Reliable Server Pooling (RSerPool)

Konrad Windszus
Betreuer: Nils Kammenhuber
Seminar Future Internet SS2009
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik
Technische Universität München
Email: windszus@in.tum.de

***Zusammenfassung*— Zur Erfüllung von Anforderungen wie Redundanz und Verfügbarkeit wurde eine neue Protokollsuite mit Namen Reliable Server Pooling standardisiert, die die Verwaltung und Zuteilung von Servern aus einem Serverpool spezifiziert. Diese Ausarbeitung soll einen Überblick über die Protokollsuite und deren Anwendungsmöglichkeiten bieten.**

***Schlüsselworte*— RSerPool, Reliable Server Pooling, ASAP, ENRP, SCTP, Round-Robin, Least-Used**

I. Einleitung

Schon seit 2002 existiert der RFC3237 [1], der die Anforderungen an eine neue Protokollsuite mit dem Namen Reliable Server Pooling beschreibt. Unter Mitarbeit von Siemens, Nokia, Motorola und Cisco entstand dieses Anforderungsdokument. Orientiert hat man sich dabei am Signaling System No. 7 (SS7), einer Protokollsuite aus der Telefonwelt [2]. Eine ähnliche Protokollsuite sollte nun auch für IP-basierte Dienste entstehen und hohe Anforderungen in Bezug auf Verfügbarkeit und Redundanz, aber auch in Bezug auf Echtzeit (Reaktion bei Serverausfall), Skalierbarkeit, Erweiterbarkeit und Einfachheit erfüllen. Insbesondere die letzte Eigenschaft sollte es möglich machen, bestimmte Entitäten der Reliable-Server-Pooling-Architektur wie z.B. die Clients auch auf kleinen Geräten (wie z.B. mobilen oder integrierten Computern) zu betreiben.

Reliable Server Pooling (abgekürzt RSerPool) wurde von der RSerPool-Arbeitsgruppe innerhalb der IETF standardisiert und im Rahmen von mehreren RFCs [3]–[7] schließlich 2008 vollständig spezifiziert.

II. Überblick

RSerPool ist eine Protokollsuite mit zwei Applikationsprotokollen ASAP und ENRP, die den Aufbau eines Serverpools und die Kommunikation mit diesem Serverpool, insbesondere die Auswahl eines Servers, beschreiben. Ein Serverpool wird gebildet von mehreren Servern, die denselben Service anbieten. Die Einsatzmöglichkeiten sind vielfältig und beeinhalten zum Beispiel Load-Balancing von Webservern oder VoIP mit SIP-Proxys. Der Serverpool kann flexibel erweitert bzw. verkleinert werden, erkennt Ausfälle von Servern automatisch und kann auch selbstständig darauf reagieren. RSerPool bietet also eine Alternative zu klassischen Hard-/Software-Load-Balancern. Methoden zum Abgleich von Inhalten sind nicht in dieser Protokollsuite enthalten und werden weiterhin applikationsspezifisch geregelt, genau wie die Kommunikation zwischen Client und Anwendungsserver.

