Stream Control Transmission Protocol (SCTP)

Violin Yanev
Betreuer: Nils Kammenhuber
Seminar Future Internet SS2010
Lehrstuhl Netzarchitekturen und Netzdienste
Fakultät für Informatik, Technische Universität München
Email: yanev@in.tum.de

KURZFASSUNG

Das Transportprotokoll SCTP wurde entwickelt, um Funktionalität anzubieten, welche die weitverbreiteten TCP und UDP nicht bieten. Dieses Protokoll weist starke Ähnlichkeiten zu TCP auf, bietet aber weitere Möglichkeiten, wie die Aufteilung des Bytestroms in einzelne Nachrichten, die Aufspaltung des Datentransfers in sogenannte Streams sowie die verbesserte Ausfallsicherheit und Angriffschutz. Dieses Paper versucht, die zusätzlichen Features, die SCTP implementiert, in einem objektiven Vergleich mit anderen Transportprotokollen zu untersuchen.

Schlüsselworte

SCTP, Transportprotokoll, Multihoming, Multistreaming, Network Resilience, TCP

1. MOTIVATION

TCP ist das zurzeit am meisten verbreitete Transport-Protokoll. Obwohl es eine erhebliche Rolle im Bereich des verlässlichen Datentransfers leistet und geleistet hat, benötigt es gewisse Eigenschaften, die das moderne Internet braucht. Aus diesem Grunde haben einzelne Nutzer von TCP versucht, durch Extensions oder sogar mittels eigener Protokolle ihre Bedürfnisse zu befriedigen.

Die Entwickler von SCTP haben versucht, anhand der Erfahrung mit TCP ein neues Transportprotokoll zu entwerfen. Obwohl es ursprünglich zum Datentransfer innerhalb von PSTN (Public Switched Telephone Network) entwickelt wurde, können auch andere Anwendungen Gebrauch von den SCTP-Features machen.

2. EINFÜHRUNG IN SCTP

Bevor die Vor- und Nachteile von SCTP gegenüber TCP in den folgenden Kapiteln untersucht werden, bietet sich hier eine kurze Einführung in die Konzepte beider Protokolle. Diese hat den Zweck, dem Leser einen Überblick über SCTP zu verschaffen sowie die im Folgenden verwendeten Begriffe zu erklären.

2.1 Begriffserklärung und Annotation

An dieser Stelle muss erklärt werden, welche Bedeutung der Begriff *Stream* in diesem Paper trägt. TCP definiert Stream als eine Bytefolge, die in Pakete aufgeteilt und transportiert wird. SCTP dagegen spaltet den gesamten Datentransfer in mehrere logische Datenströme auf, die auch als Streams bezeichnet werden. In diesem Paper wird der TCP-Begriff als *Bytestream* und der SCTP-Begriff einfach als *Stream* bezeichnet, um Verwirrung zu vermeiden.

Eine SCTP-Verbindung wird auch *Assoziation* genannt, weil ein oder beide Endpunkte mehrere Netzwerkadressen besitzen können, womit der Term „Verbindung“ nicht mehr einen eindeutigen physischen Pfad zwischen den Peers definiert.

Mit *Overhead* ist der Datenzuschlag gemeint, der durch sämtliche Header zusätzlich zu den Benutzerdaten entsteht.

ULP ist eine Abkürzung für Upper-Layer Protocol, was im Fall von SCTP die Anwendung ist.

2.2 Wer kümmert sich um SCTP?

Der Protokoll wurde im Jahr 2000 von der IETF Signaling Transport Workgroup, SIGTRAN, entwickelt. Es wird von der IETF Transport Area Workgroup betreut, unterstützt und aktualisiert. Die Referenz [1] gibt die technische Spezifikation an.

2.3 Eigenschaften von SCTP

SCTP agiert auf der Transportschicht in der Internet Protocol Suite, also zwischen der Internetschicht und der Anwendungsschicht. Es ist verbindungsorientiert wie TCP und hat als kleinste Dateneinheit eine Message (beliebige Folge von Bytes). Die Verbindung wird zwischen genau zwei Endpunkten aufgebaut und kann in beide Richtungen Daten transportieren (full duplex). Jeder Endpunkt kann aber mehrere Netzadressen besitzen, was als „Multihoming“ bezeichnet wird (s. Abschnitt 3.4).

2.4 Paket-Struktur

Das SCTP-Paket (vgl. Abbildung 1) besteht aus einem Common Header und einer beliebigen Folge von Chunks. Der 12 Byte große Header enthält die Source- und Destination Ports (je 2 Byte), den Verification-Tag (4 Byte) zur Validierung der Pakete (s. Abschnitt 3.6) und eine Prüfsumme (4 Byte) als Schutz gegen Störungen auf der Leitung. Die Chunks sind Dateneinheiten, die entweder Nutzdaten oder Kontrollinformation transportieren. Jedes Chunk beginnt mit einem Byte, das den Chunk-Typ angibt (User Data, INIT, ACK, ABORT oder andere Kontrolltypen). Danach folgen Flags (1 Byte), Länge der Chunks und die eigentlichen Daten. Für eine genauere Beschreibung der verschiedenen Chunk-Typen ist der interessierte Leser auf die genaue Spezifikation verwiesen [1].

