流控制传输协议 SCTP

摘要

SCTP 被设计用来在 IP 网上传输 PSTN 信令消息,但能有更广泛的用途。

SCTP 是一个可靠的传输层协议，运行在无连接的包网络上（如 IP 网）。它为用户提供下列服务:

-- 确保（acknowledged）用户数据无差错，无重复的传送；
-- 根据通路 MTU 的限制，进行用户数据的分段；
-- 流内用户消息的顺序递交；（可选的）每个用户消息按到达顺序（order-of-arrival）进行递交；
-- （可选的）多个用户消息捆绑到单个 SCTP 包中进行传输；
-- 通过在偶联的一端或两端提供的多归属（multi-homing）机制，提供网络级容错；

SCTP 的设计包含了避免拥塞机制和防泛播（flooding）及匿名攻击（masquerade attacks）的能力。

目录

1. Introduction..... 5
1.1 Motivation..... 6
1.2 Architectural View of SCTP..... 6
1.3 Functional View of SCTP..... 7
1.3.1 Association Startup and Takedown..... 8
1.3.2 Sequenced Delivery within Streams..... 9
1.3.3 User Data Fragmentation..... 9
1.3.4 Acknowledgement and Congestion Avoidance..... 9
1.3.5 Chunk Bundling 10
1.3.6 Packet Validation..... 10
1.3.7 Path Management..... 11
1.4 Key Terms..... 11
1.5 Abbreviations..... 15
1.6 Serial Number Arithmetic..... 15
2. Conventions..... 16
3. SCTP packet Format..... 16
3.1 SCTP Common Header Field Descriptions..... 17
3.2 Chunk Field Descriptions..... 18
3.2.1 Optional/Variable-length Parameter Format..... 20
3.3 SCTP Chunk Definitions..... 21
3.3.1 Payload Data (DATA)..... 22
3.3.2 Initiation (INIT)..... 24

3.3.2.1 Optional or Variable Length Parameters..... 26
3.3.3 Initiation Acknowledgement (INIT ACK)..... 30
3.3.3.1 Optional or Variable Length Parameters..... 33
3.3.4 Selective Acknowledgement (SACK)..... 33
3.3.5 Heartbeat Request (HEARTBEAT)..... 37
3.3.6 Heartbeat Acknowledgement (HEARTBEAT ACK)..... 38
3.3.7 Abort Association (ABORT)..... 39
3.3.8 Shutdown Association (SHUTDOWN)..... 40
3.3.9 Shutdown Acknowledgement (SHUTDOWN ACK)..... 40
3.3.10 Operation Error (ERROR)..... 41
3.3.10.1 Invalid Stream Identifier..... 42
3.3.10.2 Missing Mandatory Parameter..... 43
3.3.10.3 Stale Cookie Error..... 43
3.3.10.4 Out of Resource..... 44
3.3.10.5 Unresolvable Address..... 44
3.3.10.6 Unrecognized Chunk Type..... 44
3.3.10.7 Invalid Mandatory Parameter..... 45
3.3.10.8 Unrecognized Parameters..... 45
3.3.10.9 No User Data..... 46
3.3.10.10 Cookie Received While Shutting Down..... 46
3.3.11 Cookie Echo (COOKIE ECHO)..... 46
3.3.12 Cookie Acknowledgement (COOKIE ACK)..... 47
3.3.13 Shutdown Complete (SHUTDOWN COMPLETE)..... 48
4. SCTP Association State Diagram..... 48
5. Association Initialization..... 52
5.1 Normal Establishment of an Association..... 52
5.1.1 Handle Stream Parameters..... 54
5.1.2 Handle Address Parameters..... 54
5.1.3 Generating State Cookie..... 56
5.1.4 State Cookie Processing..... 57
5.1.5 State Cookie Authentication..... 57
5.1.6 An Example of Normal Association Establishment..... 58
5.2 Handle Duplicate or unexpected INIT, INIT ACK, COOKIE ECHO, and COOKIE ACK..... 60
5.2.1 Handle Duplicate INIT in COOKIE-WAIT or COOKIE-ECHOED States..... 60
5.2.2 Unexpected INIT in States Other than CLOSED, COOKIE-ECHOED, COOKIE-WAIT and SHUTDOWN-ACK-SENT..... 61
5.2.3 Unexpected INIT ACK..... 61
5.2.4 Handle a COOKIE ECHO when a TCB exists..... 62
5.2.4.1 An Example of a Association Restart..... 64
5.2.5 Handle Duplicate COOKIE ACK..... 66
5.2.6 Handle Stale COOKIE Error..... 66
5.3 Other Initialization Issues..... 67

5.3.1 Selection of Tag Value..... 67
6. User Data Transfer..... 67
6.1 Transmission of DATA Chunks..... 69
6.2 Acknowledgement on Reception of DATA Chunks..... 70
6.2.1 Tracking Peer's Receive Buffer Space..... 73
6.3 Management Retransmission Timer..... 75
6.3.1 RTO Calculation..... 75
6.3.2 Retransmission Timer Rules..... 76
6.3.3 Handle T3-rtx Expiration..... 77
6.4 Multi-homed SCTP Endpoints..... 78
6.4.1 Failover from Inactive Destination Address..... 79
6.5 Stream Identifier and Stream Sequence Number..... 80
6.6 Ordered and Unordered Delivery..... 80
6.7 Report Gaps in Received DATA TSNs..... 81
6.8 Adler-32 Checksum Calculation..... 82
6.9 Fragmentation..... 83
6.10 Bundling 84
7. Congestion Control 85
7.1 SCTP Differences from TCP Congestion Control..... 85
7.2 SCTP Slow-Start and Congestion Avoidance..... 87
7.2.1 Slow-Start..... 87
7.2.2 Congestion Avoidance..... 89
7.2.3 Congestion Control..... 89
7.2.4 Fast Retransmit on Gap Reports..... 90
7.3 Path MTU Discovery..... 91
8. Fault Management..... 92
8.1 Endpoint Failure Detection..... 92
8.2 Path Failure Detection..... 92
8.3 Path Heartbeat..... 93
8.4 Handle "Out of the blue" Packets..... 95
8.5 Verification Tag..... 96
8.5.1 Exceptions in Verification Tag Rules..... 97
9. Termination of Association..... 98
9.1 Abort of an Association..... 98
9.2 Shutdown of an Association..... 98
10. Interface with Upper Layer.....101
10.1 ULP-to-SCTP.....101
10.2 SCTP-to-ULP.....111
11. Security Considerations.....114
11.1 Security Objectives.....114
11.2 SCTP Responses To Potential Threats.....115
11.2.1 Countering Insider Attacks.....115
11.2.2 Protecting against Data Corruption in the Network.....115
11.2.3 Protecting Confidentiality.....115

11.2.4 Protecting against Blind Denial of Service Attacks.....116
11.2.4.1 Flooding.....116
11.2.4.2 Blind Masquerade.....118
11.2.4.3 Improper Monopolization of Services.....118
11.3 Protection against Fraud and Repudiation.....119
12. Recommended Transmission Control Block (TCB) Parameters.....120
12.1 Parameters necessary for the SCTP instance.....120
12.2 Parameters necessary per association (i.e. the TCB).....120
12.3 Per Transport Address Data.....122
12.4 General Parameters Needed.....123
13. IANA Considerations.....123
13.1 IETF-defined Chunk Extension.....123
13.2 IETF-defined Chunk Parameter Extension.....124
13.3 IETF-defined Additional Error Causes.....124
13.4 Payload Protocol Identifiers.....125
14. Suggested SCTP Protocol Parameter Values.....125
15. Acknowledgements.....126
16. Authors' Addresses.....126
17. References.....128
18. Bibliography.....129
Appendix A 131
Appendix B 132
Full Copyright Statement 134

1. 引言

本节解释了发展SCTP的缘由, SCTP提供的服务及理解协议细节所需的基本概念。

1.1 动机

作为IP网上可靠数据传输的主要手段，TCP [RFC793] 已经提供了非常好的服务（产生了大量应用）。然而，近来越来越多的应用发现TCP具有许多局限性，进而已经改用它们自己运行在UDP[RFC768]上的可靠数据传输协议。用户希望TCP解决的局限性主要有：

-- TCP保证用户数据的可靠传输及严格按传输顺序向用户递交数据。然而一些应用仅需数据的可靠传输（无需维持用户数据的顺序），而另一些也仅需部分数据按序递交。在这两种情况下，TCP可能引发的队头阻塞（head-of-line blocking）问题会造成不必要的延迟。

-- TCP面向流的本质经常是麻烦的来源。Applications must add their own record marking to delineate their messages, and must make explicit use of the push facility to ensure that a complete message is transferred in a reasonable time.

-- TCP sockets 的局限性使得利用多归属机制提供高度可得到的数据传输变的更为复杂。

-- TCP 相对易于遭受拒绝服务攻击，如 SYN 攻击。

在 IP 网上传输 PSTN 信令就是一个需要解决 TCP 所有局限性的应用。尽管这个应用直接推动了 SCTP 的开发，但是 SCTP 可能也满足其它一些应用的需要。

1.2 SCTP 结构视图

SCTP 被视为 SCTP 用户应用（简称 SCTP 用户）和无连接包网络服务（如 IP）之间的一层。本文档假定 SCTP 运行在 IP 之上。SCTP 提供的基本服务就是在对等的 SCTP 用户之间提供可靠的用户消息传输。SCTP 通过两个 SCTP 端点（endpoints）间建立的偶联提供此服务。第 10 节给出了 SCTP 和 SCTP 用户之间的 API。

SCTP 是面向连接的协议，但 SCTP 的偶联（association）是一个比 TCP 的连接更广泛的概念。在偶联建立的过程中，SCTP 能让一个 SCTP 端点向另一 SCTP 端点提供一个传输地址列表（如多个 IP 地址结合一个 SCTP 端口），通过传输地址列表 SCTP 端点可达并从它发起 SCTP 包（传输地址列表是 SCTP 端点的标识）。The association spans transfers over all of the possible source/destination combinations which may be generated from each endpoint's lists.

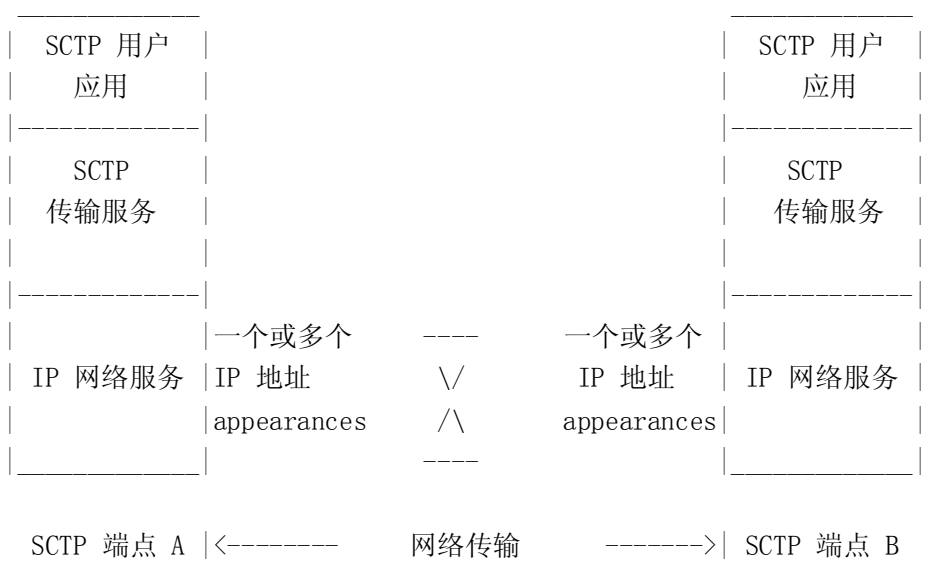


图 1: SCTP 偶联的概念

1.3 SCTP 功能视图

SCTP 提供的传输服务能被分解为几个部分，如图 2 所示。

SCTP 用户应用

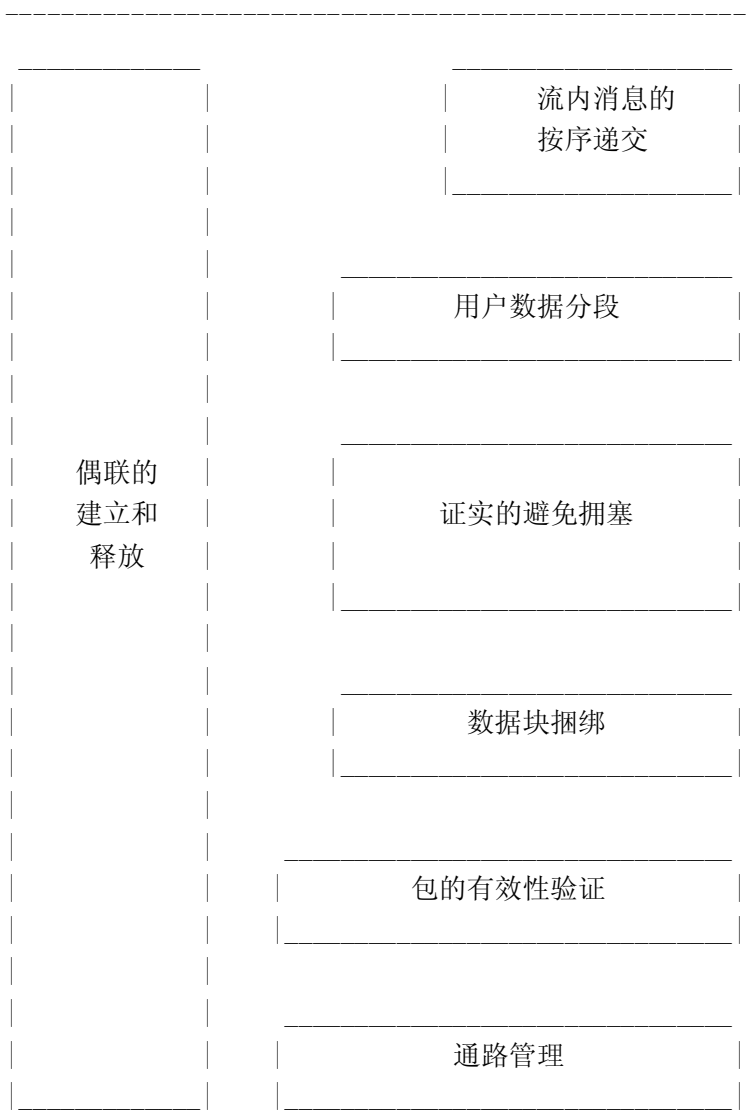


图 2: SCTP 传输服务的功能视图

1.3.1 偶联的建立和释放

偶联被来自 SCTP 用户的请求发起(见第 10 节的 ASSOCIATE (or SEND)原语描述)。

在偶联初始化过程中，一个 cookie 机制————类似于 Karn 和 Simpson 在[RFC2522]中所描述的————被运用，以提供安全保护。cookie 机制经历四次握手，后两次允许携带用户数据以便偶联的快速建立。偶联建立的消息序列在第五节中描述。

接到 SCTP 用户请求时，SCTP 对激活偶联进行正常关闭(见第 10 节的 SHUTDOWN 原语描述)。SCTP 也允许异常关闭偶联，在接到 SCTP 用户请求(ABORT 原语)或检测到 SCTP 层的内部错误时。第 9 节描述了正常和异常关闭偶联的过程。

SCTP 不支持半开（half-open）状态（TCP 支持）————一端可以继续发送数据，尽

管另一端已经关闭。当任一端点执行正常关闭（shutdown）时，两端上的偶联都将停止接收来自用户的新数据，而仅仅递交正常关闭时已在队列中的数据（见第 9 节）。

1.3.2 流内消息地按序递交

SCTP 中术语"流" 表示把用户消息依次递交给 ULP（upper-layer protocol）的顺序，而在 TCP 中术语"流"指字节的顺序。

SCTP 用户在偶联建立时能指定偶联支持流的数目，此数目要和远端（对端）进行协商(见 5.1.1)。用户消息关联于流号（流的标识）(见第 10 节 SEND, RECEIVE 原语)，SCTP 内部给 SCTP 用户传给流的每一消息都分配一个流内的序列号。在接受端，SCTP 确保流内的消息按序递交给 SCTP 用户。这样，尽管一个流可能被阻塞，但其它流的消息递交仍可继续。

SCTP 提供了一种绕开按需递交的机制，按此机制发送的用户消息在到达对端时即被递交给 SCTP 用户。

1.3.3 用户数据分段

在需要的时候，SCTP 对用户消息进行分段以确保传递给底层的 SCTP 包符合通路 MTU 的要求。接受时分段被重新组装，进而递交给 SCTP 用户。

1.3.4 消息证实和避免拥塞

SCTP 为每一个用户消息（分段的或未分段的）分配一个传输的序列号 TSN，它于流级别上的序列号无关。接收端证实所有接收到的 TSNs，即便存在消息遗漏（gaps）。通过这种方式，可靠的消息传输在功能上独立于流内的按序递交。

当适时的证实消息没有被收到时，消息证实和避免拥塞功能负责 SCTP 包的重传。包的重传受避免拥塞过程（类似于 TCP 中采用的）的制约。此功能的详细描述见第 6 节和第 7 节。

1.3.5 数据块捆绑

SCTP 包由一个公共头和一个或多个数据块组成（见第 3 节）。每个数据块可以包含用户数据，也可以包含 SCTP 控制信息。SCTP 用户可选的可以请求把多个用户消息捆绑到一个 SCTP 包中。SCTP 的数据块捆绑功能负责 SCTP 包的装配和分解。

During times of congestion an SCTP implementation MAY still perform bundling even if the user has requested that SCTP not bundle. The user's disabling of bundling only affects SCTP implementations that may delay a small period of time before transmission (to attempt to encourage bundling). When the user layer disables bundling, this small delay is prohibited but not bundling that is performed during

congestion or retransmission.

1.3.6 包的有效性验证

SCTP 包的公共头包含两个必备的字段：验证标签字段和 32 位的校验字段（Adler-32 校验的描述见附录 B）。验证标签字段的值在偶联建立时由各个端点分别选定。带有无效验证值的包被丢弃，以防止匿名攻击和来自原来偶联的过期 SCTP 包。Adler-32 校验值应该被 SCTP 包的发送者提供以防止数据在网络中被破坏，接收者丢弃带有无效 Adler-32 校验值的 SCTP 包。

1.3.7 通路管理

通过第 10 节描述的原语，SCTP 用户（sending）能管理（manipulate）用作 SCTP 包目的地的传输地址集合。SCTP 通路管理功能可以根据 SCTP 用户的指令和符合条件的目的地址集合的当前可达性状态，为每个需要发送的 SCTP 包选择一个目的传输地址。通路管理功能通过心跳（heartbeats）监视目的地址的可达性，仅当其它包的交互不足以提供此信息时；并在任何远端传输地址的可达性改变时，警告 SCTP 用户。通路管理功能也负责在偶联建立过程中向远端报告符合条件的本地传输地址集合及向 SCTP 用户报告从远端返回的传输地址集合。

在偶联建立时，每一个 SCTP 端点的主通路（primary path）被定义，被用于 SCTP 包的正常发送。

在接受端，通路管理负责在对输入（inbound）SCTP 包作进一步处理之前，验证其所属的偶联是否存在。

注意：通路管理和包验证同时进行，尽管上面分别进行了描述，但事实上它们不可能被分开执行。

1.4 主要术语

- Active destination transport address: A transport address on a peer endpoint which a transmitting endpoint considers available for receiving user messages.

- Bundling: An optional multiplexing operation, whereby more than one user message may be carried in the same SCTP packet. Each user message occupies its own DATA chunk.

- 块（数据块）：SCTP 包内的一个信息单元，由块头和块类型相关的内容组成。

- Congestion Window (cwnd): An SCTP variable that limits the data, in number of bytes, a sender can send to a particular destination transport address before receiving an acknowledgement.

- Cumulative TSN Ack Point: The TSN of the last DATA chunk acknowledged via the Cumulative TSN Ack field of a SACK.

- Idle destination address: An address that has not had user messages sent to it within some length of time, normally the HEARTBEAT interval or greater.

- Inactive destination transport address: An address which is considered inactive due to errors and unavailable to transport user messages.

- 消息（用户消息）： 由ULP提交的SCTP数据。

- Message Authentication Code (MAC): An integrity check mechanism based on cryptographic hash functions using a secret key. Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties. In SCTP it is used by an endpoint to validate the State Cookie information that is returned from the peer in the COOKIE ECHO chunk. The term "MAC" has different meanings in different contexts. SCTP uses this term with the same meaning as in [RFC2104].

- 网络字节顺序 : 最重要的字节优先, a.k.a., Big Endian.

- Ordered Message: A user message that is delivered in order with respect to all previous user messages sent within the stream the message was sent on.

- Outstanding TSN (at an SCTP endpoint): A TSN (and the associated DATA chunk) that has been sent by the endpoint but for which it has not yet received an acknowledgement.

- Path: The route taken by the SCTP packets sent by one SCTP endpoint to a specific destination transport address of its peer SCTP endpoint. Sending to different destination transport addresses does not necessarily guarantee getting separate paths.

- Primary Path: The primary path is the destination and source address that will be put into a packet outbound to the peer endpoint by default. The definition includes the source address since an implementation MAY wish to specify both destination and source address to better control the return path taken by reply

chunks and on which interface the packet is transmitted when the data sender is multi-homed.

- Receiver Window (rwnd): An SCTP variable a data sender uses to store the most recently calculated receiver window of its peer, in number of bytes. This gives the sender an indication of the space available in the receiver's inbound buffer.

- SCTP association: A protocol relationship between SCTP endpoints, composed of the two SCTP endpoints and protocol state information including Verification Tags and the currently active set of Transmission Sequence Numbers (TSNs), etc. An association can be uniquely identified by the transport addresses used by the endpoints in the association. Two SCTP endpoints MUST NOT have more than one SCTP association between them at any given time.

- SCTP endpoint: The logical sender/receiver of SCTP packets. On a multi-homed host, an SCTP endpoint is represented to its peers as a combination of a set of eligible destination transport addresses to which SCTP packets can be sent and a set of eligible source transport addresses from which SCTP packets can be received. All transport addresses used by an SCTP endpoint must use the same port number, but can use multiple IP addresses. A transport address used by an SCTP endpoint must not be used by another SCTP endpoint. In other words, a transport address is unique to an SCTP endpoint.

- SCTP packet (or packet): The unit of data delivery across the interface between SCTP and the connectionless packet network (e.g., IP). An SCTP packet includes the common SCTP header, possible SCTP control chunks, and user data encapsulated within SCTP DATA chunks.

- SCTP user application (SCTP user): The logical higher-layer application entity which uses the services of SCTP, also called the Upper-layer Protocol (ULP).

- Slow Start Threshold (ssthresh): An SCTP variable. This is the threshold which the endpoint will use to determine whether to perform slow start or congestion avoidance on a particular destination transport address. Ssthresh is in number of bytes.

- Stream: A uni-directional logical channel established from one to another associated SCTP endpoint, within which all user messages

are delivered in sequence except for those submitted to the unordered delivery service.

Note: The relationship between stream numbers in opposite directions is strictly a matter of how the applications use them. It is the responsibility of the SCTP user to create and manage these correlations if they are so desired.

- Stream Sequence Number: A 16-bit sequence number used internally by SCTP to assure sequenced delivery of the user messages within a given stream. One stream sequence number is attached to each user message.

- Tie-Tags: Verification Tags from a previous association. These Tags are used within a State Cookie so that the newly restarting association can be linked to the original association within the endpoint that did not restart.

- Transmission Control Block (TCB): An internal data structure created by an SCTP endpoint for each of its existing SCTP associations to other SCTP endpoints. TCB contains all the status and operational information for the endpoint to maintain and manage the corresponding association.

- Transmission Sequence Number (TSN): A 32-bit sequence number used internally by SCTP. One TSN is attached to each chunk containing user data to permit the receiving SCTP endpoint to acknowledge its receipt and detect duplicate deliveries.

- Transport address: A Transport Address is traditionally defined by Network Layer address, Transport Layer protocol and Transport Layer port number. In the case of SCTP running over IP, a transport address is defined by the combination of an IP address and an SCTP port number (where SCTP is the Transport protocol).

- Unacknowledged TSN (at an SCTP endpoint): A TSN (and the associated DATA chunk) which has been received by the endpoint but for which an acknowledgement has not yet been sent. Or in the opposite case, for a packet that has been sent but no acknowledgement has been received.

- Unordered Message: Unordered messages are "unordered" with respect to any other message, this includes both other unordered messages as well as other ordered messages. Unordered message might be

delivered prior to or later than ordered messages sent on the same stream.

- o User message: The unit of data delivery across the interface between SCTP and its user.

- o Verification Tag: A 32 bit unsigned integer that is randomly generated. The Verification Tag provides a key that allows a receiver to verify that the SCTP packet belongs to the current association and is not an old or stale packet from a previous association.

1.5. 缩略语

MAC - Message Authentication Code [RFC2104]

RTO - Retransmission Time-out

RTT - Round-trip Time

RTTVAR - Round-trip Time Variation

SCTP - Stream Control Transmission Protocol

SRTT - Smoothed RTT

TCB - Transmission Control Block

TLV - Type-Length-Value Coding Format

TSN - Transmission Sequence Number

ULP - Upper-layer Protocol

1.6 序列号算法

It is essential to remember that the actual Transmission Sequence Number space is finite, though very large. This space ranges from 0 to 2**32 - 1. Since the space is finite, all arithmetic dealing with Transmission Sequence Numbers must be performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 - 1 to 0 again. There are some subtleties to computer modulo arithmetic, so great care should be taken in programming the comparison of such values. When referring to TSNs,

the symbol "=<" means "less than or equal"(modulo 2**32).

Comparisons and arithmetic on TSNs in this document SHOULD use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 32.

An endpoint SHOULD NOT transmit a DATA chunk with a TSN that is more than 2**31 - 1 above the beginning TSN of its current send window. Doing so will cause problems in comparing TSNs.

Transmission Sequence Numbers wrap around when they reach 2**32 - 1. That is, the next TSN a DATA chunk MUST use after transmitting TSN = 2*32 - 1 is TSN = 0.

Any arithmetic done on Stream Sequence Numbers SHOULD use Serial Number Arithmetic as defined in [RFC1982] where SERIAL_BITS = 16. All other arithmetic and comparisons in this document uses normal arithmetic.

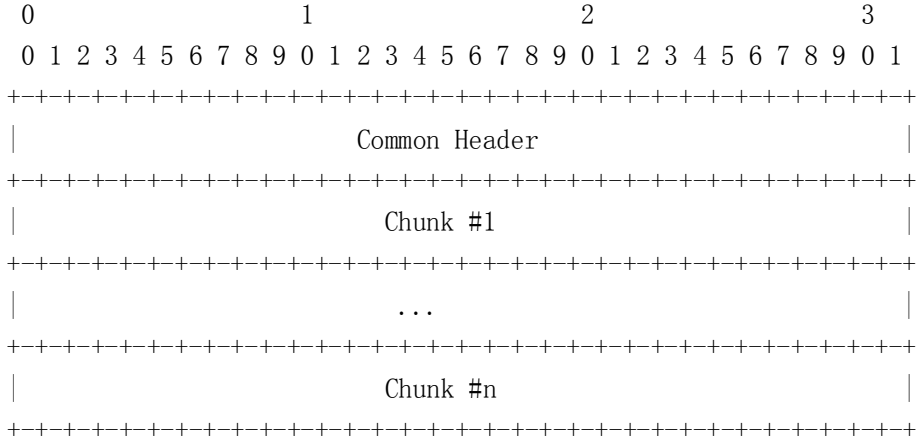
2. Conventions

The keywords MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, NOT RECOMMENDED, MAY, and OPTIONAL, when they appear in this document, are to be interpreted as described in [RFC2119].

