TAPS
Internet-Draft
Intended status: Informational
Expires: March 24, 2016

M. Welzl
University of Oslo
M. Tuexen
Muenster Univ. of Appl. Sciences
N. Khademi
University of Oslo

September 21, 2015

An Approach to Identify Services Provided by IETF Transport Protocols and Congestion Control Mechanisms
draft-welzl-taps-transports-00

Abstract

This document describes a method to identify services in transport protocols and congestion control mechanisms. It shows the approach using TCP and SCTP (base protocol) as examples.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 24, 2016.

Table of Contents

1. Introduction

This document considers every form of defined interaction between a transport protocol and its user ("upper layer protocol" or "application") as a "service". Here, the term "service" is NOT the same as the term used to specify the entire "above transport" protocol that maps to a port number or service name (which is another common meaning of the term "service" in the context of transport protocols).

The list of services in this document is strictly based on the parts of relevant protocol specifications that relate to what the protocol provides to an application using it and how the application interacts with it. It is based on text that describes what a protocol provides to the upper layer and how it is used (abstract API descriptions), given for the base protocols in [RFC0793], [RFC1122] and [RFC4960]. It does not cover API instances, for example the one given for SCTP in [RFC6458]. It also does not cover parts of the protocol that are explicitly stated as optional to implement.

The document presents a three-pass process to arrive at a list of transport protocol services. In the first pass, the relevant RFC text is discussed per protocol. In the second pass, this discussion is used to derive a list of services that are uniformly categorized across protocols. Here, an attempt is made to present services in a slightly generalized form to highlight similarities. This is, for example, achieved by renaming "commands" (or "transport primitives") of protocols or by avoiding a strict 1:1-mapping between these commands and services in the list. Finally, the third pass presents all services from pass 2, identifying which protocol implements them.

In the list resulting from the second pass, some services are missing because they are implicit in some protocols, and they only become explicit when we consider the superset of all services offered by all protocols. For example, TCP's reliability includes integrity via a checksum, but we have to include a protocol like UDP-Lite as specified in [RFC3828] (which has a configurable checksum) in the list to consider an always-on checksum as a service (it would not be a service if no protocol would allow to disable / configure the checksum). Similar arguments apply to other protocol functions (e.g. congestion control). The complete list of services across all protocols is therefore only available after pass 3.

2. General Considerations

This document discusses unicast [AUTHOR'S NOTE: for simplicity, for now. Hopefully forever, for simplicity.] transport protocols.

[AUTHOR'S NOTE: we skip "congestion control mechanisms" for now. This simplifies the discussion; the congestion control mechanisms part is about LEDBAT, which is easy to add later.] Transport protocols provide communication between processes that operate on network endpoints, which means that they allow for multiplexing of such communication between the same IP addresses, and normally such multiplexing is achieved using port numbers. Port multiplexing is therefore assumed to be always provided and not discussed as a service.

Some protocols are connection-oriented. Connection-orientation, to the user of an application, means that there is state shared between the endpoints that persists across messages. Connection-oriented protocols often use an initial call to "open" a connection before communication can progress, and require communication to be explicitly terminated by issuing a "close" call. Moreover, a "connection" is the common state that some transport primitives refer to, e.g. to adjust general configuration settings. Connections establishment, maintenance and termination are therefore used to categorize certain services of connection-oriented transport protocols in pass 2 and 3.

3. Pass 1

In this first iteration, the relevant text parts of the RFCs describing the protocols are summarized, focusing on what a protocol provides to the upper layer and how it is used (abstract API descriptions). The resulting text is somewhat heterogeneous in terminology - e.g. the user of the protocol is called "Application" in TCP and "Upper-Layer Protocol (ULP)" in SCTP, and TCP's "user commands" are called "ULP primitives" in SCTP.

3.1. Services Provided by TCP

[RFC0793] states: "TCP is a connection-oriented, end-to-end reliable protocol (..)". Section 3.8 in [RFC0793] further specifies the interaction with the application by listing several user commands. It is also assumed that the Operating System provides a means for TCP to asynchronously signal the user program. Here, we describe the relevant user commands and notifications to the application.

open: this is either active or passive, to initiate a connection or listen for incoming connections. All other commands are associated with a specific connection, which is assumed to first have been opened. An active open call contains a fully specified foreign socket (IP address + port number). A passive open call with a fully specified foreign socket waits for a particular

connection; alternatively, a passive open call can leave the foreign socket unspecified to accept any incoming connection. A fully specified passive call can later be made active by executing 'send'. Optionally, a timeout can be specified, after which TCP will abort the connection if data is not successfully delivered to the destination (else a default timeout value is used). [RFC1122] describes a procedure for aborting the connection that must be used to avoid excessive retransmissions, and states that an application must be able to control the threshold used to determine the condition for aborting -- and that this threshold may be measured in time units or as a count of retransmission. This indicates that the timeout could also be specified as a count of retransmission.