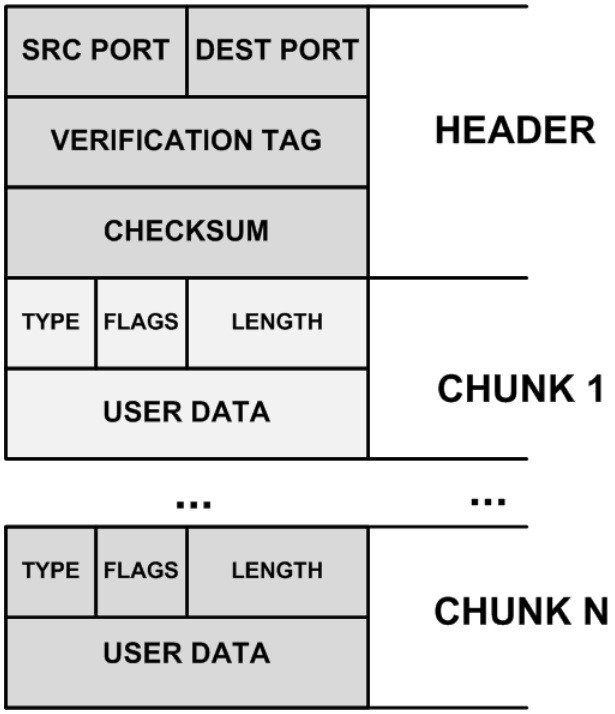

Abbildung 1: Die SCTP-Paketstruktur

3. FEATURES

In diesem Abschnitt werden kurz die Besonderheiten von SCTP dargestellt. Eine nähere Betrachtung und Bewertung der Features findet sich im Abschnitt 4 („Vergleich mit TCP“). Eine Liste findet sich auch in Tabelle 1.

3.1 Verbindungsauf- und abbau

SCTP implementiert einen 4-Way-Handshake-Verbindungsaufbau, der das Cookie-Verfahren verwendet. Das Cookie ist ein Datenblock, der die Verbindungsparameter zum Zeitpunkt des Verbindungsaufbaus enhält und zurück an den Client gesendet wird. Warum es sinnvoll ist, diese Daten im Cookie zu speichern und nicht auf dem Server selbst, wird in Abschnitt 4.3 näher beschrieben.

Feature name	UDP	TCP	SCTP
Connection oriented	No	Yes	Yes
Reliable transport	No	Yes	Yes
Unreliable transport	Yes	No	Yes
Preserve image boundary	Yes	No	Yes
Ordered delivery	No	Yes	Yes
Unordered delivery	Yes	No	Yes
Data checksum	Yes	Yes	Yes
Checksum size (bits)	16	16	32
Partial checksum	No	No	No
Path MTU	No	Yes	Yes
Congestion control	No	Yes	Yes
Multiple streams	No	No	Yes
Multihoming	No	No	Yes
Bundling	No	Yes	Yes

Tabelle 1: Unterstützte Features von UDP, TCP und SCTP

Im ersten Schritt sendet der Client ein INIT Paket (entspricht dem SYN in TCP), worauf der Server mit einem INIT-ACK Paket mit dem darin enthaltenen Cookie reagiert. Im dritten Schritt sendet der Client das erhaltene Cookie in einem COOKIE-ECHO Paket zurück, um die Verbindung zu bestätigen, daraufhin sendet der Server ein COOKIE-ACK Paket, und die Verbindung ist aufgebaut (s. Abbildung 4b). Im dritten und vierten Schritt dürfen bereits Daten übertragen werden, um eine weitere RTT (Round-Trip-Time) als Wartezeit bis zum Senden der ersten Nutzdaten zu vermeiden.

Eine Verbindung wird wie bei TCP per 3-Way-Handshake abgebaut. Dieser Vorgang wird für jeden der beiden Kommunizierenden vollzogen. Wenn beide Teilnehmer keine Daten mehr übertragen möchten, wird die Verbindung getrennt. Dadurch entsteht der Begriff „Half-Open-State“, was bedeutet, dass Daten nur noch in eine Richtung übertragen werden, bis auch der andere Peer mit der Übermittlung fertig ist.

SCTP unterscheidet sich dadurch von TCP, dass eine Assoziation keinen Half-Open-State besitzt, sondern erzwingt, dass beide Teilnehmer die Verbindung aufgeben, sobald alle gepufferten Daten erfolgreich übertragen wurden.

Es ist auch möglich, eine Verbindung mittels ABORT (analog zu RST bei TCP) zu unterbrechen, falls z.B. ein Restart erfolgt oder ein Fehler entdeckt wurde.

