On the security of reliable server pooling systems

Thomas Dreibholz*

Institute for Experimental Mathematics,
University of Duisburg-Essen,
Ellernstrasse 29, 45326 Essen, Germany
E-mail: dreibh@iem.uni-due.de
*Corresponding author

Xing Zhou

College of Information Science and Technology,
Hainan University,
Renmin Avenue 58, 570228 Haikou, Hainan, China
E-mail: zhouxing@hainu.edu.cn

Martin Becke, Jobin Pulinthanath and
Erwin P. Rathgeb

Institute for Experimental Mathematics,
University of Duisburg-Essen,
Ellernstrasse 29, 45326 Essen, Germany
E-mail: martin.becke@uni-due.de
E-mail: jp@iem.uni-due.de
E-mail: rathgeb@iem.uni-due.de

Wencai Du

College of Information Science and Technology,
Hainan University,
Renmin Avenue 58, 570228 Haikou, Hainan, China
E-mail: wencai@hainu.edu.cn

Abstract: In order to cope with the requirements of availability-critical internet services, reliable server pooling (RSerPool) has been developed as the new IETF standard for a lightweight server redundancy and session failover framework. While the service and pool management performance of RSerPool had already been the topic of various research papers, its security has not yet been widely examined. But security for availability-critical systems is crucial, since service outages – regardless of whether being caused by system failures or intentional denial of service (DoS) attacks – are not acceptable for the users of such systems.

In this article, we first introduce RSerPool as well as the underlying SCTP protocol. In an analysis of the attack threats, we will show the possibilities of an attacker to degrade the service provided by an RSerPool system. We will furthermore introduce possible countermeasures, in order to prevent attacks and improve the robustness of the systems. We will finally show the effectiveness of our proposed countermeasures using simulations. In order to validate our simulation results, we furthermore compare them to measurements from a real-world internet setup using the PlanetLab.

Keywords: reliable server pooling; RSerPool; security; attacks; robustness; performance analysis; PlanetLab.

Reference to this paper should be made as follows: Dreibholz, T., Zhou, X., Becke, M., Pulinthanath, J., Rathgeb, E.P. and Du, W. (2010) ‘On the security of reliable server pooling systems’, *Int. J. Intelligent Information and Database Systems*, Vol. 4, No. 6, pp.552–578.

Biographical notes: Thomas Dreibholz is an Assistant Professor in the Computer Networking Technology group at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany. Currently, his main research topics are reliable server pooling (RSerPool) and the stream control transmission protocol (SCTP). He is the author of various research papers at international conferences and in journals. Furthermore, he contributed multiple working group and individual submission drafts to the IETF RSerPool Working Group's standardisation process. He is co-author of multiple RFC documents published by the IETF. His research interests also include quality of service (QoS) and network security as well as concepts and protocols for the future internet.

Xing Zhou is a Full Professor of Computer Science at Hainan University and Vice Secretary-General of Hainan Information Industry Association of China. She received her Bachelor and Master degrees in Electrical Engineering from Chongqing University. She was an Advanced Visiting Scholar in Computer Science at Shanghai Jiaotong University (2001–2003) and a Visiting Scholar at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany (2006–2007). She is the author of six books, has published more than 40 articles in journals and is co-author of two drafts to the IETF RSerPool Working Group. Her research interests include RSerPool, network security, the protocols for next generation internet and web applications.

Martin Becke has studied Computer Science and Information Technology at the Osnabrueck and the Muenster Universities of Applied Sciences, Germany. He received his Master degree in Computer Science in 2006. For several years, he worked at a consulting agency as Technology Consultant on network and operating system development. In 2009, he became a Researcher at the Institute for Experimental Mathematics of the University of Duisburg-Essen to pursue his PhD. His current research focuses on transport protocols – particularly SCTP and multipath TCP – in combination with performance evaluation and network virtualisation techniques.

Jobin Pulinthanath has studied Business Informatics at the University of Duisburg-Essen, Germany and received his Diploma (Dipl.-Wirt.-Inf.) degree in 2007 for his thesis ‘Reliable transport of IPFIX-messages with the RSerPool-architecture’. Between 2007 and 2009 he was a member of the scientific staff in the Computer Networking Technology Group at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany. He is the co-author of multiple research papers on the stream control transmission protocol (SCTP). His research interests include also concepts and protocols for routing and network management.

Erwin P. Rathgeb holds the Alfried Krupp von Bohlen und Halbach-Chair for ‘Computer Networking Technology’ at the Institute for Experimental Mathematics, University of Duisburg-Essen, Germany. He is the author of a book on ATM and has published more than 50 papers in journals and at international conferences. He is a Senior Member of IEEE and a member of GI, IFIP and ITG where he is Chairman of the expert group on network security. His current research interests include network security as well as concepts and protocols for next generation internets, in particular SCTP and RSerPool.

Wencai Du is a Full Professor of Computer Science and Dean of the College of Information Science and Technology at Hainan University and President of Electronic Society of Hainan Province of China. He received his PhD in Informatics from the University of South Australia, two Master degrees in Geoinformatics from International Institute for Geo-Information Science and Earth Observation (ITC) at The Netherlands and Hohai University and Bachelor degree from Peking University of China. He has authored ten books and has published more than 80 articles. His research interests cover software engineering, the internet, computer network, interoperability, and web services.

1 Introduction and scope

When the internet was designed a long time ago, its main applications were e-mail and file transfer. On failures of servers, routers or network links, the users just waited for some time and tried again. This worked quite well for the application of that time, but new applications – which are widely used today – have a significantly higher demand for availability. For example, in the area of e-commerce, a service not being available will not gain any revenue. Also, there are many competitors on the internet. Potential customers can simply use the service of such a competitor – without ever coming back. In order to cope with the requirements of availability-critical services, the IETF has just published a generic, application-independent server pool (see Dreibholz and Rathgeb, 2008a) and session management (see Dreibholz, 2007) framework as RFCs: reliable server pooling (RSerPool) (see Lei et al., 2008). It is responsible for the required server redundancy and session management. Various research papers have already been published on the load balancing (see Dreibholz and Rathgeb, 2005b; Dreibholz, 2007) and server failure handling (see Dreibholz and Rathgeb, 2009) features of RSerPool, but there have only been simulations of some security concepts by Schöttle et al. (2008), Dreibholz et al. (2008) and a corresponding evaluation in a lab setup by Zhou et al. (2009a).

In our paper Dreibholz et al. (2009c) for the ConTEL 2009, we have presented an experimental evaluation of the mechanisms described in Zhou et al. (2009a) and Dreibholz et al. (2008) in a real-world internet setup, based on the PlanetLab (see Peterson and Roscoe, 2006). This article is the extended version of this paper, extending the content of our paper by a detailed description of the attack threats on RSerPool systems. Also, we cover the endpoint handlespace redundancy protocol (ENRP) protocol as well as the underlying stream control transmission protocol (SCTP) protocol, which have not been addressed by the paper.

This article is structured as follows. First, we introduce the SCTP protocol in Section 2. Next, we present the RSerPool framework in Section 3. In Section 4, we offer an overview of attack threats on RSerPool systems. Our proposed attack countermeasures are presented in Section 5. Finally, by using the performance metrics introduced in Section 6 and our simulative and experimental PlanetLab setups described in Section 7, we provide an evaluation of the most important security mechanisms in Section 8.

2 The SCTP protocol

The SCTP protocol – which is defined as RFC by Stewart (2007) – is a connection-oriented, general-purpose, unicast transport protocol providing the reliable transport of user messages. An SCTP connection is denoted as association. Each SCTP endpoint can use multiple IPv4 and/or IPv6 addresses to provide network fault tolerance. The addresses used by the endpoints are negotiated during association setup. This redundancy feature is called multi-homing and is illustrated in Figure 2 (see also Jungmaier et al., 2001; Dreibholz et al., 2003, for more details). User data and control information is transported in so-called chunks, which are bundled into SCTP packets.

Several SCTP extensions have been developed and standardised. The most important extensions with relevance to this article are:

- The dynamic address reconfiguration extension (Add-IP) defined as RFC by Stewart et al. (2007) provides interface and address changes during association runtime. In particular, it allows for mobility as examined by Dreibholz et al. (2003) or an interruption-free IPv6 site renumbering – or even a seamless migration from IPv4 to IPv6 as explained by Dreibholz and Rathgeb (2005a).
- The chunk authentication extension defined as RFC by Tüxen et al. (2007) provides the authenticity and integrity for the chunks of an SCTP association by using keys negotiated during association setup or pre-shared keys. Chunk authentication is required to avoid association hijacking when using Add-IP. However, it does not provide confidentiality.
- The packet drop extension defined by Stewart et al. (2009) allows for notifying a sender of dropped packets due to bit errors. In this case, retransmissions can be sent without reducing the congestion window. This feature improves the association throughput over low-quality, high-delay satellite links.
- CMT-SCTP (see Iyengar et al., (2006); Dreibholz et al., 2010b) is a concurrent multipath transfer (CMT) extension for SCTP. Unlike standard SCTP as defined in Stewart (2007), it utilises all paths for data transport (not just a designated primary path). Combined with resource pooling (RP), the CMT/RP-SCTP extension introduced by Dreibholz et al. (2010a) allows for TCP-friendly CMT transport over the internet.

Furthermore, SCTP provides some security features: first, SCTP applies a four-way handshake for association setup – in contrast to the three-way handshake of TCP which is susceptible to the SYN-flooding attack. The principle of the four-way handshake is illustrated in Figure 1: the endpoint A starts establishing an association to the endpoint B using an INIT chunk. Endpoint B stores all information about the new association into a signed cookie, which is returned to endpoint A in an INIT ACK chunk. After that, endpoint B releases all resources associated with the new association. The cookie is returned from endpoint A in a COOKIE ECHO chunk. By checking its signature, endpoint B can ensure that the cookie is valid and unaltered as well as that the other endpoint is existing and reachable (since the cookie has been returned by endpoint A, its address cannot be spoofed). Using the association information from the cookie, endpoint B can restore the corresponding data structures. The successful association establishment is finally signalised by a COOKIE ACK chunk.