Also optional, for multihomed hosts, the local IP address can be provided [RFC1122]. If it is not provided, a default choice will be made in case of active open calls. A passive open call will await incoming connection requests to all local addresses and then maintain usage of the local IP address where the incoming connection request has arrived. Finally, the 'options' parameter is explained in [RFC1122] to let the application specify IP options such as source route, record route, or timestamp. (It is not stated on which segments of a connection these options should be applied, but probably all segments, as this is also stated in a specification given for the usage of source route (section 4.2.3.8 of [RFC1122]). As the only non-optional IP option in this parameter, an application can specify a source route when it actively opens a TCP connection.

send: this command hands over a provided number of bytes that TCP should reliably send to the other side of the connection. The PUSH flag, if set, requires data to be promptly transmitted to the receiver without delaying it. Conversely, not using PUSH can reduce the number of unnecessary wakeup calls to the receiving application process. [RFC1122] states that "Generally, an interactive application protocol must set the PUSH flag at least in the last SEND call in each command or response sequence. A bulk transfer protocol like FTP should set the PUSH flag on the last segment of a file or when necessary to prevent buffer deadlock." An optional timeout parameter can be provided that updates the connection's timeout (see "open").

receive: This command allocates a receiving buffer for a provided number of bytes. It returns the number of received bytes provided in the buffer when these bytes have been received and written into the buffer by TCP.

close: This command closes one side of a connection. It is semantically equivalent to "I have no more data to send" but does not mean "I will not receive any more", as the other side may still have data to send. This call reliably delivers any data that has already been handed over to TCP (and if that fails, 'close' becomes 'abort'). Close also implies push function.

abort: This command causes all pending SENDs and RECEIVES to be aborted, the TCB to be removed, and a special RESET message to be sent to the TCP on the other side of the connection. See [RFC0793].

close event: TCP will signal a user, even if no RECEIVEs are outstanding, that the other side has closed, so the user can terminate his/her side gracefully. See [RFC0793], Section 3.5.

abort event: When TCP aborts a connection upon receiving a "Reset" from the peer, it "advises the user and goes to the CLOSED state." See [RFC0793], Section 3.4.

USER TIMEOUT event: This event, described in Section 3.9 of [RFC0793], is executed when the user timeout expires (see 'open'). All queues are flushed and the user is signaled "error: connection aborted due to user timeout".

ERROR_REPORT event: This event, described in Section 4.2.4.1 of [RFC1122], informs the application of "soft errors" that can be safely ignored, including the arrival of an ICMP error message or excessive retransmissions (reaching a threshold below the threshold where the connection is aborted).

Type-of-Service: Section 4.2.4.2 of [RFC1122] states that the application layer MUST be able to specify the Type-of-Service (TOS) for segments that are sent on a connection. The application should be able to change the TOS during the connection lifetime, and the TOS value should be passed to the IP layer unchanged. Since then, parts of the TOS field have been assigned to ECN [RFC3168] and the six most significant bits have been assigned to DiffServ by the name of DSField [RFC3260]. Staying with the intention behind the application's ability to specify the "Type of Service", this should probably be interpreted to mean the value in the DSField, which is the Differentiated Services Codepoint (DSCP). [AUTHOR's NOTE: text trying to "read between the lines" of RFCs here... this perhaps calls for an update to [RFC1122]?]

Nagle: An application can disable the Nagle algorithm on an individual connection. This algorithm delays sending data for some time to increase the likelihood of sending a full-sized segment.

3.1.1. Excluded Services

The 'send' and 'receive' commands include usage of an "URGENT" mechanism, which SHOULD NOT be implemented according to [RFC6093] and is therefore not described here. This also concerns the notification "Urgent pointer advance" in the ERROR_REPORT described in Section 4.2.4.1 of [RFC1122].

