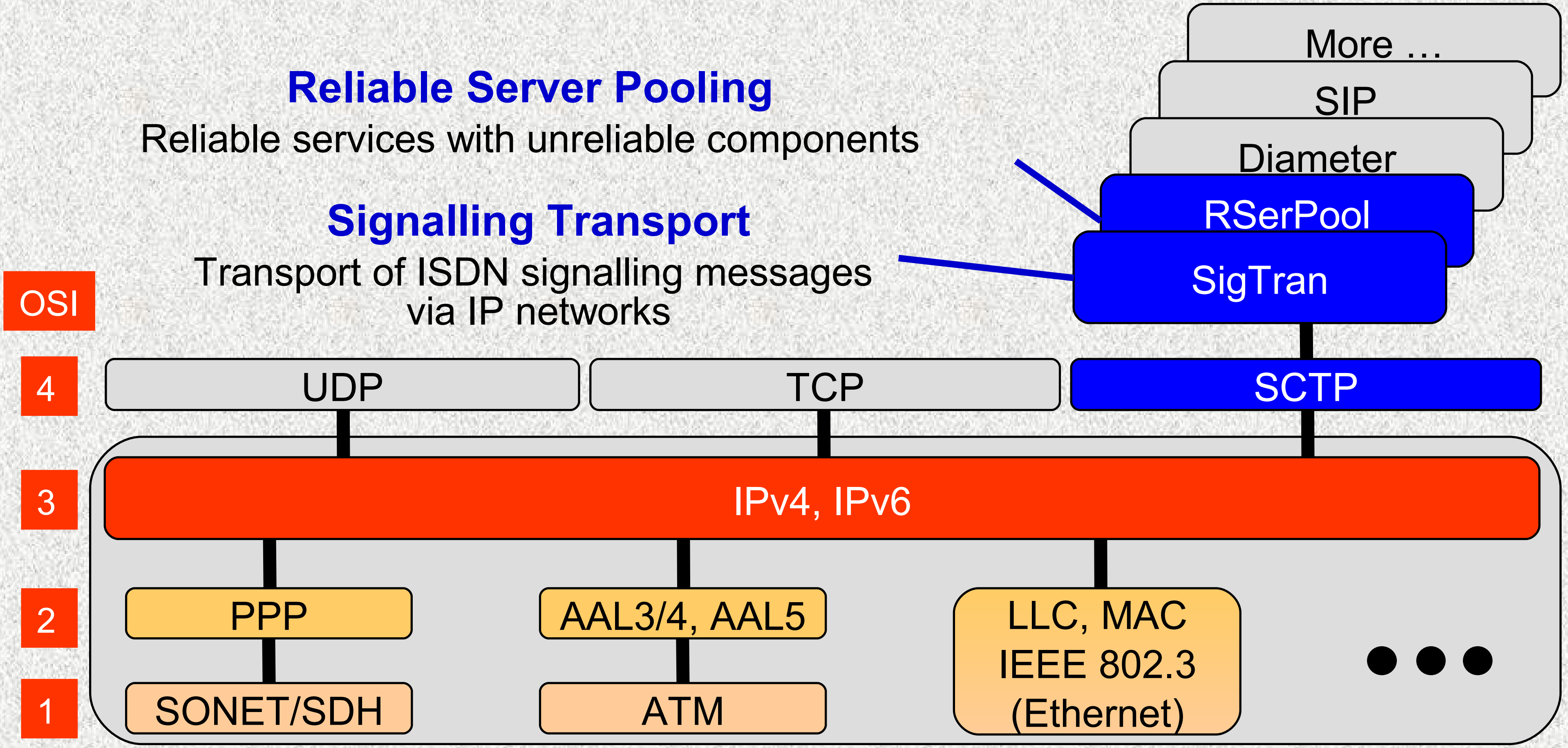
New protocols from the IETF:

- Stream Control Transmission Protocol (SCTP, RFC 4960):
  - Advanced transport protocol (i.e. next generation of TCP)
  - Important contributors:
    - Randall Stewart (Cisco Systems, U.S.A.)
    - Michael Tüxen (Münster University of Applied Sciences, Germany)
    - Andreas Jungmaier, Thomas Dreibholz (Uni. of Duisburg-Essen, Germany)
    - Hans Jürgen Schwarzbauer (Siemens, Germany)
    - Ian Rytina (Ericsson, Australia)

IETF and Standardization Background (2)