Figure 1 SCTP association establishment by four-way handshake (see online version for colours)

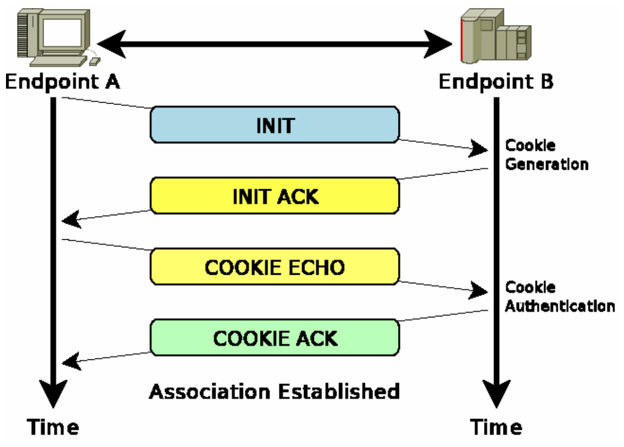
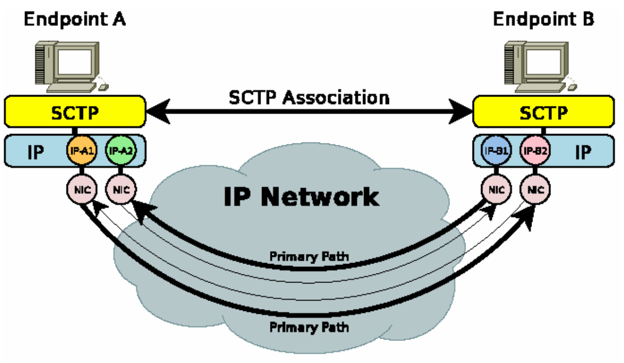


Figure 2 Multi-homing (see online version for colours)



Another security feature of SCTP against blind flooding attacks is the so-called verification tag. This is a 32-bit number which is configured for each communication direction during association establishment. An endpoint writes its verification tag into all outgoing packets; an attacker trying to inject a packet into the association (e.g., in order to abort the association similar to the RST attack of TCP) needs to guess the verification tag – which requires ample bandwidth and time for brute-force trials. The peer endpoint simply ignores packets containing a wrong verification tag.

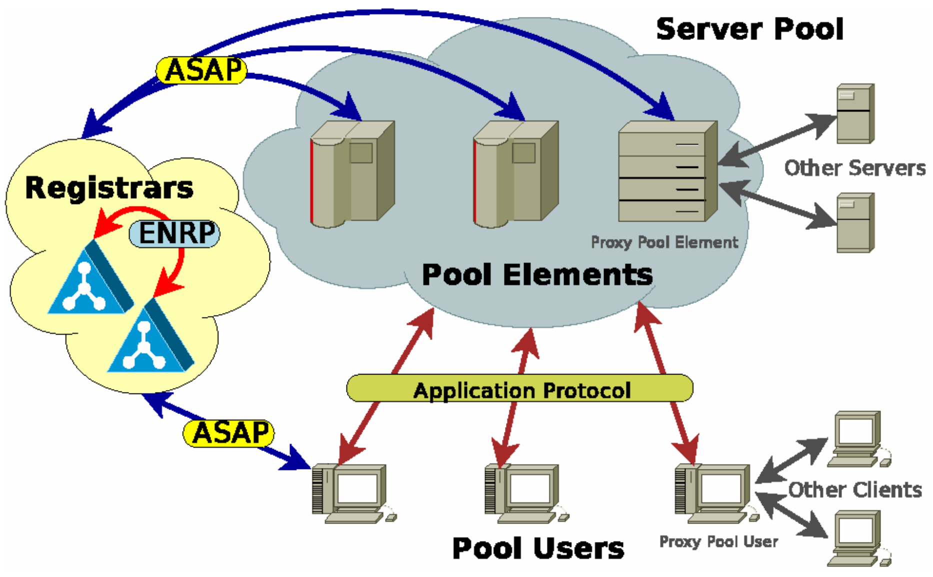
In order to support authenticity, integrity and confidentiality for the user data transport, SCTP can – similar to TCP – also be used with transport layer security (TLS) by Dierks and Rescorla (2008). But since TLS has been designed for a byte-stream-oriented, single-homed transport, it poorly supports the enhanced protocol features of SCTP – particularly its message-oriented transport, multi-homing and dynamic address reconfiguration. An enhanced version of TLS – denoted as datagram TLS (D-TLS) – and its adaptation to SCTP is described by Hohendorf et al. (2007). An alternative to TLS is IPsec by Kent and Seo (2005). However, multi-homing results in the need for a large number of security associations (SA). Using the optimisations described by Bellovin et al. (2003), an SCTP transport over IPsec can be realised efficiently. Another alternative is the secure-SCTP (S-SCTP) extension by Unurkhaan (2005), which directly realises authenticity, integrity and confidentiality within the SCTP stack.

3 The RSerPool architecture

An overview of the RSerPool architecture – which is defined as RFC by Lei et al. (2008) – is provided in Figure 3. There are three types of components:

- Pool element (PE) denotes a server in a pool. PEs in the same pool provide the same service.
- Pool user (PU) denotes a client using the service of a pool.
- Pool registrar (PR) is the management component for the pools.

Figure 3 The RSerPool architecture (see online version for colours)



The set of all pools within an operation scope (e.g., an organisation, a company or a department) is denoted as handle-space. Clearly, a single PR would be a single point of failure. Therefore, PRs also have to be redundant. Within the handle-space, each pool is identified by a unique pool handle (PH).

RSerPool provides support for non-RSerPool nodes by proxies: proxy PUs connect non-RSerPool clients to a server pool; Proxy PEs let non-RSerPool servers join a pool.

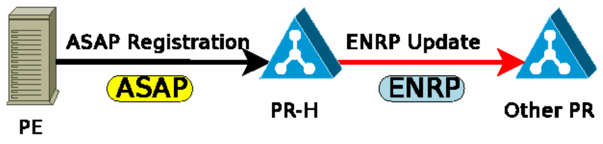
3.1 Registrar operations

The PRs of an operation scope synchronise their view of the handle-space by using the ENRP (defined as RFC by Xie et al., 2008), transported via SCTP (see Section 2). In contrast to grid computing (see Foster, 2002), an operation scope is restricted to a single administrative domain. That is, all of its components are under the control of the same authority (e.g., a company). This property leads to small management overhead (details are described by Dreibholz and Rathgeb, 2008a, 2005a), which also allows for RSerPool usage on devices having only limited memory and CPU resources (e.g., telecommunications equipment). Nevertheless, PEs may be distributed globally to continue their service even in case of localised disasters (e.g., an earthquake). Such scenarios are examined in more detail by Dreibholz and Rathgeb (2007). Each PR in the operation scope is identified by a PR ID, which is a randomly chosen 32-bit number.

3.2 PE operations

Within their operation scope, the PEs may choose an arbitrary PR to register into a pool by using the aggregate server access protocol (ASAP) (defined as RFC by Stewart et al., 2008). The registration is performed by using an ASAP registration message. Within its pool, a PE is characterised by its PE ID, which is a randomly chosen 32-bit number. Upon registration at a PR, the chosen PR becomes the home-PR (PR-H) of the newly registered PE. A PR-H is responsible for monitoring the availability of its PEs by ASAP endpoint keep alive messages (to be acknowledged by a PE via an ASAP endpoint keep alive ack message within a configured timeout). The PR-H propagates the information about its PEs to the other PRs of the operation scope via ENRP Update messages. This principle is illustrated in Figure 4.

Figure 4 The principle of PE registration (see online version for colours)



PEs re-register regularly in an interval denoted as registration lifetime and for information updates. Similar to the registration, a re-registration is performed by using another ASAP registration message. PEs may intentionally deregister from the pool by using an ASAP deregistration message. Also like for the registration, the PR-H makes the deregistration known to the other PRs within the operation scope by using an ENRP update message.

3.3 Takeover procedure

As soon as a PE detects the failure of its PR-H (i.e., its request is not answered within a given timeout), it simply tries another PR of the operation scope for its registration and deregistration requests. However, as a double safeguard, the remaining PRs also negotiate a takeover of the PEs managed by a dead PR. This ensures that each PE again gets a working PR-H as soon as possible. The PRs of an operation scope monitor the availability of each other PR by using ENRP presence messages, which are transmitted regularly. If there is no ENRP presence within a given timeout, the peer is assumed to be dead and a so-called takeover procedure (see also Zhou et al., 2009b, for details) is initiated for the PEs managed by the dead PR: from all PRs having started this takeover procedure, the PR with the highest PR ID takes over the ownership of these PEs. The PEs are informed about being taken over by their new PR-H via an ASAP endpoint keep-alive with home-flag set. The PEs are requested to adopt the sender of this home-flagged message as their new PR-H.

3.4 PU operations

In order to access the service of a pool given by its PH, a PU requests a PE selection from an arbitrary PR of the operation scope, again by using ASAP. This selection procedure is denoted as handle resolution. Upon reception of a so-called ASAP handle resolution message the PR selects the requested list of PE identities and returns them in an ASAP handle resolution response message. The pool-specific selection rule is denoted as pool policy. Two classes of load distribution policies are supported: non-adaptive and adaptive strategies (a detailed overview is provided by Dreibholz, 2007; Dreibholz and Rathgeb, 2005b, 2008a). While adaptive strategies base their selections on the current PE state (which requires up-to-date information), non-adaptive algorithms do not need such data. A basic set of adaptive and non-adaptive pool policies is defined as RFC by Dreibholz and Tüxen (2008).