The 'open' command specified in [RFC0793] can be handed optional Precedence or security/compartment information according to [RFC0793], but this was not incuded here because it is mostly irrelevant today, as explained in [RFC7414]. The 'open' command also includes a parameter "options" that is explained in [RFC1122] to let the application specify IP options such as source route, record route, or timestamp. This parameter was not included here because it is not clear which segments of a connection (all?) these options would then be applied to.

The 'status' command was not included because [RFC0793] calls this command "implementation dependent" and states that it "could be excluded without adverse effect". Moreover, while a data block containing specific information is described, it is also stated that not all of this information may always be available. The 'receive' command can (under some conditions) yield the status of the PUSH flag according to [RFC0793], but this TCP functionality is made optional in [RFC1122] and hence not considered here. Generally, section 4.2.2.2 of [RFC1122] says that PUSH on send calls MAY be implemented, which could be a reason not to consider it here. However, the text then explains that "an interactive application protocol must set the PUSH flag at least in the last SEND call in each command or response sequence", and most implementations provide some option to cause a behavior that is in some way similar to PUSH. Therefore PUSH is described as a part of SEND here. [RFC1122] also introduces keep-alives to TCP, but these are optional and hence not considered here. [RFC1122] describes that "some TCP implementations have included a FLUSH call", indicating that this call is optional to implement. It is therefore not considered here.

3.2. Services Provided by SCTP

Section 1.1 of [RFC4960] lists limitations of TCP that SCTP removes. Three of the four mentioned limitations directly translate into a

service that is visible to an application using SCTP: 1) it allows for preservation of message delineations; 2) these messages, while reliably transferred, do not require to be in order unless the application wants it; 3) multi-homing is supported. In SCTP, connections are called "association" and they can be between not only two (as in TCP) but multiple transport addresses at each end point. For SCTP running over IP, [RFC4960] defines a "transport address" as "the combination of an IP address and an SCTP port number (where SCTP is the transport protocol)".

Section 10 of [RFC4960] further specifies the interaction with the application (which RFC [RFC4960] calls the "Upper Layer Protocol" (ULP)). It is assumed that the Operating System provides a means for SCTP to asynchronously signal the user program. Here, we describe the relevant ULP primitives and notifications to the ULP process:

Initialize: Initialize creates a local SCTP instance which it binds to a set of local addresses (and, if provided, port number). Initialize needs to be called only once per set of local addresses.

Associate: This creates an association (the SCTP equivalent of a connection) between the local SCTP instance and a remote SCTP instance. Most primitives are associated with a specific association, which is assumed to first have been created. Associate can return a list of destination transport addresses so that multiple paths can later be used. One of the transport addresses from the returned destination addresses will be selected by the local endpoint as default primary path for sending SCTP packets to this peer, but this choice can be changed by the ULP using the list of destination addresses. Associate is also given the number of outgoing streams to request and optionally returns the number of outgoing streams negotiated.

Send: This sends a message of a certain length in bytes over an association. A number can be provided to later refer to the correct message when reporting an error and a stream id is provided to specify the stream to be used inside an association (we consider this as a mandatory parameter here for simplicity: if not provided, the stream id defaults to 0). An optional maximum life time can specify the time after which the message should be discarded rather than sent. A choice (advisory, i.e. not guaranteed) of the preferred path can be made by providing a destination transport address, and the message can be delivered out-of-order if the unordered flag is set. Another advisory flag indicates the ULP's preference to avoid bundling user data with other outbound DATA chunks (i.e., in the same packet). The handling of this no-bundle flags is similar to the sender side

handling of the TCP PUSH flag. A payload protocol-id can be provided to pass a value that indicates the type of payload protocol data to the peer.

Receive: Messages are received from an association, and optionally a stream within the association, with their size returned. The ULP is notified of the availability of data via a DATA ARRIVE notification. If the sender has included a payload protocol-id, this value is also returned. If the received message is only a partial delivery of a whole message, a partial flag will indicate so, in which case the stream id and a stream sequence number are provided to the ULP.

Shutdown: This primitive gracefully closes an association, reliably delivering any data that has already been handed over to SCTP. A return code informs about success or failure of this procedure.

Abort: This ungracefully closes an association, by discarding any locally queued data and informing the peer that the association was aborted. Optionally, an abort reason to be passed to the peer may be provided by the ULP. A return code informs about success or failure of this procedure.