3.2 Messages und Zuverlässigkeit der Lieferung

SCTP bietet eine verlässliche Übertragung, d.h. die Ankunft der Daten auf der anderen Seite der Verbindung ist garantiert. Das geschieht ähnlich wie bei TCP: Die Daten werden mit Sequenznummern (SN) versehen, die der Empfänger mit einem SACK Paket (selective acknowledgement) [4] bestätigt. Im Gegensatz zu TCP, das keine Teilung der Daten unternimmt, sondern lediglich eine ununterbrochene Folge von Bytes an den Peer sendet, transportiert SCTP die Daten in Form von Messages (s. Abbildung 2a-b). Diese Messages können in kleinere Stücke fragmentiert werden, wenn sie die ermittelte Path-MTU (Maximum Transmission Unit [5], die Packetgröße, die der Netzwerkpfad erlaubt) überschreiten. Für viele, kleine Messages ist Bündelung vorgesehen, so dass mehrere kleine Nachrichten (jede in einem Data-Chunk verpackt) in einem größeren SCTP-Paket zusammengefasst werden können.

Ein Unterschied zu TCP besteht darin, dass die Sequenz-Nummern bei SCTP pro Message vergeben werden, nicht pro Byte. Dadurch wird z.B. die Berechnung der Datenmenge, die sich auf dem Weg zum Peer befindet, etwas komplizierter, da die Größe der Messages zusätzlich beachtet werden muss.

3.3 Streams in SCTP

Da der Datenstrom in SCTP in einzelnen Nachrichten organisiert ist, besteht die Möglichkeit, die Nachrichten wiederum in einzelne, logisch abgetrennte Ströme aufzuspalten (s. Abbildung 2c).

TCP leistet einen geordneten Transportdienst, d.h. Daten werden in der richtigen Reihenfolge geliefert. Für manche Anwendungen ist das aber nicht für alle Daten oder für gar keine Daten notwendig. Beispielsweise ist die richtige Reihenfolge für Dateien kritisch, nicht aber für Signalnachrichten oder Online-Spiele, wo die Datenübertragung in Echtzeit stattfindet. Um diese Funktionalität zu gewährleisten, erlaubt SCTP mehrere logische Datenströme. Die Reihenfolge der Daten wird für den einzelnen Strom garantiert, und optional kann der Nutzer auch ungeordnete Ströme festlegen. So kann der Nutzer selbst angeben, welche Daten reihenfolgekritisch sind und welche nicht. Das hat auch einen weiteren Vorteil – es verhindert, dass ein Strom, dessen Pakete auf der Strecke verloren gegangen sind, den Fluss der anderen Ströme blockiert. Dieses Problem ist in der Fachwelt als *Head-Of-Line blocking* bekannt [6] und eine vergleichbare Lösung kann in TCP nur mit mehreren Verbindungen verwirklicht werden. Das erschwert aber die Instandhaltung der Verbindung: Die einzelnen Datenflüße sind trotzdem logisch abgespalten, unterliegen nicht ein und derselben Staukontrolle und generieren zusatzlichen Overhead. Außerdem kann eine Anwendung mittels zweier oder mehreren TCP-Verbindungen eine mehrfache Bandbreite beantragen, was die anderen Anwendungen bzw. Benutzer im Netz benachteiligt.

Mit einer Erweiterung [20] von SCTP kann man sogar unzuverläßige Verbindungen definieren, d.h. solche, bei denen verlorene Pakete nicht erneut vermittelt werden, ähnlich wie bei UDP. Es ist sogar möglich, einen unsicheren, aber geordneten Stream zu definieren (diese Funktionalitat bietet UDP nicht). Da SCTP die Daten in Streams einteilt, ist es moglich, unsichere mit sicheren Datenströmen innerhalb einer Assoziation zu multiplexen. Alternativ kann man eine UDP- und eine TCP-Verbindung parallel aufbauen, was aber mit einem zusätzlichen Overhead verbunden ist. Außerdem unterliegt mit SCTP auch die unsichere Verbindung einer Staukontrolle, etwas, was UDP überhaupt nicht beachtet. Diese Erweiterung heißt SCTP-PR (fur Partial Reliability) und bietet eine Reihe weiterer Möglichkeiten. Zum Beispiel kann man spezifizieren, wie lange der Sender warten muss, bevor er nicht bestätigte Pakete verwirft. Die genaue Spezifikation von SCTP-PR ist in [20] enthalten.

SCTP verwaltet die Ströme, indem jedes Data-Chunk mit einer Stream-ID versehen wird. Die maximale Anzahl von Streams wird wahrend der INIT-Phase zwischen Client und Server ausgehandelt.