Relevant for this article are the non-adaptive policies round robin (RR) and random (RAND) as well as the adaptive policies least used (LU) and least used with degradation (LUD). LU selects the least-used PE, according to up-to-date application-specific load information. RR selection is applied among multiple least-loaded PEs. LUD, which is evaluated by Zhou et al. (2008), furthermore introduces a load decrement constant which is added to the actual load each time a PE is selected. This mechanism compensates inaccurate load states due to delayed updates. An update resets the load to the actual load value. It is important to differentiate policies between stateful and stateless, as is explained by Dreibholz and Rathgeb (2005b): for a stateful policy, a selection is influenced by the previous choice. For example, the 'in turn' selection of RR is stateful. LUD is also stateful. On the order hand, LU and RAND are stateless; they will – regardless of a previous selection – return a least-loaded or random element.

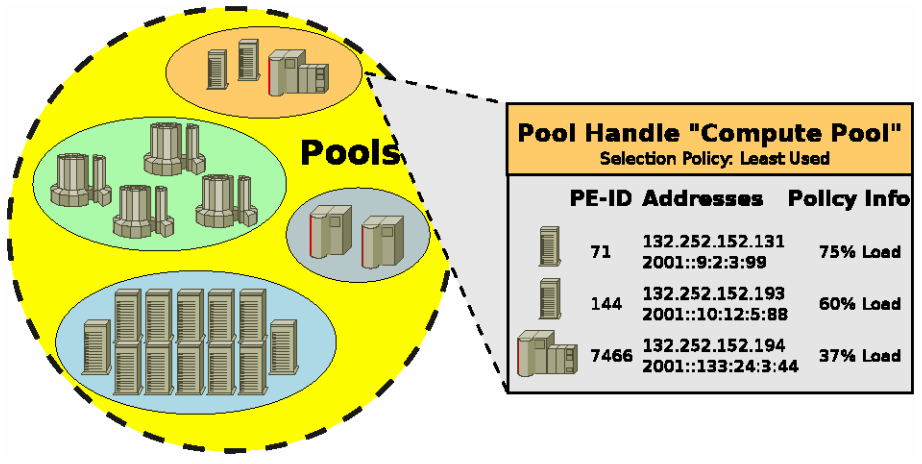
A PE may fail, e.g., due to hardware or network failures. Since there is a certain latency between the actual failure of a PE and the removal of its entry from the handle-space – depending on the interval and timeout for the ASAP endpoint keep alive monitoring – the PUs may report unreachable PEs to a PR by using an ASAP endpoint unreachable message. A PR locally counts these reports for each PE and when reaching the threshold MaxBadPEReports (default is 3, as defined in the RFC by Stewart et al., 2008), the PR may decide to remove the PE from the handle-space. The counter of a PE is reset upon its re-registration. More details on this threshold and guidelines for its configuration can be found in Dreibholz and Rathgeb (2009).

3.5 A handle-space example

Figure 5 depicts an example of a handle-space containing four pools. The pool using the PH 'compute pool' consists of 3 dual-homed PEs (IPv4 and IPv6). Since its pool policy is

LU, the handle-space also stores the latest known load state of each PE. Clearly, the next handle resolution in this pool will return PE #7466, since its load state is lowest.

Figure 5 A handlespace example (see online version for colours)


Pools
Pool Handle "Compute Pool"
Selection Policy: Least Used
PE-ID Addresses Policy Info
71 132.252.152.131 2001::9:2:3:99 75% Load
144 132.252.152.193 2001::10:12:5:88 60% Load
7466 132.252.152.194 2001::133:24:3:44 37% Load

3.6 *Automatic configuration*

RSerPool components need to know the PRs of their operation scope. While it is of course possible to configure a list of PRs into each component, RSerPool also provides an auto-configuration feature: PRs may send so-called announces, i.e., ASAP announce and ENRP Presence messages which are regularly sent over UDP via IP multicast. Unlike broadcasts, multicast messages can also be transported over routers (at least, this is easily possible within LANs). The announces of the PRs can be heard by the other components, which can maintain a list of currently available PRs. That is, RSerPool components are usually just turned on and everything works automatically.

An example is provided by Figure 6 for the ASAPbased PU/PE configuration: all PEs and PUs within the multicast domain (e.g., a company or department LAN) can learn the identity of the PE automatically. Components outside of this domain (e.g., off-site systems in the internet) need manual configuration.

3.7 *The RSerPool protocol stack*

Figure 7 presents an illustration of the RSerPool protocol stack: a PR provides ENRP and ASAP services to PRs and PEs/PUs respectively. But between PU and PE, ASAP provides a session layer protocol in the OSI model. This makes ASAP the first IETF standard for a session layer protocol. From the perspective of the application layer, the PU side establishes a session with a pool. ASAP takes care of selecting a PE of the pool, initiating and maintaining the underlying transport connection and triggering a failover procedure when the PE becomes unavailable.

The transport layer protocol is by default SCTP over possibly multi-homed IPv4 and/or IPv6 – except for the UDP-based automatic configuration announces (see Subsection 3.6) which are not shown here for readability reasons.

Figure 6 Automatic configuration by ASAP announces (see online version for colours)

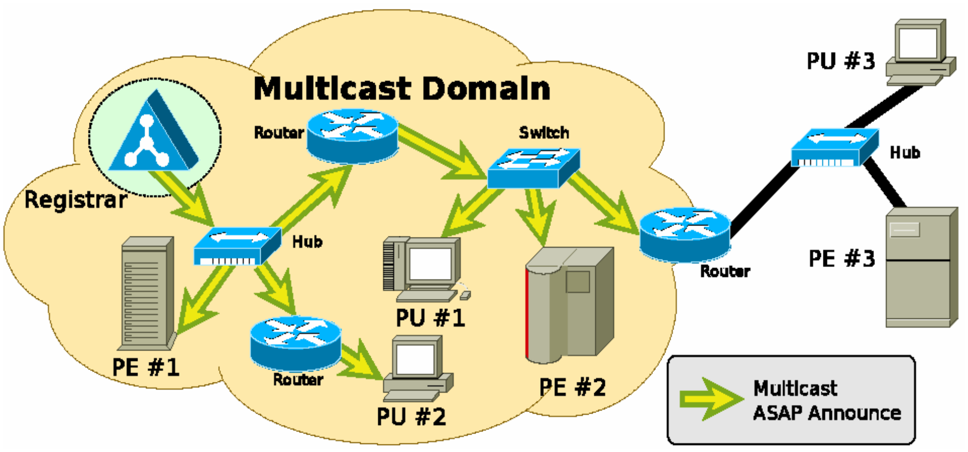
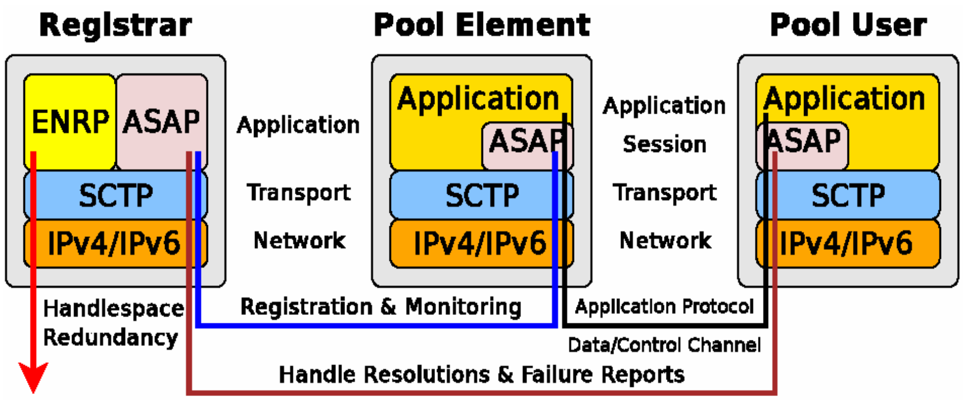


Figure 7 The RSerPool protocol stack (see online version for colours)



3.8 Application scenarios

While the initial motivation of RSerPool has been the availability of SS7 (Signalling System No. 7, see ITU-T, 1993) services over IP networks, it has been designed for application independence. Current research on the applicability and the performance of RSerPool includes application scenarios like VoIP with SIP (see Conrad et al., 2002), SCTP-based mobility (see Dreibholz et al., 2003), web server pools, e-commerce systems (see Dreibholz, 2002), video on demand (see Maharana and Rathna, 2006), battlefield networks (see Uyar et al., 2004), IP flow information export (IPFIX) (see Dreibholz et al., 2009a) and workload distribution (see Dreibholz and Rathgeb, 2008b).

A detailed comparison of the RSerPool architecture to other frameworks and protocols for load balancing and service availability – including DNS, CORBA and Layer-4/Layer-7 switching – is provided by Loughney et al. (2005). We neglect a further overview here, since this would exceed the scope of this article.

4 Attack threats on RSerPool systems

The SCTP protocol – as introduced in Section 2 – already contains the four-way handshake and the verification tag as countermeasures against blind flooding attacks (see also Hohendorf et al., 2007). Furthermore, chunk authentication can be applied to avoid association hijacking, which is particularly useful when using dynamic address reconfiguration in changing network environments. SCTP as the underlying transport protocol already prevents simple address spoofing attacks: each network-layer address under which a PE is registered must be part of the SCTP association between PE and PR. The ASAP protocol (see Stewart et al., 2008) requires the addresses to be validated by SCTP. But it is just sufficient for an attacker to hold a registration association to the PR and silently drop all incoming PU requests.

As defined for RSerPool in the RFC by Stillman et al. (2008), the application of TLS (see Jungmaier et al., 2002), IPsec (see Bellovin et al., 2003) or secure SCTP (see Unurkhaan, 2005) is required in order to ensure authenticity, integrity and confidentiality. Nevertheless, relying on these techniques alone is still not sufficient: a component having authenticated successfully may still be misbehaving – for example when it is taken over by an attacker. Therefore, it is important to analyse the implications to the service under denial of service (DoS) attack situations, in order to apply effective attack countermeasures. Such counter-measures should at least reduce the impact of attacks on the performance of the system.