Change Heartbeat / Request Heartbeat: This allows the ULP to enable/disable heartbeats and optionally specify a heartbeat frequency as well as requesting a single heartbeat to be carried out upon a function call, with a notification about success or failure of transmitting the HEARTBEAT chunk to the destination.

Set Protocol Parameters: This allows to set values for protocol parameters per association; for some parameters, a setting can be made per transport address. The set listed in [RFC4960] is: RTO.Initial; RTO.Min; RTO.Max; Max.Burst; RTO.Alpha; RTO.Beta; Valid.Cookie.Life; Association.Max.Retrans; Path.Max.Retrans; Max.Init.Retransmits; HB.interval; HB.Max.Burst.

Set Primary: This allows to set a new primary default path for an association by providing a transport address. Optionally, a default source address to be used in IP datagrams can be provided.

Status: The 'Status' primitive returns a data block with information about a specified association, containing: 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.

COMMUNICATION UP notification: When a lost communication to an endpoint is restored or when SCTP becomes ready to send or receive user messages, this notification informs the ULP process about the affected association, the type of event that has occurred, the complete set of transport addresses of the peer, the maximum number of allowed streams and the inbound stream count (the number of streams the peer endpoint has requested).

DATA ARRIVE notification: When a message is ready to be retrieved via the Receive primitive, the ULP process is informed by this notification.

SEND FAILURE notification / Receive Unsent Message / Receive Unacknowledged Message: When a message cannot be delivered via an association, the sender can be informed about it and learn whether the message has just not been acknowledged or (e.g. in case of lifetime expiry) if it has not even been sent.

NETWORK STATUS CHANGE notification: The NETWORK STATUS CHANGE notification informs the ULP about a transport address becoming active/inactive.

COMMUNICATION LOST notification: When SCTP loses communication to an endpoint (e.g. via Heartbeats or excessive retransmission) or detects an abort, this notification informs the ULP process of the affected association and the type of event (failure OR termination in response to a shutdown or abort request).

SHUTDOWN COMPLETE notification: When SCTP completes the shutdown procedures, this notification is passed to the upper layer, informing it about the affected assocation.

3.2.1. Excluded Services

For the 'Set Primary' primitive, an optional possibility to specify the source transport address to be used in outgoing IP datagrams is described, but the RFC text says "some implementations may allow you to", indicating that implementing this in SCTP is optional. This functionality is therefore not considered here. The 'Receive' primitive can also return certain additional information, but this is also left up to the implementation and therefore not considered. With a COMMUNICATION LOST notification, some more information may optionally be passed to the ULP (e.g., identification to retrieve unsent and unacknowledged data). SCTP "can invoke" a COMMUNICATION ERROR notification and "may send" a RESTART notification, making these two notifications optional to implement. The list provided under 'Status' includes "etc", indicating that more information could

be provided. The primitive 'Get SRTT Report' returns information that is included in what 'Status' provides and is therefore not discussed. Similarly, 'Set Failure Threshold' sets only one out of various possible parameters included in 'Set Protocol Parameters'. The 'Destroy SCTP Instance' primitive was excluded: it erases the SCTP instance that was created by 'Initialize', but this does not translate into a service for the ULP.

4. Pass 2

Here we categorize the services from pass 1 based on whether they relate to a connection or to data transmission. Services are presented following the nomenclature "CATEGORY.[SUBCATEGORY].SERVICENAME.PROTOCOL". We present "connection" as a general protocol-independent concept and use it to refer to both TCP's connections (which are identifiable by a unique socket pair, where a socket is defined as an IP address and TCP port) and SCTP's associations (which are identifiable by multiple IP address and port number pairs). We define the "transport address" as "the combination of an IP address and a transport protocol's port number". The "application" is the user of the protocol (called "Upper-Level Protocol (ULP)" in SCTP).

Some minor details are omitted for the sake of generalization -- e.g., for SCTP's 'close', [RFC4960] states that success or failure is returned, whereas this is not described in the same way for TCP in [RFC0793], but this detail plays no significant role for the service provided by either TCP or SCTP.

4.1. CONNECTION Related Services

ESTABLISHMENT:
Active creation of a connection from one transport address to one or more transport addresses.

- CONNECT.TCP:
  Command / event: 'open' (active) or 'open' (passive) with destination transport address, followed by 'send'
  Parameters: 1 local IP address (optional); 1 destination transport address (for active open; else the destination transport address and the local IP address of the succeeding incoming connection request will be maintained); timeout (optional); options (optional)
  Comments: If the local IP address is not provided, a default choice will automatically be made. [AUTHOR'S NOTE: [RFC1122] does not clearly state this, but it seems to be the implication of some text there.] The timeout can also be a retransmission count. The

options are IP options to be used on all segments of the connection. At least the Source Route option is mandatory for TCP to provide.