3.4 Multihoming

SCTP erlaubt, dass einer oder beide Verbindungsteilnehmer mehrere IP-Adressen besitzen können. Daduch entstehen mehrere Möglichkeiten zur Auswahl des Netzwerkpfades zum Peer. Im Falle eines Ausfalls der physischen Verbindung kann der Sender eine alternative Route wählen. Die Verbindung geht nicht verloren, solange es noch mindestens einen aktiven Pfad gibt. Im Allgemeinen bietet die Verbindung dadurch einen besseren Ausfallschutz als eine TCP-Verbindung. Das Multihoming hat eine Einschränkung – es bietet standardmäßig kein Load Sharing [7], es werden also nicht die Kapazitäten aller Netzwerkschnittstellen ausgenutzt. Der Sender wählt einen Pfad und sendet darüber alle Daten, es sei denn der Pfad wird unterbrochen. Erst

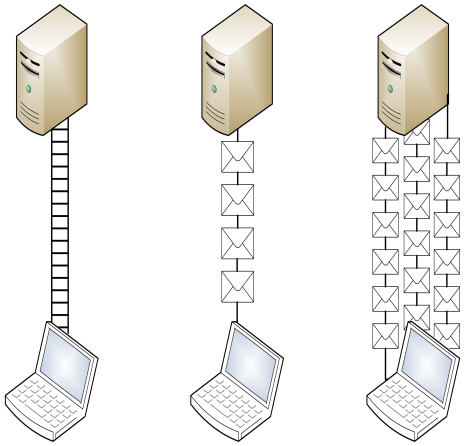

Abbildung 2: Von links nach rechts: (a) TCP mit reinem Bytestrom, (b) Message-orientiertes SCTP, (c) Message-orientiertes SCTP mit Streams

dann versucht SCTP einen anderen Pfad zu finden. Die Pfade, die aktuell nicht zur Datenübertragung verwendet werden, überprüft SCTP laufend auf Verfügbarkeit, indem es HEARTBEAT-Requests an den Peer sendet, der seinerseits mit HEARTBEAT-ACK antwortet. Wenn eine Zeitlang kein ACK ankommt, wird der Pfad als inaktiv gekennzeichnet.

Obwohl mehrere IP-Adressen eingesetzt werden, ist die Portnummer für die bestehende Assoziation eindeutig.

3.5 Stau- und Flusskontrolle

Die Staukontrolle ist ein Mechanismus, der die Überlastung des darunter liegenden Transportnetzes vermeidet und auf spontan enstandene Überlast reagiert (z.B. im Fall einer ausgefallenen Backbone-Verbindung, die den Datentransfer dramatisch verringert). Es basiert auf vier Prinzipien – slow start, congestion avoidance, fast retransmit und fast recovery. Die Staukontrolle ist ein komplexes und von der Implementierung abhängiges Konzept, dessen genauere Betrachtung den Rahmen dieses Papers sprengen würde. SCTP lehnt sich stark an die in TCP definierten [9] Techniken an, der fundamentale Unterschied besteht lediglich im Multihoming-Feature. Referenz [1] legt fest, dass die Staukontrolle pro Empfänger-IP-Adresse durchgeführt werden soll. Nähere Infos befinden sich im Kapitel Congestion control von [1].

Die Flusskontrolle schützt vor Überflutung des Empfängers mit Informationen. Dies kann vorkommen, wenn der Empfänger viel langsamer Daten verarbeitet als der Sender (zum Beispiel bei der Verbindung mit einem PDA). TCP und SCTP ähneln sich auch in der Flusskontrolle, mit der Ausnahme, dass ungeordnete Ströme keine Lücken im Transport aufweisen können, was die Kalkulation des Receiver-Windows beeinflusst [10].

3.6 Paket-Validierung

Das SCTP-Paket enthält im Common-Header einen 32 Bit langen Validierungstag (s. Abbildung 3). Pakete mit falschem Validierungstag werden schweigend verworfen. Dieser Tag hat verschiedene Anwendungen, die wichtigste davon ist ein Schutz gegen blinde Masquerade-Attacken. Es ist praktisch

unmöglich für einen blinden Angreifer (der keinen Zugriff auf die Transportverbindung hat) eine Verbindung vorzutäuschen (connection forgery), da er dafür den richtigen Validierungstag erraten müsste. Eine zusätzliche Funktion dieses Datenfelds stellt die Erkennung von Paketen aus einer alten Verbindung zwischen den gleichen Teilnehmern dar.

Man darf den Validierungstag nicht als absoluten Schutz gegen Angreifer betrachten, denn es funktioniert nur gegen absolut blinde Attacker (s. Abbildung 3). Falls der Angreifer die Pakete lesen kann (s. Abbildung 3b), kann er den Validierungstag einfach kopieren. Für eine bessere Kommunikationssicherung ist der Nutzer auf IPsec, SSL/TLS angewiesen [11].

4. VERGLEICH MIT TCP UND ANWENDBARKEIT

Nachdem wir die Fähigkeiten von SCTP schematisch beschrieben haben, versuchen wir in diesem Abschnitt eine Gegenüberstellung zu TCP herzustellen und dabei die Stärken und Schwächen von SCTP zu erkennen.