Table 1 lists the attack threats on RSerPool systems for the protocols ENRP and ASAP. These attack threats and their impact will be explained in the following subsections.

Table 1 Attack threats on the RSerPool systems

Protocol	*Attacker role*	*Threat*	*Solved by authentication*
ENRP	Registrar	Fake ENRP announce	Yes
ENRP	Registrar	Handlespace manipulation	Yes
ENRP	Registrar	Takeover manipulation	Yes
ASAP	Registrar	Fake ASAP announce	Yes
ASAP	Registrar	Hostile takeover	Yes
ASAP	Registrar	Malicious responses to PEs	Yes
ASAP	Registrar	Malicious responses to PUs	Yes
ASAP	PE	Registration hijacking	Partly
ASAP	PE	Deregistration hijacking	Partly
ASAP	PE	Registration/re-registration flooding	Partly
ASAP	PE	Deregistration flooding	Partly
ASAP	PE	Fake registration	No
ASAP	PU	Handle resolution flooding	No
ASAP	PU	PE impeachment	No

4.1 Threats on the ENRP protocol

The ENRP protocol is highly security-critical. An attacker gaining ENRP access to a PR is able to perform the following attacks:

- *Fake ENRP announce:* an attacker may announce itself as PR, using UDP-based multicast ENRP presence messages (see Subsection 3.6). Since UDP is connection-less, an attacker may even apply IP-spoofing and send such announces from an arbitrary location within the same multicast domain. The effort to perform this kind of attack is therefore very small – but its impact on the system is very high.
- *Handlespace manipulation:* The attacker is able to arbitrarily manipulate the content of the handlespace (e.g., by adding or removing PE entries, modifying policy information, etc.).
- *Takeover manipulation:* The ENRP takeover mechanism can be used to take ownership of all PEs in the handlespace. After that, the PEs will adopt the attacker's PR as their PR-H and the attacker gains control over the synchronisation of all handlespace information with the real PRs.

Under the assumption that an attacker cannot obtain a valid PR identity, the usage of authentication will solve the attack threats on ENRP. Furthermore, for redundancy reasons a number of 2 to 5 PRs is realistic for an operation scope (see also Zhou et al., 2009b). The PRs can be placed at protected places (e.g., locked server rooms, etc.), so that the possibility of attackers to directly manipulate a PR remains small. These properties make the likeliness of ENRP-based attacks quite small. However, in Subsection 5.1, we will discuss some further countermeasures against such attacks.

4.2 Threats on the ASAP protocol

While the ENRP protocol is only used among the small number of PRs, ASAP is used among all RSerPool components. In large setups, the number of PEs and Pus may easily reach several thousands of PEs and PUs (see also Dreibholz and Rathgeb, 2008a) which are located at much less protected places (e.g., even on end-user PCs). The probability of attacks on the ASAP protocol is therefore significantly higher.

4.2.1 A malicious registrar

The first four ASAP-based attack threats are attackers in the role of malicious PRs:

- *Fake ASAP announce:* Similar to ENRP-based fake announces, an attacker can send fake ASAP announces, via UDP-based multicast messages from anywhere in the multicast domain. This can lead to PEs and PUs using the attacker's system as PR.
- *Hostile takeover:* In order to let a PE adopt the attacker's PR as PR-H, it has to send ASAP endpoint keep-alive messages with home-flag set (see Subsection 3.3). To perform this attack, the attacker only has to guess the ASAP endpoint address of the PE: its IP addresses are known (they may be obtained by a handle resolution), only the 16-bit SCTP port number has to be guessed.

- *Malicious responses to PEs:* An attacker can claim to be a PR, so that it is chosen by PEs as their PR-H. By just returning valid responses to the ASAP registration messages, the PEs will not become known in the 'real' pool, i.e., their service cannot be used by PUs.
- *Malicious responses to PUs:* PUs assuming the attacker to be a PR will use it for PE selections. The attacker can simply return an empty list, which means for a PU that the pool is currently empty (and the requested service is currently unavailable).

Under the assumption that an attacker cannot obtain the identity of a PR in the operation scope, authentication will solve the problem of malicious PRs: a PR has to authenticate to a PE or PU, which ensures that the PR is valid. In Subsection 5.1, we will discuss some mechanisms to cope with untrustworthy PRs.

4.2.2 A malicious PE

An attacker in the role of a PE introduces the following threats:

- Registration hijacking registrations: and re-registrations may be performed at an arbitrary PR of the operation scope. If the PE is already registered, a further registration simply updates the existing registration. PEs are identified by their PE ID, i.e., an attacker just has to use the known ID of a PE to hijack its registration. Particularly, the registration update by the attacker could contain malicious policy information or wrong transport addresses.
- *Deregistration hijacking:* Similar to registration hijacking, the attacker can perform the same kind of attack with a deregistration. A known PE (identified by its PE ID) will be removed from the handlespace.
- Registration/re-registration flooding: An attacker can flood its PR-H with ASAP registration messages. The PR-H will propagate these updates to all other PRs in the handlespace. These operations consume CPU power and/or memory at the PRs.
- *Deregistration flooding:* Like for the registration/re-registration flooding, a similar kind of attack is possible with deregistrations.
- *Fake registration:* An attacker can create fake registrations. By appropriately configuring the policy information of such fake entries, they will be selected upon handle resolution; PUs try to contact the fake PEs which of course will not provide any (useful) service. We will demonstrate this problem in Subsection 8.1.

The authentication of PEs to their PR-H will not fully solve these problems: as soon as an attacker can obtain a valid PE authentication, [e.g., by exploiting a software bug on one of the possibly thousands of PEs in an operation scope (see Dreibholz and Rathgeb, 2008a)], its ASAP registration and ASAP deregistration messages are processed by PRs. We will discuss the impact of such attacks and countermeasures in Subsection 5.2.

4.2.3 A malicious PU

An attacker in the role of a PU introduces the following threats:

- *Handle resolution flooding:* An attacker can flood a PR with ASAP handle resolution messages. The resulting PE selections not only consume CPU power but can – in combination with a stateful pool policy (see Subsection 3.4) – also influence the selection performance. We will demonstrate this problem in Subsection 8.3.
- *PE impeachment:* When using PU-based endpoint monitoring (see Subsection 3.4), the attacker can use ASAP endpoint unreachable reports to impeach PEs, i.e., to let a PR assume them as dead and remove them from the handlespace. The performance impact of such an attack is very severe, as we will show in Subsection 8.3.

Similar to the PE-based attacks, also the authentication of PUs will not fully solve these threats: having stolen a valid PU authorisation, the ASAP handle resolution and ASAP Endpoint Unreachable messages will be processed by PRs. Countermeasures to this kind of attacks will be presented in Subsection 5.3.

5 Our attack countermeasure mechanisms

5.1 Countermeasures against malicious registrars

Under the assumption of trustworthy PRs, the ENRP-based attack treats fake ENRP announce, handlespace manipulation and takeover manipulation (see Subsection 4.1) as well as the ASAP-based threats fake ASAP announce, hostile takeover and malicious responses to PEs and PUs can be solved by applying authentication of PRs to other PRs (ENRP) and PEs as well PUs (ASAP).

In order to cope with untrustworthy PRs, ideas from the area of peer-to-peer (P2P) networks – where nodes cooperate with totally unknown and possibly hostile peers – could be adapted. Aberer and Despotovic (2001) present data structures and algorithms for P2P networks to assess trust by computing the reputation of an agent from its former interactions with other agents. But while the application of such mechanisms is also adaptable to the interaction with PRs, their application adds a significant level of complexity to the otherwise lightweight RSerPool framework. Therefore, their usefulness in the – presumably – controlled single administrative domain of an RSerPool operation scope may be limited. A more straightforward approach to cope with malicious PR behaviour in the operation scope may be to keep the PRs and the performance of the pools under a tight control by monitoring, e.g., based on the RSerPool SNMP interface defined as RFC by Dreibholz and Mulik (2009).

5.2 Countermeasures against malicious PEs

Registration and deregistration hijacking

In order to cope with the threats of registration and deregistration hijacking, it is necessary to ensure that the PE identity performing a re-registration or deregistration is the same that has been performed the original registration. It is therefore necessary to add certified identity information into the ASAP registration and ASAP deregistration messages (which currently contain PH and PE ID only). Then, the PR can distribute this information by ENRP Update messages having this information also added. This allows for checking the validity of re-registrations and deregistrations by all PRs of the operation scope.

(Re-)registration and deregistration flooding

When successfully authenticated as PE, an attacker can perform (re-)registration and deregistration flooding, i.e., trying to overload the PR's CPU capacity by such requests. However, the handlespace management of RSerPool systems can be realised very efficiently when using appropriate data structures for the information storage. Dreibholz and Rathgeb (2008a) show that even a low-performance CPU is able to handle many thousands of such operations per second. Further combining a rate threshold with the authenticated identity will solve the attack problem.

Fake registration

Without further countermeasure mechanisms, an authenticated PE may perform an almost infinite number of registrations. Using a new PE ID in each ASAP registration message, the handlespace can be flooded with fake PE entries. Therefore, the starting point for a countermeasure is a restriction of the number of PE registrations a single PE identity is allowed to create. In order to retain the 'lightweight' property of RSerPool (see Dreibholz and Rathgeb, 2008a) and to avoid synchronising such numbers among PRs, our countermeasure approach first introduces a so-called registration authorisation ticket (suggested by us in Dreibholz et al., 2008), which consists of:

1 the pool's PH and a fixed PE ID (or an ID range for proxy PEs)

2 minimum/maximum policy information settings (e.g., a lower bound on the LUD load decrement)

3 a signature of the ticket by a trustworthy authority (to be explained below).