- CONNECT.SCTP:
  Command / event: 'initialize', followed by 'associate'
  Parameters: list of local transport addresses (initialize); 1 destination transport address; outbound stream count
  Returns: destination transport address list
  Comments: 'initialize' needs to be called only once per local transport address list. One destination transport address will automatically be chosen; it can later be changed in MAINTENANCE.

AVAILABILITY:
Preparing to receive incoming connection requests.

- LISTEN.TCP:
  Command / event: 'open' (passive)
  Parameters: 1 local IP address (optional); 1 destination transport address (optional); timeout (optional)
  Comments: if the transport address and/or local IP address is provided, this waits for incoming connections from only and/or to only the provided address. Else this waits for incoming connections without this / these restraint(s). ESTABLISHMENT can later be done with 'send'.

- LISTEN.SCTP:
  Command / event: 'initialize', followed by 'COMMUNICATION UP' notification
  Parameters: list of local transport addresses (initialize)
  Returns: destination transport address list; outbound stream count; inbound stream count
  Comments: initialize needs to be called only once per local transport address list. COMMUNICATION UP can also follow a COMMUNICATION LOST notification, indicating that the lost communication is restored.

MAINTENANCE:
Adjustments made to an open connection, or notifications about it. These are out-of-band messages to the protocol that can be issued at any time, at least after a connection has been established and before it has been terminated (with one exception: CHANGE-TIMEOUT.TCP can only be issued when new data are handed over for sending).

- CHANGE-TIMEOUT.TCP:
  Command / event: 'send'
  Parameters: timeout value
  Comments: when sending data, the connection's timeout value (time after which the connection will be aborted if data cannot be delivered) can be adjusted.

- CHANGE-TIMEOUT.SCTP:
  Command / event: 'Change HeartBeat' combined with 'Set Protocol Parameters'
  Parameters: 'Change HeartBeat': heartbeat frequency; 'Set Protocol Parameters': Association.Max.Retrans (whole association) or Path.Max.Retrans (per transport address)
  Comments: Change Heartbeat can enable / disable heartbeats in SCTP as well as change their frequency. The parameter Association.Max.Retrans defines after how many unsuccessful heartbeats the connection will be terminated; thus these two commands / parameters together can yield a similar behavior to CHANGE-TIMEOUT.TCP.

- DISABLE-NAGLE.TCP:
  Command / event: not specified
  Parameters: one boolean value
  Comments: the Nagle algorithm delays data transmission to increase the chance to send a full-sized segment. An application must be able to disable this algorithm for a connection. This is related to the no-bundle flag in DATA.SEND.SCTP.

- REQUESTHEARTBEAT.SCTP:
  Command / event: 'Request HeartBeat'
  Parameters: destination transport address
  Returns: success or failure
  Comments: requests a heartbeat to be immediately carried out on a path, returning success or failure.

- SETPROTOCOLPARAMETERS.SCTP:
  Command / event: 'Set Protocol Parameters'
  Parameters: RTO.Initial; RTO.Min; RTO.Max; Max.Burst; RTO.Alpha; RTO.Beta; Valid.Cookie.Life; Association.Max.Retrans; Path.Max.Retrans; Max.Init.Retransmits; HB.interval; HB.Max.Burst

- SETPRIMARY.SCTP:
  Command / event: 'Set Primary'
  Parameters: destination transport address
  Returns: result of attempting this operation
  Comments: update the current primary address to be used, based on the set of available destination transport addresses of the association.

- ERROR.TCP:
  Command / event: 'ERROR_REPORT'
  Returns: reason (encoding not specified); subreason (encoding not specified)
  Comments: soft errors that can be ignored without harm by many applications; an application should be able to disable these notifications. The reported conditions include at least: Excessive Retransmissions and ICMP error message arrived.

- STATUS.SCTP:
  Command / event: 'Status' and 'NETWORK STATUS CHANGE' notification
  Returns: data block with information about a specified association, containing: 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. The NETWORK STATUS CHANGE notification informs the application about a transport address becoming active/inactive.

