Das *rsplib*-Projekt - Hochverfügbarkeit mit Reliable Server Pooling

Thomas Dreibholz

Institut für Experimentelle Mathematik

Universität Duisburg-Essen

dreibh@exp-math.uni-essen.de

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

Überblick

UNIVERSITÄT DUISBURG ESSEN

- Motivation
- Was ist Reliable Server Pooling?
  - Einführung
  - Anwendungsszenarien
- Unsere Implementation
  - Aufbau
  - API
  - Demo-Vorführung
- Unsere Aktivitäten im Bereich Reliable Server Pooling

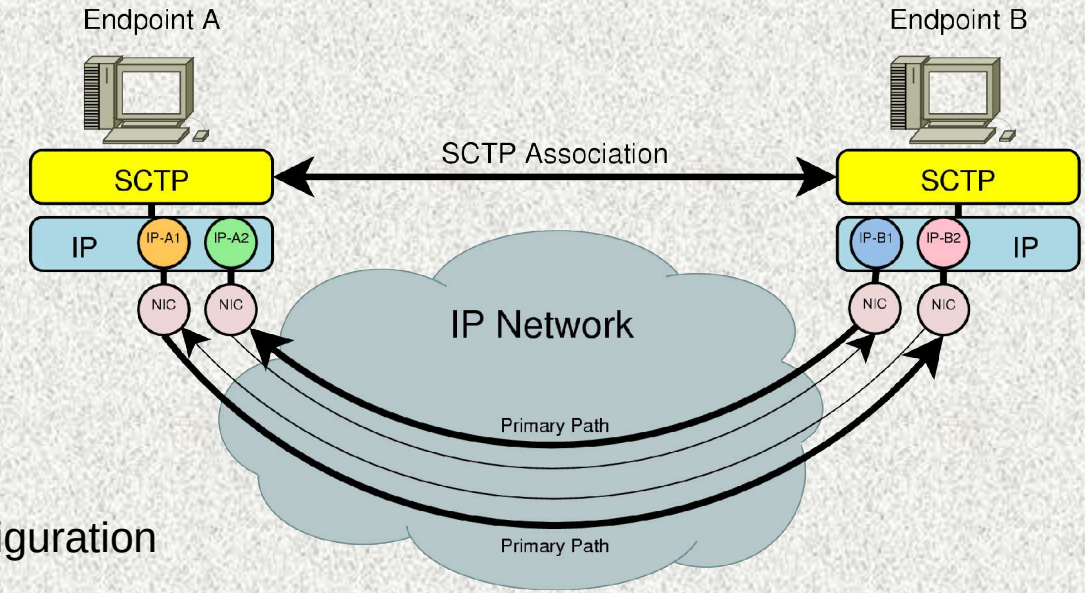
Thomas Dreibholz's Reliable Server Pooling Page

http://tdrwww.exp-math.uni-essen.de/dreibholz/rserpool/

Motivation (I)

UNIVERSITÄT DUISBURG ESSEN

- Ursprüngliche Motivation:
  - Telefonsignalisierung (SS7-Protokoll) über IP
  - Hohe Anforderungen an Verfügbarkeit
- Das Stream Control Transmission Protocol (SCTP) [RFC2960]
  - „TCP Next Generation“
  - **Multi-Homing**
  - Multi-Streaming
  - Message-Framing
  - Sicherheit vor DoS-Angriffen
    - 4-Wege-Handshake
    - Verification Tag
  - Add-IP: dynamische Adreßrekonfiguration



- SCTP schützt vor einer Vielzahl von Netzproblemen, aber ...
- ... nicht vor einem **Serverausfall** => Konzept für **Server-Redundanz** ist **notwendig**

Motivation (II)