3. SCTP 包格式

SCTP 包由一个公共头和一个或多个数据块组成。每个数据块可以包含用户数据，也可以包含 SCTP 控制信息。

SCTP 包格式显示如下:



多个块能被捆绑到一个 SCTP 包中（受限于通路 MTU 的大小），但 INIT,INIT ACK 和 SHUTDOWN COMPLETE 块除外。这些块不能和其它块捆绑在一个 SCTP 包内。数据块捆绑的更多细节见 6.10。

如果一个用户消息不能放进一个 SCTP 包中，用 6.9 定义的过程对消息进行分段放入多个块内。

SCTP 包的所有整数字段必须用网络字节顺序进行传递，除非另有说明。

3.1 SCTP 公共头字段描述

SCTP 公共头格式

源端口号: 16 bits (unsigned integer)

SCTP 发送者的端口号，能被接收者用来确定此 SCTP 包所属的偶联————结合源 IP 地址，SCTP 目的端口号及可能的目的 IP 地址。

目的端口号: 16 bits (unsigned integer)

SCTP 包被发往的 SCTP 端口号。接收主机用此端口号分发 SCTP 包到正确的接收端点（应用）。

验证标签: 32 bits (unsigned integer)

接收者用验证标签验证发送者的有效性。传输时，验证标签必须被填为偶联初始化时从对端接收到 Initiate 标签的值，下列情况例外：

- 包含 INIT 块的 SCTP 包必须将验证标签字段置零。
- 包含 T-比特被设置的 SHUTDOWN-COMPLETE 块的 SCTP 包的验证标签必须从包含 SHUTDOWN-ACK 块的 SCTP 包中拷贝。
- 包含 ABORT 块的 SCTP 包的验证标签可以从造成 ABORT 块被发送的 SCTP 包中拷贝。细节见 8.4 和 8.5。

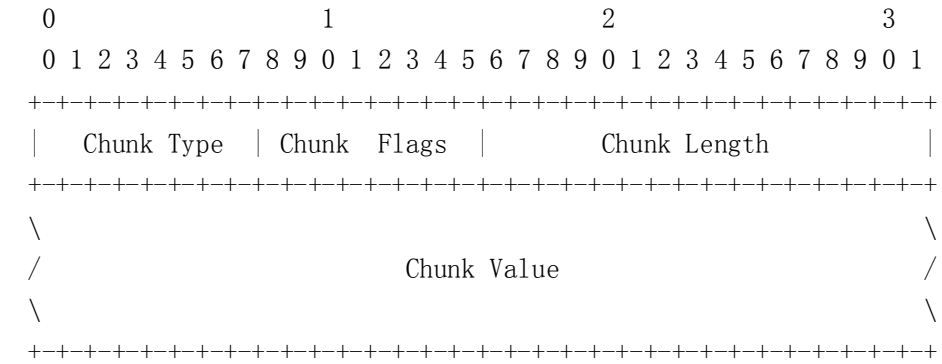
INIT 块必须是携带它的 SCTP 包内的唯一的块。

校验值: 32 bits (unsigned integer)

此字段包含 SCTP 包的校验值。它的计算在 6.8 中描述。SCTP 用附录 B 中描述的 Adler-32 算法计算校验值。

3.2 SCTP 数据块字段描述

下图是在 SCTP 包中被传送的块的字段格式。每个块具有一个块类型字段，块标记字段（与块类型有关），块长度字段和块值字段。



块类型: 8 bits (unsigned integer)

指定了包含在块值（Chunk Value）字段的信息的类型。取值范围为 0 到 254。255 被预留以备将来扩充时用。

块类性的值被定义如下:

ID 值	块类型
0	Payload Data (DATA)
1	Initiation (INIT)
2	Initiation Acknowledgement (INIT ACK)
3	Selective Acknowledgement (SACK)
4	Heartbeat Request (HEARTBEAT)
5	Heartbeat Acknowledgement (HEARTBEAT ACK)
6	Abort (ABORT)
7	Shutdown (SHUTDOWN)
8	Shutdown Acknowledgement (SHUTDOWN ACK)
9	Operation Error (ERROR)
10	State Cookie (COOKIE ECHO)
11	Cookie Acknowledgement (COOKIE ACK)
12	Reserved for Explicit Congestion Notification Echo (ECNE)

13   - Reserved for Congestion Window Reduced (CWR)
14   - Shutdown Complete (SHUTDOWN COMPLETE)
15 to 62   - reserved by IETF
63   - IETF-defined Chunk Extensions
64 to 126   - reserved by IETF
127   - IETF-defined Chunk Extensions
128 to 190 - reserved by IETF
191   - IETF-defined Chunk Extensions
192 to 254 - reserved by IETF
255   - IETF-defined Chunk Extensions

块类型的最高两个比特指定了正在处理的端点不识别块类型时必须（要）采取的行为。

00 - 停止处理并丢弃此 SCTP 包，也不处理包内的其他块。

01 - 停止处理并丢弃此 SCTP 包，也不处理包内的其他块，并用块 ERROR 或块 INIT ACK（'Unrecognized Parameter Type'）向对端报告。

10 - 跳过此块，继续处理。

11 - 跳过此块，继续处理，并用 ERROR 块（'Unrecognized Chunk Type'）向对端报告。

注意：块 ECNE 和块 CWR 类型被预留以备将来明确拥塞通知 ECN（Explicit Congestion Notification）时使用。

块标记：8 bits

这些比特的用法取决于块类型。除非特别指明，否则传输时它们被置零，接收时被忽略。

块长度：16 bits (unsigned integer)

表示块的长度（字节的个数），包含块类型，块标记，块长度和块值字段。因此，如果块值字段为零长度（块值字段不存在），则长度字段的值为 4。块长度字段不包括任何填充的字节。

块值：可变长度

块值字段包含了在块中将要传递的实际信息，其用法和格式取决于块类型。

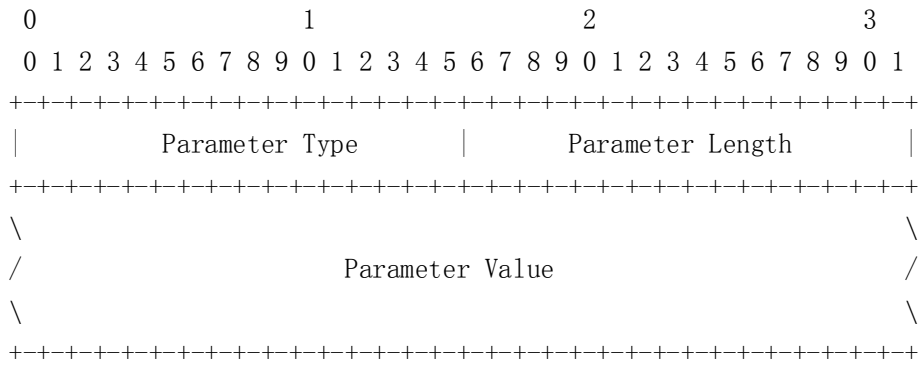
块的总长度(包含 Type, Length 和 Value 字段)必须是 4 的倍数，否则发送者必须用零字节进行填充，并且填充的字节不计入块长度字段。发送者从不会填充

多于 3 个字节，接收者忽略被填充的字节。

SCTP 定义的块在 3.3 中详细描述。The guidelines for IETF-defined chunk extensions can be found in Section 13.1 of this document.

3.2.1 可选/可变长度的参数格式

SCTP 控制块的块值字段由与块类型相关的（依赖于块类型）头部（必备的）及零或多个参数组成。包含在一个块内的可选/可变长度参数的格式被定义如下。



块参数类型： 16 bits (unsigned integer)

参数类型的标识符，取值范围为 0 到 65534。值 65535 被预留以备 IETF 定义扩展时使用，其它值（未出现在 SCTP 块描述中）被 IETF 预留。

块参数长度： 16 bits (unsigned integer)

参数的字节个数，包括参数类型，参数长度和参数值字段。具有零长度参数值字段（参数值字段不存在）的参数的长度字段将有值 4，参数长度包含任何填充的字节。

块参数值: 可变长度.

参数值字段包含了在参数中将要传递的实际信息。

参数的总长度(包含 Type, Length 和 Value 字段)必须是 4 的倍数，否则发送者必须在结尾（参数值字段之后）用零字节进行填充，并且填充的字节不计入参数长度字段。发送者从不会填充多于 3 个字节，接收者忽略被填充的字节。

参数类型的最高两个比特指定了正在处理的端点不识别参数类型时必须（要）采取的行为。

00 - 停止处理并丢弃此 SCTP 包，也不处理包内的其他块。

01 - 停止处理并丢弃此 SCTP 包，也不处理包内的其他块，并用块 ERROR 或块 INIT ACK（'Unrecognized Parameter Type'）向对端报告。

10 - 跳过此参数，继续处理。

11 - 跳过此参数，继续处理，并用块 ERROR 或块 INIT ACK（'Unrecognized Parameter Type'）向对端报告。

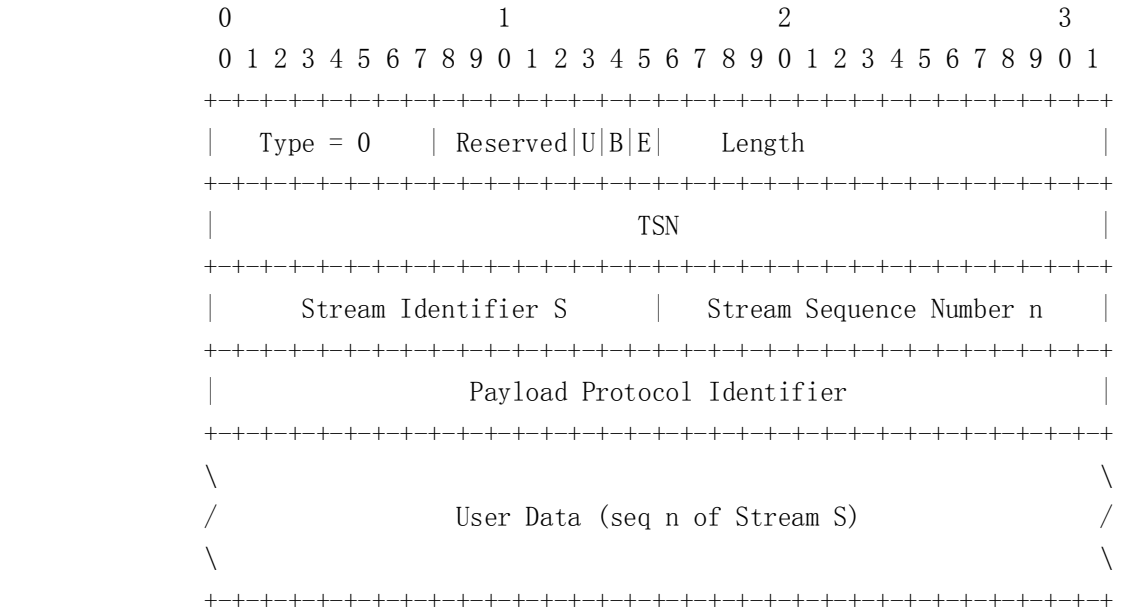
具体的 SCTP 参数被定义在相关 SCTP 块的描述部分。IETF 定义的参数扩展规则见 13.2。

3.3 SCTP 数据块定义

这一部分定义了各种 SCTP 块类型的格式。

3.3.1 Payload Data (DATA) (0)

The following format MUST be used for the DATA chunk:



Reserved: 5 bits

应该被置零，接收者忽略。

U bit: 1 bit

无序(U)nordered 消息指示符。被置'1'表示这是一个无序的 DATA 块，没有 SSN（流序列号）被分配给此 DATA 块，接收者忽略 SSN 字段。

重新组装后（如果必要的话），无序 DATA 块无须再进行重新排序，被接收者

直接分发给上层ULP。

如果无序（用户）消息被分段， 则每一片段(fragment)的U比特都得置'1'。

B bit: 1 bit

开始(B)eginning片段指示符，置'1'表示用户消息的第一个片段。

E bit: 1 bit

结尾(E)nding片段指示符，置'1'表示用户消息的最后一个片段。

不分段用户消息应把B比特和E比特都置为'1'。B比特和E比特都置为'0'表示多个片段用户消息中的一个中间片段, 总结如下表:

B	E	描述
1	0	分段用户消息的第一个片段
0	0	分段用户消息的中间片段
0	1	分段用户消息的最后一个片段
1	1	不分段用户消息

表 1: 分段描述标记

当一个（用户）消息被分为多个片段（块）进行传送时, 接收者用TSNs（传输序列号）来重新组装消息。因此为分段用户消息的每一个片断所用的TSNs必须严格有序（strictly sequential）。

长度: 16 bits (unsigned integer)

This field indicates the length of the DATA chunk in bytes from the beginning of the type field to the end of the user data field excluding any padding. A DATA chunk with no user data field will have Length set to 16 (indicating 16 bytes).

TSN : 32 bits (unsigned integer)

块DATA的传输序列号，有效范围为0到4294967295 (2**32 - 1)。TSN wraps back to 0 after reaching 4294967295.

流标识符 S: 16 bits (unsigned integer)

标识用户数据所属的流。

流序列号 n: 16 bits (unsigned integer)

用户数据在流 S 内的序列号，有效范围为 0 到 65535。

当一个用户消息被 SCTP 分段传输时，每一个片断具有相同的流序列号 SSN。

净荷协议标识符: 32 bits (unsigned integer)

This value represents an application (or upper layer) specified protocol identifier. This value is passed to SCTP by its upper layer and sent to its peer. This identifier is not used by SCTP but can be used by certain network entities as well as the peer application to identify the type of information being carried in this DATA chunk. This field must be sent even in fragmented DATA chunks (to make sure it is available for agents in the middle of the network).

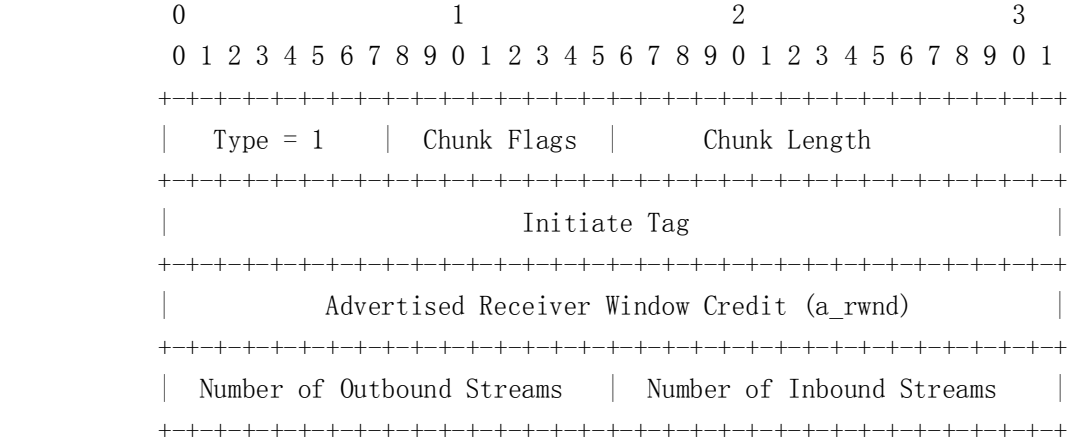
The value 0 indicates no application identifier is specified by the upper layer for this payload data.

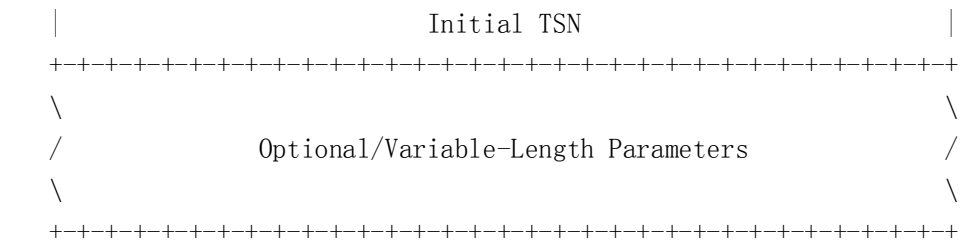
用户数据: 可变长度

字段应为 4 的倍数，否则实现必须在结尾用零字节填充，但填充的字节不计入常独字段。发送者从不会填充多于 3 个字节。

3.3.2 Initiation (INIT) (1)

用来在两个端点间发起一个 SCTP 偶联，格式如下:





INIT 块包含下列参数，除非另有说明，否则 INIT 内每一个参数只能被包含一次。

Fixed Parameters	Status
Initiate Tag	Mandatory
Advertised Receiver Window Credit	Mandatory
Number of Outbound Streams	Mandatory
Number of Inbound Streams	Mandatory
Initial TSN	Mandatory

Variable Parameters	Status	Type Value
IPv4 Address (Note 1)	Optional	5
IPv6 Address (Note 1)	Optional	6
Cookie Preservative	Optional	9
Reserved for ECN Capable (Note 2)	Optional	32768 (0x8000)
Host Name Address (Note 3)	Optional	11
Supported Address Types (Note 4)	Optional	12

注意 1: 块 INIT 能包含多个地址，可以是 IPv4 和/或 IPv6 的任意组合。

注意 2: The ECN capable field is reserved for future use of Explicit Congestion Notification.

注意 3: 块 INIT 千万不要包含多于一个的主机名地址参数。而且 INIT 的发送者也不要把 INIT 中的任何其它地址类型和主机名地址关联起来。INIT 的接收者必须忽略任何其它地址类型，如果主机名地址参数在被接收的块 INIT 内出现的话。

注意 4: 此参数，如果出现，指定了发送端点能支持的所有地址类型。不提供此参数表明发送端点能支持任何地址类型。

块 INIT 内的块标记字段被预留，发送者将所有比特置零，接收者忽略所有比特。The sequence of parameters within an INIT can be processed in any order.

Initiate 标签: 32 bits (unsigned integer)

块 INIT 的接收者(the responding end)必须记录 Initiate 标签参数的值。在偶

联内（如果偶联建立成功），此接收者必须把此值填入到它所发送的所有 SCTP 包的验证标签字段。

Initiate 标签允许有除零外的任何值，选择此标签值的更多细节见 5.3.1。

如果被接收到的 INIT 块的 Initiate 标签的值为零，接收者必须视为一个错误，关闭偶联并发送 ABORT。

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer)

This value represents the dedicated buffer space, in number of bytes, the sender of the INIT has reserved in association with this window. During the life of the association this buffer space SHOULD not be lessened (i.e. dedicated buffers taken away from this association); 然而，端点可以在 SACK 块中改变它所发送的 a_rwnd 的值。

输出流(OS)个数 : 16 bits (unsigned integer)

块 INIT 的发送者希望在此偶联中创建的输出流个数，值 0 千万不要被用。

注意: OS 被置 0 的块 INIT 的接收者应该放弃（异常关闭 abort）偶联。

输入流(MIS)个数 : 16 bits (unsigned integer)

块 INIT 的发送者允许对端在此偶联中创建的输入流的最大个数，值 0 千万不要被用。

注意: 两个端点不就偶联中支持流（输入和输出）的实际个数进行协商，而是用 min（requested, offered)。细节见 5.1.1。

注意: MIS 被置 0 的块 INIT 的接收者应该放弃（异常关闭 abort）偶联。

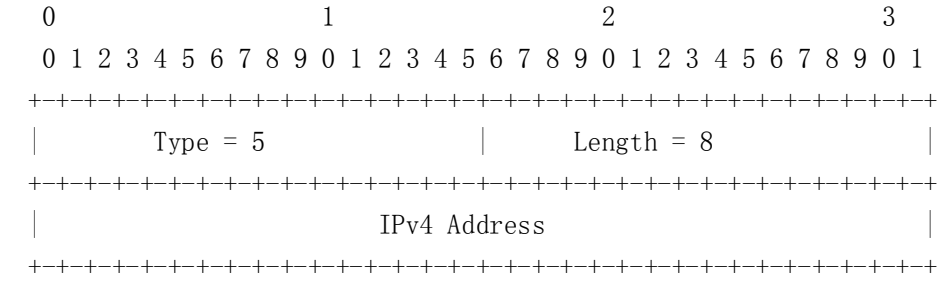
初始 TSN(I-TSN) : 32 bits (unsigned integer)

发送者将要使用的初始 TSN，有效范围为 0 到 4294967295，可被填为 Initiate 标签字段的值。

3.3.2.1 INIT 中的可选/可变长度参数

下列参数遵循 3.2.1 中定义的 Type-Length-Value 格式。任何 TLV 字段必须跟在前一部分中定义的固定长度字段的后面。

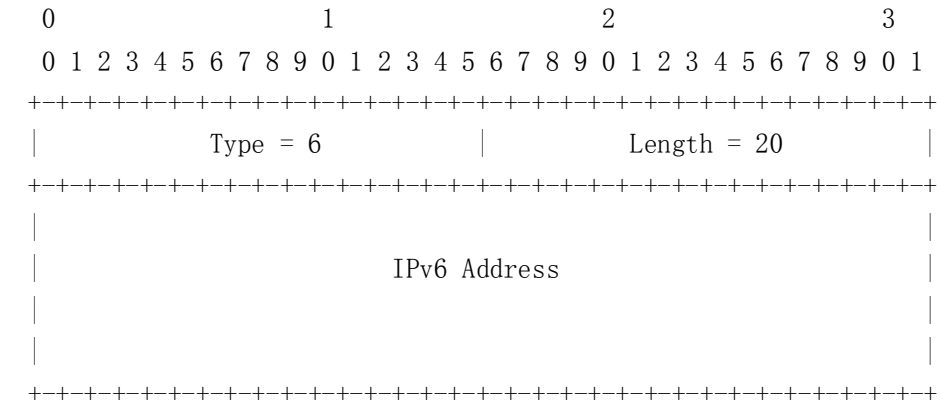
IPv4 Address Parameter (5)



IPv4 Address: 32 bits (unsigned integer)

发送端点的 IPv4 地址，It is binary encoded。

IPv6 Address Parameter (6)



IPv6 Address: 128 bit (unsigned integer)

发送端点的 IPv6 地址，It is binary encoded。

注意: A sender MUST NOT use an IPv4-mapped IPv6 address [RFC2373] but should instead use an IPv4 Address Parameter for an IPv4 address.

Combined with the Source Port Number in the SCTP common header, the value passed in an IPv4 or IPv6 Address parameter indicates a transport address the sender of the INIT will support for the association being initiated. That is, during the lifetime of this association, this IP address can appear in the source address field of an IP datagram sent from the sender of the INIT, and can be used as a destination address of an IP datagram sent from the receiver of the INIT.

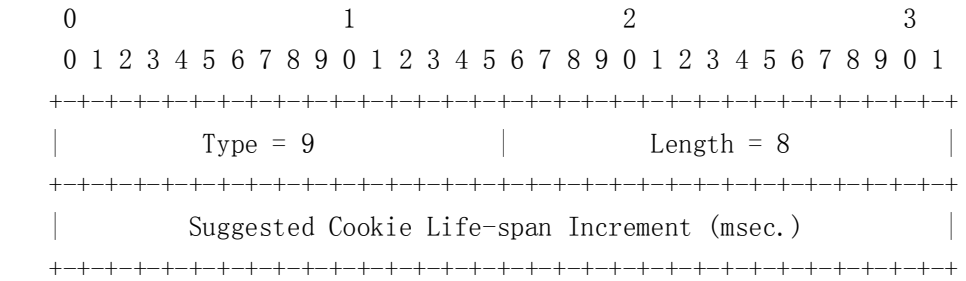
More than one IP Address parameter can be included in an INIT chunk when the INIT sender is multi-homed. Moreover, a multi-homed endpoint may have access to different types of network, thus more than one address type can be present in one INIT chunk, i.e., IPv4 and IPv6 addresses are allowed in the same INIT chunk.

If the INIT contains at least one IP Address parameter, then the source address of the IP datagram containing the INIT chunk and any additional address(es) provided within the INIT can be used as destinations by the endpoint receiving the INIT. If the INIT does not contain any IP Address parameters, the endpoint receiving the INIT MUST use the source address associated with the received IP datagram as its sole destination address for the association.

Note that not using any IP address parameters in the INIT and INIT-ACK is an alternative to make an association more likely to work across a NAT box.

Cookie Preservative (9)

块 INIT 的发送者将会用这个参数建议（suggest）块 INIT 的接收者延长 Cookie 状态（the State Cookie）的寿命。



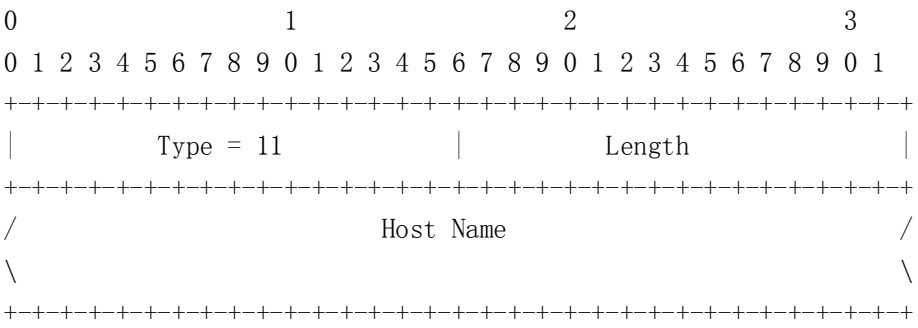
Suggested Cookie Life-span Increment: 32 bits (unsigned integer)

指定发送者希望接收者在其缺省 cookie 寿命基础上新增的额外增量（单位毫秒）。

当重新试图和对端建立偶联时，这个可选参数应该被发送者加入到 INIT 块中。This optional parameter should be added to the INIT chunk by the sender when it re-attempts establishing an association with a peer to which its previous attempt of establishing the association failed due to a stale cookie operation error. 出于自省安全的考虑，接收者可以选择忽略被建议的 cookie 寿命增量。

Host Name Address (11)

块 INIT 的发送者用这个参数把它的主机名（Host Name）传给对端(代替 IP 地址)。对端负责解析主机名。Using this parameter might make it more likely for the association to work across a NAT box.



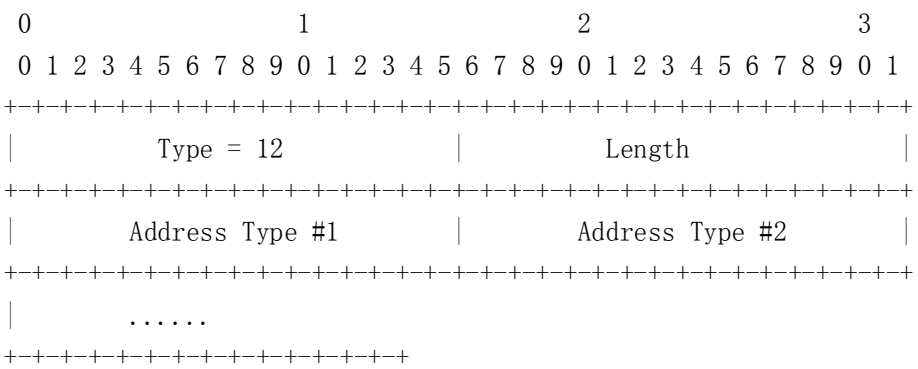
Host Name: 可变长度

This field contains a host name in "host name syntax" per RFC1123 Section 2.1 [RFC1123]. 解析主机名的方法不属于 SCTP 的范畴。

Note: At least one null terminator is included in the Host Name string and must be included in the length.

Supported Address Types (12)

块 INIT 的发送者用这个参数列举它所能支持的所有地址类型。



Address Type: 16 bits (unsigned integer)

用相应的地址 TLV 的类型值来填充(如, IPv4 = 5, IPv6 = 6, Hostname = 11)。

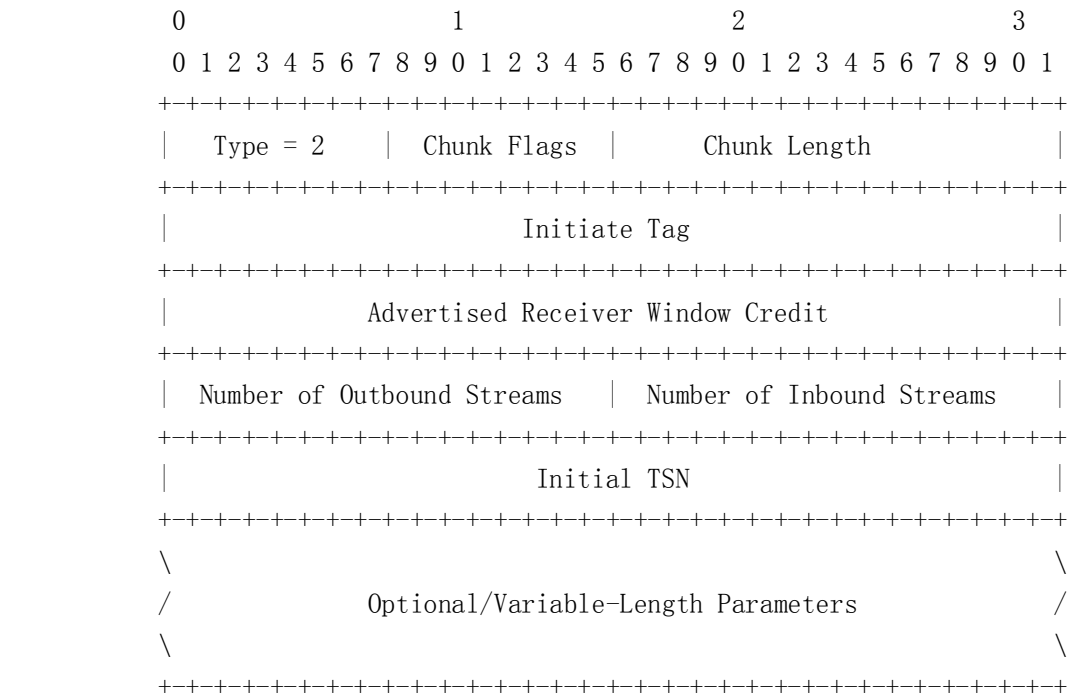
3.3.3 Initiation Acknowledgement (INIT ACK) (2):

块 INIT ACK 用来证实的 SCTP 偶联的初始化（INIT）.

块 INIT ACK 参数部分的格式类似于块 INIT。它有两个额外的可变参数: The State

Cookie 和 the Unrecognized Parameter:

块 INIT ACK 的格式如下:



Initiate Tag: 32 bits (unsigned integer)

块 INIT ACK 的接收者记录 Initiate 标签参数的值。在偶联内（如果偶联建立成功），此接收者必须把此值填入到它所发送的所有 SCTP 包的验证标签字段。

Initiate 标签允许有除零外的任何值，选择此标签值的更多细节见 5.3.1。

如果被接收到的 INIT 块的 Initiate 标签的值为零，接收者必须视为一个错误，关闭偶联并发送 ABORT。

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer)

This value represents the dedicated buffer space, in number of bytes, the sender of the INIT ACK has reserved in association with this window. During the life of the association this buffer space SHOULD not be lessened (i.e. dedicated buffers taken away from this association).

输出流(OS)个数 : 16 bits (unsigned integer)

块 INIT ACK 的发送者希望在此偶联中创建的输出流个数，值 0 千万不要被用。

注意: OS 被置 0 的块 INIT ACK 的接收者应该销毁偶联，放弃其 TCB。

输入流(MIS)个数 : 16 bits (unsigned integer)

块 INIT ACK 的发送者允许对端在此偶联中创建的输入流的最大个数，值 0 千万不要被用。

注意: 两个端点不就偶联中支持流（输入和输出）的实际个数进行协商，而是用 min（requested, offered)。细节见 5.1.1。

注意: MIS 被置 0 的块 INIT 的接收者应该销毁偶联，放弃其 TCB。

Initial TSN (I-TSN) : 32 bits (unsigned integer)

INIT-ACK 发送者将要使用的初始 TSN，有效范围为 0 到 4294967295，可被填为 Initiate 标签字段的值。

Fixed Parameters	Status
Initiate Tag	Mandatory
Advertised Receiver Window Credit	Mandatory
Number of Outbound Streams	Mandatory
Number of Inbound Streams	Mandatory
Initial TSN	Mandatory

Variable Parameters	Status	Type Value
State Cookie	Mandatory	7
IPv4 Address (Note 1)	Optional	5
IPv6 Address (Note 1)	Optional	6
Unrecognized Parameters	Optional	8
Reserved for ECN Capable (Note 2)	Optional	32768 (0x8000)
Host Name Address (Note 3)	Optional	11

注意 1: 块 INIT ACK 能包含多个 IP 地址参数，可以是 IPv4 和/或 IPv6 的任意组合。

注意 2: The ECN capable field is reserved for future use of Explicit Congestion Notification.

注意 3: 块 INIT ACK 千万不要包含多于一个的主机名地址参数。而且 INIT ACK 的发送者也不要把 INIT ACK 中的任何其它地址类型和主机名地址关联起来。INIT ACK 的接收者必须忽略任何其它地址类型，如果主机名地址参数在被接收的块 INIT ACK 内出现的话。

实现注意: 由于可变大小的 cookie 状态及可变的地址列表，实现必须准备接受一个相当长（大于 1500 字节）的 INIT ACK 块。例如，如果 INIT 的一个响应者（responder）希望发送 1000 个 IPv4 地址，在 INIT ACK 块中将至少需要 8000 字节来编码。

In combination with the Source Port carried in the SCTP common header, each IP Address parameter in the INIT ACK indicates to the receiver of the INIT ACK a valid transport address supported by the sender of the INIT ACK for the lifetime of the association being initiated.

If the INIT ACK contains at least one IP Address parameter, then the source address of the IP datagram containing the INIT ACK and any additional address(es) provided within the INIT ACK may be used as destinations by the receiver of the INIT-ACK. If the INIT ACK does not contain any IP Address parameters, the receiver of the INIT-ACK MUST use the source address associated with the received IP datagram as its sole destination address for the association.

参数 State Cookie 和 Unrecognized Parameters 使用定义在 3.2.1 的 TLV 格式，它们在下面被描述。其它字段同于 INIT 块中的相应部分。

3.3.3.1 可选/可变长度参数

State Cookie

Parameter Type Value: 7

Parameter Length: variable size, depending on Size of Cookie

Parameter Value:

对 INIT ACK 的发送者来说，此参数值必须包含创建偶联所必要的所有状态及参数信息，连同一个消息验证码(MAC)。State Cookie 定义的细节见 5.1.3。

Unrecognized Parameters:

Parameter Type Value: 8

Parameter Length: Variable Size.