- CHANGE-DSCP.TCP:
  Command / event: not specified
  Parameters: DSCP value
  Comments: This allows an application to change the DSCP value. It was only specified for the TOS field in [RFC1122], which is here interpreted to refer to the DSField as per [RFC3260].

TERMINATION:
Gracefully or forcefully closing a connection, or being informed about this event happening.

- CLOSE.TCP:
  Command / event: 'close'
  Comments: this terminates the sending side of a connection after reliably delivering all remaining data. Close also implies push function (see DATA.SEND.TCP).

- CLOSE.SCTP:
  Command / event: 'Shutdown'
  Comments: this terminates a connection after reliably delivering all remaining data.

- ABORT.TCP:
  Command / event: 'abort'
  Comments: this terminates a connection without delivering remaining data and sends an error message to the other side.

- ABORT.SCTP:
  Command / event: 'abort'
  Parameters: abort reason to be given to the peer (optional)
  Comments: this terminates a connection without delivering remaining data and sends an error message to the other side.

- TIMEOUT.TCP:
  Command / event: 'USER TIMEOUT' event
  Comments: the application is informed that the connection is aborted. This event is executed when the timeout set in CONNECTION.ESTABLISHMENT.CONNECT.TCP (and possibly adjusted in CONNECTION.MAINTENANCE.CHANGE-TIMEOOUT.TCP) expires.

- TIMEOUT.SCTP:
  Command / event: 'COMMUNICATION LOST' event
  Comments: the application is informed that the connection is aborted. this event is executed when the timeout that should be enabled by default (see beginning of section 8.3 in [RFC4960]) and was possibly adjusted in CONNECTION.MAINTENANCE.CHANGE-TIMEOOUT.SCTP expires.

- ABORT-EVENT.TCP:
  Command / event: not specified

- ABORT-EVENT.SCTP:
  Command / event: 'COMMUNICATION LOST' event
  Returns: abort reason from the peer (if available)
  Comments: the application is informed that the other side has aborted the connection using CONNECTION.TERMINATION.ABORT.SCTP.

- CLOSE-EVENT.TCP:
  Command / event: not specified

- CLOSE-EVENT.SCTP:
  Command / event: 'SHUTDOWN COMPLETE' event
  Comments: the application is informed that CONNECTION.TERMINATION.CLOSE.SCTP was successfully completed.

4.2. DATA Transfer Related Services

All commands in this section refer to an existing connection, i.e. a connection that was either established or made available for receiving data. In addition to the listed parameters, all sending commands contain a reference to a data block and all receiving commands contain a reference to available buffer space for the data.

- SEND.TCP:
  Command / event: 'send'
  Parameters: PUSH flag (optional); timeout (optional)
  Comments: If the push flag is set, the data block should promptly be transmitted to the receiver without waiting. The timeout can be configured with this call whenever data are sent (see also CONNECTION.MAINTENANCE.CHANGE-TIMEOUT.TCP).

- SEND.SCTP:
  Command / event: 'Send'
  Parameters: stream number; context (optional); life time (optional); destination transport address (optional); unordered flag (optional); no-bundle flag (optional); payload protocol-id (optional)
  Comments: the 'stream number' denotes the stream to be used. The 'context' number can later be used to refer to the correct message when an error is reported. The 'life time' specifies a time after which this data block will not be sent. The 'destination transport address' can be used to state which path should be preferred, if there are multiple paths available (see also CONNECTION.MAINTENANCE.SETPRIMARY.SCTP). The data block can be delivered out-of-order if the 'unordered flag' is set. The 'no-bundle flag' can be set to indicate a preference to avoid bundling (this is related to CONNECTION.MAINTENANCE.DISABLE-NAGLE.TCP). The 'payload protocol-id' is a number that will, if it was provided, be handed over to the receiving application.

- RECEIVE.TCP:
  Command / event: 'receive'

- RECEIVE.SCTP:
  Command / event: 'DATA ARRIVE' notification, followed by 'Receive'
  Parameters: stream number (optional)
  Returns: stream sequence number (optional), partial flag (optional)
  Comments: if the 'stream number' is provided, the call to receive only receives data on one particular stream. If a partial message arrives, this is indicated by the 'partial flag', and then the 'stream sequence number' must be provided such that an application can restore the correct order of data blocks an entire message consists of.