4.1 Chunks vs. Bytestream

Wie in Abschnitt 2.3 erwähnt, ist SCTP Message-orientiert im Gegensatz zu TCP, das Byteströme vermittelt. Das hat sowohl Vorteile als auch Nachteile. In SCTP muss sich der Nutzer nicht um die Segmentierung der gesendeten Daten kümmern. Da das ULP (Upper-Layer-Protocol) jetzt keinen Bytestrom mehr abfragt, sondern ganze Nachrichten, entsteht eine neue Möglichkeit der Kommunikation zwischen Transport- und Anwendungsschicht. Die überliegende Instanz wird benachrichtigt sobald eine neue Nachricht komplett empfangen wurde. Das macht die explizite Nachfrage an Daten überflüssig seitens der Anwendung, denn der Nutzer kann einfach im Idle Modus warten, bis er auf angekommene Daten benachrichtigt wird. Bei TCP werden Anwendungen ebenfalls blockiert, falls keine Daten im Puffer liegen, aber es ist schwierig eine Anwendung zu implementieren, die aus mehreren TCP-Verbindungen gleichzeitig lesen kann (vgl. SCTP-Streams in Kapitel 3.3).

Andererseits entsteht ein zusätzlicher Overhead bei dieser Art der Datenvermittlung, da jede Nachricht einen eigenen Header (4 Byte) hat.

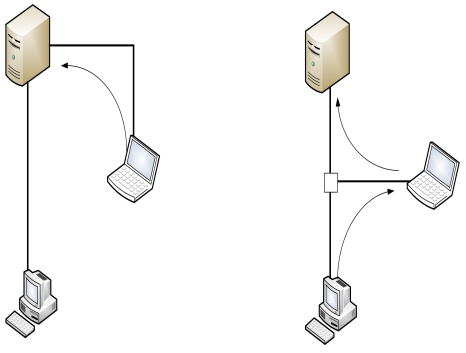

Abbildung 3: Links – ein absolut blinder Angreifer, rechts – der Anreifer kann die Kommunikation mitschneiden, aber nicht beeinflussen

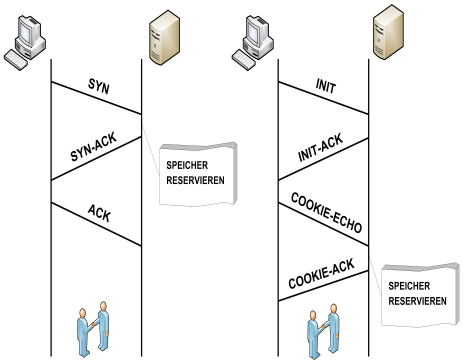

Abbildung 4: Verbindungsaufbau bei (a) TCP und (b) SCTP

4.2 Vor- und Nachteile der Streams

Streams (s. Abschnitt 3.4.3) sind gut geeignet für den Transport von mehreren Date(ie)n, deren Bytereihenfolge wichtig ist, nicht aber die Reihenfolge ihrer Ankunft. Es ist zum Beispiel unerheblich, welches Bild einer Web-Seite zuerst ankommt, es ist aber nicht egal, in welcher Reihenfolge die einzelnen Bytes eines Bildes ankommen. Eine Web-Seite ist ein gutes Beispiel, wie Streams effektiv eingesetzt werden können. Die ganze Web-Seite kann innerhalb einer Assoziation übertragen werden, wobei die (x)HTML Datei, die Bilder, Skripte, Style-Sheets, Applets etc. jeweils in eigenen Streams übertragen werden können. Wenn sich dann ein Paket auf der Strecke verzögert oder gar ausfällt, wird das die restlichen Komponenten nicht blockieren. [6] bietet eine gute praktische Evaluation der Streams in SCTP und untersucht statistisch, inwiefern sie die Performanz einer Verbindung beeinflussen.

4.3 Das Cookie

TCP stellt eine Verbindung mittels 3-Way-Handshake her: Der Client sendet ein SYN-Paket, der Server antwortet mit SYN-ACK und der Client sendet ACK zur Bestätigung der Verbindung (s. Abbildung 4a). Das hat den Nachteil, dass zu jedem SYN-Request auf der Serverseite Speicher für die entstehende Verbindung zugewiesen wird. Der in diesem Speicher enthaltene Transport Control Block (TCB) wird für eine Zeitlang behalten, in der aber ein potentieller Angreifer sehr viele verfälschte TCP-Verbindungen anfordern kann. Manche Server implementieren das sogenannte Cookie-Verfahren, um sich gegen solche Angriffe zu verteidigen: Wenn sich der Puffer auf dem Server füllt, teilt er keinen Speicher mehr zu, sondern kodiert die Verbindungsparameter in einem sog. SYN-Cookie im SYN-ACK-Paket und schickt es an den Anfrager zurück. Diese Technik verhindert Denial-of-Service-Angriffe gegen den Server, schränkt aber die Verbindung ein (die maximale Segment-Größe ist z.B. begrenzt), da das Cookie nur 32 Bit lang sein darf. Ein weiterer Nachteil ist, dass die Verbindung einfriert, wenn das ACK des Clients verloren geht, da der Server nicht weiß, dass eine Verbindung hergestellt wurde und deshalb die eingehenden Daten des Clients schweigend verwirft (s. Abbildung 5). Referenz [2] beschreibt das Verhalten von TCP unter SYN-Angriffen sowie den Cookie-Mechanismus.