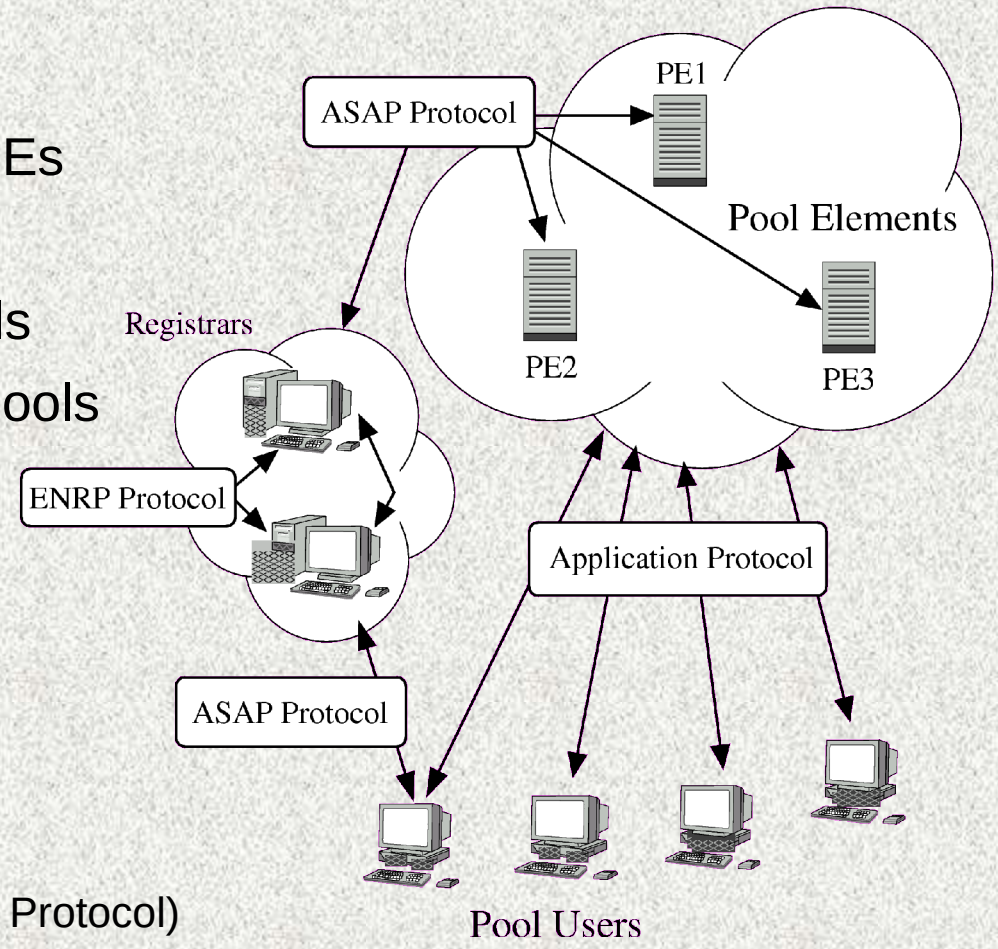
UNIVERSITÄT DUISBURG ESSEN

- Reliable Server Pooling (RSerPool)
  - Standardisierung in der IETF RSerPool WG
  - RSerPool ist ein **Framework** zur **Pool- und Sitzungsverwaltung**
- Anforderungen an RSerPool:
  - **Lightweight** (z.B. auch auf Embedded Devices nutzbar)
  - **Echtzeit** (schneller Failover)
  - **Skalierbar** (bis auf große (Firmen-)Netzwerke, nicht aber auf das globale Internet)
  - **Erweiterbar** (z.B. durch neue Serverauswahlregeln)
  - **Einfach** (automatische Konfiguration)
- Anwendungsgebiete für RSerPool:
  - Telefonsignalisierung (SS7) über IP
  - **GRID Computing /** Distributed Computing [Zha2004]
  - Mobility-Management in Verbindung mit SCTP und Add-IP [LCN2003]
  - Battlefield Networks
  - **Load Balancing** (z.B. Web-Server; zur Zeit sehr aktiv in der IETF diskutiert)

Reliable Server Pooling (RSerPool)

UNIVERSITÄT DUISBURG ESSEN

- Terminologie:
  - **Pool Element (PE)**: Server
  - Pool: Menge von PEs
  - PE ID: ID eines PEs
  - Pool Handle: ID eines Pools
  - Handlespace: Menge von Pools
  - **Registrar (PR)**
  - **Pool User (PU)**: Client



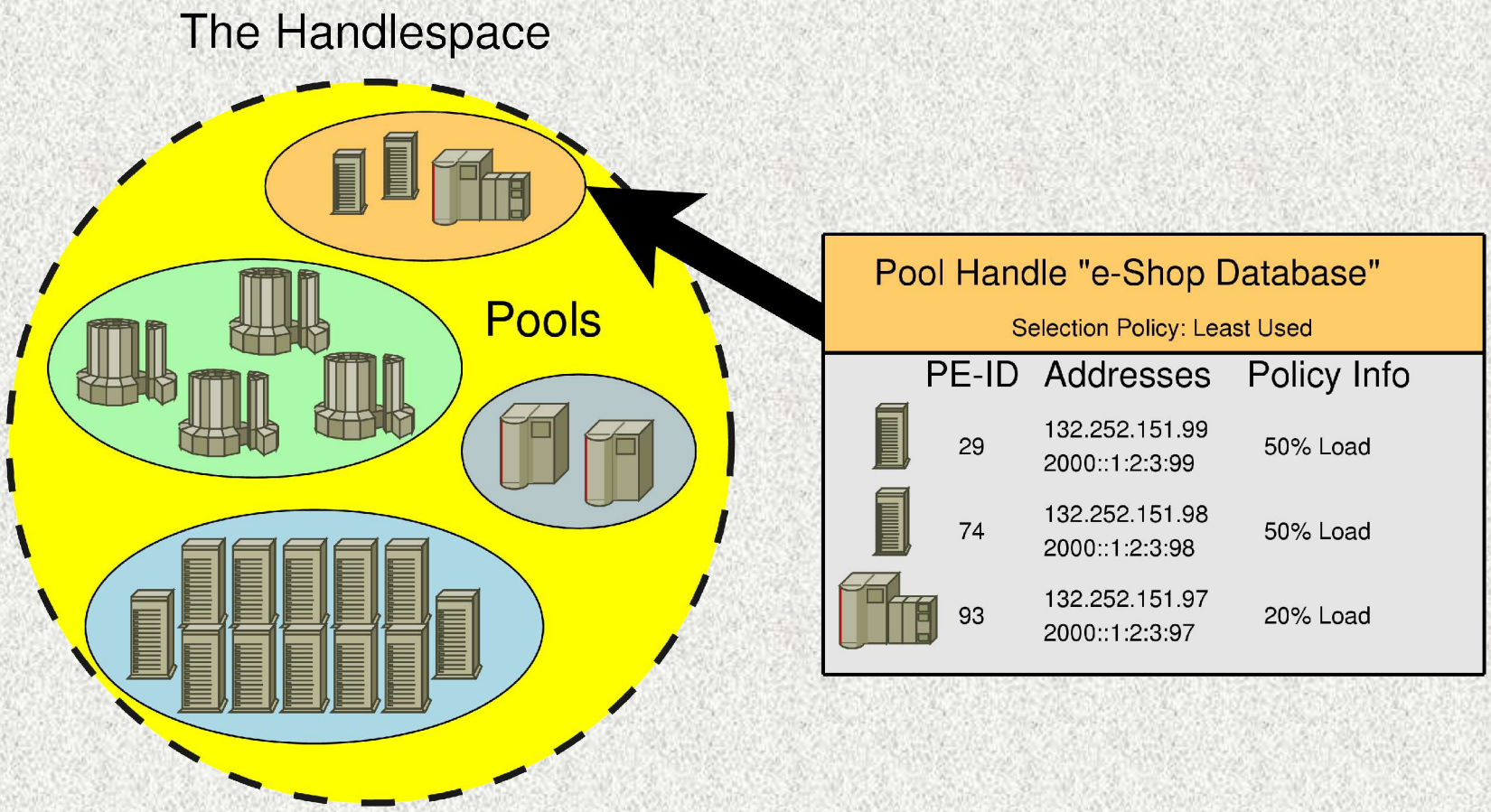
- Protokolle:

  ASAP (Aggregate Server Access Protocol)

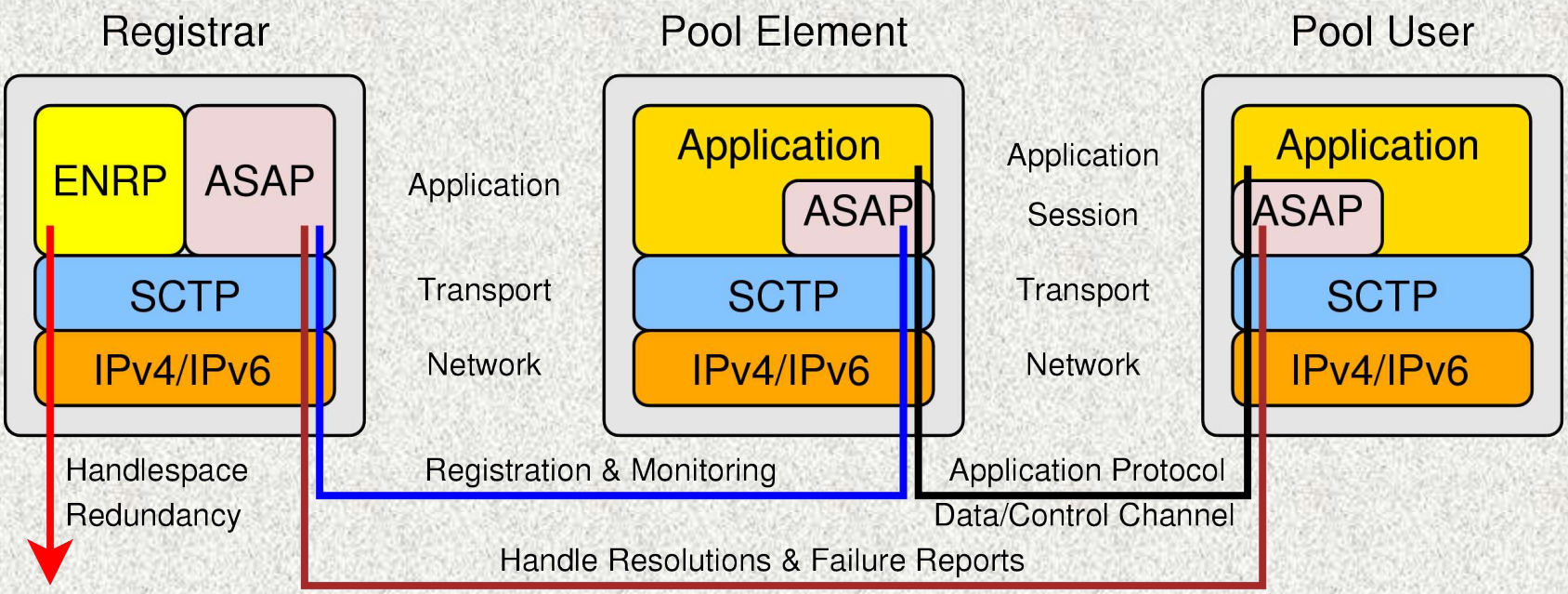
  ENRP (Endpoint Handlespace Redundancy Protocol)