Parameter Value:

This parameter is returned to the originator of the INIT chunk when the INIT contains an unrecognized parameter which has a value that indicates that it should be reported to the sender.
参数值字段将包含收到的 INIT 块中不识别参数的拷贝（包括 T,L,V 字段）

3.3.4 Selective Acknowledgement (SACK) (3):

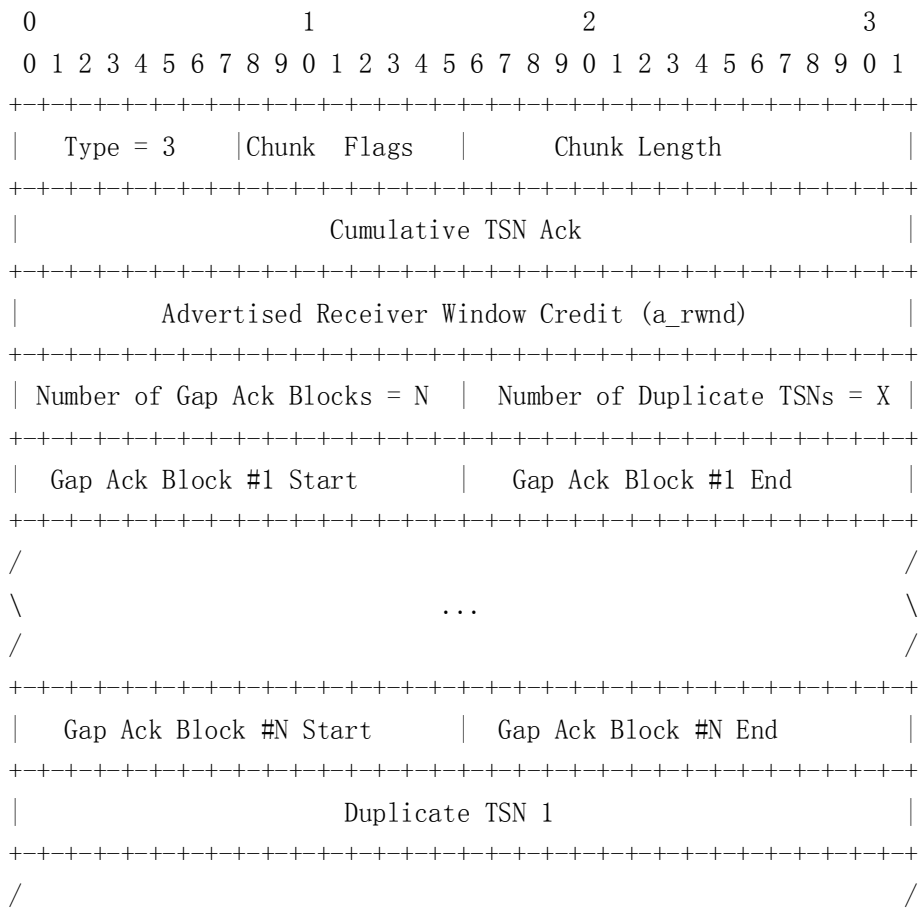
用来向对端证实已收到的 DATA 块及在收到的 DATA 块序列中的 gaps（用 TSNs 表示）。

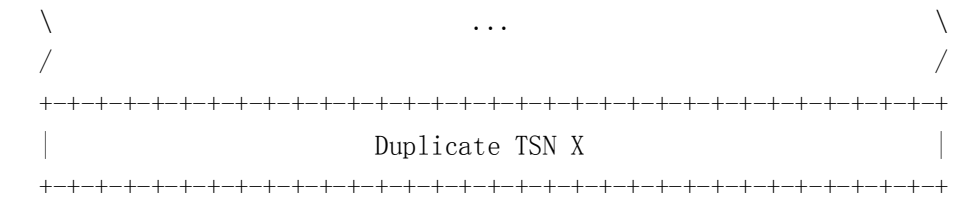
SACK 必须包含参数 Cumulative TSN Ack 及 a_rwnd。

根据定义，参数 Cumulative TSN Ack 的值为：在接收到的 TSNs 序列出现中断之前的那个 TSN；下一个 TSN 所标识的 SCTP 包还没有被发送此 SACK 块的端点收到。此参数对所有小于或等于此值的 TSN 进行了证实。

SACK 接收者对 a_rwnd 处理的细节见 6.2.1.

SACK 也包含零个或多个 Gap Ack Blocks。每一个 Gap Ack Block 对在所有接收到 TSNs 中的一个 TSNs 序列（following a break）进行了证实。根据定义，所有被 Gap Ack Blocks 证实的 TSNs 都大于 Cumulative TSN Ack 的值。





Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

Cumulative TSN Ack: 32 bits (unsigned integer)

一个 TSN，在此 TSN 之前的所有 DATA 块都已收到。

Advertised Receiver Window Credit (a_rwnd): 32 bits (unsigned integer)

用于更新 SACK 发送者的接收缓存空间（单位字节），细节见 6.2.1。

Number of Gap Ack Blocks: 16 bits (unsigned integer)

包含在 SACK 中的 Gap Ack Blocks 的个数。

Number of Duplicate TSNs: 16 bit

端点收到的重复 TSN 个数，每一个重复的 TSN 被列举在 Gap Ack Blocks 的后面。

Gap Ack Blocks:

包含 N 个（定义在 Number of Gap Ack Blocks 字段）Gap Ack Block。每一个 Gap Ack Block 中，所有(Cumulative TSN Ack + Gap Ack Block Start)<= * <=(Cumulative TSN Ack + Gap Ack Block End)的 DATA 块都已被正确的接收到。

Gap Ack Block Start: 16 bits (unsigned integer)

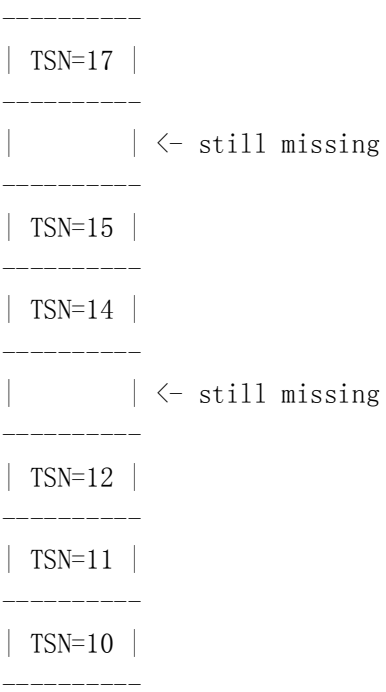
Indicates the Start offset TSN for this Gap Ack Block. To calculate the actual TSN number the Cumulative TSN Ack is added to this offset number. This calculated TSN identifies the first TSN in this Gap Ack Block that has been received.

Gap Ack Block End: 16 bits (unsigned integer)

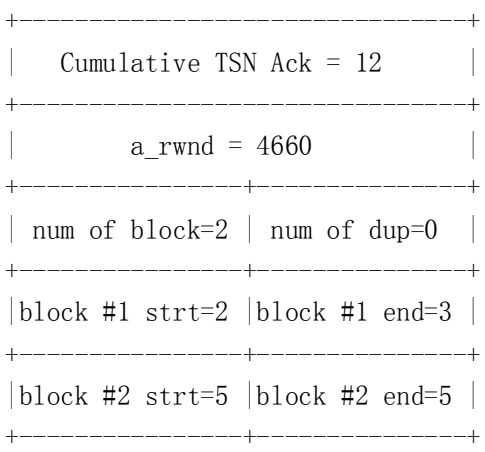
Indicates the End offset TSN for this Gap Ack Block. To calculate the actual TSN number the Cumulative TSN Ack is added to this

offset number. This calculated TSN identifies the TSN of the last DATA chunk received in this Gap Ack Block.

For example, assume the receiver has the following DATA chunks newly arrived at the time when it decides to send a Selective ACK,



then, the parameter part of the SACK MUST be constructed as follows (assuming the new a_rwnd is set to 4660 by the sender):



Duplicate TSN: 32 bits (unsigned integer)

Indicates the number of times a TSN was received in duplicate since the last SACK was sent. Every time a receiver gets a

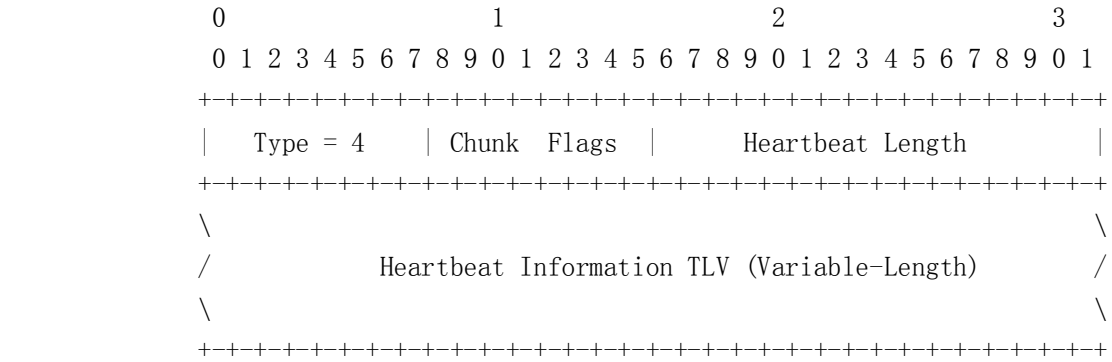
duplicate TSN (before sending the SACK) it adds it to the list of duplicates. 发送一个SACK之后，重复计数器被重新初始化为0。

For example, if a receiver were to get the TSN 19 three times it would list 19 twice in the outbound SACK. After sending the SACK if it received yet one more TSN 19 it would list 19 as a duplicate once in the next outgoing SACK.

3.3.5 Heartbeat Request (HEARTBEAT) (4):

用来向对端探查某一特定目的传输地址（当前的偶联内）的可达性。

块值域包含心跳信息————可变长度，仅被发送者理解的不透明数据结构。



Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

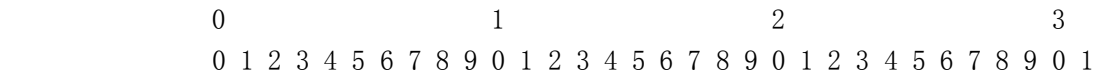
Heartbeat Length: 16 bits (unsigned integer)

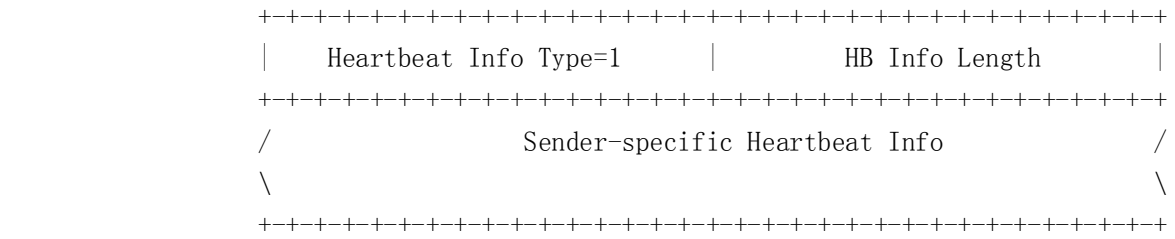
Set to the size of the chunk in bytes, including the chunk header and the Heartbeat Information field.

Heartbeat Information: variable length

Defined as a variable-length parameter using the format described in Section 3.2.1, i.e.:

Variable Parameters	Status	Type Value
Heartbeat Info	Mandatory	1



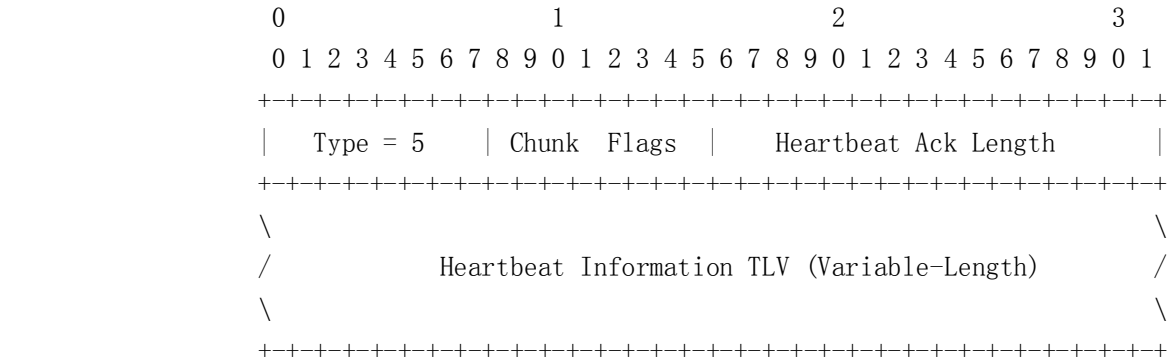


发送者相关的心跳信息通常包括：块 HEARTBEAT 发送时发送者当前时间的信息和块 HEARTBEAT 被发往的目的传输地址（见 8.3）。

3.3.6 Heartbeat Acknowledgement (HEARTBEAT ACK) (5):

作为对 HEARTBEAT 块的响应，端点应该向对端发送 HEARTBEAT ACK（见 8.3）。
A HEARTBEAT ACK is always sent to the source IP address of the IP datagram containing the HEARTBEAT chunk to which this ack is responding.

The parameter field contains a variable length opaque data structure.



Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

Heartbeat Ack Length: 16 bits (unsigned integer)

Set to the size of the chunk in bytes, including the chunk header and the Heartbeat Information field.

Heartbeat Information: variable length

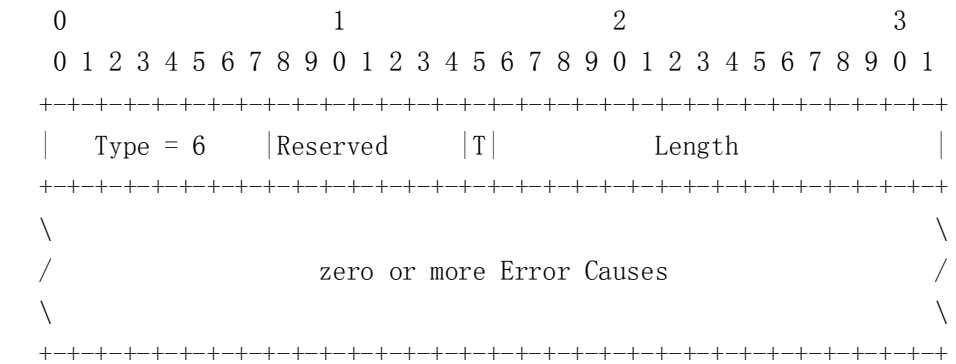
必须包含对应心跳请求的心跳信息参数。
This field MUST contain the Heartbeat Information parameter of the Heartbeat Request to which this Heartbeat Acknowledgement is responding.

Variable Parameters	Status	Type Value
Heartbeat Info	Mandatory	1

3.3.7 Abort Association (ABORT) (6):

用来异常关闭偶联，可以包含参数 Cause 通知接收者放弃的原因。块 DATA 不能和块 ABORT 捆绑在一起。控制块(除 INIT, INIT ACK 和 SHUTDOWN COMPLETE)可以和块 ABORT 捆绑，但在 SCTP 包中它们必须被放置在块 ABORT 的前面，否则将会被接收者忽略。

如果端点接收到一个格式错误或相应的偶联不存在的块 ABORT 时，必须丢弃此 ABORT 块。而且，任何情况下，端点不要对接收到的块 ABORT 进行响应（发送自己的的 ABORT 块）。



Chunk Flags: 8 bits

Reserved: 7 bits

传输时所有比特被置零，接收时忽略所有比特。

T bit: 1 bit

0 表示发送者有一个 TCB，但已被破坏；1 表示发送者没有 TCB。

注意: Special rules apply to this chunk for verification, 细节见 8.5.1。

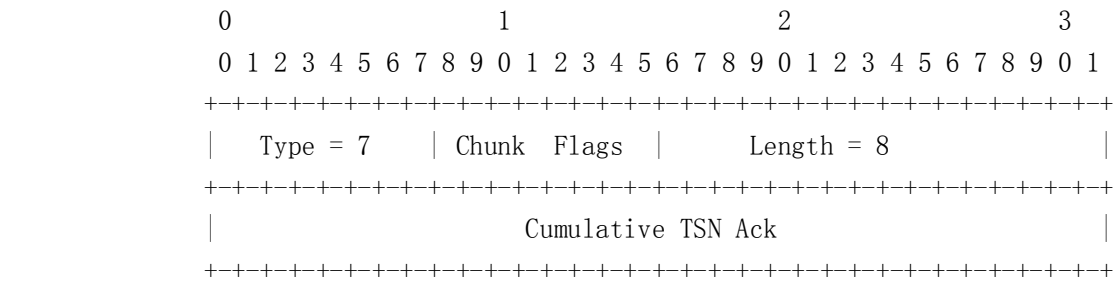
Length: 16 bits (unsigned integer)

Set to the size of the chunk in bytes, including the chunk header and all the Error Cause fields present.

See Section 3.3.10 for Error Cause definitions.

3.3.8 Shutdown Association (SHUTDOWN) (7):

用来正常关闭偶联，格式如下：



Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

Length: 16 bits (unsigned integer)

Indicates the length of the parameter. Set to 8.

Cumulative TSN Ack: 32 bits (unsigned integer)

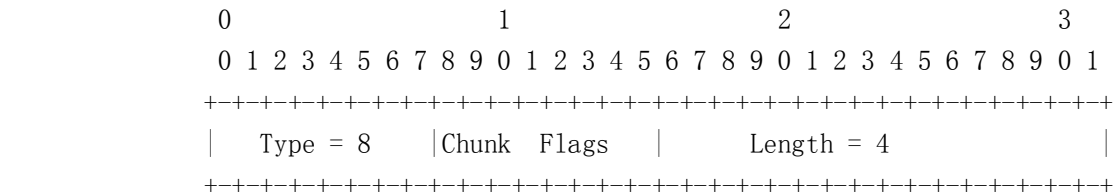
This parameter contains the TSN of the last chunk received in sequence before any gaps.

Note: Since the SHUTDOWN message does not contain Gap Ack Blocks, it cannot be used to acknowledge TSNs received out of order. In a SACK, lack of Gap Ack Blocks that were previously included indicates that the data receiver reneged on the associated DATA chunks. Since SHUTDOWN does not contain Gap Ack Blocks, the receiver of the SHUTDOWN shouldn't interpret the lack of a Gap Ack Block as a renege. (see Section 6.2 for information on reneging)

3.3.9 Shutdown Acknowledgement (SHUTDOWN ACK) (8):

用来在 shutdown 进程完成时，作为对收到 SHUTDOWN 的证实，细节见 9.2。

SHUTDOWN ACK 没有参数。

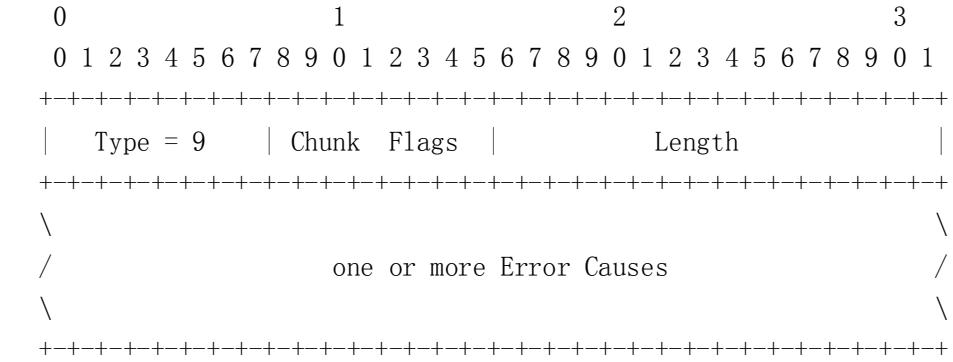


Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

3.3.10 Operation Error (ERROR) (9):

用来向对端报告一定的错误状态（error conditions），包含一个或多个错误原因。An Operation Error is not considered fatal in and of itself, but may be used with an ABORT chunk to report a fatal condition。格式如下:



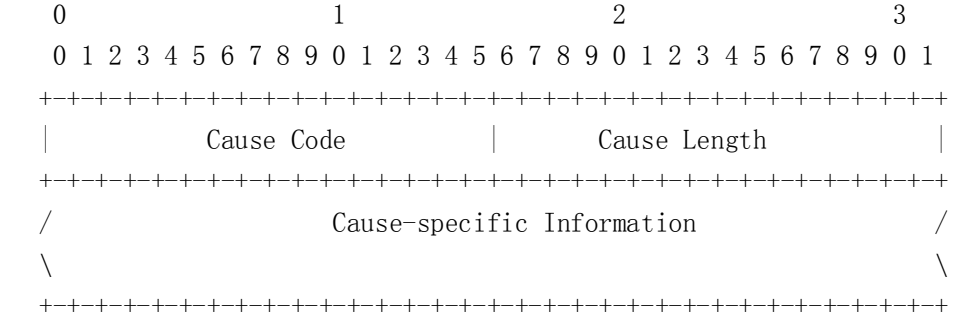
Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

Length: 16 bits (unsigned integer)

Set to the size of the chunk in bytes, including the chunk header and all the Error Cause fields present.

Error causes are defined as variable-length parameters using the format described in 3.2.1, i.e.:



Cause Code: 16 bits (unsigned integer)

被报告错误状态的类型。

Cause Code Value	Cause Code
1	Invalid Stream Identifier
2	Missing Mandatory Parameter
3	Stale Cookie Error
4	Out of Resource
5	Unresolvable Address
6	Unrecognized Chunk Type
7	Invalid Mandatory Parameter
8	Unrecognized Parameters
9	No User Data
10	Cookie Received While Shutting Down

Cause Length: 16 bits (unsigned integer)

Set to the size of the parameter in bytes, including the Cause Code, Cause Length, and Cause-Specific Information fields

Cause-specific Information: variable length

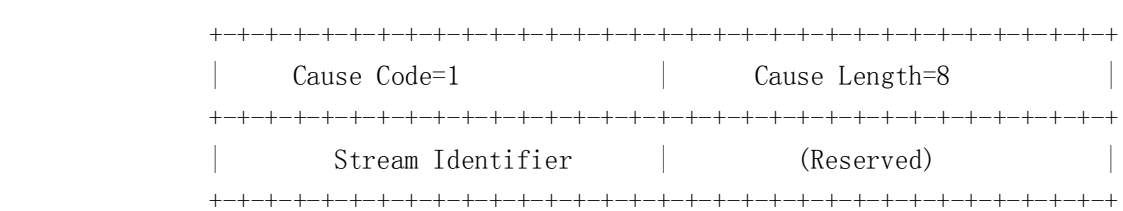
This field carries the details of the error condition.

Sections 3.3.10.1 - 3.3.10.10 define error causes for SCTP. Guidelines for the IETF to define new error cause values are discussed in Section 13.3.

3.3.10.1 Invalid Stream Identifier (1)

Cause of error

Invalid Stream Identifier: Indicates endpoint received a DATA chunk sent to a nonexistent stream.



Stream Identifier: 16 bits (unsigned integer)

Contains the Stream Identifier of the DATA chunk received in error.

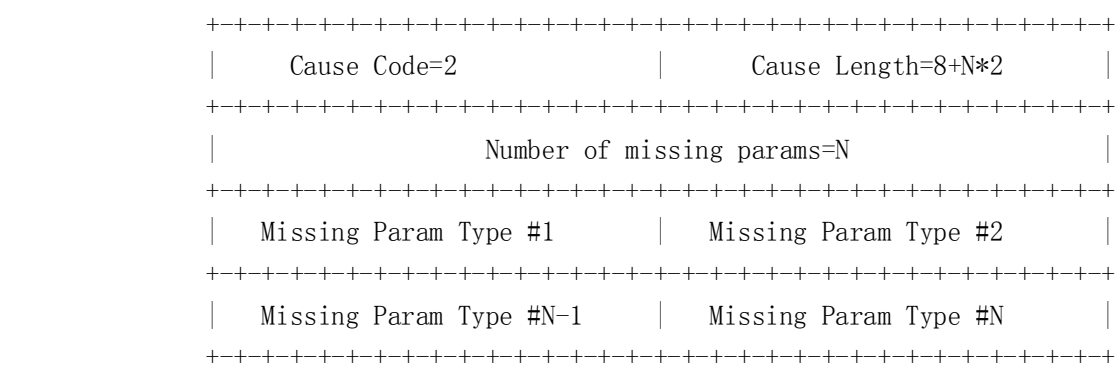
Reserved: 16 bits

This field is reserved. It is set to all 0's on transmit and Ignored on receipt.

3.3.10.2 Missing Mandatory Parameter (2)

Cause of error

缺少必备参数: 在收到的块 INIT 或 INIT ACK 中，缺少一个或多个必备的 TLV 参数。



缺少必备参数个数: 32 bits (unsigned integer)

This field contains the number of parameters contained in the Cause-specific Information field.

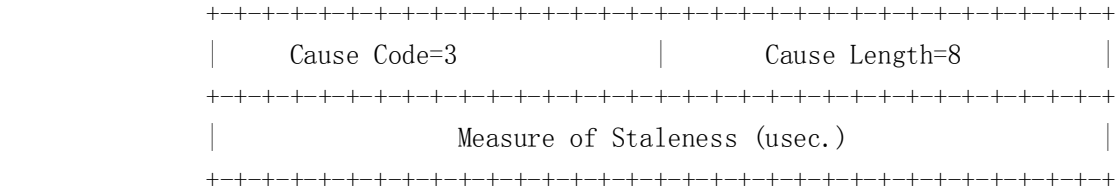
缺少必备参数类型: 16 bits (unsigned integer)

Each field will contain the missing mandatory parameter number.

3.3.10.3 Stale Cookie Error (3)

Cause of error

旧 Cookie 错误: State Cookie 已经过期。



Measure of Staleness: 32 bits (unsigned integer)

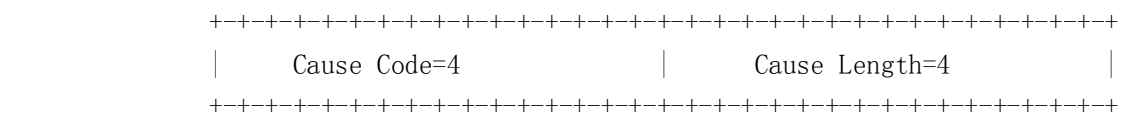
当前时刻与State Cookie过期时刻的差额（单位毫秒）。

错误原因的发送者可以选择报告State Cookie已经过期了多长时间（通过字段Measure of Staleness取非零值）。如果发送者不想提供这个消息，它应该把字段Measure of Staleness置零。

3.3.10.4 Out of Resource (4)

Cause of error

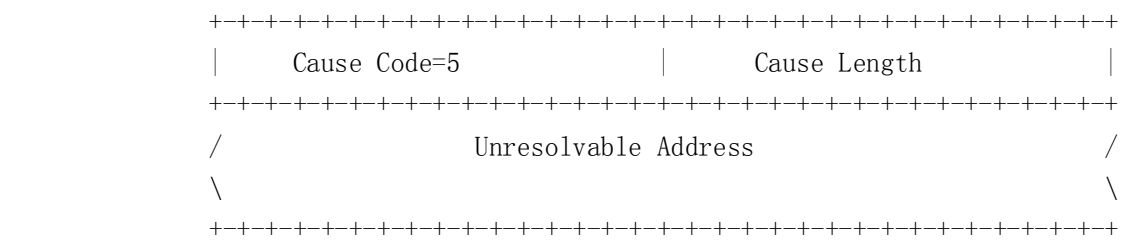
Out of Resource: Indicates that the sender is out of resource. This is usually sent in combination with or within an ABORT.



3.3.10.5 Unresolvable Address (5)

Cause of error

Unresolvable Address: Indicates that the sender is not able to resolve the specified address parameter (e.g., type of address is not supported by the sender). This is usually sent in combination with or within an ABORT.



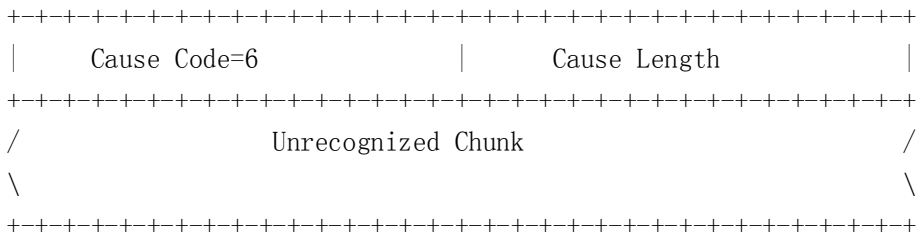
Unresolvable Address: variable length

The unresolvable address field contains the complete Type, Length and Value of the address parameter (or Host Name parameter) that contains the unresolvable address or host name.

3.3.10.6 Unrecognized Chunk Type (6)

Cause of error

Unrecognized Chunk Type: This error cause is returned to the originator of the chunk if the receiver does not understand the chunk and the upper bits of the 'Chunk Type' are set to 01 or 11.



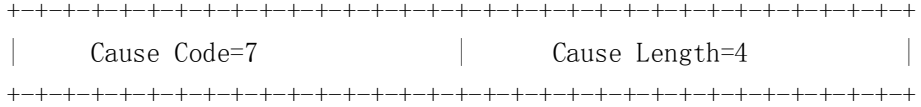
Unrecognized Chunk: variable length

The Unrecognized Chunk field contains the unrecognized Chunk from the SCTP packet complete with Chunk Type, Chunk Flags and Chunk Length.

3.3.10.7 Invalid Mandatory Parameter (7)

Cause of error

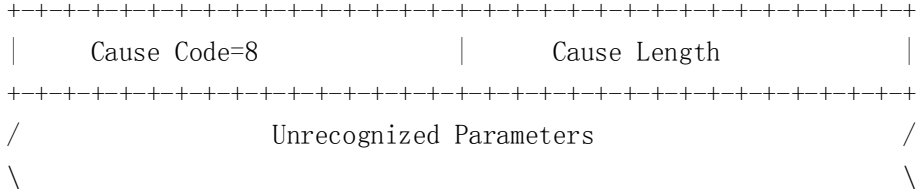
Invalid Mandatory Parameter: This error cause is returned to the originator of an INIT or INIT ACK chunk when one of the mandatory parameters is set to a invalid value.



3.3.10.8 Unrecognized Parameters (8)

Cause of error

Unrecognized Parameters: This error cause is returned to the originator of the INIT ACK chunk if the receiver does not recognize one or more Optional TLV parameters in the INIT ACK chunk.



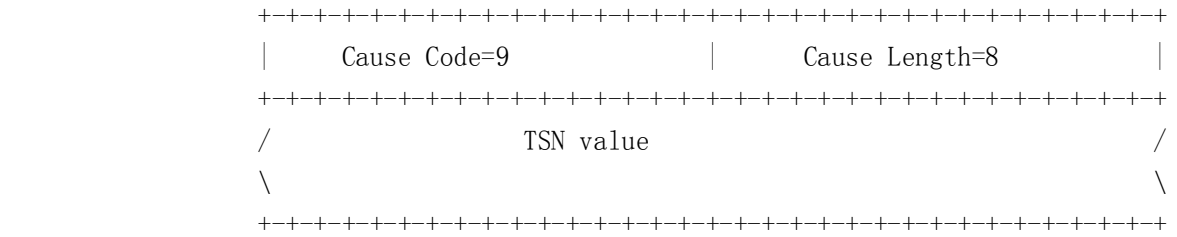
Unrecognized Parameters: variable length

The Unrecognized Parameters field contains the unrecognized parameters copied from the INIT ACK chunk complete with TLV. This error cause is normally contained in an ERROR chunk bundled with the COOKIE ECHO chunk when responding to the INIT ACK, when the sender of the COOKIE ECHO chunk wishes to report unrecognized parameters.