- SENDFAILURE-EVENT.SCTP:
  Command / event: 'SEND FAILURE' notification, optionally followed by 'Receive Unsent Message' or 'Receive Unacknowledged Message'
  Returns: cause code; context; unsent or unacknowledged message (optional)
  Comments: 'cause code' indicates the reason of the failure, and

'context' is the context number if such a number has been provided in DATA.SEND.SCTP, for later use with 'Receive Unsent Message' or 'Receive Unacknowledged Message', respectively. These commands can be used to retrieve the complete unsent or unacknowledged message if desired.

5. Pass 3

Here we present the superset of all services in all protocols, based on the list in pass 2 but also on text in pass 1 to include services that can be configured in one protocol and are static properties in another. Again, some minor details are omitted for the sake of generalization -- e.g., TCP may provide various different IP options but only supporting source route is mandatory to implement, and this detail is no longer visible in "Specify IP Options". The detail was removed because no other protocols provide this features. [AUTHOR'S NOTE: and if we find another one that does, we need that detail again.]

[AUTHOR'S NOTE: the list here looks pretty similar to the list in pass 2 for now. This will change as more protocols are added. For example, if we add UDP, we will find that UDP does not do congestion control, which is relevant to the application using it. This will have to be reflected in pass 1 and pass 2, only for UDP. In pass 3, we can derive "congestion control" as a service of TCP and SCTP because it probably does not make much sense to write that only UDP provides a congestion control related service: the "service" of not doing it -- meaning that it may require more work from the application developer.]

5.1. CONNECTION Related Services

ESTABLISHMENT:
Active creation of a connection from one transport address to one or more transport addresses.

- Specify IP Options
  Protocols: TCP

- Request multiple streams
  Protocols: SCTP

- Obtain multiple destination transport addresses
  Protocols: SCTP

AVAILABILITY:
Preparing to receive incoming connection requests.

- Listen, 1 specified local interface
  Protocols: TCP, SCTP

- Listen, N specified local interfaces
  Protocols: SCTP

- Listen, all local interfaces (unspecified)
  Protocols: TCP, SCTP

- Obtain requested number of streams
  Protocols: SCTP

MAINTENANCE:
Adjustments made to an open connection, or notifications about it.
NOTE: all services except "set primary path" in this category apply to one out of multiple possible paths (identified via destination transport addresses) in SCTP, whereas TCP uses only one path (one destination transport address).

- Change timeout for aborting connection (using retransmit limit or time value)
  Protocols: TCP, SCTP

- Disable Nagle algorithm
  Protocols: TCP
  Comments: This is available in SCTP implementations, but not specified in [RFC4960].

- Request an immediate heartbeat, returning success/failure
  Protocols: SCTP

- Set protocol parameters
  Protocols: SCTP
  SCTP parameters: RTO.Initial; RTO.Min; RTO.Max; Max.Burst; RTO.Alpha; RTO.Beta; Valid.Cookie.Life; Association.Max.Retrans; Path.Max.Retrans; Max.Init.Retransmits; HB.interval; HB.Max.Burst
  Comments: in future versions of this document, it might make sense to split out some of these parameters -- e.g., if a different protocol provides means to adjust the RTO calculation there could be a common service for them called "adjust RTO calculation".

- Notification of Excessive Retransmissions (early warning below abortion threshold)
  Protocols: TCP

- Notification of ICMP error message arrival
  Protocols: TCP

- Status (query or notification)
  Protocols: SCTP
  SCTP parameters: 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; transport address becoming active / inactive

- Set primary path
  Protocols: SCTP

- Change DSCP
  Protocols: TCP
  Comments: This is described to be changeable for SCTP too in [RFC6458].

TERMINATION:
Gracefully or forcefully closing a connection, or being informed about this event happening.

- Close after reliably delivering all remaining data, causing an event informing the application on the other side
  Protocols: TCP, SCTP
  Comments: TCP's locally only closes the connection for sending; it may still receive data afterwards.

- Abort without delivering remaining data, causing an event informing the application on the other side
  Protocols: TCP, SCTP
  Comments: In SCTP a reason can optionally be given by the application on the aborting side, which can then be received by the application on the other side.

- Timeout event when data could not be delivered for too long
  Protocols: TCP, SCTP
  Comments: the timeout is configured with CONNECTION.MAINTENANCE "Change timeout for aborting connection (using retransmit limit or time value)".