Since ASAP messages use a TLV structure (see Dreibholz, 2007, Section 3.8, for a detailed description), it is easily possible to add the registration authorisation ticket to the ASAP registration message sent from the PE to its PR-H. Using the ticket, the PR-H can verify the validity of the request by checking the signature and ensuring that the PE's policy settings are within the valid range granted by the ticket. These checks are possible with additional time complexity in $O(1)$. As a result, an attacker stealing the identity of a real PE would only be able to masquerade as this specific PE. Particularly, it is neither necessary to change the protocols (except for adding the ticket TLV structure described above) nor to perform additional ENRP-based synchronisation of authorisation information among the PRs of the operation scope. Our approach only requires a trusted authority for issuing the tickets, e.g., a Kerberos service (see Neuman et al., 2005, for details). Due to the restriction of RSerPool to a single administrative domain (as described in Section 3), the effort to fulfil this requirement is feasible at reasonable costs – and this effort is worthwhile, as we will show in Subsection 8.2.

5.3 Countermeasures against malicious PUs

PE impeachment

PU-based endpoint monitoring using ASAP endpoint unreachable messages to report failed PEs empowers a malicious PU to impeach PEs. In order to cope with this problem, we first introduce a PU identification which is certified by a trusted authority and which can be verified by the PR (similar to the registration authorisation ticket for PEs, as explained in Subsection 5.2). This allows for tracking the number of failure reports sent by a PU for a certain pool (given by its PH): the PR simply has to memorise (to be explained later) the PH for which a certain PU has reported unreachable PEs. After that, multiple reports coming for the same pool can simply be ignored. Since the unreachability count for each PE is a PR-local variable, no synchronisation among PRs is necessary. That is, an attacker is unable to cause harm to the service by simply sending its unreachability reports for the same PE to different PRs.

Storing each reported PE identity would be exploitable in the form of a so-called computational complexity attack as introduced by Crosby and Wallach (2003). An attacker would just have to send a large number of ASAP endpoint unreachables with random PE IDs to exhaust the memory of the PR. Therefore, our approach applies a hash-based per-PU report blackboard (as suggested by us in Zhou et al., 2009a): the hash function Ψ maps a PE's PH into a bucket:

$$\Psi(\text{PH}) = \Phi(\text{PH}) \text{ MOD number of buckets.}$$

Φ denotes a hash function that is not easily guessable by the attacker. This property is provided by so-called universal hash functions (see Crosby and Wallach, 2003 for details), which are – in contrast to cryptographic hash functions like MD5 (see Rivest, 1992) or SHA1 (see Eastlake and Jones, 2001) – very efficiently computable.

Each bucket contains the time stamps of the latest up to MaxEntries ASAP endpoint unreachable messages for the corresponding bucket. Then, the endpoint unreachable report rate can be calculated as:

$$\text{Rate}_{\text{EU}} = \frac{\text{Number of time stamps}}{\text{TimeStamp}_{\text{last}} - \text{TimeStamp}_{\text{first}}} \tag{1}$$

Upon reception of an ASAP Endpoint unreachable from a PU, a PR has to update the corresponding bucket entry of the reported PE. If the rate in equation (1) exceeds the configured threshold MaxEURate, the unreachability report is silently ignored. The time complexity as well as the space complexity for this operation are in $O(1)$.

Handle resolution flooding attack countermeasures

In order to cope with handle resolution flooding, a hash-based approach similar to our countermeasure against PE impeachment attacks can be applied by introducing the handle resolution rate threshold MaxHRRate. When this threshold is exceeded, the PR returns an empty list in the ASAP handle resolution response. This indicates a currently empty pool and a legitimate PU would therefore try again some time (e.g., one minute) later.

5.4 Countermeasure alternatives

Schöttle et al. (2008) presents statistical anomaly detection as an alternative approach for detecting and handling attacks: instead of specifying fixed thresholds, the behaviour of the majority of nodes is assumed to be ‘normal’. Differing behaviour – which is necessary for an effective attack – is denoted as an anomaly. However, this approach can – by definition – only detect attackers if their number is smaller than the number of legitimate components. Furthermore, obtaining the ‘normal’ behaviour is more resource-intensive than simple thresholds. But the advantage of this approach is that the system can automatically adapt to a changing environment, e.g., the deployment or testing of new applications.

6 Quantifying an RSerPool system

As application model for our quantitative performance analysis, we use the model of Dreibholz (2007): the service provider side of an RSerPool system consists of a pool of PEs. Each PE has a request handling capacity, which we define in the abstract unit of calculations per second. An application-specific view of capacity may be mapped to this definition, e.g., CPU cycles. Each request consumes a certain number of calculations; we call this number request size. A PE can handle multiple requests simultaneously – in a processor sharing mode as provided by multitasking operating systems.

On the service user side, there is a set of PUs. The number of PUs can be given by the ratio between Pus and PEs (*PU:PE ratio*), which defines the parallelism of the request handling. Each PU generates a new request in an interval denoted as request interval. Requests are queued and sequentially assigned.

The total delay for handling a request d_{Handling} *is defined* as the sum of queuing delay d_{Queuing}, startup delay d_{Startup} (dequeuing until reception of acceptance acknowledgement) and processing time $d_{\text{Processing}}$ (acceptance until finish):

$$d_{\text{Handling}} = d_{\text{Queuing}} + d_{\text{Startup}} + d_{\text{Processing}}. \tag{2}$$

That is, d_{Handling} not only incorporates the time required for processing the request, but also the latencies of queuing, server selection and message transport. The user-side performance metric is the handling speed, which is defined as:

$$\text{HandlingSpeed} = \frac{\text{RequestSize}}{d_{\text{Handling}}}.$$

For convenience, the handling speed (in calculations/s) is represented in % of the average PE capacity.

Using the definitions above, it is possible to delineate the average system utilisation U (for NumPEs servers and total pool capacity PoolCapacity) as:

$$\text{U} = \text{NumPEs*puToPERatio*}\frac{\frac{\text{RequestSize}}{\text{RequestInterval}}}{\text{PoolCapacity}}. \tag{3}$$

Obviously, the provider-side performance metric is the system utilisation, since only utilised servers gain revenue. In practise, a well-designed client/server system is dimensioned for a certain target system utilisation of e.g., 50%. By setting any two of the parameters (PU:PE ratio, request interval and request size), the value of the third one can be calculated using equation (3) (see also Dreibholz, 2007; Dreibholz and Rathgeb, 2005b).

7 System setup

For our performance analysis, we have used our OMNeT++-based RSerPool simulation model RSPSIM (see Dreibholz and Rathgeb, 2005b) as well as our implementation RSPLIB (see Dreibholz and Rathgeb, 2007; Dreibholz, 2007; Zhou et al., 2010) [which is also the RSerPool reference implementation of the IETF, see Lei et al. (2008), Chapter 5] for measurements in a PlanetLab setup. Both – simulation model and implementation – contain the protocols ASAP and ENRP, a PR module, an attacker module and PE as well as PU modules for the request handling application defined in Section 6.

The PlanetLab (see Peterson and Roscoe, 2006) setup distributes the components to different machines in the USA. This country provides a sufficient number of PlanetLab nodes and a country-wide setup is also realistic for an RSerPool setup in a large company, in order to protect a critical service against e.g., earthquakes, power failures or terrorist attacks. By ping-based tests, we have observed inter-node network delays of about 20 ms to 30 ms.

The Linux-based PlanetLab nodes only support the protocols TCP and UDP, i.e., in particular the Linux Kernel SCTP module (LK-SCTP) is not provided. Unlike for our lab measurements in Zhou et al. (2009a), we therefore had to use our own userland SCTP implementation of Jungmaier (2005). The restriction of PlanetLab to TCP and UDP furthermore made it necessary to tunnel our SCTP traffic over UDP using the 'SCTP over UDP' encapsulation defined in Tüxen and Stewart (2007). Since our ASAP and ENRP messages are small compared to the usual MTU (i.e., 1,500 bytes) and bandwidth is not the limiting factor, the additional per-packet overhead of 8 bytes is negligible.

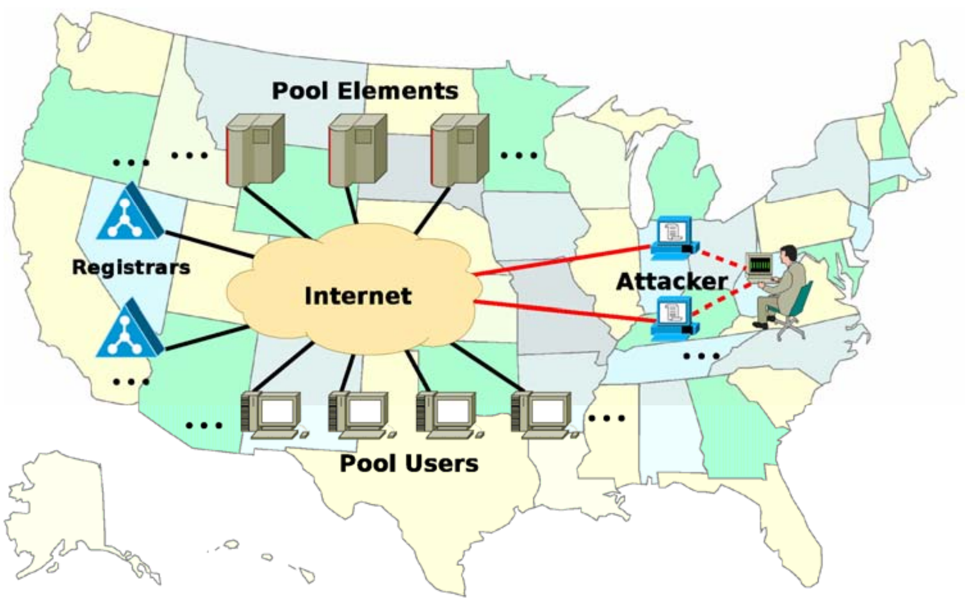
For our simulation and measurement setup, which is depicted in Figure 8, we use the following parameter settings unless otherwise specified:

- The target system utilisation is 50%. Request size and request interval are randomised using a negative exponential distribution [in order to provide a generic and application-independent analysis (see Dreibholz and Rathgeb, 2005b)]. There are 10 PEs; each one provides a capacity of 106 calculations/s.
- A PU:PE ratio of 3 is used [i.e., a non-critical setting as explained in Dreibholz (2007)].
- We use a request size:PE capacity setting of 10; i.e., being processed exclusively, the average processing takes 10s – see also Dreibholz and Rathgeb (2005b).
- There is a single PR only, since we do not examine PR failure scenarios here (see Zhou et al., 2009b for such scenarios). PEs re-register every 30s (registration lifetime) and on every load change of the adaptive LU and LUD policies.