3.3.10.9 No User Data (9)

Cause of error

No User Data: This error cause is returned to the originator of a DATA chunk if a received DATA chunk has no user data.



TSN value: 32 bits (+unsigned integer)

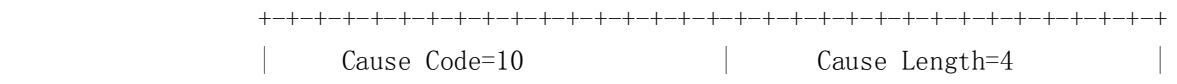
The TSN value field contains the TSN of the DATA chunk received with no user data field.

This cause code is normally returned in an ABORT chunk (see Section 6.2)

3.3.10.10 Cookie Received While Shutting Down (10)

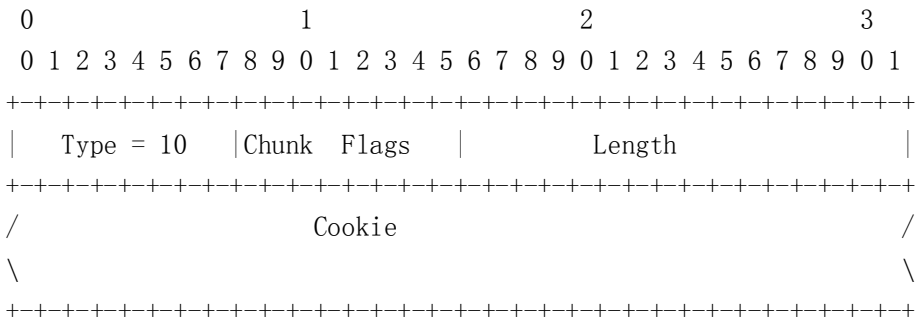
Cause of error

Cookie Received While Shutting Down: A COOKIE ECHO was received While the endpoint was in SHUTDOWN-ACK-SENT state. This error is usually returned in an ERROR chunk bundled with the retransmitted SHUTDOWN ACK.



3.3.11 Cookie Echo (COOKIE ECHO) (10):

仅仅在偶联初始化时被使用。它被偶联的发起者发往对端，以完成偶联的初始化。COOKIE ECHO 必须领先于偶联内的任何 DATA 块被发送，但可以和一个或多个 DATA 块捆绑于一个 SCTP 包内。



Chunk Flags: 8 bit

传输时所有比特被置零，接收时忽略所有比特。

Length: 16 bits (unsigned integer)

Set to the size of the chunk in bytes, including the 4 bytes of the chunk header and the size of the Cookie.

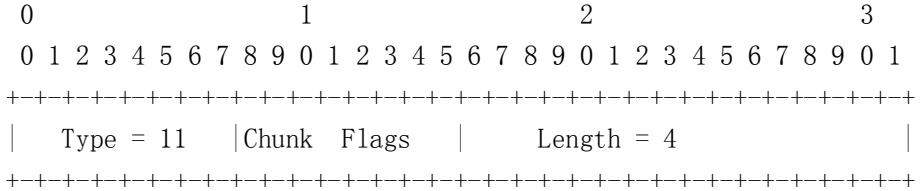
Cookie: variable size

This field must contain the exact cookie received in the State Cookie parameter from the previous INIT ACK.

实现应该使 cookie 尽可能小以确保互操作性。

3.3.12 Cookie Acknowledgement (COOKIE ACK) (11):

仅仅在偶联初始化时被使用，用来证实 COOKIE ECHO。COOKIE ACK 必须领先于偶联内的任何 DATA 或 SACK 块被发送，但可以和一个或多个 DATA 或 SACK 块捆绑于一个 SCTP 包内。



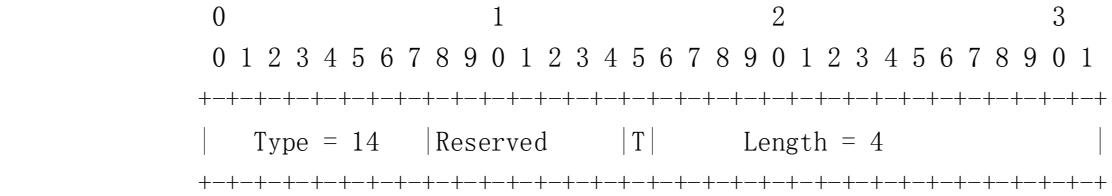
Chunk Flags: 8 bits

传输时所有比特被置零，接收时忽略所有比特。

3.3.13 Shutdown Complete (SHUTDOWN COMPLETE) (14):

用来在 shutdown 进程完成时，作为对收到 SHUTDOWN ACK 的证实，细节见 9.2。

The SHUTDOWN COMPLETE chunk has no parameters.



Chunk Flags: 8 bits

Reserved: 7 bits

传输时所有比特被置零，接收时忽略所有比特。

T bit: 1 bit

0 表示发送者有一个 TCB，但已被破坏；1 表示发送者没有 TCB。

注意: Special rules apply to this chunk for verification, 细节见 8.5.1。

4. SCTP 偶联状态图

在 SCTP 偶联的生命周期期间，受各种（外部）事件的驱动从一个状态转移到另外一个状态。可能的事件如下：

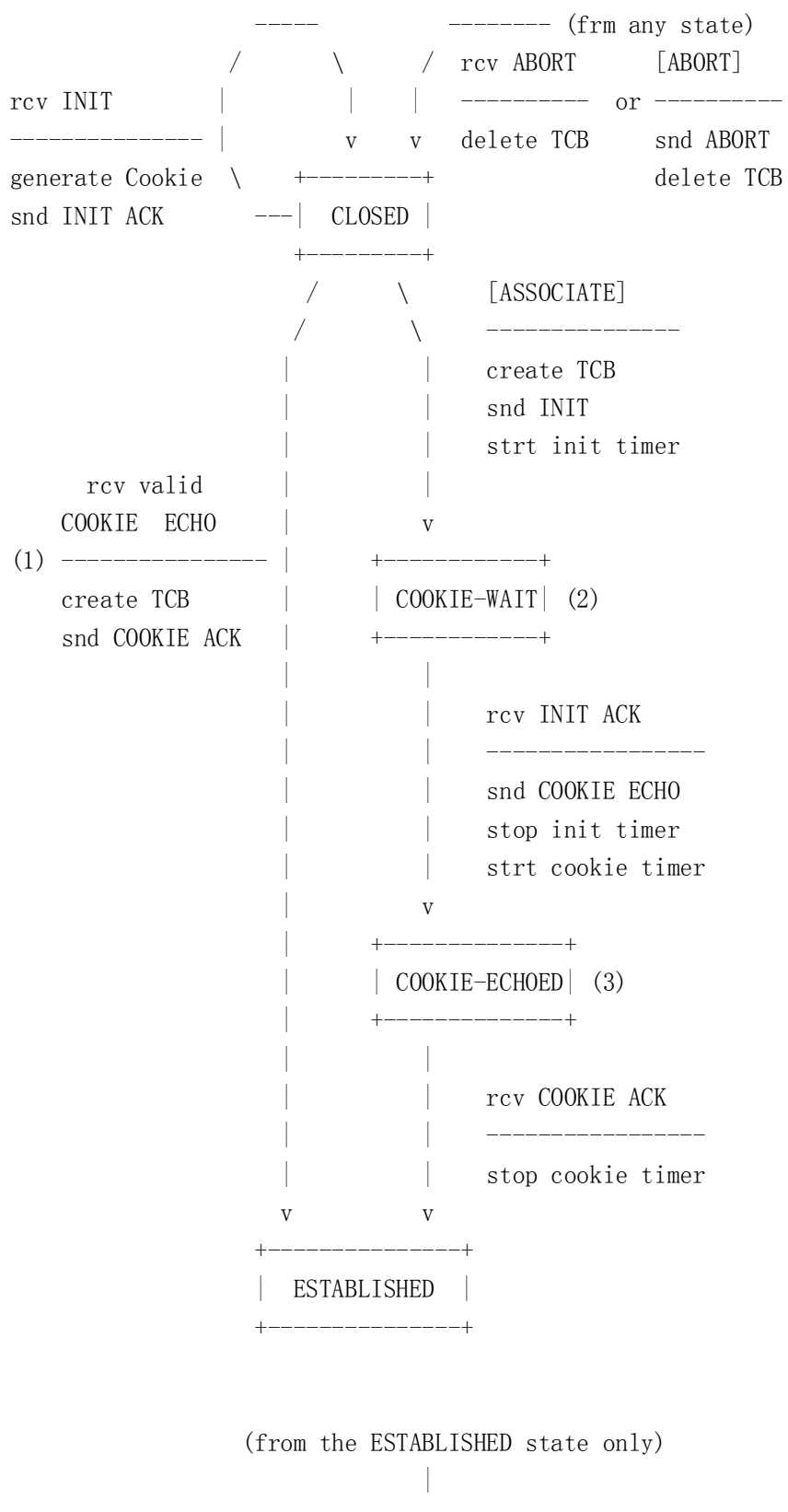
o SCTP 用户原语调用, 如, [ASSOCIATE], [SHUTDOWN], [ABORT],

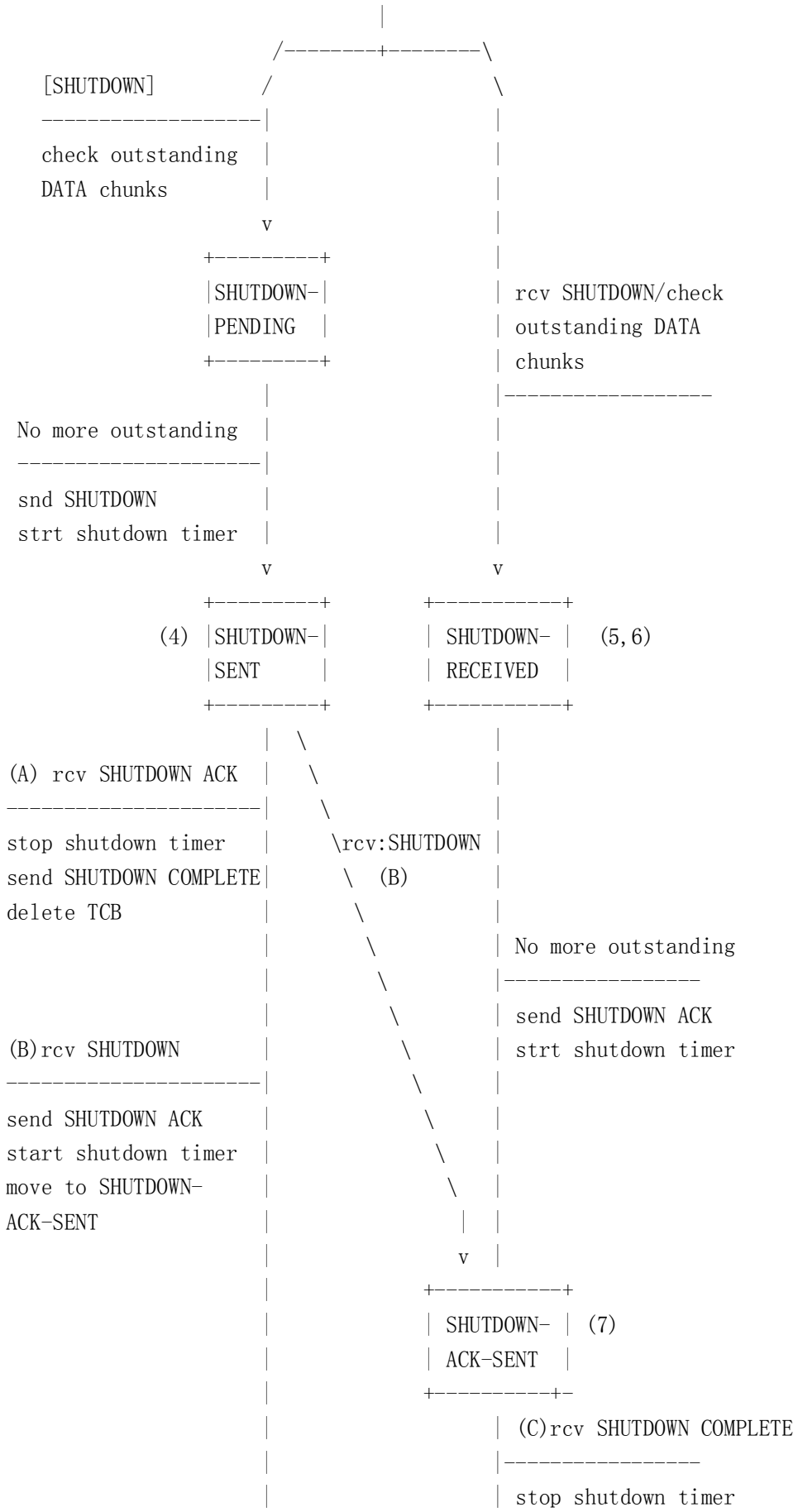
o 收到 INIT, COOKIE ECHO, ABORT, SHUTDOWN, 等控制块, 或

o 一些超时事件.

The state diagram in the figures below illustrates state changes, together with the causing events and resulting actions. Note that some of the error conditions are not shown in the state diagram. Full description of all special cases should be found in the text.

注意：块名全用大写字母，参数名仅首字母被大写，如，COOKIE ECHO 块类型 vs. State Cookie 参数。如果有多于一个的 event/message 能造成状态转移，用(A)，(B)等标记。




[SHUTDOWN]
check outstanding
DATA chunks
SHUTDOWN-
PENDING
rcv SHUTDOWN/check
outstanding DATA
chunks
No more outstanding
snd SHUTDOWN
strt shutdown timer
(4)
SHUTDOWN-
SENT
SHUTDOWN-
RECEIVED
(5,6)
(A) rcv SHUTDOWN ACK
stop shutdown timer
send SHUTDOWN COMPLETE
delete TCB
rcv:SHUTDOWN
(B)
No more outstanding
send SHUTDOWN ACK
strt shutdown timer
(B)rcv SHUTDOWN
send SHUTDOWN ACK
start shutdown timer
move to SHUTDOWN-
ACK-SENT
SHUTDOWN-
ACK-SENT
(7)
(C)rcv SHUTDOWN COMPLETE
stop shutdown timer

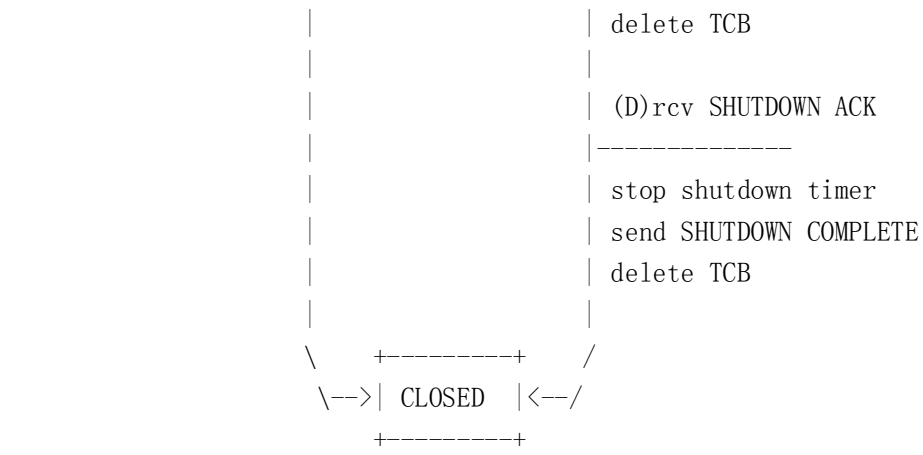


Figure 3: State Transition Diagram of SCTP

注意:

1) 如果收到的 COOKIE ECHO 中的 State Cookie 是无效的(如,无法通过完整性检查),接收者必须丢弃此 SCTP 包; 如果收到的 State Cookie 已过期(见 5.1.5), 接收者必须必须发送 ERROR 块。在两者情况下, 接收者都保持在 CLOSED 状态。

2) 如果 T1-init 定时器超期, 端点必须重传 INIT 并重起 T1-init 定时器而不改变状态。重传的最大次数为'Max.Init.Retransmits'。此后, 端点必须放弃初始化过程并向 SCTP 用户报告错误。

3) 如果 T1-cookie 定时器超期, 端点必须重传 COOKIE ECHO 并重起 T1-cookie 定时器而不改变状态。重传的最大次数为'Max.Init.Retransmits'。此后, 端点必须放弃初始化过程并向 SCTP 用户报告错误。

4) 在 SHUTDOWN-SENT 状态, 端点必须无延迟证实任何收到的 DATA 块。

5) 在 SHUTDOWN-RECEIVED 状态, 端点千万不要接收任何新的 SCTP 用户发送请求。

6) 在 SHUTDOWN-RECEIVED 状态, 端点必须传送或重传已在队列中的数据直到所有数据都被成功的传送后, 偶联离开此状态。

7) 在 SHUTDOWN-ACK-SENT 状态, 端点千万不要接收任何新的 SCTP 用户发送请求。

CLOSED 状态表明偶联没有被创建(如, 不存在)。

5. 偶联初始化

在从端点 A 到端点 Z 能发送第一个数据之前, 两个端点必须完成初始化进程以在它们之间创立一个 SCTP 偶联。

一个端点的 SCTP 用户应该用 ASSOCIATE 原语去初始化和另一个 SCTP 端点的偶联。

实现注意：从 SCTP 用户的观点来看，偶联可以被隐含的创建（无须调用 ASSOCIATE 原语）————通过端点第一次向对端发送用户数据。SCTP 对 INIT/INIT ACK 中的必备及可选参数都用缺省值。

Once the association is established, unidirectional streams are open for data transfer on both ends (see Section 5.1.1).

5.1 偶联的正常建立

初始化进程由下列步骤组成（假设 SCTP 端点"A"试图建立到 SCTP 端点"Z"的偶联，"Z"接受新偶联）：

A) "A" 发送 INIT 到"Z"。"A" 必须在 INIT 的 Initiate Tag 字段中提供它的 Verification Tag (Tag_A)。Tag_A 应该是从 1 到 4294967295 之间的一个随机数（标签值选择见 5.3.1)。发送 INIT 后，"A" 启动 T1-init 定时器并进入 COOKIE-WAIT 状态。

B) "Z" 立即响应 INIT ACK。INIT ACK 的目的 IP 地址必须被设置为 INIT（INIT ACK 正响应的）的源 IP 地址。在响应中除了填写其它参数外，"Z" 必须设置 Verification Tag 字段为 Tag_A 并在 Initiate Tag 中提供它自己的 Verification Tag (Tag_Z)。

而且, "Z" 必须产生并发送一个 State Cookie（在 INIT ACK 内）。State Cookie 的产生见 5.1.3。

注意：在发送带有 State Cookie 参数的 INIT ACK 之后，"Z" 千万不要分配任何资源，也不要保持任何新偶联的的状态。否则，"Z" 将易于受到资源攻击。

C) 接收到"Z"的 INIT ACK 后，"A" 会停止 T1-init 定时器并离开 COOKIE-WAIT 状态。接着"A" 会在 COOKIE ECHO 中发送在 INIT ACK 中收到的 State Cookie，启动 T1-cookie 定时器并进入 COOKIE-ECHOED 状态。

注意：COOKIE ECHO 能和任何未发的（pending）输出 DATA 捆绑，但必须是包内的第一个块并且在 COOKIE ACK 被返回之前，发送者不要向对端发送任何包。

D) 接收到 COOKIE ECHO 后, "Z" 构建 TCB 并进入 ESTABLISHED 状态，然后响应 COOKIE ACK。COOKIE ACK 可和任何未发的（pending）DATA（和/或 SACK）捆绑，但 COOKIE ACK 必须是包内的第一个块。

实现注意：收到一个有效的 COOKIE ECHO 时，实现可以选择发发送 Communication Up 通知给 SCTP 用户。

E) 接收到 COOKIE ACK 后，"A" 将从状态 COOKIE-ECHOED 进入到状态 ESTABLISHED，停止 T1-cookie 定时器。它也可用 Communication Up（第 10 节）通知 ULP 偶联已成功建立。

INIT 或 INIT ACK 千万不要和任何其他块进行捆绑，它们必须是携带它们的 SCTP 包的唯一块。

端点必须发送 INIT ACK 到携带它接收到的 INIT 的 IP 地址。

注意：T1-init 定时器和 T1-cookie 定时器遵守相同的规则（见 6.3.）。

如果一个端点在收到 INIT, INIT ACK, 或 COOKIE ECHO 后决定不建立新的偶联————INIT 或 INIT ACK 中缺少必备参数，无效的参数值，或缺乏本地资源等，它必须用 ABORT 响应。它也应该通过在 ABORT 块中包含 Error Causes 参数指明放弃的原因，如缺少必备参数的类型等。包含 ABORT 块的输出 SCTP 包的公共头的 Verification Tag 字段必须被置为对端 Initiate Tag 的值。

偶联中接收到第一个 DATA 块之后，端点必须立即用 SACK 进行证实。随后的证实按 6.2 描述的进行。

当 TCB 被创建后，每一端点必须将它内部的 Cumulative TSN Ack 指示器设为 Initial TSN - 1。

实现注意：IP 地址和 SCTP 端口号通常被用作在一个 SCTP 实例内查找 TCB 的关键字。

5.1.1 Handle Stream Parameters

发送者会在 INIT 和 INIT ACK 中指明希望在偶联中拥有的 OS 与能从其它端接收的 MIS。

接收到来自其它端的流配置信息后，端点会执行下列检查：如果对端的 MIS 小于本端点的 OS（对端无法支持本端想要配置的所有输出流），必须用对端的 MIS，或 abort 偶联并向它的上层报告对端资源 shortage。

偶联被初始化后，任何一端有效的输出流标识符范围为 0 到 min(local OS, remote MIS)-1.

5.1.2 Handle Address Parameters

偶联初始化过程中，端点会用下面的规则去发现并收集对端的目的传输地址(es)。

A) 如果在接受到的 INIT 或 INIT ACK 中没有任何地址参数，端点会从携带此块的 IP 包中取出源 IP 地址，与 SCTP 源端口号一起作为对端的唯一目的传输地址。

B) If there is a Host Name parameter present in the received INIT or

INIT ACK chunk, the endpoint shall resolve that host name to a list of IP address(es) and derive the transport address(es) of this peer by combining the resolved IP address(es) with the SCTP source port.

端点必须忽略任何其他 IP 地址参数，如果它们也出现在接收到的 INIT 或 INIT ACK 中。

INIT 接收者解析主机名的时机存在安全隐患。如果一收到 INIT 立即解析主机名，并且接收者所用的解析机制（如 DNS 查询）(potential）陷入长时间的等待中，那么可能接收者在能创建 State Cookie 并释放本地资源之前，因须等待解析结果而 open 自己从而受到资源攻击。

因此在此情形下，INIT 接收者必须延迟主机名解析直到从对端收到 COOKIE ECHO 。INIT 接收者应该用接收到的主机名（而不是目的传输地址）来创建 State Cookie 并发送 INIT ACK 到承载 INIT 的源 IP 地址。

接收者接收到 INIT ACK 后，总会立即试图解析地址（主机名）。

INIT 或 INIT ACK 的接收者在地址（主机名）被成功的解析之前，千万不要发送任何数据到对端(piggy-backed or stand-alone)。

如果地址（主机名）解析不成功，端点必须立即发送 ABORT（"Unresolvable Address"）到对端。ABORT 应（会）被发送到最近一次接收到的对端的包的源 IP 地址。

C) If there are only IPv4/IPv6 addresses present in the received INIT or INIT ACK chunk, the receiver shall derive and record all the transport address(es) from the received chunk AND the source IP address that sent the INIT or INIT ACK. The transport address(es) are derived by the combination of SCTP source port (from the common header) and the IP address parameter(s) carried in the INIT or INIT ACK chunk and the source IP address of the IP datagram. The receiver should use only these transport addresses as destination transport addresses when sending subsequent packets to its peer.

实现注意: 在一些情形下(如, 实现不控制用于传输的源 IP 地址)，端点可能需要在它的 INIT 或 INIT ACK 中包含所有能被用来向对端发送包的源 IP 地址。

利用上面的规则从 INIT 或 INIT ACK 得到所有的传输地址之后，端点会从中选择一个作为初始主通路。

注意: INIT-ACK 必须被发往 INIT 的源地址。

INIT 的发送者可以包含一个'Supported Address Types'参数以指示可接受的地址类型。此时，INIT 的接收者(initiatee)在响应 INIT 时必须用参数 Supported Address Types 中指定的一种地值类型；或 abort 偶联（"Unresolvable Address"）如果它不愿意或不能使用被对端指示的任何地址类型。

实现注意：在 INIT ACK 接收者因不支持地址类型而不能解析地址参数的情形下，创建偶联的发起者能放弃初始化过程，并通过在新的 INIT 中（'Supported Address Types'）指明它所 prefer 的地址类型而重新发起初始化。

5.1.3 State Cookie 的产生

当为响应 INIT 而发送 INIT ACK 时，发送者要创建一个 State Cookie 并包含在 INIT ACK 的参数 State Cookie 中。发送者应在 State Cookie 之内包括一个 MAC（see [RFC2104] for an example），State Cookie 被创建时的时间戳，State Cookie 的生命期限及所有为创建偶联而应在 State Cookie 中包含的信息。

产生 State Cookie 应采取的步骤如下:

1) Create an association TCB using information from both the received INIT and the outgoing INIT ACK chunk,

2) In the TCB, set the creation time to the current time of day, and the lifespan to the protocol parameter 'Valid.Cookie.Life',

3) From the TCB, identify and collect the minimal subset of information needed to re-create the TCB, and generate a MAC using this subset of information and a secret key (see [RFC2104] for an example of generating a MAC), and

4) Generate the State Cookie by combining this subset of information and the resultant MAC.

发送带有 State Cookie 参数的 INIT ACK 后，发送者应该删除 TCB 和任何其他与新偶联相关的本地资源，以保护资源免受攻击。

严格的说，用来产生 MAC 的哈希方法是 INIT 接收者的私有问题。MAC 的使用是必须的，以防止拒绝服务攻击。私钥应该是随机的([RFC1750]中提供了一些有关随机化指导的信息)；应该被 reasonably frequently 改变并且 State Cookie 中的时间戳可用来决定哪一个密钥应该被用来验证 MAC。

实现应该使 cookie 尽可能小以确保互操作性。

5.1.4 State Cookie Processing

当一个端点(在 COOKIE WAIT 状态)接收到一个带有 State Cookie 参数的 INIT ACK 时，它必须立即发送一个 COOKIE ECHO（带有接收到的 State Cookie）到它的对端。发送者可以在 COOKIE ECHO 块之后捆绑任何 pending 的 DATA 块于同一 SCTP 包中。

发送 COOKIE ECHO 之后，端点也会启动 T1-cookie 定时器。如果定时器超时，端点会重传 COOKIE ECHO chunk 并重起 T1-cookie，直到 COOKIE ACK 被收到或达到'Max.Init.Retransmits'(对端被标识为不可达，偶联进入 CLOSED 状态)。

5.1.5 State Cookie Authentication

当端点从还没与之建立偶联的另一端点接收到一个 COOKIE ECHO 块时，会采取下列动作（行为）:

1) Compute a MAC using the TCB data carried in the State Cookie and the secret key (note the timestamp in the State Cookie MAY be used to determine which secret key to use). Reference [RFC2104] can be used as a guideline for generating the MAC,

2) Authenticate the State Cookie as one that it previously generated by comparing the computed MAC against the one carried in the State Cookie. If this comparison fails, the SCTP packet, including the COOKIE ECHO and any DATA chunks, should be silently discarded,

3) Compare the creation timestamp in the State Cookie to the current local time. If the elapsed time is longer than the lifespan carried in the State Cookie, then the packet, including the COOKIE ECHO and any attached DATA chunks, SHOULD be discarded and the endpoint MUST transmit an ERROR chunk with a "Stale Cookie" error cause to the peer endpoint,

4) If the State Cookie is valid, create an association to the sender of the COOKIE ECHO chunk with the information in the TCB data carried in the COOKIE ECHO, and enter the ESTABLISHED state,

5) 发送 COOKIE ACK 到对端以证实收到 COOKIE ECHO。The COOKIE ACK MAY be bundled with an outbound DATA chunk or SACK chunk; however, the COOKIE ACK MUST be the first chunk in the SCTP packet.

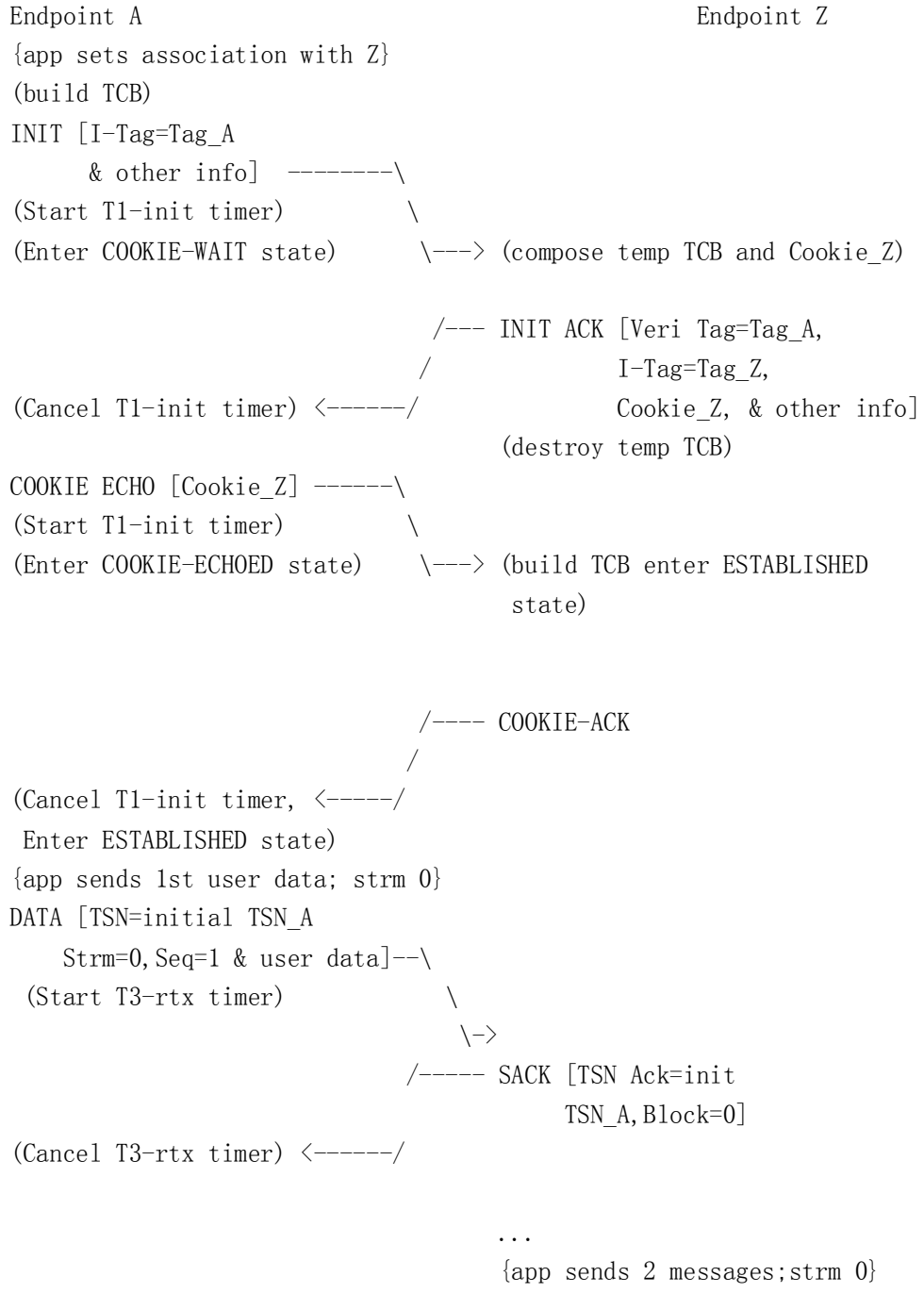
6) Immediately acknowledge any DATA chunk bundled with the COOKIE ECHO with a SACK (subsequent DATA chunk acknowledgement should follow the rules defined in Section 6.2). As mentioned in step 5), if the SACK is bundled with the COOKIE ACK, the COOKIE ACK

MUST appear first in the SCTP packet.