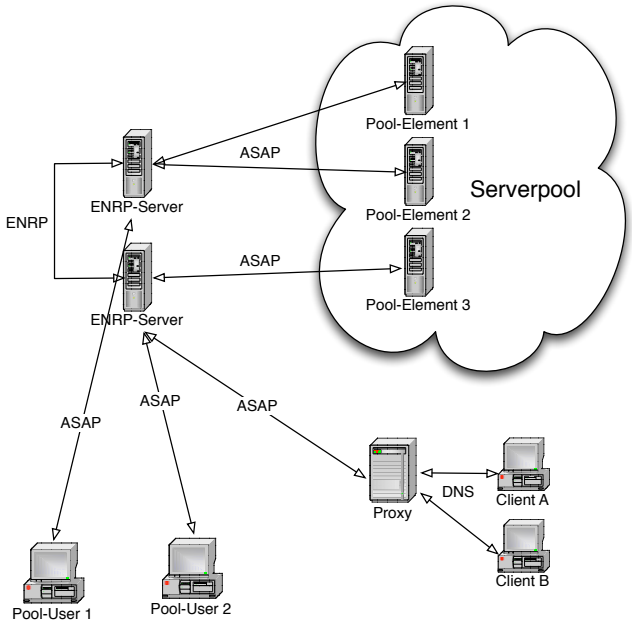

Abbildung 1. Aufbau von RSerPool

Der grundsätzliche Aufbau für RSerPool sieht folgendermaßen aus: Die einzelnen Server eines Serverpools werden als Pool-Elemente (PE) bezeichnet. Jedes PE registriert sich bei einem der ENRP-Server, der die Verwaltung des Serverpools und die Zuweisung von einzelnen PEs an die Clients vornimmt. Die Clients werden in diesem Szenario als Pool-User (PU) bezeichnet. Sowohl die Kommunikation von PE mit ENRP-Server als auch von PU mit ENRP-Server läuft über ASAP. Um Ausfallsicherheit zu gewährleisten, sind auch die ENRP-Server redundant ausgelegt. Sie gleichen sich jeweils mit ENRP-Nachrichten ab.

Nachdem die PEs in einem Serverpool registriert wurden und diesem Serverpool ein sogenanntes Pool-Handle zugewiesen wurde„ können die PUs beim ENRP-Server die Adresse eines PEs erfragen. RSerPool übernimmt dabei die Rolle des klassischen DNS und löst Anfragen auf ein Pool-Handle auf

und teilt eine bzw. mehrere IP-Adressen von dazugehörigen PEs mit.

Da RSerPool auch immer einen Teil der Logik auf den Client auslagert, sind auch Proxys denkbar, die z.B. das klassische Protokoll DNS anbieten, damit auch Clients RSerPool nutzen können, die ASAP noch nicht unterstützen.

III. Nachrichtenformat

A. Überblick

ASAP- und ENRP-Nachrichten sind sehr ähnlich aufgebaut. Alle Felder werden in Network-Byte-Order (d.h. Big-Endian) übertragen. Eine Nachricht folgt folgendem Schema, das an die SCTP-Paketstruktur angelehnt ist [5].

Tabelle I
Nachrichtenaufbau ASAP und ENRP

Feld	Länge
Typ	8 Bit
Flag	8 Bit
Länge	16 Bit
Parameter	variabel

Der Nachrichtentyp wird von einer 8-Bit-Konstante vorgegeben. Das Längenfeld umfasst die komplette Nachricht (inkl. Längenfeld selber). Die mitgegebenen Parameter sind in der Anzahl und im konkreten Typ abhängig vom Nachrichtentyp. Alle Parametertypen haben aber denselben Aufbau.

B. Parameterformat

Tabelle II
Parameteraufbau ASAP und ENRP

Feld	Länge
Typ	16 Bit
Länge	16 Bit
Wert	variabel
Füllbytes	variabel

Alle Parameter bestehen aus einem 16-Bit-Parametertyp, der die Art des übertragenen Parameters angibt. Danach folgt die Länge als 16 Bit unsigned integer. Grundsätzlich umfasst die Länge auch die Größe des Parametertyps und des Längenfeldes, berücksichtigt allerdings nicht die evtl. vorhandenen Füllbytes. Alle Parameter müssen in der Länge ein Vielfaches von 4 Byte sein. Ist das nicht der Fall, wird der Parameter mit (maximal 3) Füllbytes aufgefüllt.

IV. ASAP

A. Überblick

Das Aggregate Server Access Protocol (ASAP) ist für die Kommunikation zwischen PU und ENRP-Server zuständig, sowie für die Kommunikation zwischen PE und ENRP-Server [4]. Für Letzteres muss als Transportprotokoll SCTP genutzt werden, für Ersteres kann optional auch TCP genutzt werden.

Zunächst muss ein ENRP-Server mittels ASAP_SERVER_ANNOUNCE seine ID und seine Transport-Parameter bekanntmachen. Dies geschieht üblicherweise über einen Multicast auf einem festgelegten Kanal.