- New protocols from the IETF (continued):
  - Reliable Server Pooling (RSerPool, RFC 5351 – RFC 5356):
    - Unified architecture for server pool management
      - High availability
      - Load distribution and balancing
      - Server pool and session management
    - Important contributors:
      - **Thomas Dreibholz** (University of Duisburg-Essen)
      - Erik Guttman (Sun Microsystems, Germany)
      - Ram Gopal (Nokia Siemens Networks, U.S.A.)
      - Peter Lei (Cisco Systems, U.S.A.)
      - Lyndon Ong (Ciena, U.S.A.)
      - Aron Silverton (Motorola, U.S.A.)
      - Randall Stewart (Cisco Systems, U.S.A.)
      - Maureen Stillman (Nokia, U.S.A.)
      - Michael Tüxen (Münster University of Applied Sciences, Germany)
      - Qiaobing Xie (Motorola, U.S.A.)
      - **Xing Zhou** (Hainan University, China)

The Architectures and Protocols for the Future Internet



AAL: ATM Adaptation Layer   LLC: Logical Link Control   MAC: Medium Access Control   IETF: Internet Engineering Task Force   IP: Internet Protocol
OSI: Open Systems Interconnection   PPP: Point to Point Protocol   SDH: Synchronous Digital Hierarchy   SONET: Synchronous Optical Network

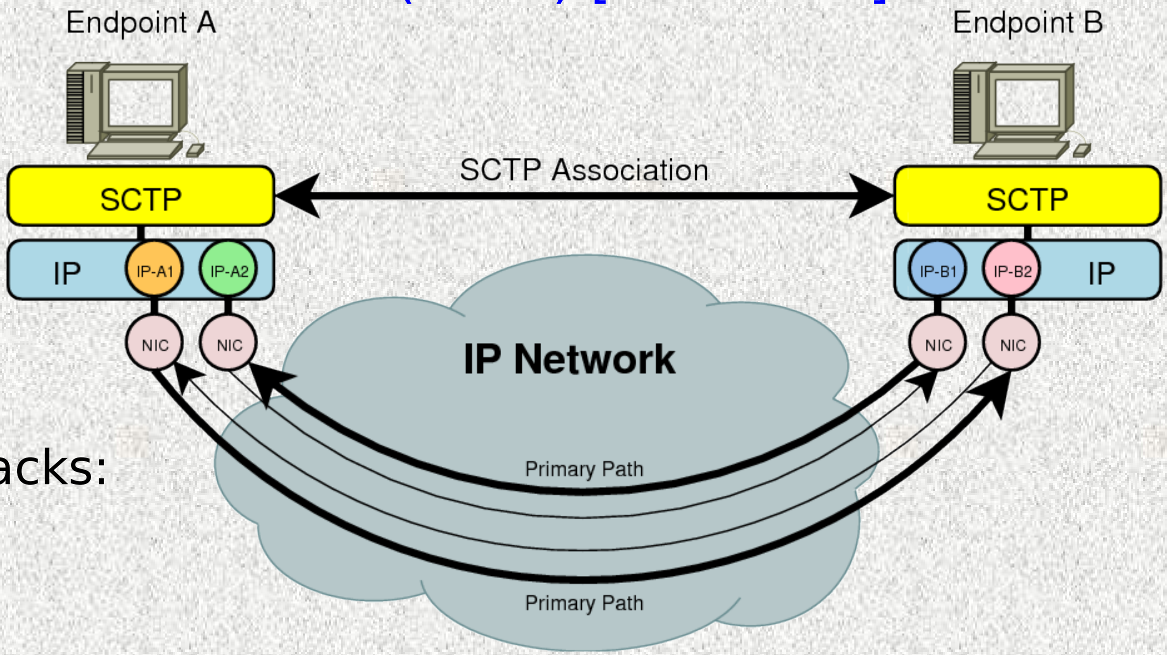
Stream Control Transmission Protocol (SCTP): Basic Features (RFC 4960)

- Flow control
  - Adaptive window size similar to TCP
- Error control
  - SCTP: retransmission with Selective Acknowledgements and Fast Retransmission
  - TCP: retransmission with Selective Acknowledgements and Fast Retransmission optional
- Security mechanisms against standard attacks
  - 4-way handshake with cryptographic signature against „Denial of Service“ attacks
  - Verification Tag in SCTP-Header against blind attacks
- Flexible message delivery
  - Out of sequence delivery possible
  - Limited number of retransmissions (Partial Reliability extension)
- Multi-Streaming
  - Multiplexing of multiple application message streams over one connection
  - Sequence integrity only within its stream => avoids „Head of Line Blocking“
- Multi-Homing
  - SCTP entities can be connected via multiple network interfaces (i.e. paths)
  - Path monitoring, rapid switch-over in case of failures
  - Dynamic addition/deletion of addresses (Add-IP)