Der Handlespace

UNIVERSITÄT DUISBURG ESSEN



Der Protokoll-Stack

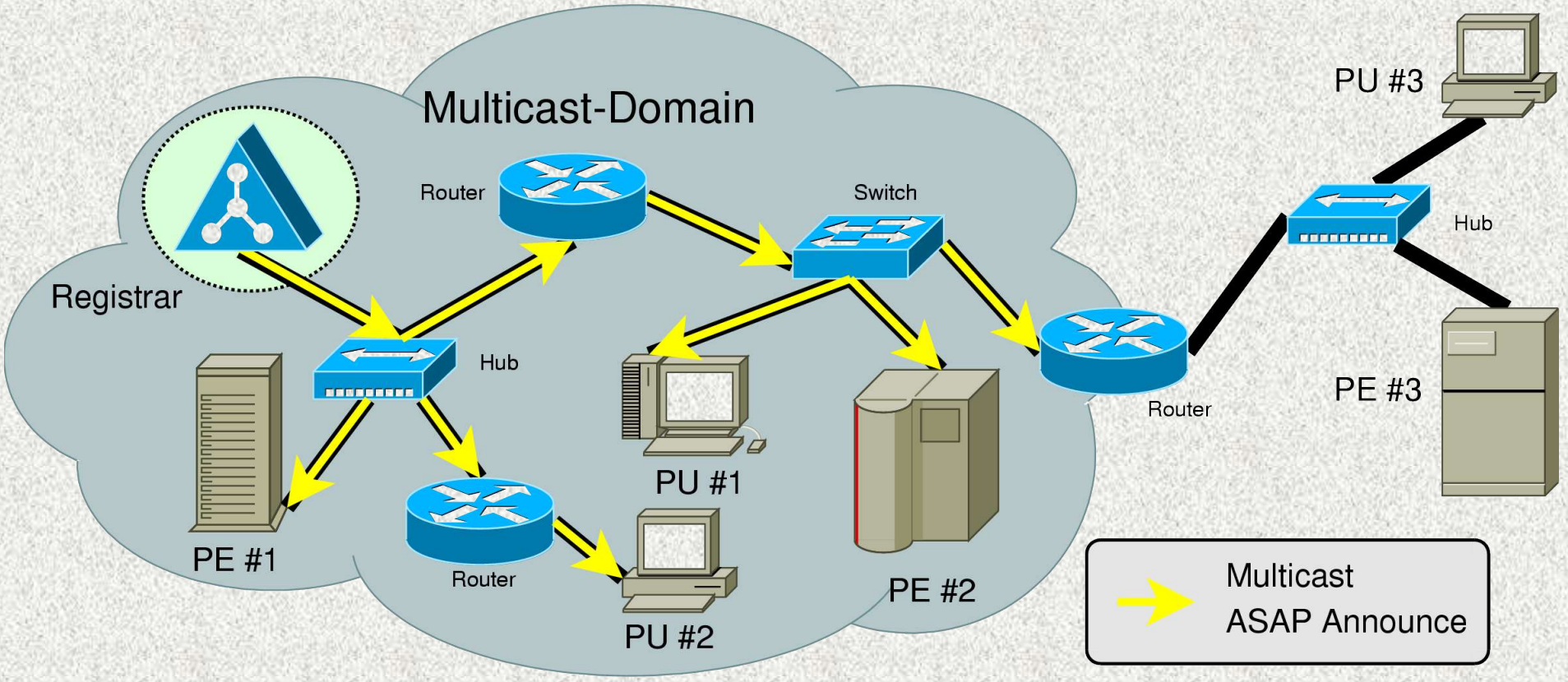


Automatische Konfiguration via PR-Announces

UNIVERSITÄT DUISBURG ESSEN

- Transport der Announces:

  als UDP-Pakete via IP-Multicast



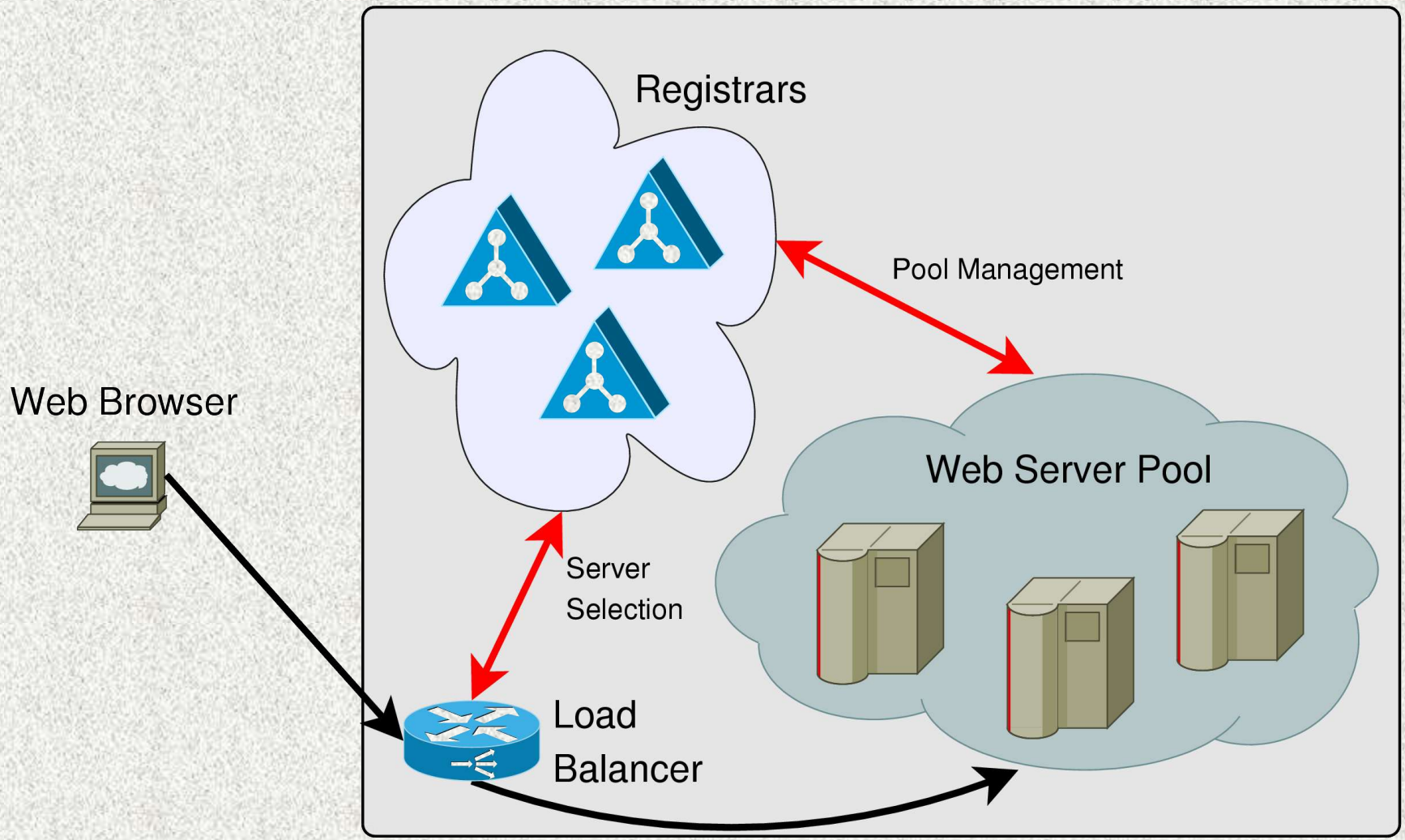
Sitzungsaufbau und -failover

UNIVERSITÄT DUISBURG ESSEN

Selektionen

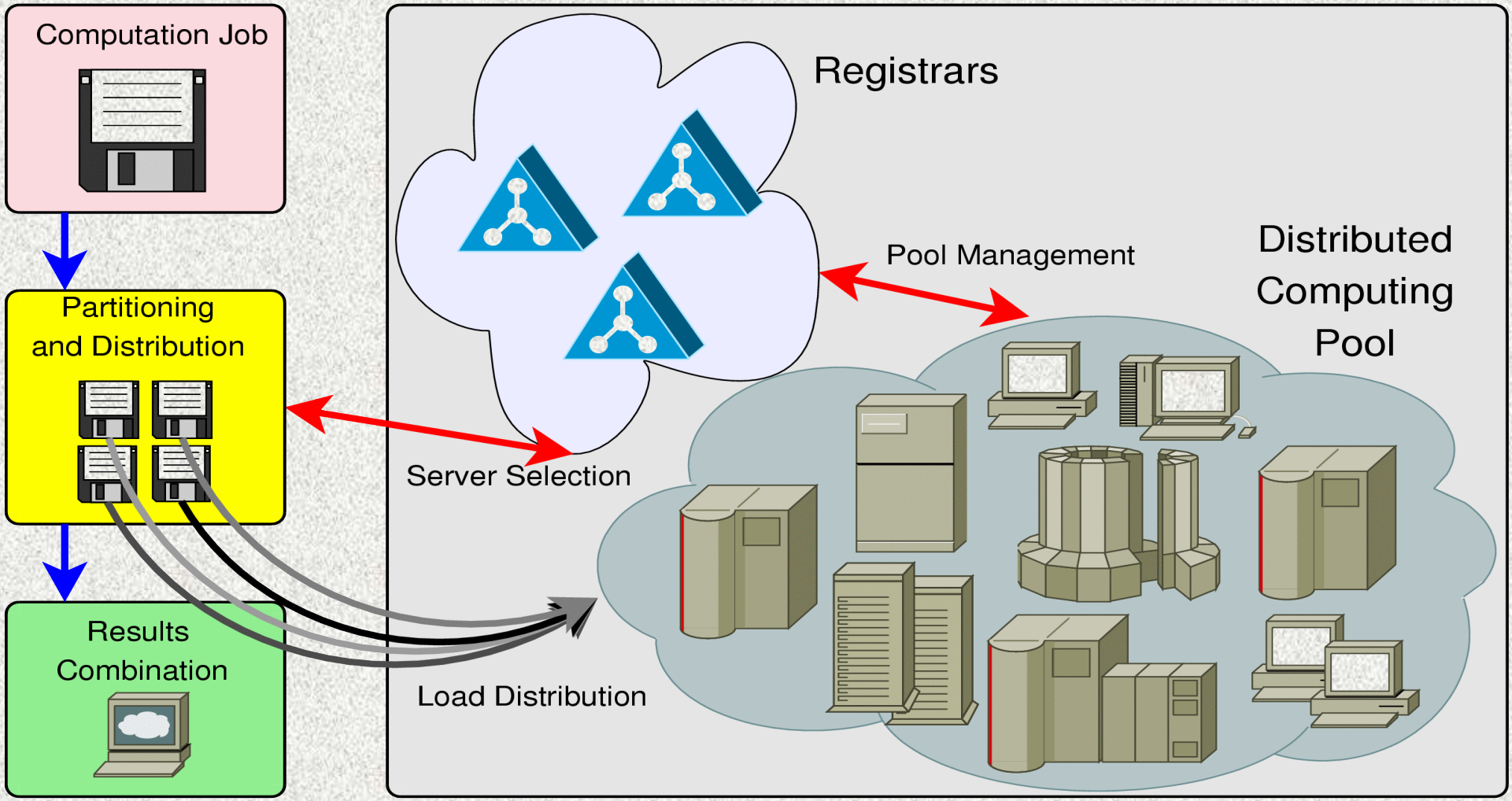
- **Sitzungsaufbau** (PU möchte Dienst eines Pools nutzen):
  - PU fragt bei beliebigem PR (z.B. im lokalen LAN) um Auflösung eines PH in eine Liste von PEs an
  - 1 **PR wählt** Menge von PEs **nach Policy** des Pools (z.B. Round Robin)
  - PU schreibt PE-Menge in seinen lokalen Cache
  - 2 **PU wählt** ein PE, wieder **nach Policy** und baut Verbindung zu diesem auf
  - Weitere Anfragen können ggf. den Cache nutzen
- **Failover** (PU behandelt Ausfall eines PEs):
  - PU soll seinen PR über den Ausfall des PEs **benachrichtigen**
    (PR kann dann das PE ggf. auf dem Handlespace entfernen)
  - PU **wählt** ein neues PE (aus dem Cache 2, oder per PR-Anfrage 1 2)
  - PU führt *applikations-spezifische* **Failover-Prozedur** durch
    (Beispiel: *Datei-Download* – Teile neuem PE Dateiname und Position mit)

Anwendungsszenario für RSerPool: Load Balancing



Anwendungsszenario für RSerPool: Distributed Computing

UNIVERSITÄT DUISBURG ESSEN



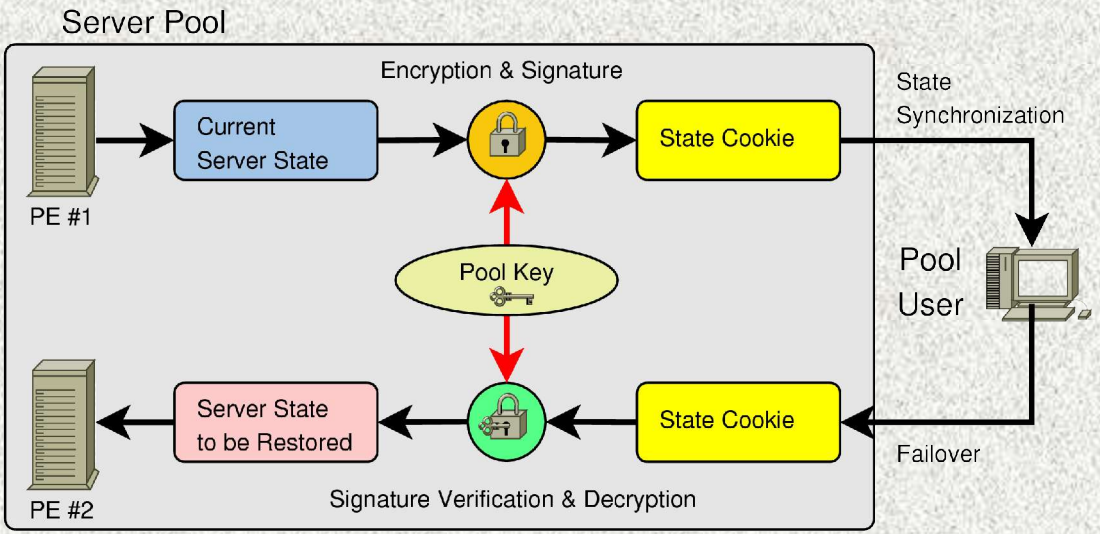
Der Session-Layer

UNIVERSITÄT DUISBURG ESSEN

- Basic Mode
  - Für PEs: Pools verwalten
  - Für PUs: PE auswählen
  - Data Channel:

    Applikation selbst ist

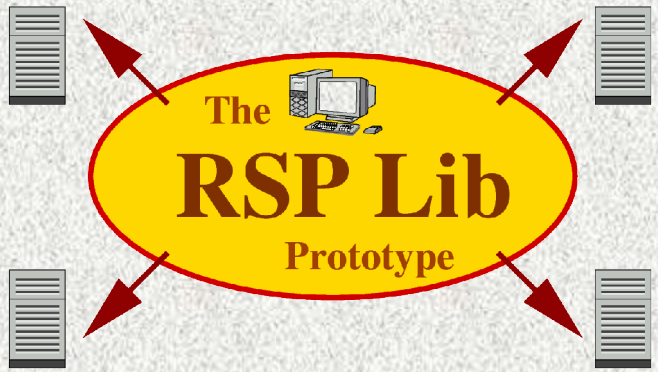
    dafür verantwortlich
- Enhanced Mode
  - Session-Layer (OSI-Modell)
  - Control/Data Channel
  - Zustandsübergang einer Session:
    - Applikationsspezifisch oder
    - Client-basiertes State-Sharing mit State Cookies



Der *rsplib*-Prototyp

UNIVERSITÄT DUISBURG ESSEN

- Designentscheidungen
  - Plattformunabhängigkeit
    Aktuell: Linux, FreeBSD, Darwin
    in Arbeit: Solaris, M$-Windows
  - Implementiert in ANSI-C
  - Open Source, GPL-Lizenz
- Grundlegende Bestandteile
  - *rsplib* Library for PUs and PEs
    - ASAP-Protokoll
  - Registrar
    - ENRP-Protokoll
  - Demo System



Entwickelt in Kooperation mit Siemens AG, München
mit Unterstützung durch BMBF und DFG

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

Der Aufbau des Registrars

UNIVERSITÄT DUISBURG ESSEN

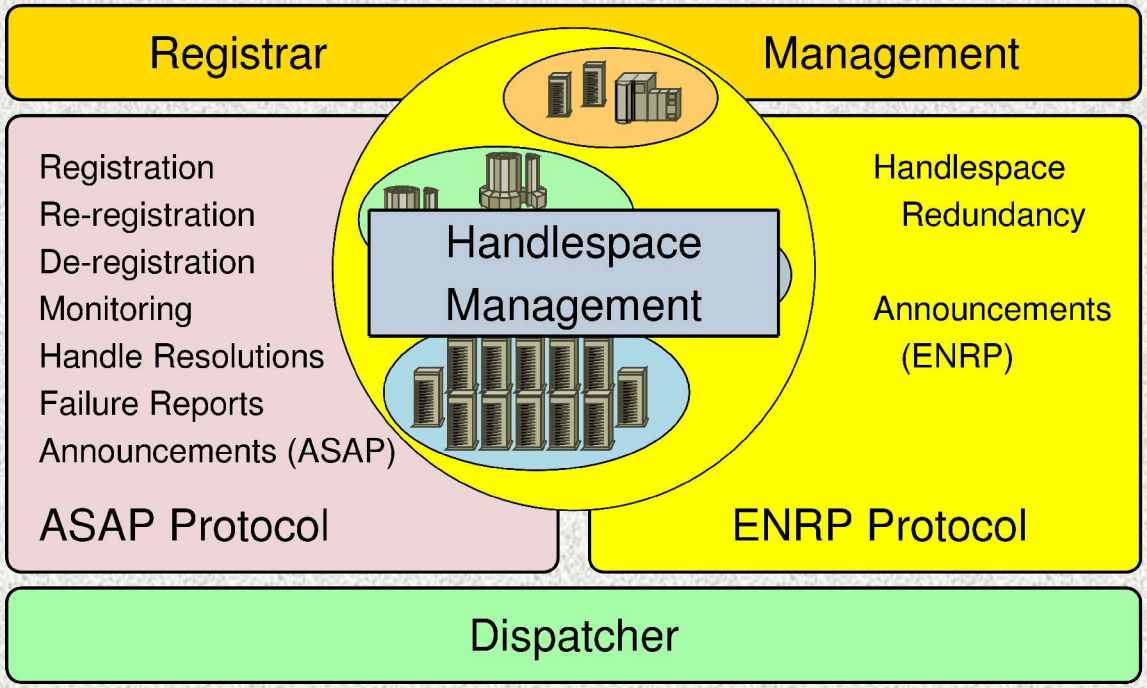
- Dispatcher:

  Plattformspez. Funktionen:

  Timer, Sockets, Threads

- Protokolle:
  - ASAP

    (PR↔PE, PR↔PU)
  - ENRP (PR↔PR)

