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
Table of Contents

- Motivation
- What is Reliable Server Pooling?
  - Demo presentation
  - Architecture and terminology
  - Protocol stack
  - Features
- Our implementation RSPLIB
  - Design goals
  - Installation and test
  - Building blocks of the components
  - Usage of the RSerPool API
  - The „Scripting Service“
- Our Reliable Server Pooling activities

Thomas Dreibholz's Reliable Server Pooling Page
http://tdrwww.exp-math.uni-essen.de/dreibholz/rservpool/
Motivation (I)

- **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**
  \( \Rightarrow \) Concept for **server redundancy** is **required**

An Introduction to Reliable Server Pooling and the RSPLIB Implementation

Thomas Dreibholz
Motivation

**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

A combination of different research areas ...

Fault Tolerance
- Redundancy
- Checkpoints

... combined into a new IETF standard!

Reliable Server Pooling

Availability
- Session Failover
- Abort and Restart

Load Balancing
- Adaptive
- Non-Adaptive
An Introduction to Reliable Server Pooling and the RSPLIB Implementation
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

- **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

ASAP is IETF's first **Session Layer** standard!
The Handlespace and its Contents

- 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

- Transport of the announces as UDP packets via IP multicast

Multicast Domain

Registrar

PE #1

PU #1

Router

Hub

Switch

PE #2

PU #2

PE #3

PU #3

Multicast ASAP Announce
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

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

- **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

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

- **Download of the source archive:**
  - [http://tdrwww.iem.uni-due.de/dreibholz/rsrerpool/](http://tdrwww.iem.uni-due.de/dreibholz/rsrerpool/)

- **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

- **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

- **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

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

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

```c
/* 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:**
  ```c
  doSomething() may contain repetitions – depending on the cookie interval!
  ```
API similar to TCP sockets server:

- For TCP sockets: `socket() -> bind() -> listen() -> accept()`
- Again: Session (RSerPool socket) instead of transport connection!

```c
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);
}
```
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

- 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!

- 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

Interested in our RSerPool research papers and presentations? Have a look at our website!
Thank You for Your Attention!
Any Questions?

Visit Our Project Homepage:
http://tdrwww.iem.uni-due.de/dreibholz/rserpool/

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