Stream Control Transmission Protocol (SCTP): Next Generation of TCP

UNIVERSITÄT DUISBURG ESSEN

- Original Motivation:
  - Telephone signalling (SS7 protocol) over IP networks
  - Strict requirements on availability
- The Stream Control Transmission Protocol (SCTP) [RFC 4960]
  - „TCP Next Generation“
  - **Multi-Homing**
  - Add-IP: dynamic address reconfiguration
  - Multi-Streaming
  - Message-Framing
  - Protection against DoS attacks:
    - 4-way handshake
    - „Verification Tag“



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

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

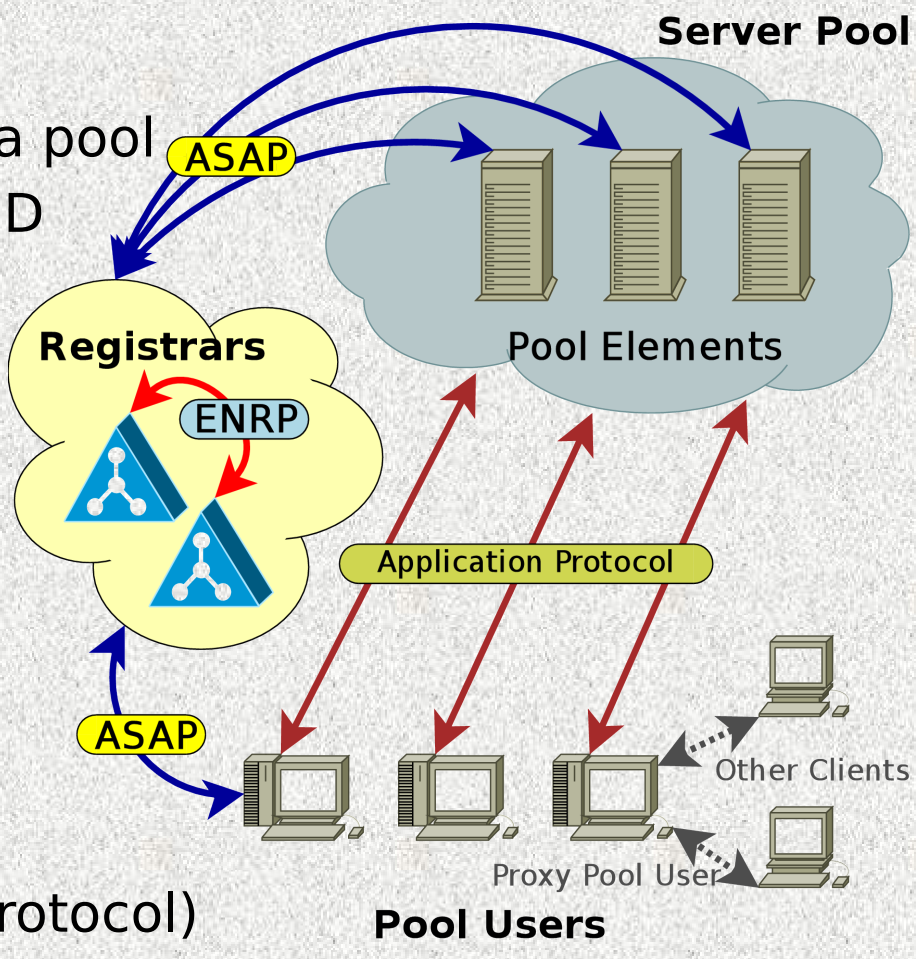
Reliable Server Pooling (RSerPool)

- Motivation of Reliable Server Pooling (RSerPool):
  - Unified, application-independent solution for service availability
    - Deployment of infrastructure once → usage for all applications
    - Significantly **reduced** development and maintenance **costs**
- Application Scenarios for RSerPool:
  - Initial motivation: **Telephone Signalling (SS7) over IP**
  - But also useful for:
    - **Load Balancing**
    - Voice over IP (VoIP) with SIP (e.g. highly available SIP proxies)
    - IP Flow Information Export (IPFIX)
  - ... and many more!
- Requirements for RSerPool:
  - **“Lightweight”** (low resource req'ts: e.g. embedded/mobile 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!”)