In SCTP ist dieses Verfahren ein Teil des Verbindungsauf-

baus (s. Abbildung 4b). Neben dem Schutz vor DoS-Angriffen entlastet es den Server und generiert kaum zusätzlichen Overhead. Das Cookie ist mit einer Checksumme (HMAC) geschützt, die mit dem Private Key des Servers berechnet wird, kann also vom Empfänger/Angreifer praktisch nicht erraten bzw. verändert werden.

4.4 Prüfsumme

Die von SCTP verwendete Prüfsumme wurde in seiner ersten Version mit dem Adler-32-Algorithmus [13] berechnet. [14] definiert diese Vorschrift neu und spezifiziert den crc32-Algorithmus als Standard, weil das für kleinere Pakete erheblich besseren Rauschschutz bietet.

4.5 Ähnlichkeiten von SCTP und TCP

Die zwei Transportprotokolle sind sich in vielen Punkten ähnlich oder gleich. Zum Beispiel verwenden beide den gleichen SACK-Ansatz, um Daten rückzumelden (SACK ist bei TCP jedoch lediglich als Extension realisiert [4]). Das Portsystem ist das Gleiche und die Stau- und Flusskontrolle basiert auf den gleichen Prinzipien, was eine Kooperation ermöglicht, falls beide Protokollinstanzen parallel laufen: z.B. ist es möglich, dass beide Protokolle die gleiche Instanz der Stau- und Flusskontrolle verwenden.

5. PROBLEME UND EINSCHRÄNKUNGEN VON SCTP

Leider hat SCTP nicht nur Vorteile, sondern auch Probleme. Hier werden einige der SCTP-Schwächen behandelt, die auf die behandelten Features zurückzuführen sind.

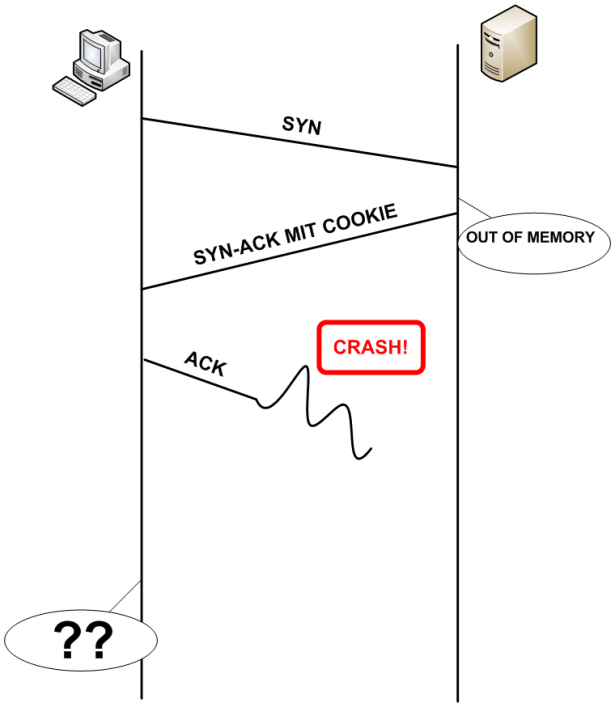

Abbildung 5: Eingefrorene Verbindung mit verlorenem TCP Cookie

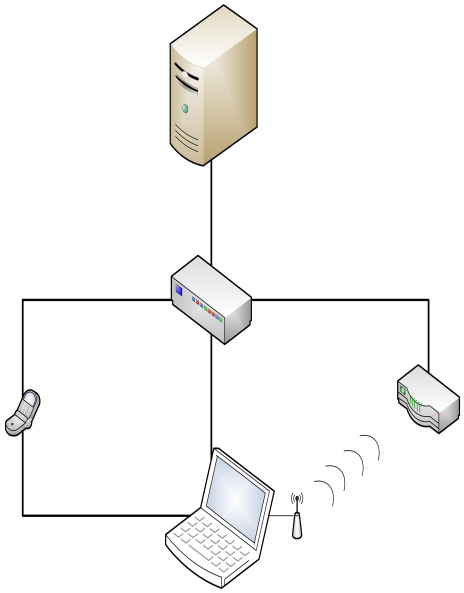

Abbildung 6: Obwohl der Client mehrere Netzwerkschnittstellen hat, verlaufen alle über den gleichen Router

5.1 Performanz

Eine SCTP-Assoziation mit nur einem Stream transportiert Daten langsamer, als eine gewöhnliche TCP-Verbindung. Die Einteilung des Datenstroms in Messages generiert zusätzlichen Overhead.