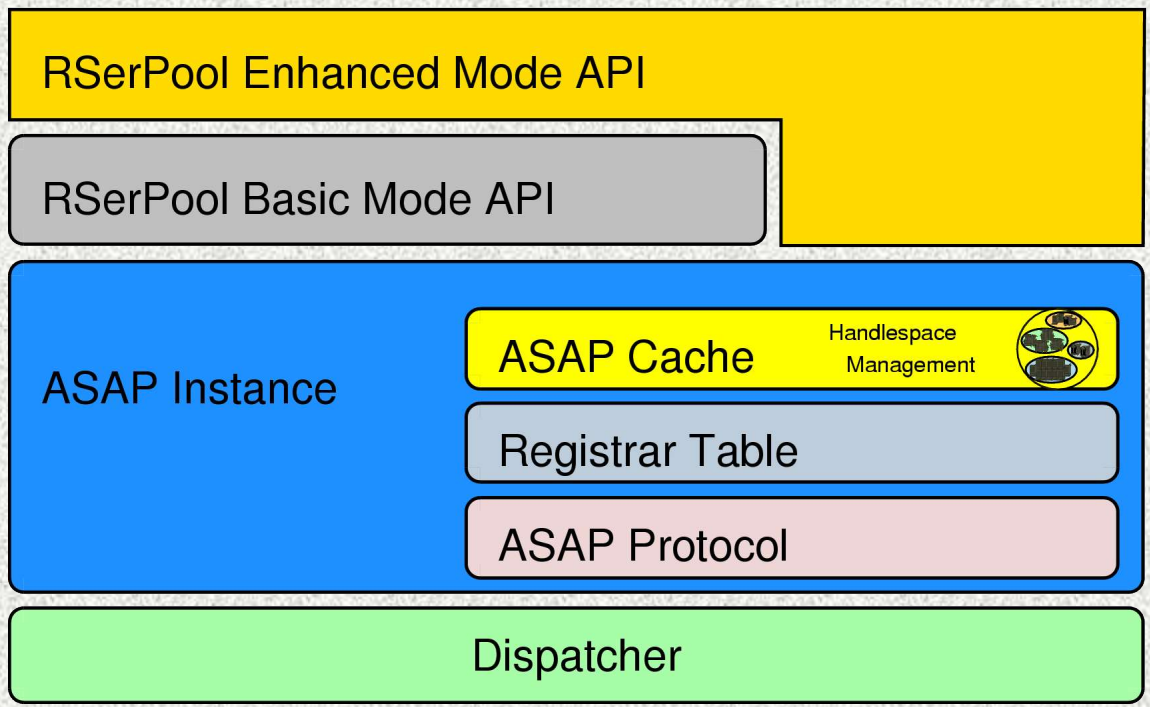


- Registrar-Verwaltung:
  - Zugriffskontrolle
  - Adreßverifikation und -filterung
- Handlespace-Management (siehe [Contel2005])

Der Aufbau der PU/PE-Library

UNIVERSITÄT DUISBURG ESSEN

- Dispatcher
- ASAP-Instance:
  - ASAP
    - PE↔PR, PU↔PR
    - PU↔PE
  - Liste von Registrars
    - Announces
  - Cache für PE-Auswahl
- RSerPool-APIs:
  - Basic Mode
  - Enhanced Mode



Das API der *rsplib*-Library: Basic Mode für PEs

UNIVERSITÄT DUISBURG ESSEN

- (Re-)Registrierungsschleife
  - z.B. als eigener Thread, damit Hauptprogramm (Server) nicht blockiert
  - Wartefunktion:
    - *rspSelect*() analog zu *select*()
    - behandelt auch Keep-Alives vom PR
- Deregistrierung
  - bei Beendigung

```
void registrationLoop() {
  struct timeval timeout;
  ...
  while(!shuttingDown) {
    rspRegister(poolHandle, ...);
    timeout.tv_sec  = reregistrationInterval;
    timeout.tv_usec = 0;
    rspSelect(0, NULL, NULL, NULL, &timeout);
  }
  rspDeregister(poolHandle, ...);
}
```

Das API der *rsplib*-Library: Basic Mode für PUs

UNIVERSITÄT DUISBURG ESSEN

- Server-Auswahl
  - API analog zu DNS-Abfrage
  - Kompatibilität zu

    getaddrinfo()
  - Statt

    Hostname -> IP-Adressen

    jetzt

    Pool Handle -> IP-Adressen
- Failover
  - Aufgabe der Applikation selbst
  - Rückmeldung an PR

```
poolHandle = "DownloadPool";
rspHandleResolution(poolHandle, strlen(poolHandle), &eai);
...
sd = socket(eai->ai_family,
                eai->ai_socktype, eai ->ai_protocol);
if(sd >= 0) {
  if(connect(sd, eai->ai_addr, eai->ai_addrlen) {
    ...
    if(failure) {
      rspReportFailure(poolHandle, strlen(poolHandle),
                            eai->ai_identifier);
      ...
    }
    ...
  }
}
```

Das API der *rsplib*-Library: Enhanced Mode für PEs

UNIVERSITÄT DUISBURG ESSEN

- API analog zu TCP-Sockets
  - Ablauf bei TCP-Sockets: *socket*() -> *bind*() -> *listen*() -> *accept*()
  - Jetzt: Sitzung statt Socket; ggf. automatischer Failover per Cookies möglich

```
void serviceThread(session) {
  rspSessionRead(session, command, ...);
  if(command is cookie) {
    Restore state;
    rspSessionRead(session, command, ...);
  }
  do {
    Handle command;
    rspSessionSendCookie(session, Current state);
    rspSessionRead(session, command, ...);
  } while(Session is alive);
}
```

```
int main(...) {
  poolElement = rspCreatePoolElement(poolHandle, ...);
  while(!shuttingDown) {
    rspSessionSelect(... poolElement ...);
    if(New Session) {
      session = rspAcceptSession(poolElement, ...);
      Create service thread for new session;
    }
  }
  rspDeletePoolElement(poolElement);
}
```

Das API der *rsplib*-Library: Enhanced Mode für PUs

UNIVERSITÄT DUISBURG ESSEN

- API analog zu TCP-Sockets
  - Ablauf bei TCP-Sockets: *socket*() -> *connect*() -> ... -> *close*()
  - Jetzt: Sitzung statt Socket; ggf. automatischer Failover per Cookies möglich