Reliable Server Pooling (RSerPool)

Terminology:

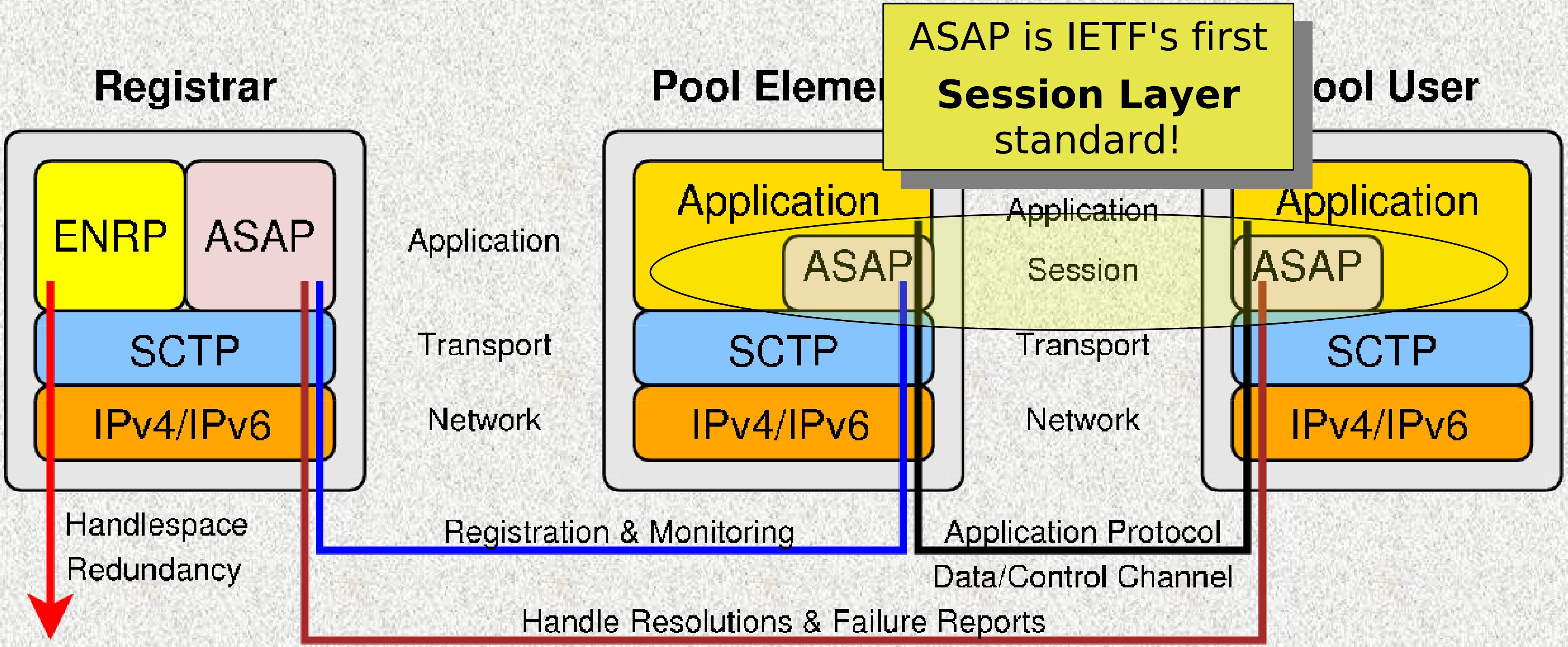
- **Pool Element (PE)**: Server
- Pool: Set of PEs
- PE ID: ID of a PE in a pool
- Pool Handle: Unique pool ID
- Handlespace: Set of pools
- Pool **Registrar (PR)**
- **Pool User (PU)**: Client
- Support for Existing Applications
  - Proxy Pool User (PPU)
  - Proxy Pool Element (PPE)



Protocols:

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

The RSerPool Protocol Stack



ASAP is IETF's first **Session Layer** standard!