5.2 Multihoming

Leider kommt es vor, dass Endpunkte A und B mehrere Netzwerkschnittstellen besitzen können, die Pfade aber verlaufen über ein und denselben Knoten innerhalb des Netzwerks und können bei einem eventuellen Ausfall dieses Knotens alle betroffen sein (s. Abbildung 6). Multihoming ist mehr eine Verbesserung der Ausfallsicherheit, als Garantie gegen den Ausfall der Transportwege.

Ein weiteres Problem ist IPsec (VPN) in Zusammenarbeit mit SCTP. Falls die Endpunkte jeweils m und n IP-Adressen haben, muss IPsec insgesamt $2 * m * n$ Sicherheitsassoziationen aufbauen. In [21] wird diskutiert, wie IPsec angepasst werden kann, um eine Multihoming-Verbindung zu erkennen.

5.3 Inkompatibilität mit NAT und Firewalls

Da ein SCTP-Endpunkt mehrere IP-Adressen haben kann, ist es genau von jenen Internet-Verfahren betroffen, die IP-Adressen verbergen, blockieren oder verändern, z.B. NAT [22]. An dieser Stelle soll nicht zu tief ins Thema eingetaucht werden, da Network Address Translation und Firewalls komplexe Instanzen sind, die sich nicht nur auf den IP-Level beschränken. Die einfachste Methode, SCTP über NAT durchzusetzen ist, indem man beim Verbindungsaufbau keine explizite IP-Adressen an den Peer mitteilt. Das bewirkt, dass die Adresse im IP-Header bezogen wird; die IP-Pakete werden vom NAT erkannt und richtig umgeleitet. Intelligente NAT-Geräte könnten SCTP-Pakete erkennen und die IP-Adressen richtig übersetzen. [15] beschreibt

die Einschränkungen, die NAT auf SCTP erzwingt, und gibt eine NAT-Implementierung an, die SCTP unterstützt.

Bei Firewalls liegt das Problem darin, dass SCTP noch nicht weit verbreitet ist, es ist also unwahrscheinlich, dass eine Firewall SCTP-Pakete durchlässt.

6. VERWANDTE ARBEITEN

Es gibt viele Artikel im World Wide Web, die SCTP mit anderen Transportprotokollen vergleichen und den Nutzen bewerten.

Die Masterarbeit von Ivan Rodriguez [8] bietet einen breiten Hintergrund zum Thema SIGTRAN, SS7 und SCTP und behandelt die hier eingeführten Konzepte detailliert (und übrigens auch in einer unterhaltsamen Form).

[3] befasst sich mit der Anwendbarkeit von SCTP in MPI (ein Netzwerkinterface für wissenschaftliche und andere hochparallele Rechenanlagen).

[7] bietet eine Möglichkeit, SCTP Multihoming zu verwenden, um Daten gleichzeitig über alle verfügbaren Pfade zu transportieren (Load Sharing).

7. ZUSAMMENFASSUNG

7.1 Ausblick – welches Potenzial hat SCTP, sich als Standard zu etablieren?

SCTP ist im Jahr 2000 als das vierte Transportprokoll neben TCP, RTP und UDP von der IETF standardisiert worden und es wird bereits von mehreren Betriebssystemen unterstützt. Es gibt eine Implementierung für den Linux-Kernel, die man auf Sourceforge herunterladen kann [16]. Die Implementierung von Randy Stewart ist im KAME-Project [17] integriert und IBM arbeitet an einer Version für AIX.

IPsec arbeitet derzeit daran, Overhead im multihomed Transport zu reduzieren [18]. Am Thema „TLS over SCTP“ wird auch gearbeitet [11]. Es ist möglich, SCTP hinter der TCP API zu verbergen, um die Einführung des neuen Transportprotokolls zu beschleunigen [19]. Es gibt auch eine Implementierung, die SCTP über UDP tunnelt.

Obwohl SCTP auf TCP basiert, das ein stabiles Verhalten über längere Zeit aufgewiesen hat, ist SCTP noch nicht weit verbreitet und dadurch auch nicht Subjekt von Angriffen und Anomalien. Es könnte durchaus andere, neue Schwächen und Probleme aufweisen, die TCP und UDP nicht haben.

7.2 Schlussfolgerung

SCTP hat die Aufgabe, einen besseren, flexibleren, zuverlässigeren Transportdienst für die Bedürfnisse des Internets in der nahen Zukunft anzubieten. Es soll die Einfachheit und Stabilität von TCP mit der Performanz von UDP kombinieren und diese mit etwas Ausfallsicherheit zu erweitern. Mit der Geschwindigkeit, mit der sich das Internet entwickelt, wird es nicht mehr lange dauern, bis Begriffe wie Multihoming, Parallel Data Transmission und Network Resilience eine globale und wichtige Rolle spielen werden. In diesem Paper wurde die Umsetzung dieser Konzepte in SCTP von einem theoretischen Standpunkt aus betrachtet; ob sie aber effizient, stabil und sicher implementiert sind, kann nur die Praxis zeigen.