Zum Registrieren eines PE wird eine ASAP_REGISTRATION-Nachricht verwendet. Dazu wird vom PE einer der bekannten ENRP-Server ausgewählt und diesem als Parameter ein eindeutiges Pool-Handle mitgeteilt. Außerdem werden als Parameter mehrere Informationen über das PE mitgegeben. Der wichtigste ist ein 32-Bit-PE-Identifier, der bei allen weiteren Nachrichten jeweils das PE identifiziert. Außerdem kann angegeben werden, wie lange die Registrierung gültig ist und welcher Algorithmus zur Auswahl eines PEs verwendet werden soll.

Ein PE kann sich explizit vom Pool abmelden. Dazu nutzt er die ASAP_DEREGISTRATION-Nachricht.

Die PUs nutzen die ASAP_HANDLE_RESOLUTION-Nachricht um zu erfahren, welche PEs zu einem Server-Pool mit einem bestimmten Handle gehören. Als Antwort erhalten sie eine Liste mit PE-Parametern, die unter anderem auch die IP-Adresse des PEs umfasst.

B. Fehlerbehandlung

Um einen Ausfall von Pool-Elementen zu erkennen, gibt es die Möglichkeit, dass entweder ein Client einen ENRP-Server explizit mit einer ASAP_ENDPOINT_UNREACHABLE-Nachricht über einen Ausfall informiert, oder dass der ENRP-Server im Zuge seiner regelmäßigen ASAP_ENDPOINT_KEEP_ALIVE-Nachrichten keine Antwort mehr vom PE erhält.

C. Session-Management

ASAP unterstützt erweitertes Session-Management, das optional auch für die Kommunikation zwischen PU und PE genutzt werden kann. Das Session-Management wird hierbei von einer erweiterten API angeboten, im Folgendem RSerPool-Service genannt. Der RSerPool-Service ist in der Lage beim Ausfall eines Servers die Verbindung auf einen anderen Server umzuleiten. Dieses Fail-Over geschieht für den Programmierer und auch Anwender transparent. Die Verbindung zwischen PU und PE muss in dem Fall über den RSerPool-Service aufgebaut werden, und umfasst neben der Datenverbindung mit dem applikationsspezifischen Protokoll für die Anwendungsdaten auch eine separate ASAP-Kontrollverbindung. Über diese zweite Verbindung wird dem PU vom PE beim Beginn einer neuen Session ein Cookie mitgeteilt, mit dessen Hilfe die Session auf anderen PEs wiederhergestellt werden kann. Dies geschieht über die ASAP_COOKIE-Nachricht. Damit das funktioniert, müssen entweder im Cookie selbst alle Sessioninformationen oder ein eindeutiger Identifier enthalten sein, mit dem jedes PE auf die Session zugreifen kann. Die Sessioninformationen selbst müssen in diesem Fall über andere Wege zwischen den PEs eines Pools ausgetauscht werden.

Außerdem werden über diesen Kanal auch ASAP_BUSINESS_CARD-Nachrichten ausgetauscht. Diese dienen dazu, das Pool-Handle zur aktuellen Verbindung zu ermitteln. Nur wenn dieses Pool-Handle bekannt ist, kann beim Ausfall auf ein anderes PE desselben Serverpools gewechselt werden. Damit ein Ausfall von der Anwendung möglichst nicht bemerkt wird, wird auch die Datenverbindung

vom RSerPool-Service gemanagt, damit auch diese beim Serverausfall transparent auf ein anderes PE umgeschaltet werden kann.

V. ENRP

A. Überblick

Das Endpoint HaNdlespace Redundancy Protocol (ENRP) wird von den ENRP-Servern genutzt, um gegenseitig die Daten abzugleichen und so die notwendige Redundanz zu schaffen. Dazu werden entweder andere Peer-ENRP-Server direkt adressiert oder Nachrichten werden über einen vorher festgelegten Multicast-Kanal ausgetauscht. Als Transportprotokoll bei ENRP ist zwingend SCTP vorgeschrieben [8]. Jeder ENRP-Server generiert zu Beginn eine 32-Bit-Server-ID, die ihn eindeutig identifiziert. Normalerweise wird dafür einfach eine Zufallszahl erzeugt. Diese ID ist unveränderlich solange der Server in Betrieb ist.