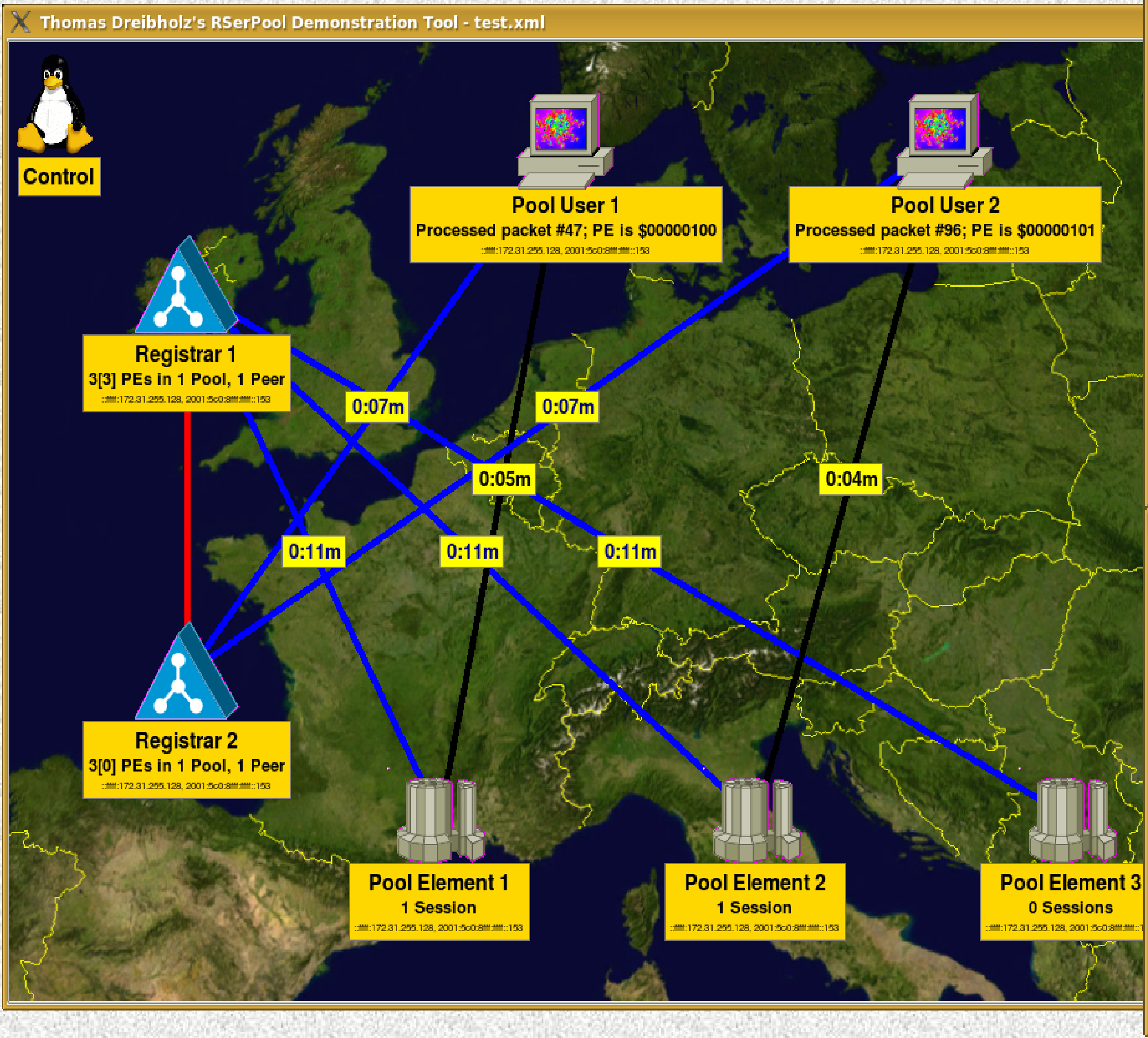
- Aggregate Server Access Protocol (ASAP)
  - PR ⇔ PE: Registration, Deregistration and Monitoring by Home-PR (PR-H)
  - PR ⇔ PU: Server Selection, Failure Reports
- Endpoint Handlespace Redundancy Protocol (ENRP)
  - PR ⇔ PR: Handlespace Synchronisation

The *RSPLIB* Implementation

- Design decisions:
  - Open Source, GPLv3 license; separate commercial licenses negotiable
  - Platform independence. Currently:
    - Systems: Linux, FreeBSD, MacOS X, Solaris
    - CPUs: x86, x86_64, PPC, MIPS
  - Implemented in ANSI-C → easy portability
- Basic components:
  - *RSPLIB* library for PUs and PEs
    - ASAP protocol (PU/PE side)
  - Registrar
    - ASAP protocol (PR side)
    - ENRP protocol
  - Many service examples

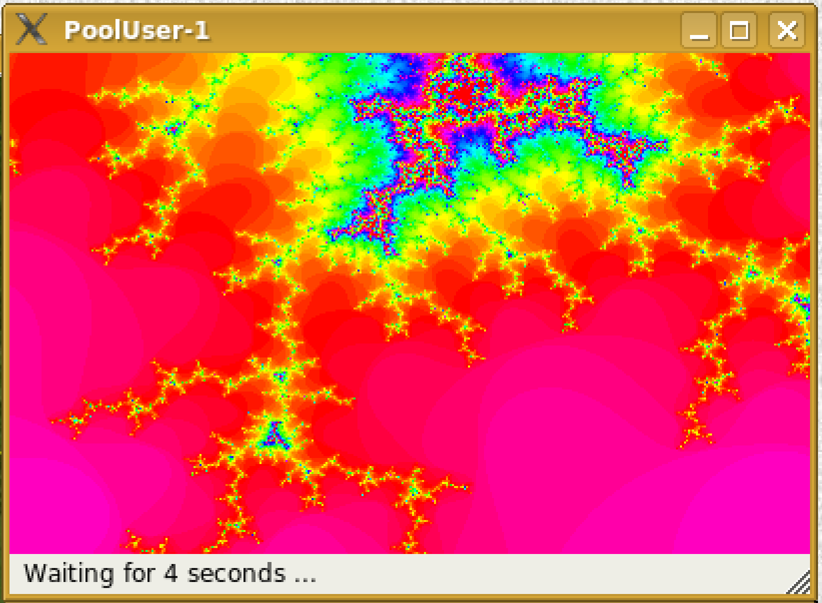
Thomas Dreibholz's Reliable Server Pooling Page
http://tdrwww.iem.uni-due.de/dreibholz/rserpool/