- MaxBadPEReports is set to 3 (default in RFC Stewart et al., 2008). A PU sends an endpoint unreachable if a contacted PE fails to respond within 10s (see also Dreibholz and Rathgeb, 2009).
- The system is attacked by a single attacker node.
- For the simulation, the simulated real-time is 120 min; each simulation run is repeated at least 24 times with a different seed in order to achieve statistical accuracy. The inter-component network delay is 25 ms, which corresponds to the PlanetLab observations above.
- Each measurement run takes 15 min; each run is repeated at least 12 times.

For statistical post-processing of the results, the SimProcTC tool-chain described by Dreibholz et al. (2009b) is used. Each resulting plot shows the average values and their 95% confidence intervals.

Figure 8 The RSerPool system setup used for the analyses (see online version for colours)



8 Performance evaluation

The PR-based attack threats on RSerPool systems (see Section 4) can be solved by just applying authentication. We neglect the possibility of malicious PRs here, since mechanisms to judge the trustworthiness of PRs (see Subsection 5.1) would exceed the scope of this article. Registration and deregistration hijacking as well as (re-)registration and deregistration flooding can be solved easily with a modified authentication procedure (as described in Subsection 5.2), which is quite obvious.

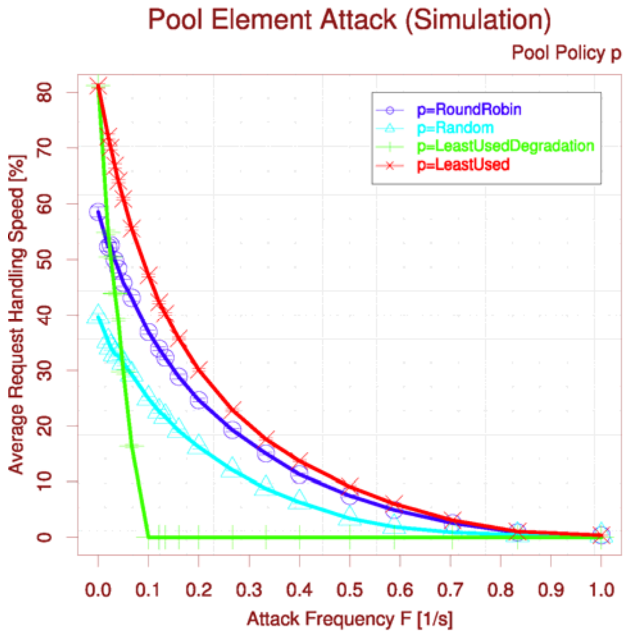
Therefore, the focus of our performance evaluation is to show the impact of the most severe attack threats – fake registration, handle resolution flooding and PE impeachment – as well as the performance of our countermeasures proposed in Section 5.

8.1 The impact of a fake registration attack without countermeasures

The intention of an attacker performing a fake registration (see Sub-subsection 4.2.2) attack is to send ASAP Registrations to a PR. Each registration request only has to contain another (e.g., randomly chosen) PE ID. The policy parameters can be set appropriately – e.g., a load of 0% (LU and LUD) and a load increment of 0% (LUD) – in order to ensure that the fake PE entry is chosen upon a handle resolution as frequently as possible.

Figure 9 presents the simulation results for the average request handling speed during a fake registration attack. Since the corresponding PlanetLab measurement results show a similar behaviour, a plot for the measurements has been omitted. The number of fake registrations per second – denoted as attack frequency F – is varied from 0s^{-1} (no attack) to 1s^{-1}. Already for $F = 0.1\text{s}^{-1}$ – which means just one fake registration every 10s – a significant degradation of the service performance can be observed. For using the LUD policy Zhou et al. (2008), this already leads to a complete DoS: an unloaded PE (i.e., its load is 0%) whose load never increases when accepting a new request (i.e., its load increment is 0%) appears to be a really good choice for a PU. As a result, the PUs will exclusively select the fake PE entries. Using $F = 0.8\text{s}^{-1}$, the attacker also achieves a complete DoS for the other policies. That is, already for sending less than one attack message per second – which is easily feasible over a modem connection – the attacker can entirely stem a service.

Figure 9 The impact of a fake registration attack without countermeasures (see online version for colours)



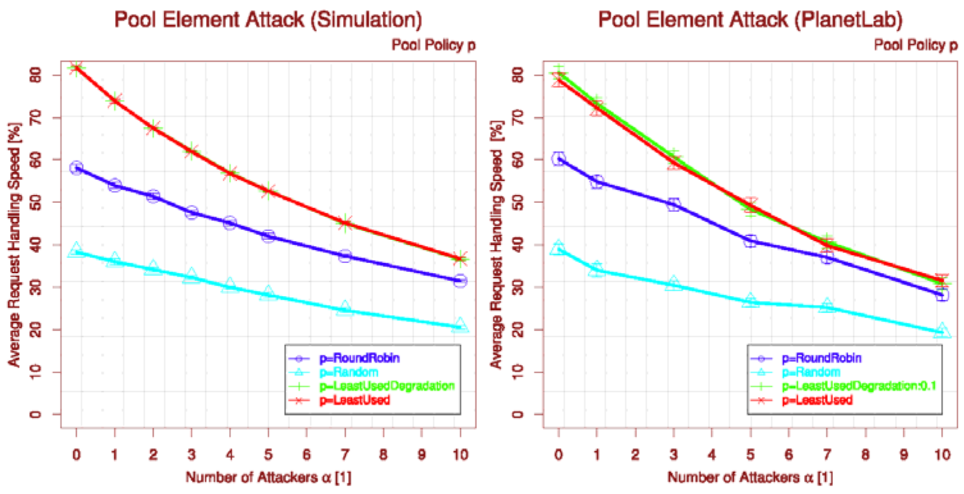
8.2 Applying countermeasures against malicious PEs

To countermeasure the fake registration attack, we now apply our approach presented in Subsection 5.2. Figure 10 presents the handling speed results for an attack frequency of $F = 10s^{-1}$ (i.e., a ten times higher attack intensity than in the DoS case of the unprotected scenario presented in Subsection 8.1) per attacker for varying the number of attackers α from 0 (i.e., no attack, for comparison) to 10. The left-hand plot presents the simulation results, the right-hand plot shows the PlanetLab measurement results. Note that $\alpha = 10$ attacker PEs means to have as many attackers as there are legitimate PEs in the pool. In particular, the attacker would have to steal 10 registration authorisation tickets in order to perform such an attack. That is, a significant effort by the attacker is necessary.

Obviously, our countermeasure approach is quite effective: even for $\alpha = 10$, the handling speed only halves at most – but the service which is provided by the 10 real PEs of the pool still remains operational and the attack impact is not even close to a DoS. The results obtained from the PlanetLab measurements correspond to the simulation results, i.e., our countermeasure approach also works effectively in a real-world internet setup as well.

It is important to note that the slightly different handling speed levels of the RSPLIB-based PlanetLab measurements in comparison to the RSPSIM simulation results are caused by the latencies of nodes (e.g., background PlanetLab slices, kernel, operating system, applications) and SCTP associations (e.g., packet scheduling, delay and jitter of the internet transport and packet retransmissions). These are – due to their complexity – not fully incorporated into the RSPSIM simulation model. Nevertheless, the tendency of the results is observed reasonably well.

Figure 10 Applying countermeasures against pool-element-based attacks (see online version for colours)



8.3 The impact of a malicious PU without countermeasures

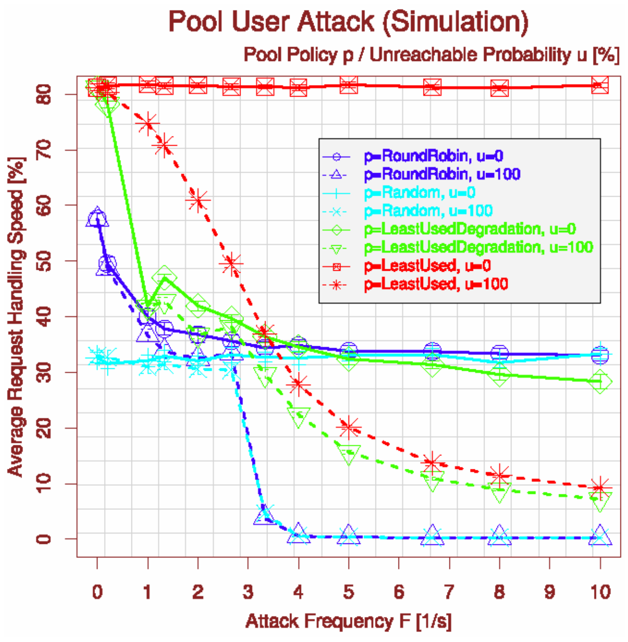
By performing a handle resolution flooding attack (see Sub-subsection 4.2.3), an attacker tries to flood the PR with handle resolution requests. Since the server selection procedure can be realised very efficiently – as shown by Dreibholz and Rathgeb (2008a) – a very high attack bandwidth would be necessary to overload the CPU of the PR. However, even by just requesting a few handle resolutions – without actually using the service of a selected PE – the performance of the service may be affected. As an example, the impact of a handle resolution attack on the request handling performance of the simulation for varying the attack frequency F (i.e., the delay between two consecutive handle resolution requests) is shown in Figure 11. Since the PlanetLab measurement results are quite similar, we omit a further plot here. For the PE entries chosen by a handle resolution, an ASAP endpoint unreachable report (see Section 3) is sent with probability u, i.e., a PE impeachment attack is performed. We present the two extreme cases: u = 0% (i.e., no unreachability reports – represented by solid lines) and u = 100% (i.e., worst case – represented by dotted lines).