Danach versucht der ENRP-Server zunächst seinen Mentor-Server zu kontaktieren. Dessen Adresse ist fest vorgegeben. Wenn mehrere Adressen bekannt sind, wird ein beliebiger Server zum Mentor-Server, der über eine der Adressen erreichbar ist. Sobald der ENRP-Server seinen Mentor-Server gefunden hat, kann er von diesem im Rahmen der Initialisierung mittels einer ENRP_LIST_REQUEST eine komplette Liste aller anderen ENRP-Server anfordern.

Um nun außerdem die Informationen über die bekannten Serverpools zu bekommen, werden ENRP_HANDLE_TABLE_REQUEST-Nachrichten genutzt. Als Antwort auf diese Nachrichten werden alle Pool-Element-Parameter nach Pool-Handles geordnet zurückgegeben.

Jedes Pool-Element kommuniziert immer nur mit einem einzigen ENRP-Server. Dieser Server wird Home-ENRP-Server in Bezug auf dieses Pool-Element genannt. Updates einzelner Pool-Elemente in einem Server-Pool werden vom jeweiligen Home-ENRP-Server an alle anderen ENRP-Server mittels einer ENRP_HANDLE_UPDATE-Nachricht verbreitet.

B. Fehlerbehandlung

Jeder Server verwaltet intern eine Liste aller bekannten Peer-Server. Sobald eine ENRP-Nachricht von einem bis dato unbekannten Peer-Server kommt, wird dieser mittels einer ENRP_PRESENCE-Nachricht aufgefordert, sich bei dem anfragenden Server zu identifizieren. Außerdem senden alle Server in regelmäßigem Abstand ebenfalls eine ENRP_PRESENCE-Nachricht an alle ihre Peers. Wenn ein Server diese Nachricht auch nach einer expliziten Aufforderung nicht mehr sendet, werden die Aufgaben des Servers von demjenigen Peer übernommen, der zuerst bemerkt hat, dass der Server nicht mehr antwortet. Damit nicht mehrere Server gleichzeitig versuchen, die Aufgaben vom toten Peer zu übernehmen, wird zunächst eine ENRP_INIT_TAKEOVER-Nachricht verschickt, die von allen anderen ENRP-Servern bestätigt werden muss. Erst dann beginnt die Übernahme des nicht mehr antwortenden Servers. Dazu wird jedes Pool-Element, benachrichtigt, das den übernommenen Server als Home-ENRP-Server genutzt hat. Dies geschieht über ASAP_ENDPOINT_KEEP_ALIVE-Nachrichten. Alle anderen ENRP-Server streichen den übernommenen ENRP-Server von ihrer internen Liste.

VI. Regeln zur Serverauswahl

Um die Last zwischen den Pool-Elementen möglichst optimal zu verteilen, existieren unterschiedliche Vorgaben zur Serverauswahl, die im RFC5356 spezifiziert sind [7]. Allen Vorgaben ist gemein, dass die Serverauswahl aufgesplittet wird zwischen PU und ENRP-Server. Der ENRP-Server gibt zunächst eine Liste von in Frage kommenden Servern zurück, die er mithilfe einer Regel ausgewählt hat. Der PU wählt dann aus dieser Liste wiederum einen einzigen Server aus. Diese doppelte Auswahl hat den Vorteil, das bei Ausfall eines Servers nicht erneut der ENRP-Server kontaktiert werden muss. In einem solchen Fall kann vom Client aus seiner Serverliste einfach ein anderer Server ausgewählt werden [9].

Grundsätzlich wird zwischen nicht-adaptiven und adaptiven Techniken unterschieden.