8. LITERATUR

[1] Stewart, R., *„Stream Control Transmission Protocol“*, RFC 4960, IETF, September 2007, `http://tools.ietf.org/html/rfc4960`

[2] Nakashima, T., Sueyoshi, T., *„Performance Estimation of TCP under SYN Flood Attacks“*, Complex, Intelligent and Software Intensive Systems, 2007. CISIS 2007. First International Conference, pp.92-99, 10-12 April 2007 doi: 10.1109/CISIS.2007.48

[3] Kamal, H., Penoff, B., Wagner, A., *„SCTP versus TCP for MPI“*, Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, pp. 30- 30, 12-18 November 2005 doi: 10.1109/SC.2005.63

[4] Mathis, M., Mahdavi, J., Floyd S., Romanow, A., *„TCP Selective Acknowledgment Options“*, IETF, October 1996, `http://tools.ietf.org/html/rfc2018`

[5] Mogul, J., Deering, S., *„Path MTU Discovery“*, RFC 1191, IETF, November 1990, `http://tools.ietf.org/rfc/rfc1191`

[6] Grinnemo, K.-J., Andersson, T., Brunstrom, A., *„Performance Benefits of Avoiding Head-of-Line Blocking in SCTP“*, Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference, pp.44-44, 23-28 October 2005 doi: 10.1109/ICAS-ICNS.2005.73

[7] Abd El Al, A., Saadawi, T., Myung Lee, *„Bandwidth aggregation in stream control transmission protocol“*, Computers and Communications, 2004. Proceedings. ISCC 2004. Ninth International Symposium, vol.2, pp. 975-980, 28 June-1 July 2004

[8] Rodríguez, I. A., Kantola, R., Loughney, J., *„Stream Control Transmission Protocol. The design of a new reliable transport protocol for IP networks“*, Helsinki University of Technology, February 2002

[9] Allman, M., Paxton, V, Stevens, W., *„TCP congestion control“*, RFC 2581, IETF, April 1999, `http://www.ietf.org/rfc/rfc2581`

[10] Jiemin Liu, Hongxing Zou, Jingxin Dou, Yuan Gao, *„Reducing Receive Buffer Blocking In Concurrent Multipath Transfer“*, Circuits and Systems for Communications, 2008. ICCSC 2008. 4th IEEE International Conference, pp.367-371, 26-28 May 2008 doi: 10.1109 / ICCSC.2008.85

[11] Jungmaier, A., Rescorla, E., Tuexen, M., *„Transport Layer Security over Stream Control Transmission Protocol“*, RFC 3436, IETF, September 2007 `http://www.ietf.org/rfc/rfc3436`

[12] Jungmaier, A., Pathgeb, E., *„On SCTP multi-homing performance“*, Telecommunication Systems, Springer Netherlands, vol. 31, no. 2-3 / March 2006

[13] Deutsch, P., Gailly, J-L., *„ZLIB Compressed Data Format Specification version 3.3“*, RFC 1950, IETF, May 1996 `http://tools.ietf.org/html/rfc1950`

[14] Stone, J., Stewart, R., Otis, D., *„Stream Control*

Transmission Protocol (SCTP) Checksum Change“, RFC 3309, IETF, September 2002 http://tools.ietf.org/html/rfc3309

[15] Tuxen, M., Rungeler, I., Stewart, R., Rathgeb, E., *„Network Address Translation for the Stream Control Transmission Protocol“*, Network, IEEE , vol.22, no.5, pp.26-32, September-October 2008 doi: 10.1109 / MNET.2008.4626229

[16] *„Linux Kernel SCTP“*, http://sourceforge.net/projects/lksctp/Df

[17] *„The KAME Project“*, http://www.kame.net/

[18] Cano, M-D., Romero, J. A., Cerdan, F., *„Experimental Tests on SCTP over IPSec“*, Network and Parallel Computing, 2008. NPC 2008. IFIP International Conference, pp.96-102, 18-21 October 2008 doi: 10.1109 / NPC.2008.92

[19] Bickhart, R., Amer, P., Stewart, R., *„Transparent TCP-to-SCTP Translation Shim Layer“*, EuroBSDCon 2007, Copenhagen, 5/07 http://www.cis.udel.edu/~amer/PEL/poc/pdf/EuroBSDCon2007-bickhart-SCTP-Shim-layer.pd

[20] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., Conrad, P., *„Stream Control Transmission Protocol Partial Reliability Extension“*, RFC 3758, IETF, May 2004 http://www.ietf.org/rfc/rfc3758

[21] Bellovin, S., Ioannidis, J., Keromytis, A., Stewart, R., *„On the use of SCTP with IPsec“*, RFC 3554, IETF , July 2003 http://www.ietf.org/rfc/rfc3554

[22] Egevang K., Francis P., *„The IP Network Address Translator (NAT)“*, RFC 1631, IETF, May 1994, http://tools.ietf.org/html/rfc1631