5.2. DATA Transfer Related Services

All services in this section refer to an existing connection, i.e. a connection that was either established or made available for receiving data. In addition to the listed parameters, all sending commands contain a reference to a data block and all receiving commands contain a reference to available buffer space for the data. Reliable data transfer entails delay -- e.g. for the sender to wait until it can transmit data, or due to retransmission in case of packet loss.

5.2.1. Sending Data

All services in this section are provided by DATA.SEND from pass 2. DATA.SEND is given a data block from the application, which we here call a "message".

- Reliably transfer data
  Protocols: TCP, SCTP

- Notifying the receiver to promptly hand over data to application
  Protocols: TCP
  Comments: This seems unnecessary in SCTP, where data arrival causes an event for the application.

- Message identification
  Protocols: SCTP

- Choice of stream
  Protocols: SCTP

- Choice of path (destination address)
  Protocols: SCTP

- Message lifetime
  Protocols: SCTP

- Choice between unordered (potentially faster) or ordered delivery
  Protocols: SCTP

- Request not to bundle messages
  Protocols: SCTP

- Specifying a "payload protocol-id" (handed over as such by the receiver)
  Protocols: SCTP

5.2.2. Receiving Data

All services in this section are provided by DATA.RECEIVE from pass 2. DATA.RECEIVE fills a buffer provided to the application, with what we here call a "message".

- Receive data
  Protocols: TCP, SCTP

- Choice of stream to receive on
  Protocols: SCTP

- Message identification
  Protocols: SCTP
  Comments: In SCTP, this is optionally achieved with a "stream sequence number". The stream sequence number is always provided in case of partial message arrival.

- Information about partial message arrival
  Protocols: SCTP
  Comments: In SCTP, partial messages are combined with a stream sequence number so that the application can restore the correct order of data blocks an entire message consists of.

5.2.3. Errors

This section describes sending failures that are associated with a specific call to DATA.SEND from pass 2.

- Notification of unsent messages
  Protocols: SCTP

- Notification of unacknowledged messages
  Protocols: SCTP

6. Acknowledgements

The authors would like to thank Joe Touch for comments on the TCP part. This work has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 644334 (NEAT). The views expressed are solely those of the author(s).

7. IANA Considerations

XX RFC ED - PLEASE REMOVE THIS SECTION XXX

This memo includes no request to IANA.

8. Security Considerations

Security will be considered in future versions of this document.

9. References

9.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <http://www.rfc-editor.org/info/rfc793>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <http://www.rfc-editor.org/info/rfc1122>.

[RFC4960] Stewart, R., Ed., "Stream Control Transmission Protocol", RFC 4960, DOI 10.17487/RFC4960, September 2007, <http://www.rfc-editor.org/info/rfc4960>.

9.2. Informative References

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <http://www.rfc-editor.org/info/rfc3168>.

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, DOI 10.17487/RFC3260, April 2002, <http://www.rfc-editor.org/info/rfc3260>.

[RFC3828] Larzon, L-A., Degermark, M., Pink, S., Jonsson, L-E., Ed., and G. Fairhurst, Ed., "The Lightweight User Datagram Protocol (UDP-Lite)", RFC 3828, DOI 10.17487/RFC3828, July 2004, <http://www.rfc-editor.org/info/rfc3828>.

[RFC6093] Gont, F. and A. Yourtchenko, "On the Implementation of the TCP Urgent Mechanism", RFC 6093, DOI 10.17487/RFC6093, January 2011, <http://www.rfc-editor.org/info/rfc6093>.

[RFC6458] Stewart, R., Tuexen, M., Poon, K., Lei, P., and V. Yasevich, "Sockets API Extensions for the Stream Control Transmission Protocol (SCTP)", RFC 6458, DOI 10.17487/RFC6458, December 2011, <http://www.rfc-editor.org/info/rfc6458>.

[RFC7414] Duke, M., Braden, R., Eddy, W., Blanton, E., and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 7414, DOI 10.17487/RFC7414, February 2015, <http://www.rfc-editor.org/info/rfc7414>.

Authors' Addresses

Michael Welzl
University of Oslo
PO Box 1080 Blindern
Oslo, N-0316
Norway

Phone: +47 22 85 24 20
Email: michawe@ifi.uio.no

Michael Tuexen
Muenster University of Applied Sciences
Stegerwaldstrasse 39
Steinfurt 48565
Germany

Email: tuexen@fh-muenster.de

Naeem Khademi
University of Oslo
PO Box 1080 Blindern
Oslo, N-0316
Norway

Email: naeemk@ifi.uio.no