What is „Reliable Server Pooling“? System Demonstration



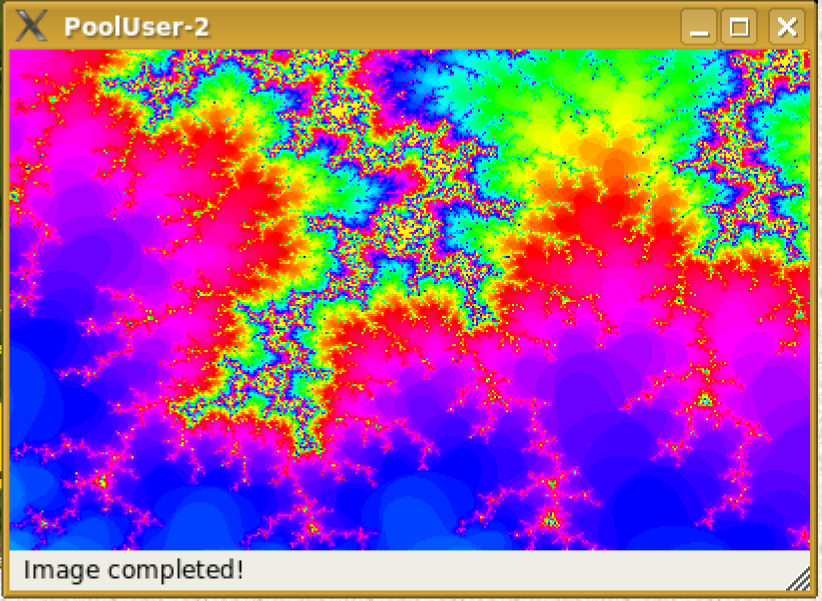
Thomas Dreibholz's RSerPool Demonstration Tool - test.xml

Control
Pool User 1
Processed packet #47; PE is $00000100
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
Pool User 2
Processed packet #96; PE is $00000101
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
Registrar 1
3[3] PEs in 1 Pool, 1 Peer
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
0:07m
0:07m
0:05m
0:04m
0:11m
0:11m
0:11m
Registrar 2
3[0] PEs in 1 Pool, 1 Peer
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
Pool Element 1
1 Session
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
Pool Element 2
1 Session
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::153
Pool Element 3
0 Sessions
::ffff:172.31.255.128, 2001:5c0:8fff:ffff::1



PoolUser-1

Waiting for 4 seconds ...