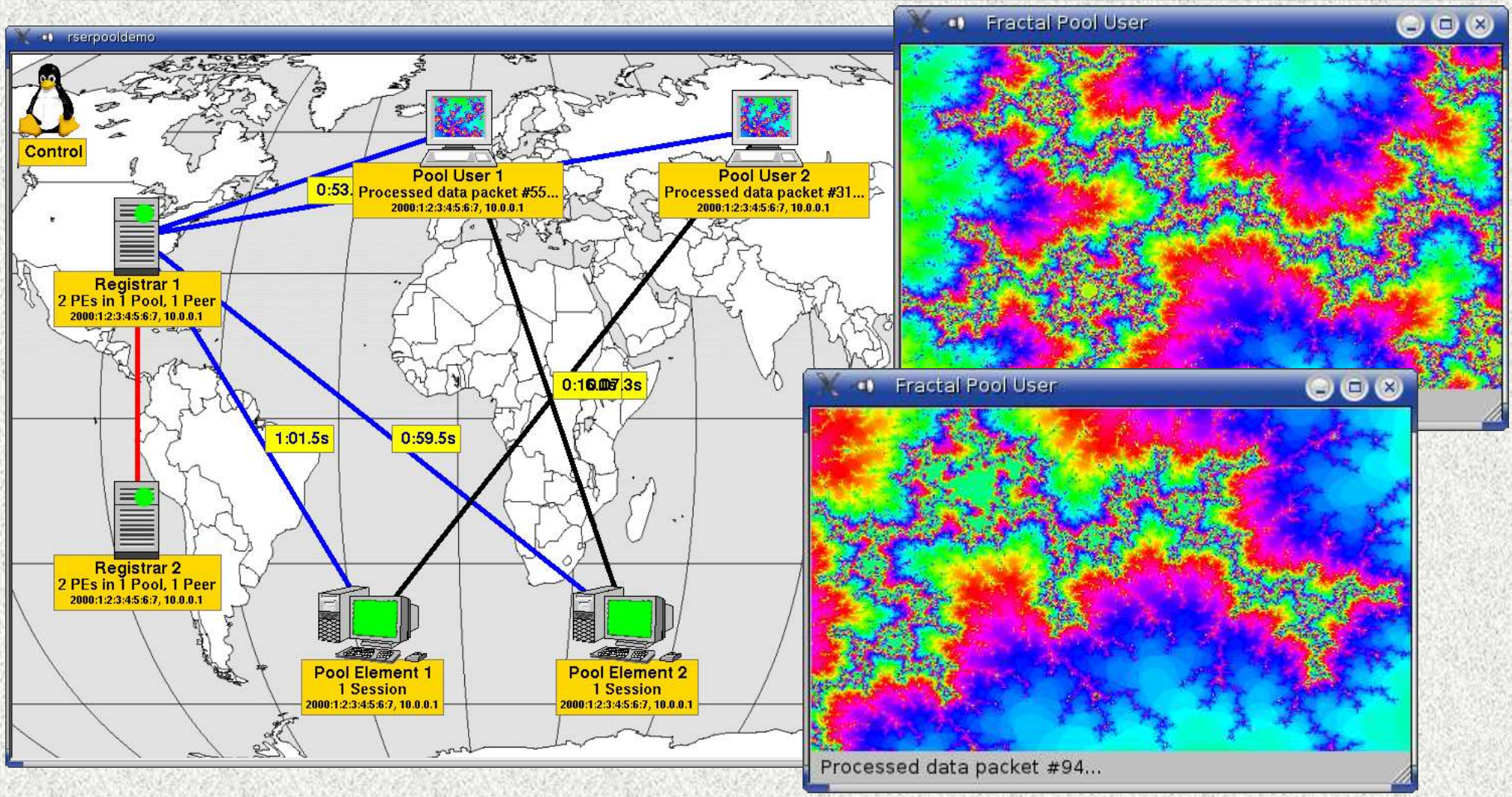
```
sd = rspCreateSession("DownloadPool", ...);
rspSessionWrite(sd, "GET Linux-CD.iso HTTP/1.0\r\n\r\n");
while((length = rspSessionRead(sd, buffer, ...)) > 0) {
   doSomething(buffer, length);
}
rspSessionClose(sd);
```

  - Anmerkung:

    doSomething() erhält ggf. Wiederholungen – je nach Cookie-Intervall!

rsplib-Demo!

UNIVERSITÄT DUISBURG ESSEN



Unsere RSerPool-Aktivitäten: Forschung

UNIVERSITÄT DUISBURG ESSEN

- Unsere Forschung wird seit Oktober 2004 durch die DFG gefördert
- Das *rspsim*-Simulationsmodell (in OMNeT++)
  - Abgeschlossene Aktivitäten:
    - Performanz von Pool Policys: [ICN2005]
    - Failover-Behandlung: [LCN2002] und [EuroMicro2005]
    - Handlespace-Verwaltung: [Contel2005]
  - Laufende Aktivitäten:
    - Performanz von Pool Policys in Ausfallszenarien
    - ENRP-Funktionalität und Skalierbarkeit
- Der *rsplib*-Prototyp
  - Abgeschlossene Aktivitäten:
    - Anwendbarkeit und Performanz für SCTP-Mobilität: [LCN2003]
    - Anwendbarkeit für Distributed Computing:
      Diplomarbeit [Zha2004], verifiziert Ideen aus [LCN2002] und [LCA2003]
  - Laufende Aktivitäten:
    - Verifikation der Performanz von Pool Policys aus unseren Simulationen
    - Anwendbarkeit für große Pools (Distributed Computing): Tests im PLANET LAB

von Simulation in die Realität

Unsere RSerPool-Aktivitäten: Standardisierung

UNIVERSITÄT DUISBURG ESSEN

- Standardisierung in der IETF
  - Unsere Standardisierungsbeiträge:
    - *draft-ietf-rserpool-policies-01.txt*

      beinhaltet Ergebnisse aus unseren Policy-Untersuchungen
    - *draft-ietf-rserpool-mib-01.txt*

      SNMP-MIB-Definition für RSerPool-Komponenten
    - API-Draft (noch nicht veröffentlicht)
  - IETF-Standardisierung erfordert „running code“ - den haben wir!
  - *rsplib*-Prototyp war die erste verfügbare RSerPool-Implementation

    Referenzimplementation der IETF RSerPool WG
  - Erste **erfolgreiche Interoperabilitätstests** mit proprietärer Motorola-Implementation beim 60th IETF, August 2004

    => Beschleunigung der Standardisierung

von Forschung zur Anwendung

Vielen Dank für Ihr Interesse!

Noch Fragen?

Projekt-Homepage:

http://tdrwww.exp-math.uni-essen.de/dreibholz/rserpool/

Thomas Dreibholz, dreibh@exp-math.uni-essen.de