Due to the stateful operation of the RR policy, the performance is even degraded for this policy by a setting of u = 0%: some PEs of the 'in turn' selection are skipped (since they are not actually used for processing a request), leading to the usage of less appropriate PEs for real requests. LUD is affected in a similar way by increased load values. Since LU and RAND are 'stateless', they are not affected by this kind of attack.

The impact of reporting all PEs as being unreachable by ASAP endpoint unreachables – i.e., u = 100% – is dramatic: PEs are kicked out of the handlespace, and the handling speed quickly sinks and leads – here at about $F = 10s^{-1}$, i.e., only 10 reports/s – to a complete DoS.

Again, a modem or ISDN connection provide sufficient bandwidth to perform this kind of attack.

Figure 11 The impact of a malicious endpoint unreachable report attack without countermeasures (see online version for colours)



8.4 Applying countermeasures against malicious PUs

To countermeasure the handle resolution and PE impeachment attacks, we apply our approach presented in Subsection 5.3 by setting MaxHRRate = 1s^{-1} (i.e., 60 times more than the actual handle resolution rate of the application) and MaxEURate = 1s^{-1} (i.e., by orders of magnitude higher than a realistic PE failure rate) for varying the attack frequency of one attacker. The resulting request handling speed performance is shown in Figure 12; the left-hand plot presents the simulation results and the right-hand plot displays the corresponding PlanetLab measurement results. The probabilities for sending ASAP endpoint unreachable reports (i.e., for actually performing PE impeachment attacks) are u = 100% (worst case) and u = 0% (for comparison).

For the stateful policies RR and LUD, the attacker is able to reduce the handling speed until triggering the countermeasure mechanism. After that, the attacker is ignored and the performance remains as for attacker-free scenarios. For the stateless RAND policy, the attack has no impact; for the LU policy, the attacker only has an impact when using unreachability reports. Interestingly, this effect is stronger for the PlanetLab measurements than for the simulation: this effect is caused by the latency of the PE load state updates in the internet (which may vary due to node latencies and congestion): since the load value of a PE only changes on re-registration, a least-loaded PE entry may be selected multiple times in sequence. That is, the attacker will send multiple unreachability reports for the same PE. As long as the rate threshold is not yet reached, the PE entry may be impeached. However, the attacker is ignored and the performance returns to the original level of an attacker-free system setup as soon as the countermeasure threshold is reached.

Figure 12 Applying countermeasures against handle resolution flooding and PE impeachment attacks (see online version for colours)

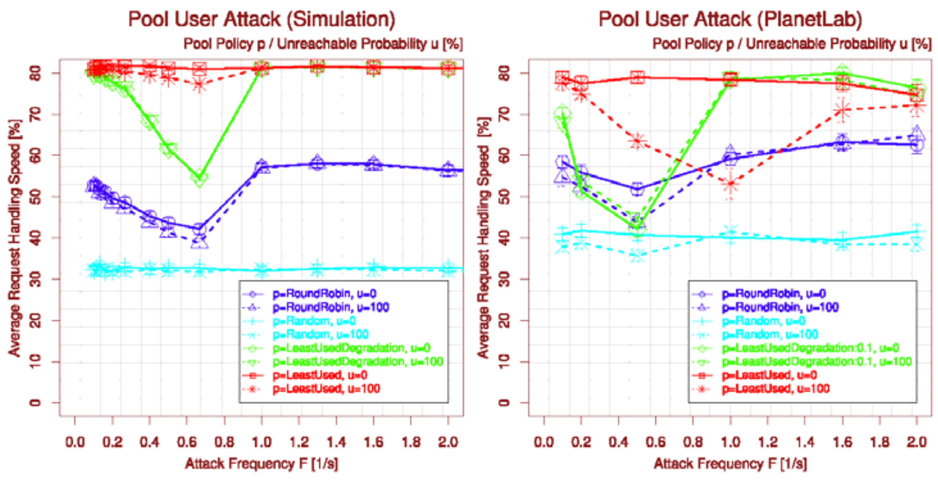
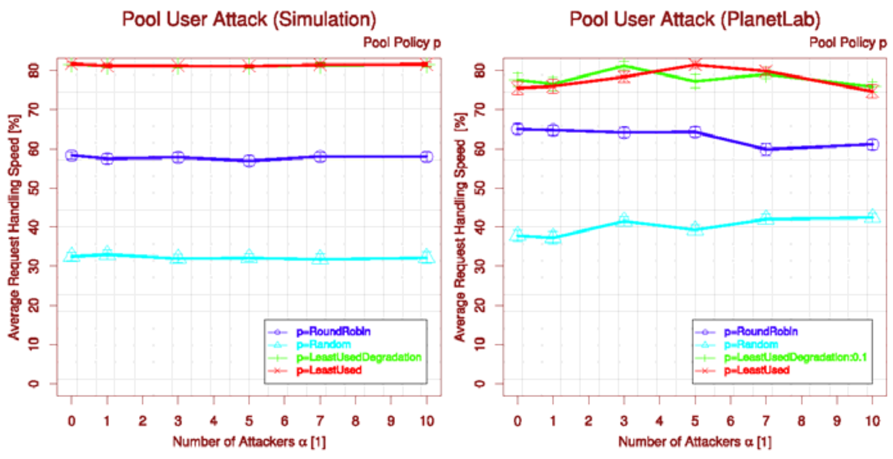


Figure 13 Varying the number of pool-user-based attackers with applying countermeasures (see online version for colours)



Just a single attacker using an attack frequency of $F = 10s^{-1}$ has been able to cause a complete DoS when not applying countermeasures (see Subsection 8.3). Therefore, Figure 13 presents the performance results for applying our countermeasures at $F = 10s^{-1}$ and $u = 100\%$ (i.e., the worst case of a PE impeachment attack) and a varying number of attackers α from 0 (i.e., no attack, for comparison) to 10. Again, the simulation results are presented in the left-hand plot while the PlanetLab measurement results can be found in the right-hand plot. Obviously, the presented results show that even $\alpha = 10$ attackers have no significant impact on the system performance – neither in the simulation nor in the real-world internet setup – any more. Furthermore, ten attackers means that the authorisation data of ten legitimate components has been stolen. Inside the restricted operation scope of an RSerPool setup (see Section 3), the effort of applying such an attack is assumed to be quite high and non-trivial.

9 Conclusions

In this article, we have given an overview of the DoS attack threats on RSerPool systems and have shown countermeasures to improve the robustness of the systems against them. Authentication of components alone cannot fully prevent such attacks. We have shown that fake registration, handle resolution flooding and PE impeachment attacks can easily cause a complete DoS. In order to cope with this threat, we have introduced countermeasure approaches which have shown to be effective – in simulations as well as in reality. The simulation results correspond to the measurements, i.e., we have also validated the correctness of our simulation model. Furthermore, our approaches are efficiently realisable – which is necessary to keep the RSerPool architecture lightweight.

The IETF standardisation process for RSerPool has just reached a major milestone by publication of its basic protocol documents as RFCs. Since the early beginnings of this process we have contributed our ideas, evaluations and improvements for the RSerPool framework. The goal of our ongoing work is to provide comprehensive security and configuration guidelines for application developers and users of the new RSerPool standard.

Acknowledgements

This article has been funded by the German Research Foundation (Deutsche Forschungsgemeinschaft) and the State Administration of Foreign Experts Affairs, China (Funding Number 20084600036).

References

Aberer, K. and Despotovic, Z. (2001) 'Managing trust in a peer-2-peer information system', in Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM), New York, USA., ACM, pp.310–317, ISBN 1-58113-436-3, available at http://doi.acm.org/10.1145/502585.502638.

Bellovin, S., Ioannidi, J., Keromytis, A. and Stewart, R. (2003) 'On the use of stream control transmission protocol (SCTP) with IPsec', *Standards Track RFC 3554*, IETF, July.

Conrad, P., Jungmaier, A., Ross, C., Sim, W-C. and Tüxen, M. (2002) 'Reliable IP telephony applications with SIP using RSerPool', in *Proceedings of the State Coverage Initiatives, Mobile/Wireless Computing and Communication Systems II*, Orlando, Florida, USA, July, Vol. X, ISBN 980-07-8150-1.

Crosby, S.A. and Wallach, D.S. (2003) 'Denial of service via algorithmic complexity attacks', in *Proceedings of the 12th USENIX Security Symposium*, Washington, DC, USA, August, pp.29–44.

Dierks, T. and Rescorla, E. (2008) 'The transport layer security (TLS) protocol version 1.2', *Standards Track RFC 5246*, IETF, August.

Dreibholz, T. (2002) 'An efficient approach for state sharing in server pools', in Proceedings *of the 27th IEEE Local Computer Networks Conference (LCN)*, Tampa, Florida, USA, October, pp.348–352, ISBN 0-7695-1591-6.

Dreibholz, T. (2007) 'Reliable server pooling – evaluation, optimization and extension of a novel IETF architecture', PhD thesis, University of Duisburg-Essen, Faculty of Economics, Institute for Computer Science and Business Information Systems, March.

Dreibholz, T. and Mulik, J. (2009) 'Reliable server pooling MIB module definition', RFC 5525, IETF, RSerPool Working Group, April.

Dreibholz, T. and Rathgeb, E.P. (2005a) 'Implementing the reliable server pooling framework', in *Proceedings of the 8th IEEE International Conference on Telecommunications (ConTEL)*, Zagreb, Croatia, June, Vol. 1, pp.21–28, ISBN 953-184-081-4.