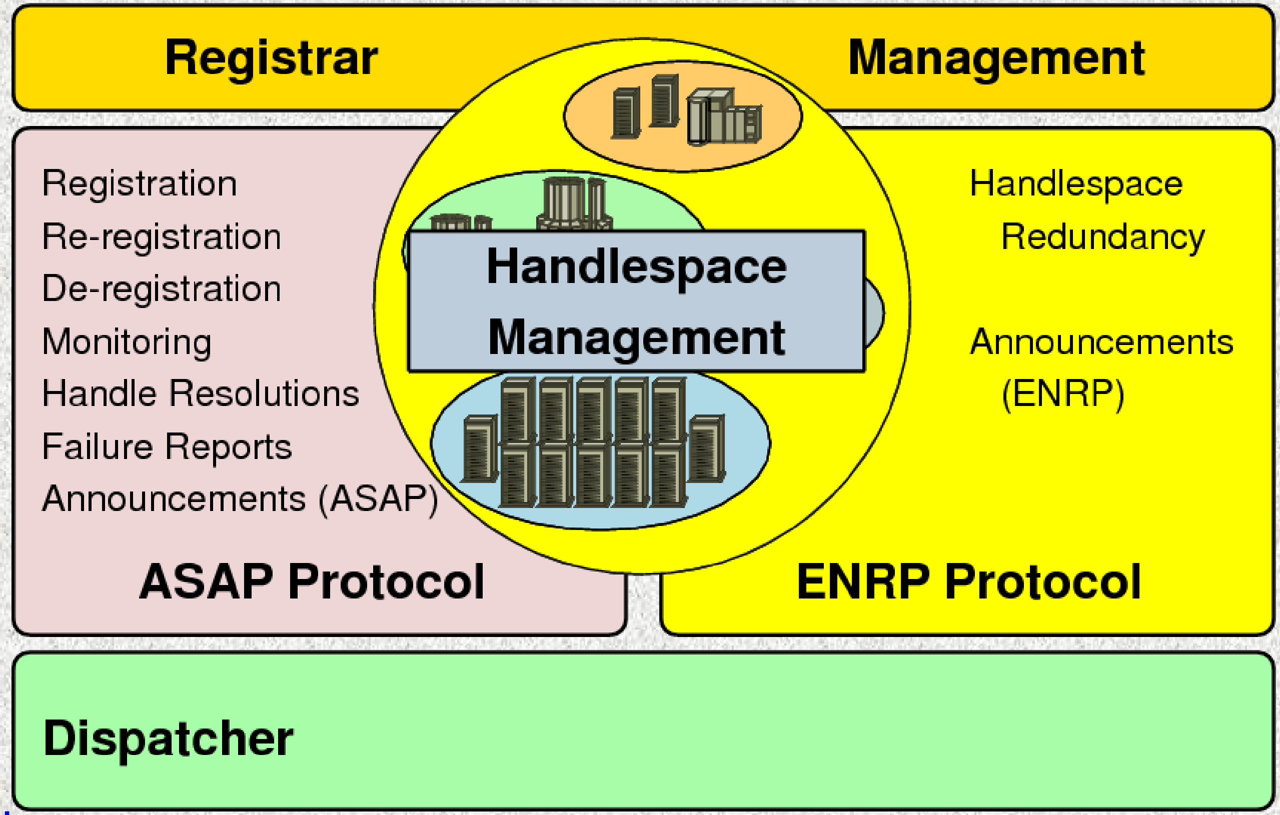


PoolUser-2

Image completed!