A. Nicht-adaptive Verfahren

Zu den nicht-adaptiven Verfahren gehört Round-Robin, das die einzelnen PEs reihum an die PUs verteilt. Haben alle PEs bereits einen PU erhalten, geht es wieder beim ersten Server los. Eine Verbesserung dieses Verfahrens stellt die Gewichtung der einzelnen PEs aufgrund ihrer unterschiedlichen Kapazität dar [10]. Diese Gewichtung ist statisch, und muss nur einmal bei der Registrierung den ENRP-Servern mitgeteilt werden. Bei beiden Verfahren wird vom ENRP-Server jeweils eine vollständige Liste von PEs an den PU übermittelt. Dieser ermittelt dann mithilfe von Round-Robin einen Eintrag. Damit nicht jeder PU denselben Server auswählt, ist bei jeder Anfrage jeweils das nächste PE am Anfang der Liste. Diese Verfahren erfordern deshalb einen Status, in dem gespeichert wird, welches PE zuletzt ausgewählt wurde bzw. am Anfang der Liste stand. Standen alle PEs einmal am Anfang der Liste, wird wieder mit dem ersten PE begonnen. Die Liste enthält PEs mit höherem Gewicht entsprechend mehrmals, so dass diese häufiger ausgewählt werden, als diejenigen mit niedrigerem Gewicht. Die mehrfachen Servereinträge werden möglichst mit äquidistantem Abstand zueinander in der Liste eingefügt. Die relative Häufigkeit der Einträge pro PE entspricht dabei deren relativem Gewicht.

Ein zustandsloses Verfahren stellt das Zufallsverfahren (RAND) dar, das ebenso in einer gewichteten Ausprägung spezifiziert ist. Es funktioniert ähnlich wie Round-Robin, allerdings erfolgt jede Auswahl unabhängig von der vorherigen. Deshalb muss weder der ENRP-Server noch der PU einen Status vorhalten. Der ENRP-Server gibt eine Liste von PEs zurück, in denen kein PE doppelt vorkommt. Die Auswahl dieser Liste erfolgt bei jeder Anfrage neu und die Wahrscheinlichkeit, dass ein PE in der Liste landet, muss gleich dessen relativem Gewicht sein. Aus dieser Liste wiederum wählt der PU zufällig ein Element aus.

Als letztes nicht-adaptives Verfahren ist ein Prioritätenverfahren spezifiziert, bei dem vom ENRP-Server immer eine

Liste von PEs Reihenfolge ihrer Priorität zurückgegeben wird. Die Priorität der PEs wird dabei bei deren Registrierung einmal festgelegt. Die PUs wählen immer den ersten Eintrag, d.h. den Eintrag mit der höchsten Priorität aus. Dieses Verfahren kann genutzt werden, damit standardmäßig immer ein PE angesprochen wird. Erst im Fehlerfall wird das PE mit der nächstniedrigeren Priorität genutzt.

B. Adaptive Verfahren

Die zweite Gruppe der Verfahren passt die Serverauswahl an bestimmte, sich ändernde Rahmenbedingungen an, oft an die Auslastung der einzelnen PEs. Diese melden in regelmäßigen Abständen ihre aktuelle Auslastung über Registrierungsnachrichten an die ENRP-Server, die diese Informationen bei der Auswahl berücksichtigen.

Die Least-Used-Policy ist ein solches Verfahren, bei der die ENRP-Server eine Liste von Servern mit geringer Last an den PU melden. Dieser wählt dann aus der Liste denjenigen Server mit der geringsten Last aus. Haben mehrere PEs die gleiche Auslastung gemeldet, wird ein PE nach Round-Robin ausgewählt.

Die Aktualisierung der Auslastung erfolgt allerdings sehr selten, so dass die Lastinformationen meistens nicht aktuell sind. Darum wurde ein Verfahren entwickelt, bei dem die Auswahl eines PEs automatisch die Auslastung des ausgewählten PEs um einen konstanten Wert hebt. Diese Least-Used-With-Degradation-Policy (LUD) funktioniert nur auf ENRP-Server-Seite. Die PUs wählen einfach den ersten Eintrag aus der Liste aus, ohne dass das Einfluss auf die Auslastungsinformationen hätte. Wenn beim ENRP-Server wieder eine Aktualisierung der Auslastung in Form einer Registrierungsnachricht eintrifft, wird der lokale Cache mit Auslastungsinformationen wieder überschrieben.