Dreibholz, T. and Rathgeb, E.P. (2005b) 'On the performance of reliable server pooling systems', in *Proceedings of the IEEE Conference on Local Computer Networks (LCN) 30th Anniversary*, Sydney, Australia, November, pp.200–208, ISBN 0-7695-2421-4.

Dreibholz, T. and Rathgeb, E.P. (2007) 'On improving the performance of reliable server pooling systems for distance-sensitive distributed applications', in *Proceedings of the 15. ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, Bern, Switzerland, February, pp.39–50, ISBN 978-3-540-69962-0.

Dreibholz, T. and Rathgeb, E.P. (2008a) 'An evaluation of the pool maintenance overhead in reliable server pooling systems', *SERSC International Journal on Hybrid Information Technology (IJHIT)*, April, Vol. 1, No. 2, pp.17–32, ISSN 1738-9968.

Dreibholz, T. and Rathgeb, E.P. (2008b) 'a powerful tool-chain for setup, distributed processing, analysis and debugging of OMNeT++ simulations', in *Proceedings of the 1st ACM/ICST OMNeT++ Workshop*, Marseille, France, March, ISBN 978-963-9799-20-2.

Dreibholz, T. and Rathgeb, E.P. (2009) 'Overview and evaluation of the server redundancy and session failover mechanisms in the reliable server pooling framework', *International Journal on Advances in Internet Technology (IJAIT)*, June, Vol. 2, No. 1, pp.1–14, ISSN 1942-2652.

Dreibholz, T. and Tüxen, M. (2008) 'Reliable server pooling policies', *RFC 5356*, IETF, September.

Dreibholz, T., Becke, M., Pulinthanath, J. and Rathgeb, E.P. (2010a) 'Applying TCP-friendly congestion control to concurrent multipath transfer', in *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, Perth, Australia, April, pp.312–319.

Dreibholz, T., Becke, M., Pulinthanath, J. and Rathgeb, E.P. (2010b) 'Implementation and evaluation of concurrent multipath transfer for SCTP in the INET framework', in *Proceedings of the 3rd ACM/ICST OMNeT++ Workshop*, Málaga, Spain, March, ISBN 78-963-9799-87-5.

Dreibholz, T., Coene, L. and Conrad, P. (2009a) 'Reliable server pooling applicability for IP flow information exchange', *Internet-Draft Version 08*, IETF, Individual Submission, July.

Dreibholz, T., Jungmaier, A. and Tüxen, M. (2003) 'A new scheme for IP-based internet mobility', in *Proceedings of the 28th IEEE Local Computer Networks Conference (LCN)*, Königswinter, Germany, November, pp.99–108, ISBN 0-7695-2037-5.

Dreibholz, T., Rathgeb, E.P. and Zhou, X. (2008) 'On robustness and countermeasures of reliable server pooling systems against denial of service attacks', in *Proceedings of the IFIP Networking*, Singapore, May, pp.586–598, ISBN 978-3-540-79548-3.

Dreibholz, T., Zhou, X. and Rathgeb, E.P. (2009b) 'SimProcTC – the design and realization of a powerful tool-chain for OMNeT++ simulations', in *Proceedings of the 2nd ACM/ICST OMNeT++ Workshop*, Rome, Italy, March, ISBN 978-963-9799-45-5.

Dreibholz, T., Zhou, X., Rathgeb, E.P. and Du, W. (2009c) 'A PlanetLab-based performance analysis of RSerPool security mechanisms', in *Proceedings of the 10th IEEE International Conference on Telecommunications (ConTEL)*, Zagreb, Croatia, June, ISBN 978-953-184-131-3.

Eastlake, D. and Jones, P. (2001) *US Secure Hash Algorithm 1 (SHA1). Informational RFC 3174*, IETF, September.

Foster, I. (2002) 'What is the grid? A three point checklist', *GRID Today*, July.

Hohendorf, C., Rathgeb, E.P., Unurkhaan, E. and Tüxen, M. (2007) 'Secure end-to-end transport over SCTP', *Journal of Computers*, June, Vol. 2, No. 4, pp.31–40, ISSN 1796-203X.

ITU-T. (1993) 'Introduction to CCITT signalling system No. 7', Technical Report Recommendation Q.700, International Telecommunication Union, March.

Iyengar, J.R., Amer, PD. and Stewart, R. (2006) 'Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths', *IEEE/ACM Transactions on Networking*, October, Vol. 14, No. 5, pp.951–964, ISSN 1063-6692, available at http://dx.doi.org/10.1109/TNET.2006.882843.

Jungmaier, A. (2005) 'Das transportprotokoll SCTP', PhD thesis, Universität Duisburg-Essen, Institut für Experimentelle Mathematik, August.

Jungmaier, A., Rathgeb, E.P., Schopp, M. and Tüxen, M. (2001) 'A multi-link end-to-end protocol for IP-based networks', AEÜ – International Journal of Electronics and Communications, January, Vol. 55, No. 1, pp.46–54, ISSN 1434-8411.

Jungmaier, A., Rescorla, E. and Tüxen M. (2002) 'Transport layer security over stream control transmission protocol', *Standards Track RFC 3436*, IETF, December 2002.

Kent, S. and Seo, K. (2005) 'Security architecture for the internet protocol', *Standards Track RFC 4301*, IETF, December 2005.

Lei, P., Ong, L., Tüxen, M. and Dreibholz, T. (2008) 'An overview of reliable server pooling protocols', *Informational RFC 5351*, IETF, September.

Loughney, J., Stillman, M., Xie, Q., Stewart, R. and Silverton, A. (2005) 'Comparison of protocols for reliable server pooling', *Internet-Draft Version 10*, IETF, RSerPool Working Group, July.

Maharana, A. and Rathna, G.N. (2006) 'Fault-tolerant video on demand in RSerPool architecture', in *Proceedings of the International Conference on Advanced Computing and Communications (ADCOM)*, Bangalore, India, December, pp.534–539, ISBN 1-4244-0716-8.

Neuman, C., Yu, T., Hartman, S. and Raeburn, K. (2005) 'The Kerberos network authentication service (V5)', *Standards Track RFC 4120*, IETF, July.

Peterson, L. and Roscoe, T. (2006) 'The design principles of PlanetLab', *Operating Systems Review*, January, Vol. 40, No. 1, pp.11–16, ISSN 0163-5980.

Rivest, R. (1992) 'The MD5 message-digest algorithm', *Informational RFC 1321*, IETF, April.

Schöttle, P., Dreibholz, T. and Rathgeb, E.P. (2008) 'On the application of anomaly detection in reliable server pooling systems for improved robustness against denial of service attacks', in *Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN)*, Montréal, Québec, Canada, October, pp.207–214, ISBN 978-1-4244-2413-9.

Stewart, R. (2007) 'Stream control transmission protocol', *Standards Track RFC 4960*, IETF, September.

Stewart, R., Lei, P. and Tüxen, M. (2009) 'Stream control transmission protocol (SCTP) packet drop reporting', *Internet-Draft Version 09*, IETF, Individual Submission, December.

Stewart, R., Xie, Q., Stillman, M. and Tüxen, M. (2008) 'Aggregate server access protcol (ASAP)', *RFC 5352*, IETF, September.

Stewart, R., Xie, Q., Tüxen, M., Maruyama, S. and Kozuka, M. (2007) 'Stream control transmission protocol (SCTP) dynamic address reconfiguration', *Standards Track RFC 5061*, IETF, September.

Stillman, M., Gopal, R., Guttman, E., Holdrege, M. and Sengodan, S. (2008) 'Threats introduced by RSerPool and requirements for security', *RFC 5355*, IETF, September.

Tüxen, M. and Stewart, R. (2007) 'UDP encapsulation of SCTP packets', *Internet-Draft Version 02*, IETF, Individual Submission, November.

Tüxen, M., Stewart, R., Lei, P. and Rescorla, E. (2007) 'Authenticated chunks for the stream control transmission protocol (SCTP)', *Standards Track RFC 4895*, IETF, August.

Unurkhaan, E. (2005) 'Secure end-to-end transport – a new security extension for SCTP', PhD thesis, University of Duisburg-Essen, Institute for Experimental Mathematics, July.

Uyar, Ü., Zheng, J., Fecko, M.A., Samtani, S. and Conrad, P. (2004) 'Evaluation of architectures for reliable server pooling in wired and wireless environments', *IEEE JSAC Special Issue on Recent Advances in Service Overlay Networks*, Vol. 22, No. 1, pp.164–175.

Xie, Q., Stewart, R., Stillman, M., Tüxen, M. and Silverton, A. (2008) 'Endpoint handlespace redundancy protocol (ENRP)', *RFC 5353*, IETF, September.

Zhou, X., Dreibholz, T. and Rathgeb, E.P. (2008) 'A new server selection strategy for reliable server pooling in widely distributed environments', in *Proceedings of the 2nd IEEE International Conference on Digital Society (ICDS)*, Sainte Luce, Martinique, February, pp.171–177, ISBN 978-0-7695-3087-1.

Zhou, X., Dreibholz, T., Becke, M., Pulinthanath, J., Rathgeb, E.P. and Du, W. (2010) 'The software modeling and implementation of reliable server pooling and RSPLIB', in *Proceedings of the 8th ACIS Conference on Software Engineering Research, Management and Applications (SERA)*, Montréal, Québec, Canada, May.

Zhou, X., Dreibholz, T., Du, W. and Rathgeb, E.P. (2009a) 'Evaluation of attack countermeasures to improve the DoS robustness of RSerPool systems by simulations and measurements', in *Proceedings of the 16, ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS)*, Kassel, Germany, March, pp.217–228, ISBN 978-3-540-92665-8.

Zhou, X., Dreibholz, T., Fa, F., Du, W. and Rathgeb, E.P. (2009b) 'Evaluation and optimization of the registrar redundancy handling in reliable server pooling systems', in *Proceedings of the IEEE 23rd International Conference on Advanced Information Networking and Applications (AINA)*, Bradford, United Kingdom, May, pp.256–262, ISBN 978-0-7695-3638-5.