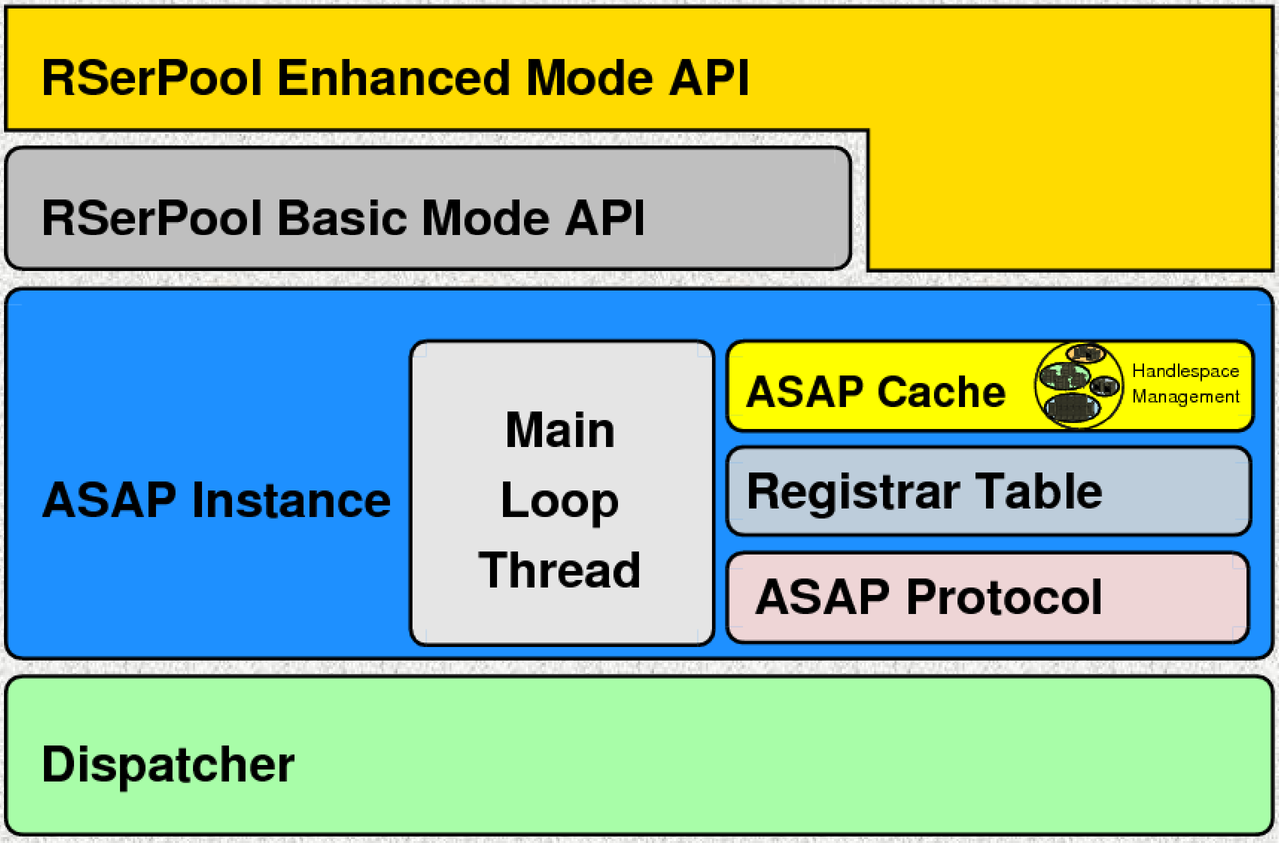
The Building Blocks of the Registrar

- 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
- Note: to adapt RSPLIB to Microsoft Windows, only Dispatcher needs changes



The Building Blocks of the *RSPLIB* Library

- 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

- Basic Mode API
  - Only core functionalities (registration, deregistration, handle resolution)
  - PU ↔ 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

Adapting Existing Applications to RSerPool: Client Side using Enhanced Mode API

- 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: same API for new applications based on RSerPool

Adapting Existing Applications to RSerPool: Server Side using Enhanced Mode API

- 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);
}
```

- Note: same API for new applications based on RSerPool

RSPLIB Example Services

- Self-designed RSerPool service protocols:
  - Fractal Generator Service (FGP):
    - PE provides computation of fractal images
    - Illustrative demonstration of RSerPool/RSPLIB features
  - Scripting Service (SSP):
    - Remote script execution
    - Workload distribution
    - Used e.g. for distributed simulation and ray tracing
  - CalcApp Service (CalcAppProtocol):
    - Simulates computation service
    - Useful to obtain load distribution/balancing performance
  - PingPong Service (PPP):
    - Simple demonstration of cookie-based failover
- Echo/Discard/Daytime/CharGen Services:
  - Like the similar TCP test services ...
  - ... but providing failover capabilities
- Note: Wireshark includes packet dissectors for all these services!

The Scripting Service: Using RSerPool in Shell Scripts

- 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 examples:
  - **Distribution** of **simulation runs**
    - Realized with only about 50 lines of *bash* shell code
  - Distribution of workload from low-power device (e.g. mobile or PDA) to powerful machines
- Deployment:
  - Used for simulation distribution in a pool of more than 30 PEs ...
  - ... at University of Duisburg-Essen and Hainan University
  - Tested in PlanetLab setups of up to 500 PEs

Summary

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

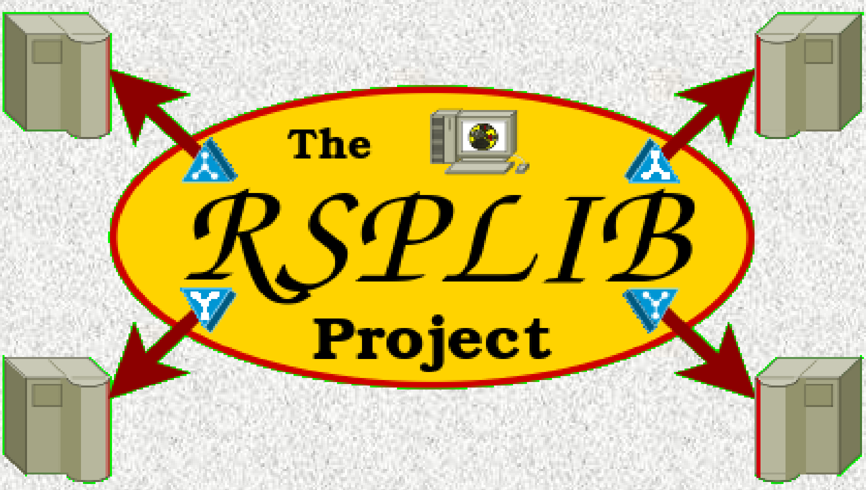
Interested in our RSerPool research papers and presentations?
Have a look at our website!

- Standardization in the IETF
  - Contribution of 4 drafts of RSerPool Working Group ...
    - draft-ietf-rserpool-overview → RFC 5351
    - draft-ietf-rserpool-policies → RFC 5356
    - draft-ietf-rserpool-mib
    - draft-ietf-rserpool-api
  - ... and various **Individual Submissions**
  - IETF standardization relies on „running code“ - we have it!
  - *RSPLIB* is the world's first complete RSerPool implementation
    - **Open Source** (GPLv3 license; commercial on request)
    - **Reference implementation** of the IETF RSerPool WG

from simulation
to reality

from research
to application

Thank You for Your Attention! Any Questions?



To be continued ...

Visit Our Project Homepage:

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

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