Bei diesem Verfahren wird allerdings nicht berücksichtig, wie stark die Last steigt, wenn ein Server ausgewählt werden würde. Deshalb wurde die Priority-Least-Used-Policy aufgenommen, bei der die Server sortiert werden nach der Reihenfolge ihrer Auslastung, die sie im Falle einer Anfrage hätten. Durch das Verfahren kann also die Last noch besser verteilt werden.

Daneben existiert noch das Randomized-Least-Used-Verfahren (RLU) das dem Zufallsverfahren gleicht, allerdings die Last mit in die Wahrscheinlichkeit der Auswahl einfließen lässt.

Eine Bewertung der Verfahren von Thomas Dreibholz hat ergeben, dass die adaptiven Verfahren wie erwartet, besser mit sich stark ändernden Bedingungen umgehen können [11]. Bei hoher Netzwerk-Latenz verschwinden diese Vorteile jedoch und die Ergebnisse nähern sich denen der nicht-adaptiven Verfahren an, da auch die adaptiven Verfahren nicht schnell genug auf Änderungen reagieren können . Ebenso wurde dort gezeigt, dass lokales Caching von PEs bei den PUs selten sinnvoll ist, da dann die ENRP-Server die Last nicht mehr gleichmäßig verteilen können.

VII. Sicherheitsaspekte

ASAP und ENRP bieten für sich genommen keine Sicherheit in der Kommunikation. Darum wurden im RFC 5355 [6] Möglichkeiten beschrieben, wie bestimmte Angriffsszenarien zu unterbinden sind. Verpflichtend ist die Nutzung von Transport Layer Security (TLS) mit einer AES128-Verschlüsselung für alle Verbindungen. Sowohl PEs als auch ENRP-Server müssen sich gegenseitig mithilfe eines gemeinsamen geheimen Schlüssels (PSK) authentifizieren. PUs müssen ENRP-Server mithilfe von Zertifikaten authentifizieren. TLS muss auf Basis von SCTP unterstützt werden, wie es im RFC3436 spezifiziert ist [12].

Mit dieser Maßnahme kann verhindert werden, dass beliebige Rechner Registrierungs- und Deregistrierungsnachrichten verschicken. Auch ein Statusupdate erfolgt erst dann, wenn das PE authentifiziert wurde. Damit werden Angriffe von nicht-authentifizierten Rechnern wirkungslos. Auch ein neuer ENRP-Server kann nicht einfach eingeschleust werden, da sich auch die Peer-ENRPs zunächst gegenseitig authentifizieren. Man-in-the-Middle-Attacken zwischen ENRP-Server und Pool-User werden durch Zertifikate in Kombination mit Verschlüsselung verhindert.

PUs selber nutzen keine Zertifikate und dementsprechend muss bei der Kommunikation vom ENRP-Servern darauf geachtet werden, dass Meldungen von PUs nicht unbedingt vertrauenswürdig sind. Auch ein Flooding von ASAP_ENDPOINT_UNREACHABLE-Nachrichten wird verhindert, indem die ENRP-Server nur eine bestimmte Anzahl von diesen Nachrichten innerhalb einer Zeitspanne von einem PU zulassen. Jede dieser Nachrichten wird außerdem noch vom ENRP-Server verifiziert.

Im Gegensatz zum DNS wurde also bei RSerPool von Anfang an darauf geachtet, dass die Protokolle besonders sicher sind. Durch die Nutzung von TLS ist das besonders einfach, da die darüberliegende Schicht von ASAP/ENRP davon so gut wie nichts mitbekommt.

VIII. SCTP

SCTP ist ein Transportprotokoll auf Schicht 4 des OSI-Modells. Es wurde im RFC4960 [13] spezifiziert. Das Protokoll wurde entwickelt, um einen besonders zuverlässigen Transport von Telefonsignalen über IP zu ermöglichen. Deshalb wurde besonders viel Wert auf Verfügbarkeit gelegt. Aus diesem Grund ist das Protokoll für RSerPool als Transportprotokoll vorgeschrieben.

SCTP ähnelt stark TCP, verhindert allerdings bestimmte Angriffsszenarien wie z.B. SYN-Flooding durch einen 4-Wege-Handshake. Der Header von SCTP besteht aus Quell- und Zielport sowie einer CRC32-Checksumme. Außerdem enthält er ein Verifikationstag, mit dem der Absender sich authentifiziert. Dieses Tag wird initial beim Handshake ausgetauscht und wird danach in allen Paketen die zu der Verbindung gehören mitübertragen.