如果COOKIE ECHO的接收者与其发送者之间已经存在一个偶联，the procedures in Section 5.2 should be followed.

5.1.6 An Example of Normal Association Establishment

In the following example, "A" initiates the association and then sends a user message to "Z", then "Z" sends two user messages to "A" later (assuming no bundling or fragmentation occurs):



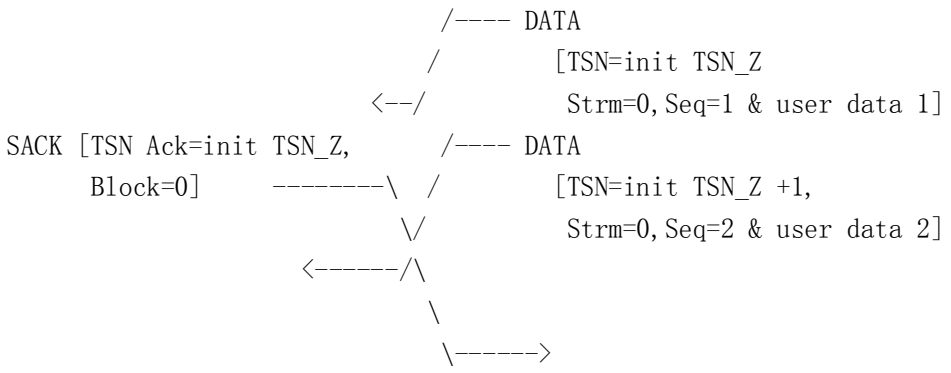


Figure 4: INITiation Example

If the T1-init timer expires at "A" after the INIT or COOKIE ECHO chunks are sent, the same INIT or COOKIE ECHO chunk with the same Initiate Tag (i.e., Tag_A) or State Cookie shall be retransmitted and

the timer restarted. This shall be repeated Max.Init.Retransmits times before "A" considers "Z" unreachable and reports the failure to its upper layer (and thus the association enters the CLOSED state). When retransmitting the INIT, the endpoint MUST follow the rules defined in 6.3 to determine the proper timer value.

5.2 Handle Duplicate or Unexpected INIT, INIT ACK, COOKIE ECHO, and COOKIE ACK

During the lifetime of an association (in one of the possible states), an endpoint may receive from its peer endpoint one of the setup chunks (INIT, INIT ACK, COOKIE ECHO, and COOKIE ACK). The receiver shall treat such a setup chunk as a duplicate and process it as described in this section.

Note: An endpoint will not receive the chunk unless the chunk was sent to a SCTP transport address and is from a SCTP transport address associated with this endpoint. Therefore, the endpoint processes such a chunk as part of its current association.

The following scenarios can cause duplicated or unexpected chunks:

A) The peer has crashed without being detected, re-started itself and sent out a new INIT chunk trying to restore the association,

B) Both sides are trying to initialize the association at about the same time,

C) The chunk is from a stale packet that was used to establish the present association or a past association that is no longer in existence,

D) The chunk is a false packet generated by an attacker, or

E) The peer never received the COOKIE ACK and is retransmitting its COOKIE ECHO.

The rules in the following sections shall be applied in order to identify and correctly handle these cases.

5.2.1 INIT received in COOKIE-WAIT or COOKIE-ECHOED State (Item B)

This usually indicates an initialization collision, i.e., each endpoint is attempting, at about the same time, to establish an association with the other endpoint.

Upon receipt of an INIT in the COOKIE-WAIT or COOKIE-ECHOED state, an endpoint MUST respond with an INIT ACK using the same parameters it

sent in its original INIT chunk (including its Initiation Tag, unchanged). These original parameters are combined with those from the newly received INIT chunk. The endpoint shall also generate a State Cookie with the INIT ACK. The endpoint uses the parameters sent in its INIT to calculate the State Cookie.

After that, the endpoint MUST NOT change its state, the T1-init timer shall be left running and the corresponding TCB MUST NOT be destroyed. The normal procedures for handling State Cookies when a TCB exists will resolve the duplicate INITs to a single association.

For an endpoint that is in the COOKIE-ECHOED state it MUST populate its Tie-Tags with the Tag information of itself and its peer (see section 5.2.2 for a description of the Tie-Tags).

5.2.2 Unexpected INIT in States Other than CLOSED, COOKIE-ECHOED, COOKIE-WAIT and SHUTDOWN-ACK-SENT

Unless otherwise stated, upon reception of an unexpected INIT for this association, the endpoint shall generate an INIT ACK with a State Cookie. In the outbound INIT ACK the endpoint MUST copy its current Verification Tag and peer's Verification Tag into a reserved place within the state cookie. We shall refer to these locations as

the Peer's-Tie-Tag and the Local-Tie-Tag. The outbound SCTP packet containing this INIT ACK MUST carry a Verification Tag value equal to the Initiation Tag found in the unexpected INIT. And the INIT ACK MUST contain a new Initiation Tag (randomly generated see Section 5.3.1). Other parameters for the endpoint SHOULD be copied from the existing parameters of the association (e.g. number of outbound streams) into the INIT ACK and cookie.

After sending out the INIT ACK, the endpoint shall take no further actions, i.e., the existing association, including its current state, and the corresponding TCB MUST NOT be changed.

Note: Only when a TCB exists and the association is not in a COOKIE-WAIT state are the Tie-Tags populated. For a normal association INIT (i.e. the endpoint is in a COOKIE-WAIT state), the Tie-Tags MUST be set to 0 (indicating that no previous TCB existed). The INIT ACK and State Cookie are populated as specified in section 5.2.1.

5.2.3 Unexpected INIT ACK

If an INIT ACK is received by an endpoint in any state other than the COOKIE-WAIT state, the endpoint should discard the INIT ACK chunk. An unexpected INIT ACK usually indicates the processing of an old or duplicated INIT chunk.

5.2.4 Handle a COOKIE ECHO when a TCB exists

When a COOKIE ECHO chunk is received by an endpoint in any state for an existing association (i.e., not in the CLOSED state) the following rules shall be applied:

1) Compute a MAC as described in Step 1 of Section 5.1.5,

2) Authenticate the State Cookie as described in Step 2 of Section 5.1.5 (this is case C or D above).

3) Compare the timestamp in the State Cookie to the current time. If the State Cookie is older than the lifespan carried in the State Cookie and the Verification Tags contained in the State Cookie do not match the current association's Verification Tags, the packet, including the COOKIE ECHO and any DATA chunks, should be discarded. The endpoint also MUST transmit an ERROR chunk with a "Stale Cookie" error cause to the peer endpoint (this is case C or D in section 5.2).

If both Verification Tags in the State Cookie match the Verification Tags of the current association, consider the State Cookie valid (this is case E of section 5.2) even if the lifespan is exceeded.

4) If the State Cookie proves to be valid, unpack the TCB into a temporary TCB.

5) Refer to Table 2 to determine the correct action to be taken.

Local Tag	Peer's Tag	Local-Tie-Tag	Peer's-Tie-Tag	Action/ Description
X	X	M	M	(A)
M	X	A	A	(B)
M	0	A	A	(B)
X	M	0	0	(C)
M	M	A	A	(D)

Table 2: Handling of a COOKIE ECHO when a TCB exists

Legend:

X - Tag does not match the existing TCB
M - Tag matches the existing TCB.
0 - No Tie-Tag in Cookie (unknown).
A - All cases, i.e. M, X or 0.

Note: For any case not shown in Table 2, the cookie should be silently discarded.

Action

A) In this case, the peer may have restarted. When the endpoint recognizes this potential 'restart', the existing session is treated the same as if it received an ABORT followed by a new COOKIE ECHO with the following exceptions:

- Any SCTP DATA Chunks MAY be retained (this is an implementation specific option).

- A notification of RESTART SHOULD be sent to the ULP instead of a "COMMUNICATION LOST" notification.

All the congestion control parameters (e.g., cwnd, ssthresh) related to this peer MUST be reset to their initial values (see Section 6.2.1).

After this the endpoint shall enter the ESTABLISHED state.

If the endpoint is in the SHUTDOWN-ACK-SENT state and recognizes the peer has restarted (Action A), it MUST NOT setup a new association but instead resend the SHUTDOWN ACK and send an ERROR chunk with a "Cookie Received while Shutting Down" error cause to its peer.

B) In this case, both sides may be attempting to start an association at about the same time but the peer endpoint started its INIT after responding to the local endpoint's INIT. Thus it may have picked a new Verification Tag not being aware of the previous Tag it had sent this endpoint. The endpoint should stay in or enter the ESTABLISHED state but it MUST update its peer's Verification Tag from the State Cookie, stop any init or cookie timers that may running and send a COOKIE ACK.

C) In this case, the local endpoint's cookie has arrived late. Before it arrived, the local endpoint sent an INIT and received an INIT-ACK and finally sent a COOKIE ECHO with the peer's same tag but a new tag of its own. The cookie should be silently discarded. The endpoint SHOULD NOT change states and should leave any timers running.

D) When both local and remote tags match the endpoint should always enter the ESTABLISHED state, if it has not already done so. It should stop any init or cookie timers that may be running and send a COOKIE ACK.

Note: The "peer's Verification Tag" is the tag received in the Initiate Tag field of the INIT or INIT ACK chunk.

5.2.4.1 An Example of a Association Restart

In the following example, "A" initiates the association after a restart has occurred. Endpoint "Z" had no knowledge of the restart until the exchange (i.e. Heartbeats had not yet detected the failure of "A"). (assuming no bundling or fragmentation occurs):

```
Endpoint A                                              Endpoint Z
<-------------- Association is established---------------------->
Tag=Tag_A                                                Tag=Tag_Z
<------------------------------------------------------------------>
{A crashes and restarts}
{app sets up a association with Z}
(build TCB)
INIT [I-Tag=Tag_A'
      & other info]  --------\
(Start T1-init timer)         \
(Enter COOKIE-WAIT state)      \---> (find a existing TCB
                                     compose temp TCB and Cookie_Z
                                     with Tie-Tags to previous
                                     association)
                               /--- INIT ACK [Veri Tag=Tag_A',
                              /               I-Tag=Tag_Z',
(Cancel T1-init timer) <-----/                Cookie_Z[TieTags=
                                              Tag_A,Tag_Z
                                               & other info]
                                    (destroy temp TCB,leave original
                                     in place)
COOKIE ECHO [Veri=Tag_Z',
             Cookie_Z
             Tie=Tag_A,
             Tag_Z]----------\
(Start T1-init timer)         \
(Enter COOKIE-ECHOED state)    \---> (Find existing association,
                                     Tie-Tags match old tags,
                                     Tags do not match i.e.
                                     case X X M M above,
                                     Announce Restart to ULP
                                     and reset association).
                              /---- COOKIE-ACK
                             /
(Cancel T1-init timer, <-----/
 Enter ESTABLISHED state)
{app sends 1st user data; strm 0}
DATA [TSN=initial TSN_A
```

```
     Strm=0,Seq=1 & user data]--\
(Start T3-rtx timer)            \
                                 \->
                               /----- SACK [TSN Ack=init TSN_A,Block=0]
(Cancel T3-rtx timer) <------/
```

Figure 5: A Restart Example

5.2.5 Handle Duplicate COOKIE-ACK.

At any state other than COOKIE-ECHOED, an endpoint should silently discard a received COOKIE ACK chunk.

5.2.6 Handle Stale COOKIE Error

Receipt of an ERROR chunk with a "Stale Cookie" error cause indicates one of a number of possible events:

A) That the association failed to completely setup before the State Cookie issued by the sender was processed.

B) An old State Cookie was processed after setup completed.

C) An old State Cookie is received from someone that the receiver is not interested in having an association with and the ABORT chunk was lost.

When processing an ERROR chunk with a "Stale Cookie" error cause an endpoint should first examine if an association is in the process of being setup, i.e. the association is in the COOKIE-ECHOED state. In all cases if the association is not in the COOKIE-ECHOED state, the ERROR chunk should be silently discarded.

If the association is in the COOKIE-ECHOED state, the endpoint may elect one of the following three alternatives.

1) Send a new INIT chunk to the endpoint to generate a new State Cookie and re-attempt the setup procedure.

2) Discard the TCB and report to the upper layer the inability to setup the association.

3) Send a new INIT chunk to the endpoint, adding a Cookie Preservative parameter requesting an extension to the lifetime of

the State Cookie. When calculating the time extension, an implementation SHOULD use the RTT information measured based on the previous COOKIE ECHO / ERROR exchange, and should add no more than 1 second beyond the measured RTT, due to long State Cookie lifetimes making the endpoint more subject to a replay attack.

5.3 其它初始化问题

5.3.1 标记值的选择（Selection of Tag Value）

Initiate Tag 值应该从范围 1 —— 2**32 - 1 内选择。Initiate Tag 值的随机化对防止"man in the middle" 和 "sequence number"攻击是非常重要的。[RFC1750]中描述的方法能被用于 Initiate Tag 值的随机化。Initiate Tags 的仔细选择也是必需的————防止把来自早先偶联的旧的重复的包错当作属于当前偶联的包而进行处理。

而且, 在给定偶联的生命周期内，任何一端所用的 Verification Tag 值千万不要改变；当一个端点宕掉而重新建立语相同对端的偶联时，必须用新的 Verification Tag 值。

6. 用户数据传递

数据传输只能发生在 ESTABLISHED, SHUTDOWN-PENDING 和 SHUTDOWN-RECEIVED 状态。唯一的例外是：在 COOKIE-WAIT 状态 DATA 块允许和一个即将输出的 COOKIE ECHO 块捆绑而传输。

在 ESTABLISHED, SHUTDOWN-PENDING 和 SHUTDOWN-RECEIVED 状态，DATA 块只能根据下面的规则进行接收。在 CLOSED 状态接收到的 DATA 块是 out of the blue (OOTB)（见 8.4 定义），应按 8.4 中的规则进行处理。任何其他状态接收到的 DATA 块应被丢弃。

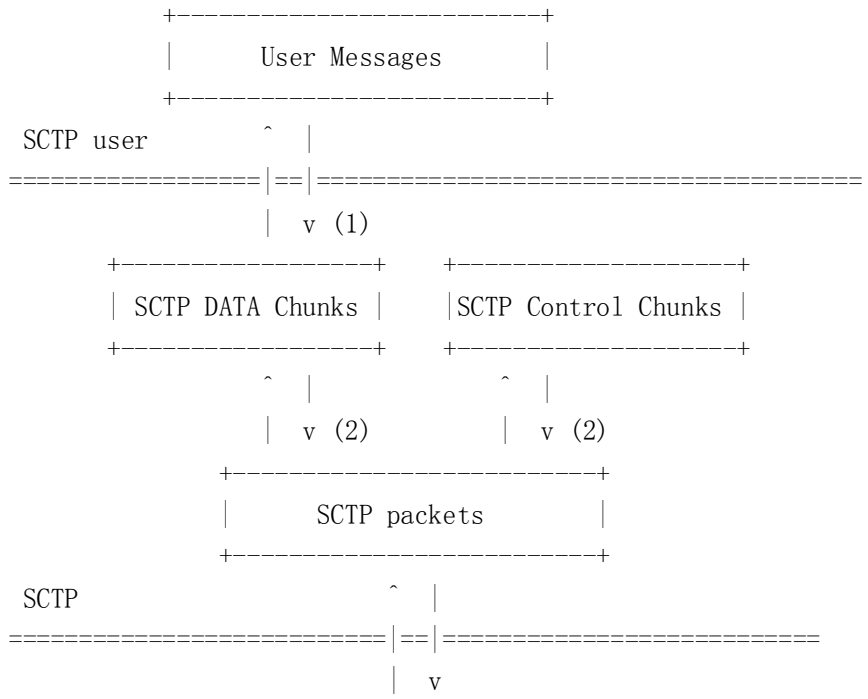
SACK 块在 ESTABLISHED, SHUTDOWN-PENDING 和 SHUTDOWN-RECEIVED 状态必须被处理，在 COOKIE-ECHOED 状态可被处理。在 CLOSED 状态接收到的 SACK 块是 out of the blue(OOTB)，应按 8.4 中的规则进行处理。任何其他状态接收到的 SACK 块应被丢弃。

SCTP 接收者必须具有接收一个至少 1500 字节的 SCTP 包的能力。这意味着，SCTP 端点在被发送的 INIT 或 INIT ACK 的 a_rwnd 中指定的数目千万不要小于 1500 字节。

出于传输效率的考虑，SCTP 提供了小用户消息的捆绑及大用户消息的分段机制。下面的图描述了用户消息通过 SCTP 的流程。

In this section the term "data sender" refers to the endpoint that transmits a DATA chunk and the term "data receiver" refers to the

endpoint that receives a DATA chunk. A data receiver will transmit SACK chunks.



注意:

1) When converting user messages into DATA chunks, an endpoint will fragment user messages larger than the current association path MTU into multiple DATA chunks. The data receiver will normally reassemble the fragmented message from DATA chunks before delivery to the user (see Section 6.9 for details).

2) Multiple DATA and control chunks may be bundled by the sender into a single SCTP packet for transmission, as long as the final size of the packet does not exceed the current path MTU. The receiver will unbundle the packet back into the original chunks. 在包内控制块必须位于 DATA 块之前。

Figure 6: Illustration of User Data Transfer

分段和捆绑机制，细节见 6.9 和 6.10，对 data sender 是可选的，但它们必须被 data receiver 实现, 如端点必须正确的接受和处理被捆绑及被分段的（用户）数据。

6.1 Transmission of DATA Chunks

本文档假定每个目的传输地址有一个重传定时器，但实现可为每个 DATA 块配置一个重传定时器。

发送者在发送和/或重传 DATA 块时，必须应用下列规则：

A) At any given time, the data sender MUST NOT transmit new data to any destination transport address if its peer's rwnd indicates that the peer has no buffer space (i.e. rwnd is 0, see Section 6.2.1). However, regardless of the value of rwnd (including if it is 0), the data sender can always have one DATA chunk in flight to the receiver if allowed by cwnd (see rule B below). This rule allows the sender to probe for a change in rwnd that the sender missed due to the SACK having been lost in transit from the data receiver to the data sender.

B) At any given time, the sender MUST NOT transmit new data to a given transport address if it has cwnd or more bytes of data outstanding to that transport address.

C) When the time comes for the sender to transmit, before sending new DATA chunks, the sender MUST first transmit any outstanding DATA chunks which are marked for retransmission (limited by the current cwnd).

D) Then, the sender can send out as many new DATA chunks as Rule A and Rule B above allow.

Multiple DATA chunks committed for transmission MAY be bundled in a single packet. Furthermore, DATA chunks being retransmitted MAY be bundled with new DATA chunks, as long as the resulting packet size does not exceed the path MTU. A ULP may request that no bundling is performed but this should only turn off any delays that a SCTP implementation may be using to increase bundling efficiency. It does not in itself stop all bundling from occurring (i.e. in case of congestion or retransmission).

Before an endpoint transmits a DATA chunk, if any received DATA chunks have not been acknowledged (e.g., due to delayed ack), the sender should create a SACK and bundle it with the outbound DATA chunk, as long as the size of the final SCTP packet does not exceed the current MTU. See Section 6.2.

IMPLEMENTATION NOTE: When the window is full (i.e., transmission is disallowed by Rule A and/or Rule B), the sender MAY still accept send

requests from its upper layer, but MUST transmit no more DATA chunks until some or all of the outstanding DATA chunks are acknowledged and transmission is allowed by Rule A and Rule B again.

Whenever a transmission or retransmission is made to any address, if the T3-rtx timer of that address is not currently running, the sender MUST start that timer. If the timer for that address is already running, the sender MUST restart the timer if the earliest (i.e., lowest TSN) outstanding DATA chunk sent to that address is being retransmitted. Otherwise, the data sender MUST NOT restart the timer.

When starting or restarting the T3-rtx timer, the timer value must be adjusted according to the timer rules defined in Sections 6.3.2, and 6.3.3.

Note: The data sender SHOULD NOT use a TSN that is more than 2**31 - 1 above the beginning TSN of the current send window.

6.2 Acknowledgement on Reception of DATA Chunks

The SCTP endpoint MUST always acknowledge the reception of each valid DATA chunk.

The guidelines on delayed acknowledgement algorithm specified in Section 4.2 of [RFC2581] SHOULD be followed. Specifically, an acknowledgement SHOULD be generated for at least every second packet (not every second DATA chunk) received, and SHOULD be generated within 200 ms of the arrival of any unacknowledged DATA chunk. In some situations it may be beneficial for an SCTP transmitter to be more conservative than the algorithms detailed in this document allow. However, an SCTP transmitter MUST NOT be more aggressive than the following algorithms allow.

A SCTP receiver MUST NOT generate more than one SACK for every incoming packet, other than to update the offered window as the receiving application consumes new data.

IMPLEMENTATION NOTE: The maximum delay for generating an acknowledgement may be configured by the SCTP administrator, either statically or dynamically, in order to meet the specific timing requirement of the protocol being carried.

An implementation MUST NOT allow the maximum delay to be configured

to be more than 500 ms. In other words an implementation MAY lower this value below 500ms but MUST NOT raise it above 500ms.

Acknowledgements MUST be sent in SACK chunks unless shutdown was requested by the ULP in which case an endpoint MAY send an acknowledgement in the SHUTDOWN chunk. A SACK chunk can acknowledge the reception of multiple DATA chunks. See Section 3.3.4 for SACK chunk format. In particular, the SCTP endpoint MUST fill in the Cumulative TSN Ack field to indicate the latest sequential TSN (of a valid DATA chunk) it has received. Any received DATA chunks with TSN greater than the value in the Cumulative TSN Ack field SHOULD also be reported in the Gap Ack Block fields.

Note: The SHUTDOWN chunk does not contain Gap Ack Block fields. Therefore, the endpoint should use a SACK instead of the SHUTDOWN chunk to acknowledge DATA chunks received out of order .

When a packet arrives with duplicate DATA chunk(s) and with no new DATA chunk(s), the endpoint MUST immediately send a SACK with no delay. If a packet arrives with duplicate DATA chunk(s) bundled with new DATA chunks, the endpoint MAY immediately send a SACK. Normally receipt of duplicate DATA chunks will occur when the original SACK chunk was lost and the peer's RTO has expired. The duplicate TSN number(s) SHOULD be reported in the SACK as duplicate.

When an endpoint receives a SACK, it MAY use the Duplicate TSN information to determine if SACK loss is occurring. Further use of this data is for future study.

The data receiver is responsible for maintaining its receive buffers. The data receiver SHOULD notify the data sender in a timely manner of changes in its ability to receive data. How an implementation manages its receive buffers is dependent on many factors (e.g., Operating System, memory management system, amount of memory, etc.). However, the data sender strategy defined in Section 6.2.1 is based on the assumption of receiver operation similar to the following:

A) At initialization of the association, the endpoint tells the peer how much receive buffer space it has allocated to the association in the INIT or INIT ACK. The endpoint sets a_rwnd to this value.

B) As DATA chunks are received and buffered, decrement a_rwnd by the number of bytes received and buffered. This is, in effect,

closing rwnd at the data sender and restricting the amount of data it can transmit.

C) As DATA chunks are delivered to the ULP and released from the receive buffers, increment a_rwnd by the number of bytes delivered to the upper layer. This is, in effect, opening up rwnd on the data sender and allowing it to send more data. The

data receiver SHOULD NOT increment a_rwnd unless it has released bytes from its receive buffer. For example, if the receiver is holding fragmented DATA chunks in a reassembly queue, it should not increment a_rwnd.

D) When sending a SACK, the data receiver SHOULD place the current value of a_rwnd into the a_rwnd field. The data receiver SHOULD take into account that the data sender will not retransmit DATA chunks that are acked via the Cumulative TSN Ack (i.e., will drop from its retransmit queue).

Under certain circumstances, the data receiver may need to drop DATA chunks that it has received but hasn't released from its receive buffers (i.e., delivered to the ULP). These DATA chunks may have been acked in Gap Ack Blocks. For example, the data receiver may be holding data in its receive buffers while reassembling a fragmented user message from its peer when it runs out of receive buffer space. It may drop these DATA chunks even though it has acknowledged them in Gap Ack Blocks. If a data receiver drops DATA chunks, it MUST NOT include them in Gap Ack Blocks in subsequent SACKs until they are received again via retransmission. In addition, the endpoint should take into account the dropped data when calculating its a_rwnd.

An endpoint SHOULD NOT revoke a SACK and discard data. Only in extreme circumstance should an endpoint use this procedure (such as out of buffer space). The data receiver should take into account that dropping data that has been acked in Gap Ack Blocks can result in suboptimal retransmission strategies in the data sender and thus in suboptimal performance.

The following example illustrates the use of delayed acknowledgements:

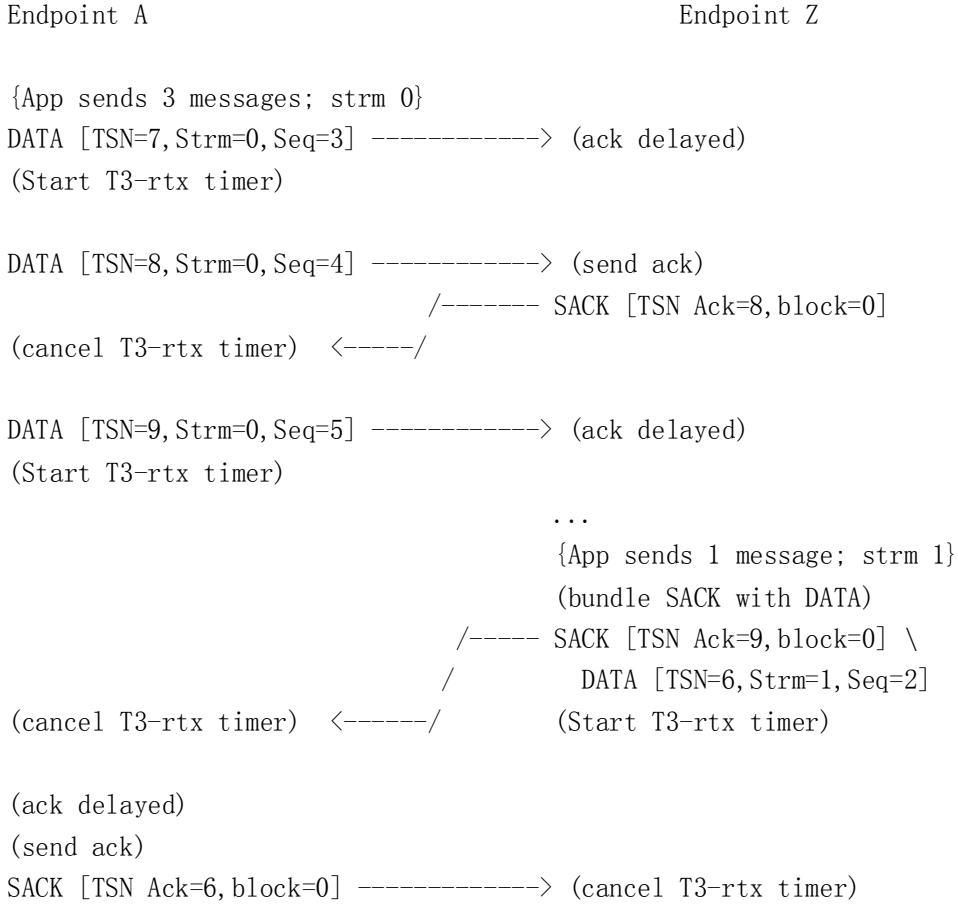


Figure 7: Delayed Acknowledgment Example

If an endpoint receives a DATA chunk with no user data (i.e., the Length field is set to 16) it MUST send an ABORT with error cause set to "No User Data".

An endpoint SHOULD NOT send a DATA chunk with no user data part.

6.2.1 Processing a Received SACK

Each SACK an endpoint receives contains an a_rwnd value. This value represents the amount of buffer space the data receiver, at the time of transmitting the SACK, has left of its total receive buffer space (as specified in the INIT/INIT ACK). Using a_rwnd, Cumulative TSN Ack and Gap Ack Blocks, the data sender can develop a representation of the peer's receive buffer space.

One of the problems the data sender must take into account when

processing a SACK is that a SACK can be received out of order. That is, a SACK sent by the data receiver can pass an earlier SACK and be received first by the data sender. If a SACK is received out of order, the data sender can develop an incorrect view of the peer's receive buffer space.

Since there is no explicit identifier that can be used to detect out-of-order SACKs, the data sender must use heuristics to determine if a SACK is new.

An endpoint SHOULD use the following rules to calculate the rwnd, using the a_rwnd value, the Cumulative TSN Ack and Gap Ack Blocks in a received SACK.

A) At the establishment of the association, the endpoint initializes the rwnd to the Advertised Receiver Window Credit (a_rwnd) the peer specified in the INIT or INIT ACK.

B) Any time a DATA chunk is transmitted (or retransmitted) to a peer, the endpoint subtracts the data size of the chunk from the rwnd of that peer.

C) Any time a DATA chunk is marked for retransmission (via either T3-rtx timer expiration (Section 6.3.3)or via fast retransmit (Section 7.2.4)), add the data size of those chunks to the rwnd.

Note: If the implementation is maintaining a timer on each DATA chunk then only DATA chunks whose timer expired would be marked for retransmission.