Die Inhalte werden in Datenblöcken (sogenannten Chunks) übertragen, die wiederum aus einem Header bestehen, der den Typ, einige Flags und die Länge des Chunks spezifiert. Hinter

Tabelle III
SCTP-PACKET-FORMAT

Feld	Länge
Quellport	16bit
Zielport	16bit
Verifikationstag	32bit
Checksumme	32bit
Chunktyp	8bit
Chunkflags	8bit
Chunklänge	16bit
Chunkinhalt	variabel
Chunk 2 - n	variabel

dem Header stehen die eigentlichen Daten. Chunks werden dabei nicht nur für die Datenübertragung genutzt, sondern auch für den Verbindungsauf- und abbau. Dafür sind bestimmte Chunk-Typen reserviert.

Durch die Unterteilung in Chunks ist es möglich, dass mehrere Applikationen sich eine Verbindung teilen (Multi-Streaming). Optional ist es möglich die Chunks zu nummerieren, so dass sie in der gleichen Reihenfolge beim Empfänger ankommen, wie sie losgeschickt wurden. Für andere Anwendungen (beispielsweise Echtzeit-Anwendungen) ist es allerdings wichtiger, dass die Pakete möglichst schnell ankommen. Darum kann SCTP auch als Alternative zu UDP genutzt werden.

Der Verbindungsaufbau ist im Gegensatz zu TCP vierstufig, so dass ein SYN-Flooding-Angriff unmöglich ist. Während des Handshakes wird kein Status auf dem Server gehalten, sondern nur in einem Cookie zwischen Client und Server ausgetauscht. Der Client initiiert die Verbindung mit einem INIT, auf dass der Server mit einem INIT-ACK antwortet. In dieser Nachricht wird bereits ein Cookie übermittelt, das der Client mithilfe der COOKIE-ECHO-Nachricht wiederum an den Server verschicken muss. Erst dann werden die Resourcen für die Verbindung vom Server reserviert und dieser antwortet mit einem COOKIE-ACK. Damit der Verbindungsaufbau den Datenfluss nicht zu sehr verzögert, ist es bereits in den letzten beiden Nachrichten möglich Daten mitzuschicken.

Zur Flusskontrolle nutzt SCTP wie TCP eine adaptive Fenstergröße und zur Fehlerkontrolle selektive Bestätigungen (Sel ACK). Fast Retransmission ist im Gegensatz zu TCP nicht mehr optional, sondern vorgeschrieben.

SCTP unterstützt das Multihoming, bei dem Endpunkte über mehrere IP-Adressen erreichbar sind (beispielsweise ein Server mit mehreren Ethernet-Schnittstellen, die unterschiedlich geroutet werden). Davon wird üblicherweise zwar nur eine genutzt, um die Auslastung der Netze gering zu halten, beim Ausfall dieser Verbindung kann aber auf die andere Schnittstelle gewechselt werden, ohne dass die Verbindung von der Anwendung neu aufgebaut werden muss. Im Gegensatz zu TCP sind bei SCTP auch One-To-Many-Verbindungen möglich, die im Zusammenhang mit ENRP verwendet werden, um gleichzeitig alle anderen ENRP-Server zu adressieren.

Fast alle Unix-Betriebssysteme unterstützen SCTP, allerdings ist das Protokoll weder in Windows Vista noch in Max OS X enthalten. Für dieses Betriebssystem gibt es aber Implementierungen von Fremdanbietern [14].

IX. ZUSAMMENFASSUNG UND AUSBLICK

RSerPool ist noch eine sehr neue Protokollsuite, die bis jetzt noch keine weite Verbreitung außerhalb des akademischen Umfelds gefunden hat. Insbesondere der obligatorische Einsatz von SCTP als Transportprotokoll stand einem Einsatz bis jetzt im Wege. RSerPool stellt aber häufig einen kostengünstigeren und flexibleren Ansatz als klassische Load-Balancer dar und wird deshalb gerade im Umfeld von Webservices seine Anwendung finden.

Kritisch zu hinterfragen ist jedoch die Zeitverzögerung der Adressauflösung durch RSerPool gegenüber dem DNS bzw. direkter IP-Adressierung. Zwar ist das System von der Anzahl der Nachrichten mit dem DNS zu vergleichen, allerdings ist es im Gegensatz zum klassischen DNS aufgrund der Lastverteilung häufig nicht sinnvoll, ASAP-Adressauflösungen zu cachen. Auch das 4-Way-Handshake der SCTP-Verbindung verzögert den Verbindungsaufbau, verglichen mit den DNS-Anfragen über UDP.

Mit rsplib steht schon eine leistungsfähige Implementierung unter der GPL bereit und auch Motorola hat bereits eine Closed-Source Implementierung entwickelt. Für das normale Web-Umfeld wird DNS in Kombination mit klassischen Load-Balancern wohl das Mittel der Wahl bleiben, da nicht mit einer baldigen Implementierung von ASAP in den Webbrowsern zu rechnen ist.

LITERATUR

[1] M. Tuexen, Q. Xie, R. Stewart, M. Shore, L. Ong, J. Loughney, and M. Stillman, "Requirements for Reliable Server Pooling," RFC 3237 (Informational), Jan. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3237.txt

[2] M. T. T. Dreibholz, "High Availability using Reliable Server Pooling," in *Linux Conference Australia (LCA'2003)*, Januar 2003. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/rserpool/rserpool-publications/RSerPool-Paper.pdf

[3] P. Lei, L. Ong, M. Tuexen, and T. Dreibholz, "An Overview of Reliable Server Pooling Protocols," RFC 5351 (Informational), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5351.txt

[4] R. Stewart, Q. Xie, M. Stillman, and M. Tuexen, "Aggregate Server Access Protocol (ASAP)," RFC 5352 (Experimental), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5352.txt

[5] ——, "Aggregate Server Access Protocol (ASAP) and Endpoint Handlespace Redundancy Protocol (ENRP) Parameters," RFC 5354 (Experimental), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5354.txt

[6] M. Stillman, R. Gopal, E. Guttman, S. Sengodan, and M. Holdrege, "Threats Introduced by Reliable Server Pooling (RSerPool) and Requirements for Security in Response to Threats," RFC 5355 (Informational), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5355.txt

[7] T. Dreibholz and M. Tuexen, "Reliable Server Pooling Policies," RFC 5356 (Experimental), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5356.txt

[8] Q. Xie, R. Stewart, M. Stillman, M. Tuexen, and A. Silverton, "Endpoint Handlespace Redundancy Protocol (ENRP)," RFC 5353 (Experimental), Sep. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5353.txt

[9] T. Dreibholz, "Das rsplib-Projekt - Hochverfügbarkeit mit Reliable Server Pooling," in *LinuxTag 2005, Karlsruhe*, Jun. 2005. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/rserpool/rserpool-publications/LinuxTag2005.pdf

[10] M. T. T. Dreibholz, E. P. Rathgeb, “Load Distribution Performance of the Reliable Server Pooling Framework,” in *IEEE International Conference on Networking (ICN 2005)*, April 2005. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/rserpool/rserpool-publications/ICN2005.pdf

[11] E. P. R. Thomas Dreibholz, “On the performance of reliable server pooling systems,” in *30th IEEE Local Computer Networks Conference*, November 2005. [Online]. Available: http://tdrwww.iem.uni-due.de/dreibholz/rserpool/rserpool-publications/LCN2005.pdf

[12] A. Jungmaier, E. Rescorla, and M. Tuexen, “Transport Layer Security over Stream Control Transmission Protocol,” RFC 3436 (Proposed Standard), Dec. 2002. [Online]. Available: http://www.ietf.org/rfc/rfc3436.txt

[13] R. Stewart, “Stream Control Transmission Protocol,” RFC 4960 (Proposed Standard), Sep. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4960.txt

[14] R. Stewart, “SCTP Implementations.” [Online]. Available: http://www.sctp.org/implementations.html