D) Any time a SACK arrives, the endpoint performs the following:

i) If Cumulative TSN Ack is less than the Cumulative TSN Ack Point, then drop the SACK. Since Cumulative TSN Ack is monotonically increasing, a SACK whose Cumulative TSN Ack is less than the Cumulative TSN Ack Point indicates an out-of-order SACK.

ii) Set rwnd equal to the newly received a_rwnd minus the number of bytes still outstanding after processing the Cumulative TSN Ack and the Gap Ack Blocks.

iii) If the SACK is missing a TSN that was previously acknowledged via a Gap Ack Block (e.g., the data receiver reneged on the data), then mark the corresponding DATA chunk as available for retransmit: Mark it as missing for fast retransmit as described in Section 7.2.4 and if no retransmit

timer is running for the destination address to which the DATA chunk was originally transmitted, then T3-rtx is started for that destination address.

6.3 Management of Retransmission Timer

An SCTP endpoint uses a retransmission timer T3-rtx to ensure data delivery in the absence of any feedback from its peer. The duration of this timer is referred to as RTO (retransmission timeout).

When an endpoint's peer is multi-homed, the endpoint will calculate a separate RTO for each different destination transport address of its peer endpoint.

The computation and management of RTO in SCTP follows closely how TCP manages its retransmission timer. To compute the current RTO, an endpoint maintains two state variables per destination transport address: SRTT (smoothed round-trip time) and RTTVAR (round-trip time variation).

6.3.1 RTO Calculation

The rules governing the computation of SRTT, RTTVAR, and RTO are as follows:

C1) Until an RTT measurement has been made for a packet sent to the given destination transport address, set RTO to the protocol parameter 'RTO.Initial'.

C2) When the first RTT measurement R is made, set SRTT <- R, RTTVAR <- R/2, and RTO <- SRTT + 4 * RTTVAR.

C3) When a new RTT measurement R' is made, set

RTTVAR <- (1 - RTO.Beta) * RTTVAR + RTO.Beta * |SRTT - R'| SRTT <- (1 - RTO.Alpha) * SRTT + RTO.Alpha * R'

Note: The value of SRTT used in the update to RTTVAR is its value before updating SRTT itself using the second assignment.

After the computation, update RTO <- SRTT + 4 * RTTVAR.

C4) When data is in flight and when allowed by rule C5 below, a new RTT measurement MUST be made each round trip. Furthermore, new RTT measurements SHOULD be made no more than once per round-trip

for a given destination transport address. There are two reasons for this recommendation: First, it appears that measuring more frequently often does not in practice yield any significant benefit [ALLMAN99]; second, if measurements are made more often, then the values of RTO.Alpha and RTO.Beta in rule C3 above should be adjusted so that SRTT and RTTVAR still adjust to changes at roughly the same rate (in terms of how many round trips it takes

them to reflect new values) as they would if making only one measurement per round-trip and using RTO.Alpha and RTO.Beta as given in rule C3. However, the exact nature of these adjustments remains a research issue.

C5) Karn's algorithm: RTT measurements MUST NOT be made using packets that were retransmitted (and thus for which it is ambiguous whether the reply was for the first instance of the packet or a later instance).

C6) Whenever RTO is computed, if it is less than RTO.Min seconds then it is rounded up to RTO.Min seconds. The reason for this rule is that RTOs that do not have a high minimum value are susceptible to unnecessary timeouts [ALLMAN99].

C7) A maximum value may be placed on RTO provided it is at least RTO.max seconds.

There is no requirement for the clock granularity G used for computing RTT measurements and the different state variables, other than:

G1) Whenever RTTVAR is computed, if RTTVAR = 0, then adjust RTTVAR <- G.

Experience [ALLMAN99] has shown that finer clock granularities (<= 100 msec) perform somewhat better than more coarse granularities.

6.3.2 Retransmission Timer Rules

The rules for managing the retransmission timer are as follows:

R1) Every time a DATA chunk is sent to any address (including a retransmission), if the T3-rtx timer of that address is not running, start it running so that it will expire after the RTO of that address. The RTO used here is that obtained after any doubling due to previous T3-rtx timer expirations on the corresponding destination address as discussed in rule E2 below.

R2) Whenever all outstanding data sent to an address have been acknowledged, turn off the T3-rtx timer of that address.

R3) Whenever a SACK is received that acknowledges the DATA chunk with the earliest outstanding TSN for that address, restart T3-rtx timer for that address with its current RTO (if there is still outstanding data on that address).

R4) Whenever a SACK is received missing a TSN that was previously acknowledged via a Gap Ack Block, start T3-rtx for the destination address to which the DATA chunk was originally transmitted if it is not already running.

The following example shows the use of various timer rules (assuming the receiver uses delayed acks).

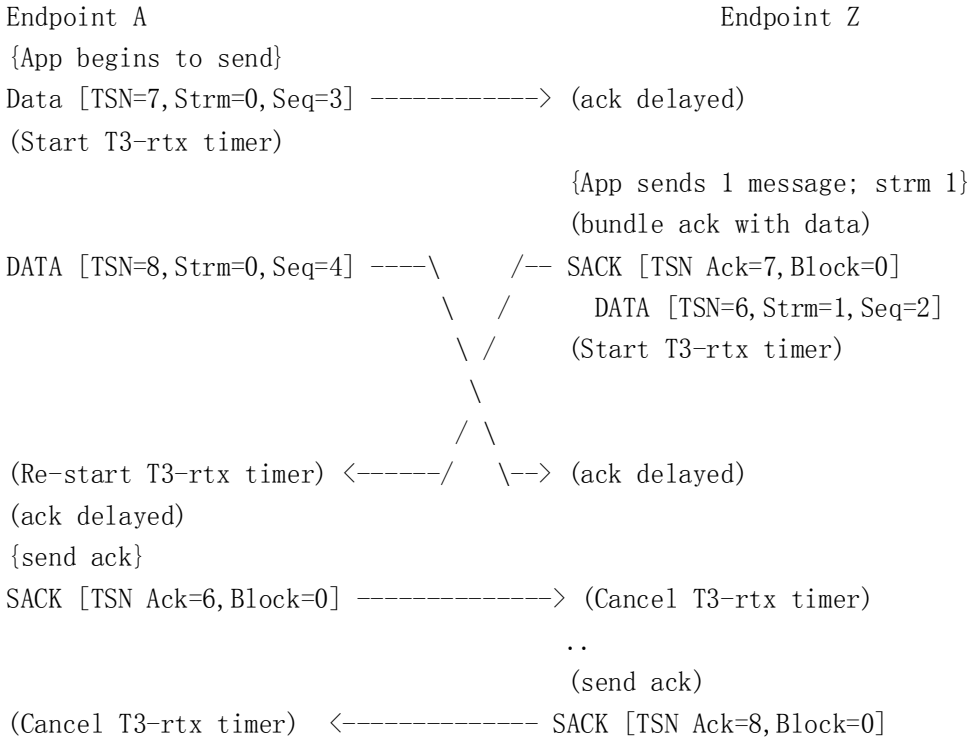


Figure 8 - Timer Rule Examples

6.3.3 Handle T3-rtx Expiration

Whenever the retransmission timer T3-rtx expires for a destination address, do the following:

E1) For the destination address for which the timer expires, adjust its ssthresh with rules defined in Section 7.2.3 and set the cwnd <- MTU.

E2) For the destination address for which the timer expires, set RTO <- RTO * 2 ("back off the timer"). The maximum value discussed in rule C7 above (RTO.max) may be used to provide an upper bound

to this doubling operation.

E3) Determine how many of the earliest (i.e., lowest TSN) outstanding DATA chunks for the address for which the T3-rtx has expired will fit into a single packet, subject to the MTU constraint for the path corresponding to the destination transport address to which the retransmission is being sent (this may be different from the

address for which the timer expires [see Section 6.4]). Call this value K. Bundle and retransmit those K DATA chunks in a single packet to the destination endpoint.

E4) Start the retransmission timer T3-rtx on the destination address to which the retransmission is sent, if rule R1 above indicates to do so. The RTO to be used for starting T3-rtx should be the one for the destination address to which the retransmission is sent, which, when the receiver is multi-homed, may be different from the destination address for which the timer expired (see Section 6.4 below).

After retransmitting, once a new RTT measurement is obtained (which can happen only when new data has been sent and acknowledged, per rule C5, or for a measurement made from a HEARTBEAT [see Section 8.3]), the computation in rule C3 is performed, including the computation of RTO, which may result in "collapsing" RTO back down after it has been subject to doubling (rule E2).

Note: Any DATA chunks that were sent to the address for which the T3-rtx timer expired but did not fit in one MTU (rule E3 above), should be marked for retransmission and sent as soon as cwnd allows (normally when a SACK arrives).

The final rule for managing the retransmission timer concerns failover (see Section 6.4.1):

F1) Whenever an endpoint switches from the current destination transport address to a different one, the current retransmission timers are left running. As soon as the endpoint transmits a packet containing DATA chunk(s) to the new transport address, start the timer on that transport address, using the RTO value of the destination address to which the data is being sent, if rule R1 indicates to do so.

6.4 Multi-homed SCTP Endpoints

An SCTP endpoint is considered multi-homed if there are more than one transport address that can be used as a destination address to reach that endpoint.

Moreover, the ULP of an endpoint shall select one of the multiple destination addresses of a multi-homed peer endpoint as the primary path (see Sections 5.1.2 and 10.1 for details).

By default, an endpoint SHOULD always transmit to the primary path, unless the SCTP user explicitly specifies the destination transport address (and possibly source transport address) to use.

An endpoint SHOULD transmit reply chunks (e.g., SACK, HEARTBEAT ACK, etc.) to the same destination transport address from which it received the DATA or control chunk to which it is replying. This rule should also be followed if the endpoint is bundling DATA chunks together with the reply chunk.

However, when acknowledging multiple DATA chunks received in packets from different source addresses in a single SACK, the SACK chunk may be transmitted to one of the destination transport addresses from which the DATA or control chunks being acknowledged were received.

When a receiver of a duplicate DATA chunk sends a SACK to a multi-homed endpoint it MAY be beneficial to vary the destination address and not use the source address of the DATA chunk. The reason being that receiving a duplicate from a multi-homed endpoint might indicate that the return path (as specified in the source address of the DATA chunk) for the SACK is broken.

Furthermore, when its peer is multi-homed, an endpoint SHOULD try to retransmit a chunk to an active destination transport address that is different from the last destination address to which the DATA chunk was sent.

Retransmissions do not affect the total outstanding data count. However, if the DATA chunk is retransmitted onto a different destination address, both the outstanding data counts on the new destination address and the old destination address to which the data chunk was last sent shall be adjusted accordingly.

6.4.1 Failover from Inactive Destination Address

Some of the transport addresses of a multi-homed SCTP endpoint may become inactive due to either the occurrence of certain error conditions (see Section 8.2) or adjustments from SCTP user.

When there is outbound data to send and the primary path becomes inactive (e.g., due to failures), or where the SCTP user explicitly requests to send data to an inactive destination transport address, before reporting an error to its ULP, the SCTP endpoint should try to send the data to an alternate active destination transport address if one exists.

When retransmitting data, if the endpoint is multi-homed, it should consider each source-destination address pair in its retransmission selection policy. When retransmitting the endpoint should attempt to pick the most divergent source-destination pair from the original source-destination pair to which the packet was transmitted.

Note: Rules for picking the most divergent source-destination pair are an implementation decision and is not specified within this document.

6.5 Stream Identifier and Stream Sequence Number

Every DATA chunk MUST carry a valid stream identifier. If an endpoint receives a DATA chunk with an invalid stream identifier, it shall acknowledge the reception of the DATA chunk following the normal procedure, immediately send an ERROR chunk with cause set to "Invalid Stream Identifier" (see Section 3.3.10) and discard the DATA chunk. The endpoint may bundle the ERROR chunk in the same packet as the SACK as long as the ERROR follows the SACK.

The stream sequence number in all the streams shall start from 0 when the association is established. Also, when the stream sequence number reaches the value 65535 the next stream sequence number shall be set to 0.

6.6 Ordered and Unordered Delivery

Within a stream, an endpoint MUST deliver DATA chunks received with the U flag set to 0 to the upper layer according to the order of their stream sequence number. If DATA chunks arrive out of order of their stream sequence number, the endpoint MUST hold the received DATA chunks from delivery to the ULP until they are re-ordered.

However, an SCTP endpoint can indicate that no ordered delivery is required for a particular DATA chunk transmitted within the stream by setting the U flag of the DATA chunk to 1.

When an endpoint receives a DATA chunk with the U flag set to 1, it must bypass the ordering mechanism and immediately deliver the data to the upper layer (after re-assembly if the user data is fragmented by the data sender).

This provides an effective way of transmitting "out-of-band" data in a given stream. Also, a stream can be used as an "unordered" stream by simply setting the U flag to 1 in all DATA chunks sent through that stream.

IMPLEMENTATION NOTE: When sending an unordered DATA chunk, an implementation may choose to place the DATA chunk in an outbound packet that is at the head of the outbound transmission queue if possible.

The 'Stream Sequence Number' field in a DATA chunk with U flag set to 1 has no significance. The sender can fill it with arbitrary value, but the receiver MUST ignore the field.

Note: When transmitting ordered and unordered data, an endpoint does not increment its Stream Sequence Number when transmitting a DATA chunk with U flag set to 1.

6.7 Report Gaps in Received DATA TSNs

Upon the reception of a new DATA chunk, an endpoint shall examine the continuity of the TSNs received. If the endpoint detects a gap in the received DATA chunk sequence, it SHOULD send a SACK with Gap Ack Blocks immediately. The data receiver continues sending a SACK after receipt of each SCTP packet that doesn't fill the gap.

Based on the Gap Ack Block from the received SACK, the endpoint can calculate the missing DATA chunks and make decisions on whether to retransmit them (see Section 6.2.1 for details).

Multiple gaps can be reported in one single SACK (see Section 3.3.4).

When its peer is multi-homed, the SCTP endpoint SHOULD always try to send the SACK to the same destination address from which the last DATA chunk was received.

Upon the reception of a SACK, the endpoint MUST remove all DATA chunks which have been acknowledged by the SACK's Cumulative TSN Ack from its transmit queue. The endpoint MUST also treat all the DATA chunks with TSNs not included in the Gap Ack Blocks reported by the SACK as "missing". The number of "missing" reports for each outstanding DATA chunk MUST be recorded by the data sender in order to make retransmission decisions. See Section 7.2.4 for details.

The following example shows the use of SACK to report a gap.

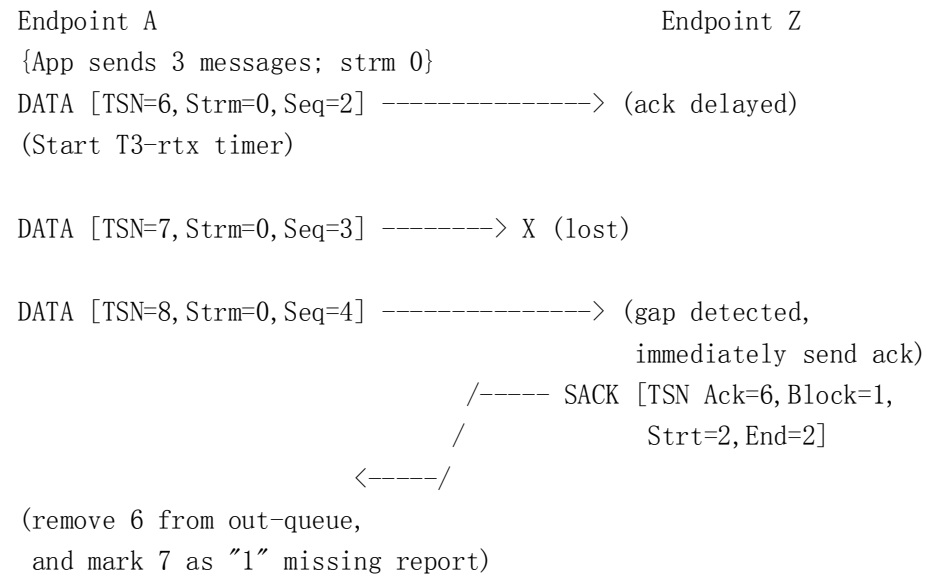


Figure 9 - Reporting a Gap using SACK

The maximum number of Gap Ack Blocks that can be reported within a single SACK chunk is limited by the current path MTU. When a single SACK can not cover all the Gap Ack Blocks needed to be reported due to the MTU limitation, the endpoint MUST send only one SACK, reporting the Gap Ack Blocks from the lowest to highest TSNs, within the size limit set by the MTU, and leave the remaining highest TSN numbers unacknowledged.

6.8 Adler-32 Checksum Calculation

When sending an SCTP packet, the endpoint MUST strengthen the data integrity of the transmission by including the Adler-32 checksum value calculated on the packet, as described below.

After the packet is constructed (containing the SCTP common header and one or more control or DATA chunks), the transmitter shall:

1) Fill in the proper Verification Tag in the SCTP common header and initialize the checksum field to 0's.

2) Calculate the Adler-32 checksum of the whole packet, including the SCTP common header and all the chunks. Refer to appendix B for details of the Adler-32 algorithm. And,

3) Put the resultant value into the checksum field in the common header, and leave the rest of the bits unchanged.

When an SCTP packet is received, the receiver MUST first check the Adler-32 checksum:

1) Store the received Adler-32 checksum value aside,

2) Replace the 32 bits of the checksum field in the received SCTP packet with all '0's and calculate an Adler-32 checksum value of the whole received packet. And,

3) Verify that the calculated Adler-32 checksum is the same as the received Adler-32 checksum. If not, the receiver MUST treat the packet as an invalid SCTP packet.

The default procedure for handling invalid SCTP packets is to silently discard them.

6.9 Fragmentation and Reassembly

An endpoint MAY support fragmentation when sending DATA chunks, but MUST support reassembly when receiving DATA chunks. If an endpoint supports fragmentation, it MUST fragment a user message if the size of the user message to be sent causes the outbound SCTP packet size to exceed the current MTU. If an implementation does not support fragmentation of outbound user messages, the endpoint must return an error to its upper layer and not attempt to send the user message.

IMPLEMENTATION NOTE: In this error case, the Send primitive discussed in Section 10.1 would need to return an error to the upper layer.

If its peer is multi-homed, the endpoint shall choose a size no larger than the association Path MTU. The association Path MTU is the smallest Path MTU of all destination addresses.

Note: Once a message is fragmented it cannot be re-fragmented. Instead if the PMTU has been reduced, then IP fragmentation must be used. Please see Section 7.3 for details of PMTU discovery.

When determining when to fragment, the SCTP implementation MUST take into account the SCTP packet header as well as the DATA chunk header(s). The implementation MUST also take into account the space required for a SACK chunk if bundling a SACK chunk with the DATA chunk.

Fragmentation takes the following steps:

1) The data sender MUST break the user message into a series of DATA chunks such that each chunk plus SCTP overhead fits into an IP datagram smaller than or equal to the association Path MTU.

2) The transmitter MUST then assign, in sequence, a separate TSN to each of the DATA chunks in the series. The transmitter assigns the same SSN to each of the DATA chunks. If the user indicates

that the user message is to be delivered using unordered delivery, then the U flag of each DATA chunk of the user message MUST be set to 1.

3) The transmitter MUST also set the B/E bits of the first DATA chunk in the series to '10', the B/E bits of the last DATA chunk in the series to '01', and the B/E bits of all other DATA chunks in the series to '00'.

An endpoint MUST recognize fragmented DATA chunks by examining the B/E bits in each of the received DATA chunks, and queue the fragmented DATA chunks for re-assembly. Once the user message is reassembled, SCTP shall pass the re-assembled user message to the specific stream for possible re-ordering and final dispatching.

Note: If the data receiver runs out of buffer space while still waiting for more fragments to complete the re-assembly of the message, it should dispatch part of its inbound message through a partial delivery API (see Section 10), freeing some of its receive buffer space so that the rest of the message may be received.

6.10 Bundling

An endpoint bundles chunks by simply including multiple chunks in one outbound SCTP packet. The total size of the resultant IP datagram, including the SCTP packet and IP headers, MUST be less or equal to the current Path MTU.

If its peer endpoint is multi-homed, the sending endpoint shall choose a size no larger than the latest MTU of the current primary path.

When bundling control chunks with DATA chunks, an endpoint MUST place control chunks first in the outbound SCTP packet. The transmitter MUST transmit DATA chunks within a SCTP packet in increasing order of TSN.

Note: Since control chunks must be placed first in a packet and since DATA chunks must be transmitted before SHUTDOWN or SHUTDOWN ACK chunks, DATA chunks cannot be bundled with SHUTDOWN or SHUTDOWN ACK chunks.

Partial chunks MUST NOT be placed in an SCTP packet.

An endpoint MUST process received chunks in their order in the packet. The receiver uses the chunk length field to determine the end of a chunk and beginning of the next chunk taking account of the fact that all chunks end on a 4 byte boundary. If the receiver detects a partial chunk, it MUST drop the chunk.

An endpoint MUST NOT bundle INIT, INIT ACK or SHUTDOWN COMPLETE with any other chunks.

7. 拥塞控制

Congestion control is one of the basic functions in SCTP. For some applications, it may be likely that adequate resources will be allocated to SCTP traffic to assure prompt delivery of time-critical data - thus it would appear to be unlikely, during normal operations, that transmissions encounter severe congestion conditions. However SCTP must operate under adverse operational conditions, which can develop upon partial network failures or unexpected traffic surges. In such situations SCTP must follow correct congestion control steps to recover from congestion quickly in order to get data delivered as soon as possible. In the absence of network congestion, these preventive congestion control algorithms should show no impact on the protocol performance.

IMPLEMENTATION NOTE: As far as its specific performance requirements are met, an implementation is always allowed to adopt a more conservative congestion control algorithm than the one defined below.

The congestion control algorithms used by SCTP are based on [RFC2581]. This section describes how the algorithms defined in RFC2581 are adapted for use in SCTP. We first list differences in protocol designs between TCP and SCTP, and then describe SCTP's congestion control scheme. The description will use the same terminology as in TCP congestion control whenever appropriate.

SCTP congestion control is always applied to the entire association, and not to individual streams.

7.1 SCTP Differences from TCP Congestion control

Gap Ack Blocks in the SCTP SACK carry the same semantic meaning as

the TCP SACK. TCP considers the information carried in the SACK as advisory information only. SCTP considers the information carried in the Gap Ack Blocks in the SACK chunk as advisory. In SCTP, any DATA chunk that has been acknowledged by SACK, including DATA that arrived at the receiving end out of order, are not considered fully delivered until the Cumulative TSN Ack Point passes the TSN of the DATA chunk (i.e., the DATA chunk has been acknowledged by the Cumulative TSN Ack

field in the SACK). Consequently, the value of cwnd controls the amount of outstanding data, rather than (as in the case of non-SACK TCP) the upper bound between the highest acknowledged sequence number and the latest DATA chunk that can be sent within the congestion window. SCTP SACK leads to different implementations of fast-retransmit and fast-recovery than non-SACK TCP. As an example see [FALL96].

The biggest difference between SCTP and TCP, however, is multi-homing. SCTP is designed to establish robust communication associations between two endpoints each of which may be reachable by more than one transport address. Potentially different addresses may lead to different data paths between the two endpoints, thus ideally one may need a separate set of congestion control parameters for each of the paths. The treatment here of congestion control for multi-homed receivers is new with SCTP and may require refinement in the future. The current algorithms make the following assumptions:

- The sender usually uses the same destination address until being instructed by the upper layer otherwise; however, SCTP may change to an alternate destination in the event an address is marked inactive (see Section 8.2). Also, SCTP may retransmit to a different transport address than the original transmission.

- The sender keeps a separate congestion control parameter set for each of the destination addresses it can send to (not each source-destination pair but for each destination). The parameters should decay if the address is not used for a long enough time period.

- For each of the destination addresses, an endpoint does slow-start upon the first transmission to that address.

Note: TCP guarantees in-sequence delivery of data to its upper-layer protocol within a single TCP session. This means that when TCP notices a gap in the received sequence number, it waits until the gap is filled before delivering the data that was received with sequence numbers higher than that of the missing data. On the other hand, SCTP can deliver data to its upper-layer protocol even if there is a gap in TSN if the Stream Sequence Numbers are in sequence for a particular stream (i.e., the missing DATA chunks are for a different

stream) or if unordered delivery is indicated. Although this does not affect cwnd, it might affect rwnd calculation.

7.2 SCTP Slow-Start and Congestion Avoidance

The slow start and congestion avoidance algorithms MUST be used by an endpoint to control the amount of data being injected into the network. The congestion control in SCTP is employed in regard to the association, not to an individual stream. In some situations it may be beneficial for an SCTP sender to be more conservative than the algorithms allow; however, an SCTP sender MUST NOT be more aggressive than the following algorithms allow.

Like TCP, an SCTP endpoint uses the following three control variables to regulate its transmission rate.

- Receiver advertised window size (rwnd, in bytes), which is set by the receiver based on its available buffer space for incoming packets.

  Note: This variable is kept on the entire association.

- Congestion control window (cwnd, in bytes), which is adjusted by the sender based on observed network conditions.

  Note: This variable is maintained on a per-destination address basis.

- Slow-start threshold (ssthresh, in bytes), which is used by the sender to distinguish slow start and congestion avoidance phases.

  Note: This variable is maintained on a per-destination address basis.

SCTP also requires one additional control variable, partial_bytes_acked, which is used during congestion avoidance phase to facilitate cwnd adjustment.

Unlike TCP, an SCTP sender MUST keep a set of these control variables cwnd, ssthresh and partial_bytes_acked for EACH destination address of its peer (when its peer is multi-homed). Only one rwnd is kept for the whole association (no matter if the peer is multi-homed or has a single address).

7.2.1 Slow-Start

Beginning data transmission into a network with unknown conditions or after a sufficiently long idle period requires SCTP to probe the network to determine the available capacity. The slow start algorithm is used for this purpose at the beginning of a transfer, or after repairing loss detected by the retransmission timer.

- The initial cwnd before DATA transmission or after a sufficiently long idle period MUST be <= 2*MTU.

- The initial cwnd after a retransmission timeout MUST be no more than 1*MTU.

- The initial value of ssthresh MAY be arbitrarily high (for example, implementations MAY use the size of the receiver advertised window).

- Whenever cwnd is greater than zero, the endpoint is allowed to have cwnd bytes of data outstanding on that transport address.

- When cwnd is less than or equal to ssthresh an SCTP endpoint MUST use the slow start algorithm to increase cwnd (assuming the current congestion window is being fully utilized). If an incoming SACK advances the Cumulative TSN Ack Point, cwnd MUST be increased by at most the lesser of 1) the total size of the previously outstanding DATA chunk(s) acknowledged, and 2) the destination's path MTU. This protects against the ACK-Splitting attack outlined in [SAVAGE99].

In instances where its peer endpoint is multi-homed, if an endpoint receives a SACK that advances its Cumulative TSN Ack Point, then it should update its cwnd (or cwnds) apportioned to the destination addresses to which it transmitted the acknowledged data. However if the received SACK does not advance the Cumulative TSN Ack Point, the endpoint MUST NOT adjust the cwnd of any of the destination addresses.

Because an endpoint's cwnd is not tied to its Cumulative TSN Ack Point, as duplicate SACKs come in, even though they may not advance the Cumulative TSN Ack Point an endpoint can still use them to clock out new data. That is, the data newly acknowledged by the SACK diminishes the amount of data now in flight to less than cwnd; and so the current, unchanged value of cwnd now allows new data to be sent.

On the other hand, the increase of cwnd must be tied to the Cumulative TSN Ack Point advancement as specified above. Otherwise the duplicate SACKs will not only clock out new data, but also will adversely clock out more new data than what has just left the network, during a time of possible congestion.

- When the endpoint does not transmit data on a given transport address, the cwnd of the transport address should be adjusted to max(cwnd/2, 2*MTU) per RTO.

7.2.2 Congestion Avoidance

When cwnd is greater than ssthresh, cwnd should be incremented by 1*MTU per RTT if the sender has cwnd or more bytes of data outstanding for the corresponding transport address.

In practice an implementation can achieve this goal in the following way:

- partial_bytes_acked is initialized to 0.

- Whenever cwnd is greater than ssthresh, upon each SACK arrival that advances the Cumulative TSN Ack Point, increase partial_bytes_acked by the total number of bytes of all new chunks acknowledged in that SACK including chunks acknowledged by the new Cumulative TSN Ack and by Gap Ack Blocks.

- When partial_bytes_acked is equal to or greater than cwnd and before the arrival of the SACK the sender had cwnd or more bytes of data outstanding (i.e., before arrival of the SACK, flightsize was greater than or equal to cwnd), increase cwnd by MTU, and reset partial_bytes_acked to (partial_bytes_acked - cwnd).

- Same as in the slow start, when the sender does not transmit DATA on a given transport address, the cwnd of the transport address should be adjusted to max(cwnd / 2, 2*MTU) per RTO.

- When all of the data transmitted by the sender has been acknowledged by the receiver, partial_bytes_acked is initialized to 0.

7.2.3 Congestion Control

Upon detection of packet losses from SACK (see Section 7.2.4), An

endpoint should do the following:

```
ssthresh = max(cwnd/2, 2*MTU)
cwnd = ssthresh
```

Basically, a packet loss causes cwnd to be cut in half.

When the T3-rtx timer expires on an address, SCTP should perform slow start by:

```
ssthresh = max(cwnd/2, 2*MTU)
cwnd = 1*MTU
```

and assure that no more than one SCTP packet will be in flight for that address until the endpoint receives acknowledgement for successful delivery of data to that address.

7.2.4 Fast Retransmit on Gap Reports

In the absence of data loss, an endpoint performs delayed acknowledgement. However, whenever an endpoint notices a hole in the arriving TSN sequence, it SHOULD start sending a SACK back every time a packet arrives carrying data until the hole is filled.

Whenever an endpoint receives a SACK that indicates some TSN(s) missing, it SHOULD wait for 3 further miss indications (via subsequent SACK's) on the same TSN(s) before taking action with regard to Fast Retransmit.

When the TSN(s) is reported as missing in the fourth consecutive SACK, the data sender shall:

1) Mark the missing DATA chunk(s) for retransmission,

2) Adjust the ssthresh and cwnd of the destination address(es) to which the missing DATA chunks were last sent, according to the formula described in Section 7.2.3.

3) Determine how many of the earliest (i.e., lowest TSN) DATA chunks marked for retransmission will fit into a single packet, subject to constraint of the path MTU of the destination transport address to which the packet is being sent. Call this value K. Retransmit those K DATA chunks in a single packet.

4) Restart T3-rtx timer only if the last SACK acknowledged the lowest outstanding TSN number sent to that address, or the endpoint is retransmitting the first outstanding DATA chunk sent to that address.

Note: Before the above adjustments, if the received SACK also acknowledges new DATA chunks and advances the Cumulative TSN Ack Point, the cwnd adjustment rules defined in Sections 7.2.1 and 7.2.2 must be applied first.

A straightforward implementation of the above keeps a counter for each TSN hole reported by a SACK. The counter increments for each consecutive SACK reporting the TSN hole. After reaching 4 and starting the fast retransmit procedure, the counter resets to 0.

Because cwnd in SCTP indirectly bounds the number of outstanding TSN's, the effect of TCP fast-recovery is achieved automatically with no adjustment to the congestion control window size.

7.3 Path MTU Discovery

[RFC1191] specifies "Path MTU Discovery", whereby an endpoint maintains an estimate of the maximum transmission unit (MTU) along a given Internet path and refrains from sending packets along that path which exceed the MTU, other than occasional attempts to probe for a change in the Path MTU (PMTU). RFC 1191 is thorough in its discussion of the MTU discovery mechanism and strategies for determining the current end-to-end MTU setting as well as detecting changes in this value. [RFC1981] specifies the same mechanisms for IPv6. An SCTP sender using IPv6 MUST use Path MTU Discovery unless all packets are less than the minimum IPv6 MTU [RFC2460].

An endpoint SHOULD apply these techniques, and SHOULD do so on a per-destination-address basis.

There are 4 ways in which SCTP differs from the description in RFC 1191 of applying MTU discovery to TCP:

1) SCTP associations can span multiple addresses. An endpoint MUST maintain separate MTU estimates for each destination address of its peer.

2) Elsewhere in this document, when the term "MTU" is discussed, it refers to the MTU associated with the destination address

corresponding to the context of the discussion.

3) Unlike TCP, SCTP does not have a notion of "Maximum Segment Size". Accordingly, the MTU for each destination address SHOULD be initialized to a value no larger than the link MTU for the local interface to which packets for that remote destination address will be routed.

4) Since data transmission in SCTP is naturally structured in terms of TSNs rather than bytes (as is the case for TCP), the discussion in Section 6.5 of RFC 1191 applies: When retransmitting an IP datagram to a remote address for which the IP datagram appears too large for the path MTU to that address, the IP datagram SHOULD be retransmitted without the DF bit set, allowing it to possibly be fragmented. Transmissions of new IP datagrams MUST have DF set.

5) The sender should track an association PMTU which will be the smallest PMTU discovered for all of the peer's destination addresses. When fragmenting messages into multiple parts this association PMTU should be used to calculate the size of each fragment. This will allow retransmissions to be seamlessly sent to an alternate address without encountering IP fragmentation.

Other than these differences, the discussion of TCP's use of MTU discovery in RFCs 1191 and 1981 applies to SCTP on a per-destination-address basis.

Note: For IPv6 destination addresses the DF bit does not exist, instead the IP datagram must be fragmented as described in [RFC2460].

8. 容错管理

8.1 端点故障检测

端点会用一个计数器记录到对端连续重传的总个数(如果是多归属的，则包含到对端所有目的传输地址的重传个数)。如果此计数器的值超过协议参数'Association.Max.Retrans'的值，端点会认为对端不可达并会停止传输任何数据给它（偶联进入CLOSED状态)。而且，端点会向上层报告错误，并optionally report back all outstanding user data remaining in its outbound queue. 对端不可达时，偶联自动关闭。

每当发往对端的DATA块被证实（接收到SACK)，或收到对端的HEARTBEAT-ACK，计数器都将被重置。

8.2 通路故障检测

当对端是多归属时，端点应为对端的每个目的传输地址配置一个 error 计数器。

每当任何地址的 T3-rtx 定时器超时时，或发往空闲地址的 HEARTBEAT 在 RTO 内不被证实时，相应目的地址的 error 计数器将增 1。当此计数器的值超过协议参数'Path.Max.Retrans'的值时，端点应把相应的目的传输地址标记为 inactive,并且一个通知应被发往上层。

When an outstanding TSN is acknowledged or a HEARTBEAT sent to that address is acknowledged with a HEARTBEAT ACK, the endpoint shall clear the error counter of the destination transport address to which the DATA chunk was last sent (or HEARTBEAT was sent). When the peer endpoint is multi-homed and the last chunk sent to it was a retransmission to an alternate address, there exists an ambiguity as to whether or not the acknowledgement should be credited to the address of the last chunk sent. However, this ambiguity does not seem to bear any significant consequence to SCTP behavior. If this ambiguity is undesirable, the transmitter may choose not to clear the error counter if the last chunk sent was a retransmission.

Note: When configuring the SCTP endpoint, the user should avoid having the value of 'Association.Max.Retrans' larger than the summation of the 'Path.Max.Retrans' of all the destination addresses for the remote endpoint. Otherwise, all the destination addresses may become inactive while the endpoint still considers the peer endpoint reachable. When this condition occurs, how the SCTP chooses to function is implementation specific.

When the primary path is marked inactive (due to excessive retransmissions, for instance), the sender MAY automatically transmit new packets to an alternate destination address if one exists and is active. If more than one alternate address is active when the primary path is marked inactive only ONE transport address SHOULD be chosen and used as the new destination transport address.

8.3 通路心跳

缺省的, SCTP 端点会周期性的向空闲（idle）目的传输地址(es)发送 HEARTBEAT 监视它们的可达性。

A destination transport address is considered "idle" if no new chunk which can be used for updating path RTT (usually including first transmission DATA, INIT, COOKIE ECHO, HEARTBEAT etc.) and no

HEARTBEAT has been sent to it within the current heartbeat period of that address. 这对 active 和 inactive 目的地址都适用。

上层能可选的发起下列功能:

A) Disable heartbeat on a specific destination transport address of a given association,

B) Change the HB.interval,

C) Re-enable heartbeat on a specific destination transport address of a given association, and,

D) Request an on-demand HEARTBEAT on a specific destination transport address of a given association.

The endpoint should increment the respective error counter of the destination transport address each time a HEARTBEAT is sent to that address and not acknowledged within one RTO.

When the value of this counter reaches the protocol parameter 'Path.Max.Retrans', the endpoint should mark the corresponding destination address as inactive if it is not so marked, and may also optionally report to the upper layer the change of reachability of this destination address. 此后, 端点应继续在此目的地址上进行 HEARTBEAT 但应停止增加计数器的值。

HEARTBEAT 块的发送者应该在块的 Heartbeat Information 字段中包含包被发送时的时间及包被发往的目的地址。

IMPLEMENTATION NOTE: An alternative implementation of the heartbeat mechanism that can be used is to increment the error counter variable every time a HEARTBEAT is sent to a destination. Whenever a HEARTBEAT ACK arrives, the sender SHOULD clear the error counter of the destination that the HEARTBEAT was sent to. This in effect would clear the previously stroked error (and any other error counts as well).

HEARTBEAT 块的接收者应立即用 HEARTBEAT ACK（包含从接收到的 HEARTBEAT 块中拷贝来的 Heartbeat Information 字段）响应。

一接收到 HEARTBEAT ACK，HEARTBEAT 的发送者应重置 HEARTBEAT 被发往的目的传输地址的 error 计数器，并标记目的传输地址为 active，如果它没被如此标记的话。端点可选的向上层报告这一情况（The endpoint may optionally report

to the upper layer when an inactive destination address is marked as active due to the reception of the latest HEARTBEAT ACK)。HEARTBEAT ACK 的接收者也必须重置偶联的（全局）error 计数器(见 8.1)。

HEARTBEAT ACK 的接收者也应根据在块 HEARTBEAT ACK 中携带的时间值对相应目的传输地址的 RTT 重新进行权衡（measurement)。

On an idle destination address that is allowed to heartbeat, a HEARTBEAT chunk is RECOMMENDED to be sent once per RTO of that destination address plus the protocol parameter 'HB.interval', with jittering of +/- 50%, and exponential back-off of the RTO if the previous HEARTBEAT is unanswered.

A primitive is provided for the SCTP user to change the HB.interval and turn on or off the heartbeat on a given destination address. The heartbeat interval set by the SCTP user is added to the RTO of that destination (including any exponential backoff). Only one heartbeat should be sent each time the heartbeat timer expires (if multiple destinations are idle). It is a implementation decision on how to choose which of the candidate idle destinations to heartbeat to (if more than one destination is idle).

Note: When tuning the heartbeat interval, there is a side effect that SHOULD be taken into account. When this value is increased, i.e. the HEARTBEAT takes longer, the detection of lost ABORT messages takes longer as well. If a peer endpoint ABORTs the association for any reason and the ABORT chunk is lost, the local endpoint will only discover the lost ABORT by sending a DATA chunk or HEARTBEAT chunk (thus causing the peer to send another ABORT). This must be considered when tuning the HEARTBEAT timer. If the HEARTBEAT is disabled only sending DATA to the association will discover a lost ABORT from the peer.

8.4 "Out of the blue" 包处理

一个 SCTP 包被称为"out of the blue" (OOTB)包，如果它 correctly formed, 如, 通过了接收者的 Adler-32 检察(见 6.8)，但接收者不能确定这个包所属的偶联。

The receiver of an OOTB packet MUST do the following:

1) If the OOTB packet is to or from a non-unicast address, silently discard the packet. Otherwise,

2) If the OOTB packet contains an ABORT chunk, the receiver MUST silently discard the OOTB packet and take no further action. Otherwise,

3) If the packet contains an INIT chunk with a Verification Tag set to '0', process it as described in Section 5.1. Otherwise,

4) If the packet contains a COOKIE ECHO in the first chunk, process it as described in Section 5.1. Otherwise,

5) If the packet contains a SHUTDOWN ACK chunk, the receiver should respond to the sender of the OOTB packet with a SHUTDOWN COMPLETE. When sending the SHUTDOWN COMPLETE, the receiver of the OOTB packet must fill in the Verification Tag field of the outbound packet with the Verification Tag received in the SHUTDOWN ACK and set the T-bit in the Chunk Flags to indicate that no TCB was found. Otherwise,

6) If the packet contains a SHUTDOWN COMPLETE chunk, the receiver should silently discard the packet and take no further action. Otherwise,

7) If the packet contains a "Stale cookie" ERROR or a COOKIE ACK the SCTP Packet should be silently discarded. Otherwise,

8) The receiver should respond to the sender of the OOTB packet with an ABORT. When sending the ABORT, the receiver of the OOTB packet MUST fill in the Verification Tag field of the outbound packet with the value found in the Verification Tag field of the OOTB packet and set the T-bit in the Chunk Flags to indicate that no TCB was found. After sending this ABORT, the receiver of the OOTB packet shall discard the OOTB packet and take no further action.

8.5 Verification Tag

本部分定义的Verification Tag规则仅仅适用于发送或接收的SCTP包中不包含INIT, SHUTDOWN COMPLETE, COOKIE ECHO (见5.1), ABORT or SHUTDOWN ACK块的情形。包含这些块时的规则在8.5.1中被分别讨论。

When sending an SCTP packet, the endpoint MUST fill in the Verification Tag field of the outbound packet with the tag value in the Initiate Tag parameter of the INIT or INIT ACK received from its peer.

When receiving an SCTP packet, the endpoint MUST ensure that the value in the Verification Tag field of the received SCTP packet matches its own Tag. If the received Verification Tag value does not match the receiver's own tag value, the receiver shall silently discard the packet and shall not process it any further except for those cases listed in Section 8.5.1 below.

8.5.1 Exceptions in Verification Tag Rules

A) Rules for packet carrying INIT:

- The sender MUST set the Verification Tag of the packet to 0.

- When an endpoint receives an SCTP packet with the Verification Tag set to 0, it should verify that the packet contains only an INIT chunk. Otherwise, the receiver MUST silently discard the packet.

B) Rules for packet carrying ABORT:

- The endpoint shall always fill in the Verification Tag field of the outbound packet with the destination endpoint's tag value if it is known.

- If the ABORT is sent in response to an OOTB packet, the endpoint MUST follow the procedure described in Section 8.4.

- The receiver MUST accept the packet if the Verification Tag matches either its own tag, OR the tag of its peer. Otherwise, the receiver MUST silently discard the packet and take no further action.

C) Rules for packet carrying SHUTDOWN COMPLETE:

- When sending a SHUTDOWN COMPLETE, if the receiver of the SHUTDOWN ACK has a TCB then the destination endpoint's tag MUST be used. Only where no TCB exists should the sender use the Verification Tag from the SHUTDOWN ACK.

- The receiver of a SHUTDOWN COMPLETE shall accept the packet if the Verification Tag field of the packet matches its own tag OR it is set to its peer's tag and the T bit is set in the Chunk Flags. Otherwise, the receiver MUST silently discard the packet and take no further action. An endpoint MUST ignore the

SHUTDOWN COMPLETE if it is not in the SHUTDOWN-ACK-SENT state.

D) Rules for packet carrying a COOKIE ECHO

- When sending a COOKIE ECHO, the endpoint MUST use the value of the Initial Tag received in the INIT ACK.

- The receiver of a COOKIE ECHO follows the procedures in Section 5.

E) Rules for packet carrying a SHUTDOWN ACK

- If the receiver is in COOKIE-ECHOED or COOKIE-WAIT state the procedures in section 8.4 SHOULD be followed, in other words it should be treated as an Out Of The Blue packet.

9. 偶联终止

An endpoint should terminate its association when it exits from service. An association can be terminated by either abort or shutdown. An abort of an association is abortive by definition in that any data pending on either end of the association is discarded and not delivered to the peer. A shutdown of an association is considered a graceful close where all data in queue by either endpoint is delivered to the respective peers. However, in the case of a shutdown, SCTP does not support a half-open state (like TCP) wherein one side may continue sending data while the other end is closed. When either endpoint performs a shutdown, the association on each peer will stop accepting new data from its user and only deliver data in queue at the time of sending or receiving the SHUTDOWN chunk.

9.1 Abort of an Association

当端点决定 abort 一个已存在的偶联时，它会发送 ABORT 块到对端。发送者必须在输出包中填写对端的 Verification Tag 并且不要把 ABORT 和任何 DATA 捆绑。

端点千万不要响应任何包含 ABORT 块的包(见 8.4)。

接收 ABORT 的端点会应用描述在 8.5.1 中的 Verification Tag 检察规则。

在检查了 Verification Tag 之后，接受端点将删掉相应的偶联记录，并向上层报告偶联终止。

9.2 Shutdown of an Association

使用 SHUTDOWN 原语(见 10.1)，端点的上层能正常关闭偶联。这允许来自对端的所有未被递交的 DATA 块在偶联终止之前被递交。

接收到来自上层的 SHUTDOWN 原语后，端点进入 SHUTDOWN-PENDING 状态并保持直至所有 outstanding data 被对端证实。此时，端点不从它的上层接收任何新的数据，但会对 gaps 进行重传。

一旦所有的 outstanding data 已被证实，端点会向对端发送 SHUTDOWN 块，并在 Cumulative TSN Ack 字段中包含它从对端接收到的最近（last）的 TSN。接着，它会启动 T2-shutdown 定时器并进入 SHUTDOWN-SENT 状态。如果定时器超时，端点必须向对端重传 SHUTDOWN，但带有最新的它从对端接收到的最近（last）的 TSN。

T2-shutdown 定时器如何取值必须遵守 6.3 中的规则。为了指明 TSN 中的任何 gaps，端点也可以在同一个 SCTP 包内捆绑一个 SACK 块。

SHUTDOWN 块被重传的次数取决于协议参数'Association.Max.Retrans'。如果超过此值，端点应销毁 TCB 并向上层报告对端不可达(偶联进入 CLOSED 状态)。来自对端的任何包(如，对端发送它对类中的 DATA 块)都应该重置端点的重传计数器并重起 T2-Shutdown 定时器，giving its peer ample opportunity to transmit all of its queued DATA chunks that have not yet been sent.

一旦接收到 SHUTDOWN, 对端会

- enter the SHUTDOWN-RECEIVED state,

- stop accepting new data from its SCTP user

- verify, by checking the Cumulative TSN Ack field of the chunk, that all its outstanding DATA chunks have been received by the SHUTDOWN sender.

Once an endpoint as reached the SHUTDOWN-RECEIVED state it MUST NOT send a SHUTDOWN in response to a ULP request, and should discard subsequent SHUTDOWN chunks.

如果仍有 outstanding DATA 块,SHUTDOWN 的接收者会继续进行正常的数据传输过程(定义在第 6 节)，直到所有的 outstanding DATA 块被证实；然而，SHUTDOWN 的接收者千万不要再从它的上层接收新数据。

While in SHUTDOWN-SENT state, the SHUTDOWN sender MUST immediately respond to each received packet containing one or more DATA chunk(s) with a SACK, a SHUTDOWN chunk, and restart the T2-shutdown timer. If

it has no more outstanding DATA chunks, the SHUTDOWN receiver shall send a SHUTDOWN ACK and start a T2-shutdown timer of its own, entering the SHUTDOWN-ACK-SENT state. If the timer expires, the endpoint must re-send the SHUTDOWN ACK.

SHUTDOWN ACK 块被重传的次数取决于协议参数'Association.Max.Retrans'。如果超过此值，端点应销毁 TCB 并向上层报告对端不可达(偶联进入 CLOSED 状态)。

接收到 SHUTDOWN ACK 之后, SHUTDOWN 的发送这会停止 T2-shutdown 定时器，向对端发送 SHUTDOWN COMPLETE 并删掉偶联的所有记录。

接收到 SHUTDOWN COMPLETE 后，端点将检查自己是否处于 SHUTDOWN-ACK-SENT 状态。若不是，块被丢弃；否则端点应该停止 T2-shutdown 定时器，删掉偶联的所有记录(偶联进入 CLOSED 状态)。

端点应该确保在发起 shutdown 程序之前，它的所有 outstanding DATA 块已被证实。

处于 SHUTDOWN-PENDING, SHUTDOWN-SENT, SHUTDOWN-RECEIVED, 或 SHUTDOWN-ACK-SENT 状态的端点应拒绝任何新的来自上层的数据。

处于 SHUTDOWN-ACK-SENT 状态的端点接收到 INIT(如, SHUTDOWN COMPLETE 丢失)块（匹配于此偶联)，应丢弃 INIT 并重传 SHUTDOWN ACK。

注意: Receipt of an INIT with the same source and destination IP addresses as used in transport addresses assigned to an endpoint but with a different port number indicates the initialization of a separate association.

The sender of the INIT or COOKIE ECHO should respond to the receipt of a SHUTDOWN-ACK with a stand-alone SHUTDOWN COMPLETE in an SCTP packet with the Verification Tag field of its common header set to the same tag that was received in the SHUTDOWN ACK packet. This is considered an Out of the Blue packet as defined in Section 8.4. The sender of the INIT lets T1-init continue running and remains in the COOKIE-WAIT or COOKIE-ECHOED state. Normal T1-init timer expiration will cause the INIT or COOKIE chunk to be retransmitted and thus start a new association.

If a SHUTDOWN is received in COOKIE WAIT or COOKIE ECHOED states the SHUTDOWN chunk SHOULD be silently discarded.

If an endpoint is in SHUTDOWN-SENT state and receives a SHUTDOWN chunk from its peer, the endpoint shall respond immediately with a SHUTDOWN ACK to its peer, and move into a SHUTDOWN-ACK-SENT state

restarting its T2-shutdown timer.

If an endpoint is in the SHUTDOWN-ACK-SENT state and receives a SHUTDOWN ACK, it shall stop the T2-shutdown timer, send a SHUTDOWN COMPLETE chunk to its peer, and remove all record of the association.

10. 与上层的接口

ULP 通过原语请求 SCTP 提供服务并就各种事件接收 SCTP 的通知。

本部分所描述的原语和通知应该被作为实现 SCTP 的一个指导。下列 ULP 接口原语的功能描述仅是说明性的，不同的 SCTP 实现可以有不同的 ULP 接口。然而，所有 SCTP 实现必须提供一个最小集合的服务以保证它们实现能支持相同的协议体系。

10.1 ULP-to-SCTP

下面功能性的描述了 ULP/SCTP 接口，符号类似于大多数高级语言的过程或函数调用。

下面描述的 ULP 原语指定了 SCTP 必需实现的基本功能以支持进程间通信。不同的实现可以定义它们自己的具体格式，并且可以在单个调用中提供基本功能的组合或子集。

A) Initialize

Format: INITIALIZE ([local port], [local eligible address list]) -> local SCTP instance name

允许 SCTP 初始化内部数据结构，分配必需的资源以创建 SCTP 的运营环境。一旦 SCTP 被初始化，ULP 能直接和其他端点通信而无须重新调用这个原语。

SCTP will return a local SCTP instance name to the ULP.

Mandatory attributes:

None.

Optional attributes:

The following types of attributes may be passed along with the primitive:

o local port - SCTP port number, if ULP wants it to be specified;

o local eligible address list - An address list that the local SCTP endpoint should bind. By default, if an address list is not included, all IP addresses assigned to the host should be used by the local endpoint.

实现注意: 如果可选的属性被实现支持，实现将确保由这个端点发出的任何 SCTP 包的 IP 源地址字段包含在本地有效地址列表中的一个 IP 地址。

B) Associate

Format: ASSOCIATE(local SCTP instance name, destination transport addr, outbound stream count)
-> association id [,destination transport addr list] [,outbound stream count]

允许上层发起一个到指定对端的偶联。

对端由标识端点的一个传输地址指定(见 1.4)。如果本地 SCTP 实例还没有被初始化，ASSOCIATE 被认为一个错误。

偶联成功建立后，返回一个偶联 id（SCTP 偶联的本地句柄)。如果 SCTP 不能和对端建立偶联，一个错误被返回。

其它的偶联参数也可被返回，包括对端完整的传输地址列表，本地端点的输出流个数（实际值，min(local wish, peer offer))。本地端点从返回的对端传输地址列表中选择一个作为发往对端的 SCTP 包的缺省主通路。ULP 可以根据对端传输地址列表改变缺省主通路或强制一个包发往一个特定的传输地址。

实现注意: 如果 ASSOCIATE 被实现为一个阻塞函数调用，偶联建立成功时能返回所有信息（偶联 id 和偶联参数)。如果被实现为一个非阻塞函数调用，仅仅偶联 id 被返回，偶联参数会通过 COMMUNICATION UP 通知传递。

Mandatory attributes:

o local SCTP instance name - obtained from the INITIALIZE operation.

o destination transport addr - specified as one of the transport addresses of the peer endpoint with which the association is to be established.

o outbound stream count - ULP 希望打开到对端的输出流的个数。

Optional attributes:

None.

C) Shutdown

Format: SHUTDOWN(association id)
-> result

正常关闭偶联。任何已在队列中的用户数据都将被递送到对端。仅当对端证实了所有被发送的 SCTP 包后才终止偶联。偶联正常终止成功时返回一个成功码，失败时返回一个错误码。

Mandatory attributes:

o association id - local handle to the SCTP association

Optional attributes:

None.

D) Abort

Format: ABORT(association id [, cause code])
-> result

异常关闭偶联。任何已在队列中的用户数据都将被丢弃，并且一个 ABORT 块被发往对端。偶联异常终止成功时返回一个成功码，失败时返回一个错误码。

Mandatory attributes:

o association id - local handle to the SCTP association

Optional attributes:

o cause code - reason of the abort to be passed to the peer.

None.

E) Send

Format: SEND(association id, buffer address, byte count [,context]
        [,stream id] [,life time] [,destination transport address]
        [,unorder flag] [,no-bundle flag] [,payload protocol-id])
-> result

This is the main method to send user data via SCTP.

Mandatory attributes:

o association id - local handle to the SCTP association

o buffer address - the location where the user message to be transmitted is stored;

o byte count - The size of the user data in number of bytes;

Optional attributes:

o context - 可选的 32 位整数。此次用户消息传输失败时，被包含在发往 ULP 的失败通知中。

o stream id - to indicate which stream to send the data on. If not specified, stream 0 will be used.

o life time - 指定用户数据的生存期限。生存期限过期后用户数据将不会被 SCTP 尝试发送。此参数能被用来避免发送旧（过期）的用户数据。如果用户数据在生存期限内没有被尝试传输，SCTP 将通知 ULP。然而,如果在生存期限过期之前 SCTP 已经进行了尝试传送，则用户数据将被传送。

实现注意: 为了更好的支持数据生存期限选项，发送者可以直到最后时刻才分配 TSN 给输出的 DATA 块。并且,为实现简单，一旦 TSN 被分配，发送者应该认为发送 DATA 块的请求已被提交，关联于此 DATA 块的生存期限选项无效。

o destination transport address - 指定对端的一个传输地址作为这个包的目的地。无论怎样，SCTP 应该用这个目的传输地址（而不是主通路）发送这个包。

o unorder flag - this flag, if present, indicates that the user would like the data delivered in an unordered fashion to the peer (i.e., the U flag is set to 1 on all DATA chunks carrying this message).

o no-bundle flag - 指示 SCTP 不要把此用户数据和其它的输出 DATA 块进行捆绑。但当网络拥赛时，SCTP 仍然可以捆绑，即使此标记出现。

o payload protocol-id - 32 位无符号整数，传递给对端用来指定被传递净荷协议数据的类型。它被 SCTP 透明传送。

F) Set Primary

Format: SETPRIMARY(association id, destination transport address,
[source transport address])
-> result

指示 SCTP 用指定的目的传输地址作为发送包的主通路。

此操作的结果应被返回。如果指定的目的传输地址不出现在 Associate 命令返回的或 COMMUNICATION UP 通知中的"目的传输地址列表"中，一个错误将被返回。

Mandatory attributes:

o association id - local handle to the SCTP association

o destination transport address - specified as one of the transport addresses of the peer endpoint, which should be used as primary address for sending packets. This overrides the current primary address information maintained by the local SCTP endpoint.

Optional attributes:

o source transport address - 可选的, 一些实现允许你设置缺省源地址（出现在所有的输出 IP 包中)。

G) Receive

Format: RECEIVE(association id, buffer address, buffer size
[,stream id])
-> byte count [,transport address] [,stream id] [,stream sequence number] [,partial flag] [,delivery number] [,payload protocol-id]

从 SCTP 的输入队列把第一个用户消息（如果存在的话）读入到由 ULP 指定的缓存中。被读的消息大小（字节）将被返回。依赖于实现，也可返回其它信息，如：发送者的地址，消息被接收的流号及是否存在更多的消息可读取等。如是有序消息的话，SSN 也可被返回。

依赖于实现, 如果输入队列中没有消息可读取，实现应该返回一个此状态的指示（非阻塞调用）或阻塞直到用户数据变的可得到（阻塞调用)。

Mandatory attributes:

o association id - local handle to the SCTP association

o buffer address - the memory location indicated by the ULP to store

the received message.

o buffer size - the maximum size of data to be received, in bytes.

Optional attributes:

o stream id - to indicate which stream to receive the data on.

o stream sequence number - the stream sequence number assigned by the sending SCTP peer.

o partial flag - 如果此标记（被返回）为 1,则此次 Receive 只包含整个消息的一部分。此时，流号 S 与 SSN 必须一起返回。如果为 0, 表明此 SSN 没有更多的片段将被接收。

o payload protocol-id - 32 位无符号整数，从对端接收，用来指定接收到的净荷协议数据的类型。它被 SCTP 透明传送。

H) Status

Format: STATUS(association id)
-> status data

此原语应该返回包含下列信息的数据块：
association connection state,
destination transport address list,
destination transport address reachability states,
current receiver window size,
current congestion window sizes,
number of unacknowledged DATA chunks,
number of DATA chunks pending receipt,
primary path,
most recent SRTT on primary path,
RTO on primary path,
SRTT and RTO on other destination addresses, etc.

Mandatory attributes:

o association id - local handle to the SCTP association

Optional attributes:

None.

I) Change Heartbeat

Format: CHANGEHEARTBEAT(association id, destination transport address,
        new state [,interval])
-> result

指示 SCTP 在指定的目的传输地址上启动或停止心跳。

The result of attempting this operation shall be returned.

注意: Even when enabled, heartbeat will not take place if the
destination transport address is not idle.

Mandatory attributes:

o association id - local handle to the SCTP association

o destination transport address - specified as one of the transport
   addresses of the peer endpoint.

o new state - the new state of heartbeat for this destination
   transport address (either enabled or disabled).

Optional attributes:

o interval - if present, indicates the frequency of the heartbeat if
   this is to enable heartbeat on a destination transport address.
   This value is added to the RTO of the destination transport
   address. This value, if present, effects all destinations.

J) Request HeartBeat

Format: REQUESTHEARTBEAT(association id, destination transport
        address)
-> result

指示本地端点对给定偶联上指定目的传输地址执行一次心跳。返回结果表明发往目的地址的 HEARTBEAT 块是否成功。

Mandatory attributes:

o association id - local handle to the SCTP association

o destination transport address - the transport address of the

association on which a heartbeat should be issued.

K) Get SRTT Report

Format: GETSRTTREPORT(association id, destination transport address)
-> srtt result

指示本地端点报告给定偶联上指定目的传输地址的当前 SRTT measurement。
The returned result can be an integer containing the most recent SRTT in milliseconds.

Mandatory attributes:

o association id - local handle to the SCTP association

o destination transport address - the transport address of the association on which the SRTT measurement is to be reported.

L) Set Failure Threshold

Format: SETFAILURETHRESHOLD(association id, destination transport address, failure threshold)
-> result

This primitive allows the local SCTP to customize the reachability failure detection threshold 'Path.Max.Retrans' for the specified destination address.

Mandatory attributes:

o association id - local handle to the SCTP association

o destination transport address - the transport address of the association on which the failure detection threshold is to be set.

o failure threshold - the new value of 'Path.Max.Retrans' for the destination address.

M) Set Protocol Parameters

Format: SETPROTOCOLPARAMETERS(association id, [,destination transport address,] protocol parameter list)
-> result

允许客户化协议参数。

Mandatory attributes:

o association id - local handle to the SCTP association

o protocol parameter list - The specific names and values of the protocol parameters (e.g., Association.Max.Retrans [see Section 14]) that the SCTP user wishes to customize.

Optional attributes:

o destination transport address - 一些协议参数可以在目的传输地址的层次上被设置。

N) Receive unsent message

Format: RECEIVE_UNSENT(data retrieval id, buffer address, buffer size [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

o data retrieval id - The identification passed to the ULP in the failure notification.

o buffer address - the memory location indicated by the ULP to store the received message.

o buffer size - the maximum size of data to be received, in bytes.

Optional attributes:

o stream id - this is a return value that is set to indicate which stream the data was sent to.

o stream sequence number - this value is returned indicating the stream sequence number that was associated with the message.

o partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and stream sequence number MUST accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this stream sequence number.

- payload protocol-id - The 32 bit unsigned integer that was sent to be sent to the peer indicating the type of payload protocol of the received data.

O) Receive unacknowledged message

Format: RECEIVE_UNACKED(data retrieval id, buffer address, buffer size, [,stream id] [, stream sequence number] [,partial flag] [,payload protocol-id])

- data retrieval id - The identification passed to the ULP in the failure notification.

- buffer address - the memory location indicated by the ULP to store the received message.

- buffer size - the maximum size of data to be received, in bytes.

Optional attributes:

- stream id - this is a return value that is set to indicate which stream the data was sent to.

- stream sequence number - this value is returned indicating the stream sequence number that was associated with the message.

- partial flag - if this returned flag is set to 1, then this message is a partial delivery of the whole message. When this flag is set, the stream id and stream sequence number MUST accompany this receive. When this flag is set to 0, it indicates that no more deliveries will be received for this stream sequence number.

- payload protocol-id - The 32 bit unsigned integer that was sent to be sent to the peer indicating the type of payload protocol of the received data.

P) Destroy SCTP instance

Format: DESTROY(local SCTP instance name)

- local SCTP instance name - this is the value that was passed to the application in the initialize primitive and it indicates which SCTP instance to be destroyed.

10.2 SCTP-to-ULP

It is assumed that the operating system or application environment provides a means for the SCTP to asynchronously signal the ULP process. When SCTP does signal an ULP process, certain information is passed to the ULP.

IMPLEMENTATION NOTE: In some cases this may be done through a separate socket or error channel.

A) DATA ARRIVE notification

SCTP shall invoke this notification on the ULP when a user message is successfully received and ready for retrieval.

The following may be optionally be passed with the notification:

- association id - local handle to the SCTP association
- stream id - to indicate which stream the data is received on.

B) SEND FAILURE notification

If a message can not be delivered SCTP shall invoke this notification on the ULP.

The following may be optionally be passed with the notification:

- association id - local handle to the SCTP association
- data retrieval id - an identification used to retrieve unsent and unacknowledged data.
- cause code - indicating the reason of the failure, e.g., size too large, message life-time expiration, etc.
- context - optional information associated with this message (see D in Section 10.1).

C) NETWORK STATUS CHANGE notification

When a destination transport address is marked inactive (e.g., when SCTP detects a failure), or marked active (e.g., when SCTP detects a

recovery), SCTP shall invoke this notification on the ULP.

The following shall be passed with the notification:

- association id - local handle to the SCTP association
- destination transport address - This indicates the destination transport address of the peer endpoint affected by the change;
- new-status - This indicates the new status.

D) COMMUNICATION UP notification

This notification is used when SCTP becomes ready to send or receive user messages, or when a lost communication to an endpoint is restored.

IMPLEMENTATION NOTE: If ASSOCIATE primitive is implemented as a blocking function call, the association parameters are returned as a result of the ASSOCIATE primitive itself. In that case, COMMUNICATION UP notification is optional at the association initiator's side.

The following shall be passed with the notification:

- association id - local handle to the SCTP association
- status - This indicates what type of event has occurred
- destination transport address list - the complete set of transport addresses of the peer
- outbound stream count - the maximum number of streams allowed to be used in this association by the ULP
- inbound stream count - the number of streams the peer endpoint has requested with this association (this may not be the same number as 'outbound stream count').

E) COMMUNICATION LOST notification

When SCTP loses communication to an endpoint completely (e.g., via Heartbeats) or detects that the endpoint has performed an abort operation, it shall invoke this notification on the ULP.

The following shall be passed with the notification:

o association id - local handle to the SCTP association

o status - This indicates what type of event has occurred; The status may indicate a failure OR a normal termination event occurred in response to a shutdown or abort request.

The following may be passed with the notification:

o data retrieval id - an identification used to retrieve unsent and unacknowledged data.

o last-acked - the TSN last acked by that peer endpoint;

o last-sent - the TSN last sent to that peer endpoint;

F) COMMUNICATION ERROR notification

When SCTP receives an ERROR chunk from its peer and decides to notify its ULP, it can invoke this notification on the ULP.

The following can be passed with the notification:

o association id - local handle to the SCTP association

o error info - this indicates the type of error and optionally some additional information received through the ERROR chunk.

G) RESTART notification

When SCTP detects that the peer has restarted, it may send this notification to its ULP.

The following can be passed with the notification:

o association id - local handle to the SCTP association

H) SHUTDOWN COMPLETE notification

When SCTP completes the shutdown procedures (section 9.2) this notification is passed to the upper layer.

The following can be passed with the notification:

o association id - local handle to the SCTP association

11. 安全考虑

11.1 Security Objectives

As a common transport protocol designed to reliably carry time-sensitive user messages, such as billing or signaling messages for telephony services, between two networked endpoints, SCTP has the following security objectives.

- availability of reliable and timely data transport services
- integrity of the user-to-user information carried by SCTP

11.2 SCTP Responses To Potential Threats

SCTP may potentially be used in a wide variety of risk situations. It is important for operator(s) of systems running SCTP to analyze their particular situations and decide on the appropriate counter-measures.

Operators of systems running SCTP should consult [RFC2196] for guidance in securing their site.

11.2.1 Countering Insider Attacks

The principles of [RFC2196] should be applied to minimize the risk of theft of information or sabotage by insiders. Such procedures include publication of security policies, control of access at the physical, software, and network levels, and separation of services.

11.2.2 Protecting against Data Corruption in the Network

Where the risk of undetected errors in datagrams delivered by the lower layer transport services is considered to be too great, additional integrity protection is required. If this additional protection were provided in the application-layer, the SCTP header would remain vulnerable to deliberate integrity attacks. While the existing SCTP mechanisms for detection of packet replays are considered sufficient for normal operation, stronger protections are needed to protect SCTP when the operating environment contains significant risk of deliberate attacks from a sophisticated adversary.

In order to promote software code-reuse, to avoid re-inventing the wheel, and to avoid gratuitous complexity to SCTP, the IP Authentication Header [RFC2402] SHOULD be used when the threat environment requires stronger integrity protections, but does not require confidentiality.

A widely implemented BSD Sockets API extension exists for applications to request IP security services, such as AH or ESP from an operating system kernel. Applications can use such an API to request AH whenever AH use is appropriate.

11.2.3 Protecting Confidentiality

In most cases, the risk of breach of confidentiality applies to the signaling data payload, not to the SCTP or lower-layer protocol overheads. If that is true, encryption of the SCTP user data only might be considered. As with the supplementary checksum service, user data encryption MAY be performed by the SCTP user application.

Alternately, the user application MAY use an implementation-specific API to request that the IP Encapsulating Security Payload (ESP) [RFC2406] be used to provide confidentiality and integrity.

Particularly for mobile users, the requirement for confidentiality might include the masking of IP addresses and ports. In this case ESP SHOULD be used instead of application-level confidentiality. If ESP is used to protect confidentiality of SCTP traffic, an ESP cryptographic transform that includes cryptographic integrity protection MUST be used, because if there is a confidentiality threat there will also be a strong integrity threat.

Whenever ESP is in use, application-level encryption is not generally required.

Regardless of where confidentiality is provided, the ISAKMP [RFC2408] and the Internet Key Exchange (IKE) [RFC2409] SHOULD be used for key management.

Operators should consult [RFC2401] for more information on the security services available at and immediately above the Internet Protocol layer.

11.2.4 Protecting against Blind Denial of Service Attacks

A blind attack is one where the attacker is unable to intercept or otherwise see the content of data flows passing to and from the target SCTP node. Blind denial of service attacks may take the form of flooding, masquerade, or improper monopolization of services.

11.2.4.1 Flooding

The objective of flooding is to cause loss of service and incorrect behavior at target systems through resource exhaustion, interference with legitimate transactions, and exploitation of buffer-related software bugs. Flooding may be directed either at the SCTP node or at resources in the intervening IP Access Links or the Internet. Where the latter entities are the target, flooding will manifest itself as loss of network services, including potentially the breach of any firewalls in place.

In general, protection against flooding begins at the equipment design level, where it includes measures such as:

- avoiding commitment of limited resources before determining that the request for service is legitimate

- giving priority to completion of processing in progress over the acceptance of new work

- identification and removal of duplicate or stale queued requests for service.

- not responding to unexpected packets sent to non-unicast addresses.

Network equipment should be capable of generating an alarm and log if a suspicious increase in traffic occurs. The log should provide information such as the identity of the incoming link and source address(es) used which will help the network or SCTP system operator to take protective measures. Procedures should be in place for the operator to act on such alarms if a clear pattern of abuse emerges.

The design of SCTP is resistant to flooding attacks, particularly in its use of a four-way start-up handshake, its use of a cookie to defer commitment of resources at the responding SCTP node until the handshake is completed, and its use of a Verification Tag to prevent insertion of extraneous packets into the flow of an established association.

The IP Authentication Header and Encapsulating Security Payload might be useful in reducing the risk of certain kinds of denial of service attacks."

The use of the Host Name feature in the INIT chunk could be used to flood a target DNS server. A large backlog of DNS queries, resolving the Host Name received in the INIT chunk to IP addresses, could be accomplished by sending INIT's to multiple hosts in a given domain. In addition, an attacker could use the Host Name feature in an indirect attack on a third party by sending large numbers of INITs to random hosts containing the host name of the target. In addition to the strain on DNS resources, this could also result in large numbers of INIT ACKs being sent to the target. One method to protect against this type of attack is to verify that the IP addresses received from DNS include the source IP address of the original INIT. If the list of IP addresses received from DNS does not include the source IP address of the INIT, the endpoint MAY silently discard the INIT. This last option will not protect against the attack against the DNS.

11.2.4.2 Blind Masquerade

Masquerade can be used to deny service in several ways:

- by tying up resources at the target SCTP node to which the impersonated node has limited access. For example, the target node may by policy permit a maximum of one SCTP association with the impersonated SCTP node. The masquerading attacker may attempt to establish an association purporting to come from the impersonated node so that the latter cannot do so when it requires it.

- by deliberately allowing the impersonation to be detected, thereby provoking counter-measures which cause the impersonated node to be locked out of the target SCTP node.

- by interfering with an established association by inserting extraneous content such as a SHUTDOWN request.

SCTP reduces the risk of blind masquerade attacks through IP spoofing by use of the four-way startup handshake. Man-in-the-middle masquerade attacks are discussed in Section 11.3 below. Because the initial exchange is memoryless, no lockout mechanism is triggered by blind masquerade attacks. In addition, the INIT ACK containing the State Cookie is transmitted back to the IP address from which it received the INIT. Thus the attacker would not receive the INIT ACK containing the State Cookie. SCTP protects against insertion of extraneous packets into the flow of an established association by use of the Verification Tag.

Logging of received INIT requests and abnormalities such as unexpected INIT ACKs might be considered as a way to detect patterns of hostile activity. However, the potential usefulness of such logging must be weighed against the increased SCTP startup processing it implies, rendering the SCTP node more vulnerable to flooding attacks. Logging is pointless without the establishment of operating procedures to review and analyze the logs on a routine basis.

11.2.4.3 Improper Monopolization of Services

Attacks under this heading are performed openly and legitimately by

the attacker. They are directed against fellow users of the target SCTP node or of the shared resources between the attacker and the target node. Possible attacks include the opening of a large number of associations between the attacker's node and the target, or transfer of large volumes of information within a legitimately-established association.

Policy limits should be placed on the number of associations per adjoining SCTP node. SCTP user applications should be capable of detecting large volumes of illegitimate or "no-op" messages within a given association and either logging or terminating the association as a result, based on local policy.

11.3 Protection against Fraud and Repudiation

The objective of fraud is to obtain services without authorization and specifically without paying for them. In order to achieve this objective, the attacker must induce the SCTP user application at the target SCTP node to provide the desired service while accepting invalid billing data or failing to collect it. Repudiation is a related problem, since it may occur as a deliberate act of fraud or simply because the repudiating party kept inadequate records of service received.

Potential fraudulent attacks include interception and misuse of authorizing information such as credit card numbers, blind masquerade and replay, and man-in-the middle attacks which modify the packets passing through a target SCTP association in real time.

The interception attack is countered by the confidentiality measures discussed in Section 11.2.3 above.

Section 11.2.4.2 describes how SCTP is resistant to blind masquerade attacks, as a result of the four-way startup handshake and the Verification Tag. The Verification Tag and TSN together are protections against blind replay attacks, where the replay is into an existing association.

However, SCTP does not protect against man-in-the-middle attacks where the attacker is able to intercept and alter the packets sent and received in an association. For example, the INIT ACK will have sufficient information sent on the wire for an adversary in the middle to hijack an existing SCTP association. Where a significant possibility of such attacks is seen to exist, or where possible repudiation is an issue, the use of the IPSEC AH service is recommended to ensure both the integrity and the authenticity of the SCTP packets passed.

SCTP also provides no protection against attacks originating at or beyond the SCTP node and taking place within the context of an existing association. Prevention of such attacks should be covered by appropriate security policies at the host site, as discussed in Section 11.2.1.

12. Recommended Transmission Control Block (TCB) Parameters

This section details a recommended set of parameters that should be contained within the TCB for an implementation. This section is for illustrative purposes and should not be deemed as requirements on an implementation or as an exhaustive list of all parameters inside an SCTP TCB. Each implementation may need its own additional parameters for optimization.

12.1 Parameters necessary for the SCTP instance

Associations: A list of current associations and mappings to the data consumers for each association. This may be in the form of a hash table or other implementation dependent structure. The data consumers may be process identification information such as file descriptors, named pipe pointer, or table pointers dependent on how SCTP is implemented.

Secret Key: A secret key used by this endpoint to compute the MAC. This SHOULD be a cryptographic quality random number with a sufficient length. Discussion in [RFC1750] can be helpful in selection of the key.

Address List: The list of IP addresses that this instance has bound. This information is passed to one's peer(s) in INIT and INIT ACK chunks.

SCTP Port: The local SCTP port number the endpoint is bound to.

12.2 Parameters necessary per association (i.e. the TCB)

Peer Verification Tag : Tag value to be sent in every packet and is received in the INIT or INIT ACK chunk.

My Verification Tag : Tag expected in every inbound packet and sent in the INIT or INIT ACK chunk.

State : A state variable indicating what state the association

: is in, i.e. COOKIE-WAIT, COOKIE-ECHOED, ESTABLISHED,
: SHUTDOWN-PENDING, SHUTDOWN-SENT, SHUTDOWN-RECEIVED,
: SHUTDOWN-ACK-SENT.

Note: No "CLOSED" state is illustrated since if a
association is "CLOSED" its TCB SHOULD be removed.

```
Peer          : A list of SCTP transport addresses that the peer is
Transport     : bound to.  This information is derived from the INIT or
Address       : INIT ACK and is used to associate an inbound packet
List          : with a given association.  Normally this information is
              : hashed or keyed for quick lookup and access of the TCB.

Primary       : This is the current primary destination transport
Path          : address of the peer endpoint.  It may also specify a
              : source transport address on this endpoint.

Overall       : The overall association error count.
Error Count   :

Overall       : The threshold for this association that if the Overall
Error         : Error Count reaches will cause this association to be
Threshold     : torn down.

Peer Rwnd     : Current calculated value of the peer's rwnd.

Next TSN      : The next TSN number to be assigned to a new DATA chunk.
              : This is sent in the INIT or INIT ACK chunk to the peer
              : and incremented each time a DATA chunk is assigned a
              : TSN (normally just prior to transmit or during
              : fragmentation).

Last Rcvd     : This is the last TSN received in sequence.  This value
TSN           : is set initially by taking the peer's Initial TSN,
              : received in the INIT or INIT ACK chunk, and
              : subtracting one from it.

Mapping       : An array of bits or bytes indicating which out of
Array         : order TSN's have been received (relative to the
              : Last Rcvd TSN).  If no gaps exist, i.e. no out of order
              : packets have been received, this array will be set to
              : all zero.  This structure may be in the form of a
              : circular buffer or bit array.

Ack State     : This flag indicates if the next received packet
              : is to be responded to with a SACK.  This is initialized
              : to 0.  When a packet is received it is incremented.
              : If this value reaches 2 or more, a SACK is sent and the
```

: value is reset to 0. Note: This is used only when no
: DATA chunks are received out of order. When DATA chunks
: are out of order, SACK's are not delayed (see Section
: 6).

Inbound Streams : An array of structures to track the inbound streams. Normally including the next sequence number expected and possibly the stream number.

Outbound Streams : An array of structures to track the outbound streams. Normally including the next sequence number to be sent on the stream.

Reasm Queue : A re-assembly queue.

Local Transport Address List : The list of local IP addresses bound in to this association.

Association PMTU : The smallest PMTU discovered for all of the peer's transport addresses.

12.3 Per Transport Address Data

For each destination transport address in the peer's address list derived from the INIT or INIT ACK chunk, a number of data elements needs to be maintained including:

Error count : The current error count for this destination.

Error Threshold : Current error threshold for this destination i.e. what value marks the destination down if Error count reaches this value.

cwnd : The current congestion window.

ssthresh : The current ssthresh value.

RTO : The current retransmission timeout value.

SRTT : The current smoothed round trip time.

RTTVAR : The current RTT variation.

partial : The tracking method for increase of cwnd when in

bytes acked : congestion avoidance mode (see Section 6.2.2)

state : The current state of this destination, i.e. DOWN, UP,
: ALLOW-HB, NO-HEARTBEAT, etc.

PMTU : The current known path MTU.

Per Destination Timer : A timer used by each destination.

RTO-Pending : A flag used to track if one of the DATA chunks sent to this address is currently being used to compute a RTT. If this flag is 0, the next DATA chunk sent to this destination should be used to compute a RTT and this flag should be set. Every time the RTT calculation completes (i.e. the DATA chunk is SACK'd) clear this flag.

last-time used : The time this destination was last sent to. This can be : used to determine if a HEARTBEAT is needed.

12.4 General Parameters Needed

Out Queue : A queue of outbound DATA chunks.

In Queue : A queue of inbound DATA chunks.

13. IANA Considerations

This protocol will require port reservation like TCP for the use of "well known" servers within the Internet. All current TCP ports shall be automatically reserved in the SCTP port address space. New requests should follow IANA's current mechanisms for TCP.

This protocol may also be extended through IANA in three ways:

-- through definition of additional chunk types,
-- through definition of additional parameter types, or
-- through definition of additional cause codes within ERROR chunks

In the case where a particular ULP using SCTP desires to have its own ports, the ULP should be responsible for registering with IANA for getting its ports assigned.

13.1 IETF-defined Chunk Extension

The definition and use of new chunk types is an integral part of SCTP. Thus, new chunk types are assigned by IANA through an IETF Consensus action as defined in [RFC2434].

The documentation for a new chunk code type must include the following information:

a) A long and short name for the new chunk type;

b) A detailed description of the structure of the chunk, which MUST conform to the basic structure defined in Section 3.2;

c) A detailed definition and description of intended use of each field within the chunk, including the chunk flags if any;

d) A detailed procedural description of the use of the new chunk type within the operation of the protocol.

The last chunk type (255) is reserved for future extension if necessary.

13.2 IETF-defined Chunk Parameter Extension

The assignment of new chunk parameter type codes is done through an IETF Consensus action as defined in [RFC2434]. Documentation of the chunk parameter MUST contain the following information:

a) Name of the parameter type.

b) Detailed description of the structure of the parameter field. This structure MUST conform to the general type-length-value format described in Section 3.2.1.

c) Detailed definition of each component of the parameter value.

d) Detailed description of the intended use of this parameter type, and an indication of whether and under what circumstances multiple instances of this parameter type may be found within the same chunk.

13.3 IETF-defined Additional Error Causes

Additional cause codes may be allocated in the range 11 to 65535 through a Specification Required action as defined in [RFC2434]. Provided documentation must include the following information:

a) Name of the error condition.

b) Detailed description of the conditions under which an SCTP endpoint should issue an ERROR (or ABORT) with this cause code.

c) Expected action by the SCTP endpoint which receives an ERROR (or ABORT) chunk containing this cause code.

d) Detailed description of the structure and content of data fields which accompany this cause code.

The initial word (32 bits) of a cause code parameter MUST conform to the format shown in Section 3.3.10, i.e.:

-- first two bytes contain the cause code value
-- last two bytes contain length of the Cause Parameter.

13.4 Payload Protocol Identifiers

Except for value 0 which is reserved by SCTP to indicate an unspecified payload protocol identifier in a DATA chunk, SCTP will not be responsible for standardizing or verifying any payload protocol identifiers; SCTP simply receives the identifier from the upper layer and carries it with the corresponding payload data.

The upper layer, i.e., the SCTP user, SHOULD standardize any specific protocol identifier with IANA if it is so desired. The use of any specific payload protocol identifier is out of the scope of SCTP.

14. Suggested SCTP Protocol Parameter Values

The following protocol parameters are RECOMMENDED:

RTO.Initial	- 3 seconds
RTO.Min	- 1 second
RTO.Max	- 60 seconds
RTO.Alpha	- 1/8
RTO.Beta	- 1/4
Valid.Cookie.Life	- 60 seconds
Association.Max.Retrans	- 10 attempts
Path.Max.Retrans	- 5 attempts (per destination address)
Max.Init.Retransmits	- 8 attempts
HB.interval	- 30 seconds

IMPLEMENTATION NOTE: The SCTP implementation may allow ULP to customize some of these protocol parameters (see Section 10).

Note: RTO.Min SHOULD be set as recommended above.

15. Acknowledgements

The authors wish to thank Mark Allman, R.J. Atkinson, Richard Band, Scott Bradner, Steve Bellovin, Peter Butler, Ram Dantu, R. Ezhirpavai, Mike Fisk, Sally Floyd, Atsushi Fukumoto, Matt Holdrege, Henry Houh, Christian Huitema, Gary Lehecka, Jonathan Lee, David Lehmann, John Loughney, Daniel Luan, Barry Nagelberg, Thomas Narten, Erik Nordmark, Lyndon Ong, Shyamal Prasad, Kelvin Porter, Heinz Prantner, Jarno Rajahalme, Raymond E. Reeves, Renee Revis, Ivan Arias Rodriguez, A. Sankar, Greg Sidebottom, Brian Wyld, La Monte Yarroll, and many others for their invaluable comments.

16. Authors' Addresses

Randall R. Stewart
24 Burning Bush Trail.
Crystal Lake, IL 60012
USA

Phone: +1-815-477-2127
EMail: rrs@cisco.com

Qiaobing Xie
Motorola, Inc.
1501 W. Shure Drive, #2309
Arlington Heights, IL 60004
USA

Phone: +1-847-632-3028
EMail: qxie1@email.mot.com

Ken Morneault
Cisco Systems Inc.
13615 Dulles Technology Drive
Herndon, VA. 20171
USA

Phone: +1-703-484-3323
EMail: kmorneau@cisco.com

Chip Sharp
Cisco Systems Inc.
7025 Kit Creek Road
Research Triangle Park, NC 27709
USA

Phone: +1-919-392-3121
EMail: chsharp@cisco.com

Hanns Juergen Schwarzbauer
SIEMENS AG
Hofmannstr. 51
81359 Munich
Germany

Phone: +49-89-722-24236
EMail: HannsJuergen.Schwarzbauer@icn.siemens.de

Tom Taylor
Nortel Networks
1852 Lorraine Ave.
Ottawa, Ontario
Canada K1H 6Z8

Phone: +1-613-736-0961
EMail: taylor@nortelnetworks.com

Ian Rytina
Ericsson Australia
37/360 Elizabeth Street
Melbourne, Victoria 3000
Australia

Phone: +61-3-9301-6164
EMail: ian.rytina@ericsson.com

Malleswar Kalla
Telcordia Technologies
3 Corporate Place
PYA-2J-341
Piscataway, NJ 08854
USA

Phone: +1-732-699-3728
EMail: mkalla@telcordia.com

Lixia Zhang
UCLA Computer Science Department
4531G Boelter Hall
Los Angeles, CA 90095-1596
USA

Phone: +1-310-825-2695
EMail: lixia@cs.ucla.edu

Vern Paxson
ACIRI
1947 Center St., Suite 600,
Berkeley, CA 94704-1198
USA

Phone: +1-510-666-2882
EMail: vern@aciri.org

17. References

[RFC768] Postel, J. (ed.), "User Datagram Protocol", STD 6, RFC 768, August 1980.

[RFC793] Postel, J. (ed.), "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1123] Braden, R., "Requirements for Internet hosts - application and support", STD 3, RFC 1123, October 1989.

[RFC1191] Mogul, J. and S. Deering, "Path MTU Discovery", RFC 1191, November 1990.

[RFC1700] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[RFC1981] McCann, J., Deering, S. and J. Mogul, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, August 1996.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2408] Maughan, D., Schertler, M., Schneider, M. and J. Turner, "Internet Security Association and Key Management Protocol", RFC 2408, November 1998.

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2581] Allman, M., Paxson, V. and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

18. Bibliography

[ALLMAN99] Allman, M. and Paxson, V., "On Estimating End-to-End Network Path Properties", Proc. SIGCOMM'99, 1999.

[FALL96] Fall, K. and Floyd, S., Simulation-based Comparisons of

Tahoe, Reno, and SACK TCP, Computer Communications Review, V. 26 N. 3, July 1996, pp. 5-21.

[RFC1750] Eastlake, D. (ed.), "Randomness Recommendations for Security", RFC 1750, December 1994.

[RFC1950] Deutsch P. and J. Gailly, "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, May 1996.

[RFC2104] Krawczyk, H., Bellare, M. and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, March 1997.

[RFC2196] Fraser, B., "Site Security Handbook", FYI 8, RFC 2196, September 1997.

[RFC2522] Karn, P. and W. Simpson, "Photuris: Session-Key Management Protocol", RFC 2522, March 1999.

[SAVAGE99] Savage, S., Cardwell, N., Wetherall, D., and Anderson, T., "TCP Congestion Control with a Misbehaving Receiver", ACM Computer Communication Review, 29(5), October 1999.

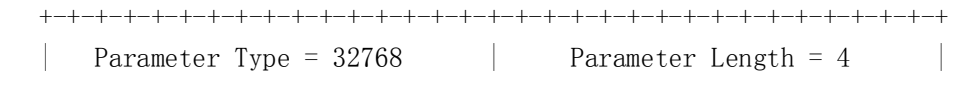
Appendix A: Explicit Congestion Notification

ECN (Ramakrishnan, K., Floyd, S., "Explicit Congestion Notification", RFC 2481, January 1999) describes a proposed extension to IP that details a method to become aware of congestion outside of datagram loss. This is an optional feature that an implementation MAY choose to add to SCTP. This appendix details the minor differences implementers will need to be aware of if they choose to implement this feature. In general RFC 2481 should be followed with the following exceptions.

Negotiation:

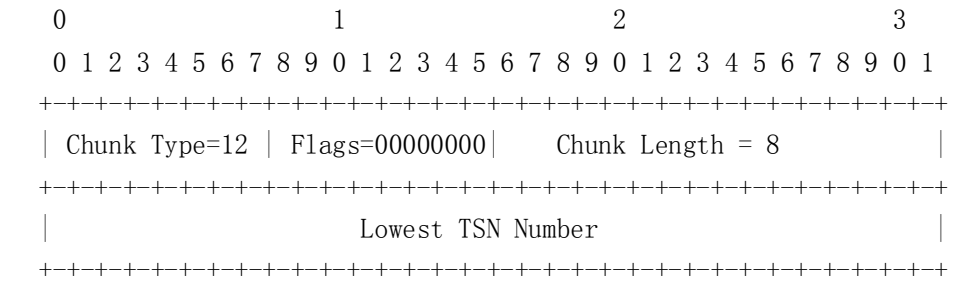
RFC2481 details negotiation of ECN during the SYN and SYN-ACK stages of a TCP connection. The sender of the SYN sets two bits in the TCP flags, and the sender of the SYN-ACK sets only 1 bit. The reasoning behind this is to assure both sides are truly ECN capable. For SCTP this is not necessary. To indicate that an endpoint is ECN capable an endpoint SHOULD add to the INIT and or INIT ACK chunk the TLV reserved for ECN. This TLV contains no parameters, and thus has the following format:

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```



ECN-Echo:

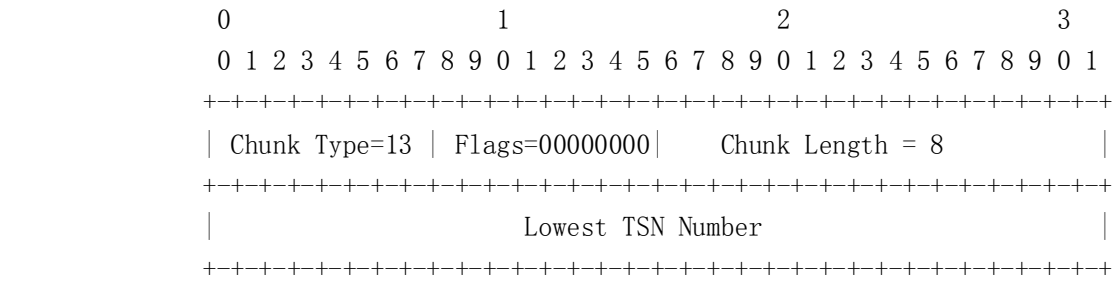
RFC 2481 details a specific bit for a receiver to send back in its TCP acknowledgements to notify the sender of the Congestion Experienced (CE) bit having arrived from the network. For SCTP this same indication is made by including the ECNE chunk. This chunk contains one data element, i.e. the lowest TSN associated with the IP datagram marked with the CE bit, and looks as follows:



Note: The ECNE is considered a Control chunk.

CWR:

RFC 2481 details a specific bit for a sender to send in the header of its next outbound TCP segment to indicate to its peer that it has reduced its congestion window. This is termed the CWR bit. For SCTP the same indication is made by including the CWR chunk. This chunk contains one data element, i.e. the TSN number that was sent in the ECNE chunk. This element represents the lowest TSN number in the datagram that was originally marked with the CE bit.



Note: The CWR is considered a Control chunk.

Appendix B Alder 32 bit checksum calculation

The Adler-32 checksum calculation given in this appendix is copied from [RFC1950].

Adler-32 is composed of two sums accumulated per byte: s1 is the sum of all bytes, s2 is the sum of all s1 values. Both sums are done modulo 65521. s1 is initialized to 1, s2 to zero. The Adler-32 checksum is stored as s2*65536 + s1 in network byte order.

The following C code computes the Adler-32 checksum of a data buffer. It is written for clarity, not for speed. The sample code is in the ANSI C programming language. Non C users may find it easier to read with these hints:

& Bitwise AND operator.
>> Bitwise right shift operator. When applied to an unsigned quantity, as here, right shift inserts zero bit(s) at the left.
<< Bitwise left shift operator. Left shift inserts zero

```
       bit(s) at the right.
++     "n++" increments the variable n.
%      modulo operator: a % b is the remainder of a divided by b.
 #define BASE 65521 /* largest prime smaller than 65536 */
 /*
   Update a running Adler-32 checksum with the bytes buf[0..len-1]
   and return the updated checksum.  The Adler-32 checksum should be
   initialized to 1.

    Usage example:

      unsigned long adler = 1L;

      while (read_buffer(buffer, length) != EOF) {
        adler = update_adler32(adler, buffer, length);
      }
      if (adler != original_adler) error();
   */
   unsigned long update_adler32(unsigned long adler,
      unsigned char *buf, int len)
   {
     unsigned long s1 = adler & 0xffff;
     unsigned long s2 = (adler >> 16) & 0xffff;
     int n;

     for (n = 0; n < len; n++) {
       s1 = (s1 + buf[n]) % BASE;
       s2 = (s2 + s1)     % BASE;
     }
     return (s2 << 16) + s1;
   }

   /* Return the adler32 of the bytes buf[0..len-1] */
   unsigned long adler32(unsigned char *buf, int len)
   {
     return update_adler32(1L, buf, len);
   }
```

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

//cjb added

协议参数

'Max.Init.Retransmits' 初始化（INIT，COOKIE ECHO)的最大重试次数，超过此值放弃初始化过程，并向 SCTP 用户报告错误。

'Valid.Cookie.Life' cookie 的生命期限，超过此值此 cookie 无效。

'Association.Max.Retrans' 偶联的最大重传次数，超过此值认为对端不可达。

'Path.Max.Retrans' 通路的最大重传次数，超过此值通路被标记为 inactive。