Group C
STP Protocol Draft Version:C

Annika Hallensleben, Quingyun He,
Leon Spitzer, Ján Tugsbayar
RWTH Aachen

SPP Transport Protocol
Version C

prepared for
RWTH Informatik i4 COMSYS
Ahornstraße 55 - building E3
52074 Aachen
Germany

by
SPP Group C:
Annika Hallensleben
Quingyun He
Leon Spitzer
Ján Tugsbayar

Table of Content

1 Introduction

This document lays out the specifications and standards for the SPP Transport Protocol, from this point shortened as STP.
STP is designed as a viable replacement for the well known TCP[1] protocol, but it features its own custom API, to which the user application has free access. As a replacement for TCP, STP must offer reliable host-to-host communication in packet-switched networks, built in flow control and guaranteed in-order arrival. The protocol is designed to be extensible, room for additional optional features is taken into account for future expansion.
STP works in the application layer and builds upon the UDP[2] protocol as it makes calls for UDP network functions. Fully implemented UDP, with all its specified functions, such as creation of receive ports, receive data function and send data function are required for STP to function properly. In this document it is assumed that UDP is readily available for all users wishing to use STP.

STP is part of the "Softwareprojekt Praktikum: Entwurf eines Transportprotokolls" subject at RWTH Aachen, the goal of the subject is to develop in teams a transport protocol which should offer similar functionalities as TCP, but works on the application layer. STP is result of the collective group effort of all the students in the subject, namely this specific standard is written by Group C: Annika Hallensleben, Quingyun He, Leon Spitzer, Ján Tugsbayar.

In the following sections we will introduce the terminology used in this document, give an overview of the protocol and lastly explain the functional specifications of STP.

[1]Specifications of TCP: https://tools.ietf.org/html/rfc793
[2]Specifications of UDP: https://tools.ietf.org/html/rfc768

2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

If not specified otherwise, all numbers in this standard must be stored in the network order (Big endian).

Send Sequence Variables
SND.UNACK - the first sequence number that has been sent not acknowledged
SND.NXT - the sequence number of the next packet to be sent
SND.WND - send window

Receive Sequence Variables
RCV.CUR - sequence number of currently received packet
RCV.EXP - expected sequence number of the next packet
RCV.WND - receive window
RCV.TOLWND - a space of sequence numbers that is to be tolerated by the receiver. Ranging from (RCV.CUR - 50) modulo 2^{16} to (RCV.EXP + 50) modulo 2^{16}.

Other Variables
TOACK - the sequence number that the sender acknowledges

Option Slot: an option slot is defined as a 32-bits word after the header that is used to exchange information about a certain option.

3 Protocol Overview

3.1 Goals

Version C of STP inherits all goals of STP as specified in the standard of STP protocol, in addition to that, Version C extends STP with the following Functionalities:

Data compression Since internet has became a very popular infrastructure for file sharing, large amount of data is sent through the internet. Version C of STP tries to reduce the amount of packets in the network infrastructure in comparison to the basic version of STP by providing the possibility of compressing data before they are sent and decompress them at the receiver's end.

Congestion Control Since neither IP or UDP provide a tool to avoid congestion in the network infrastructure, version C of STP tries to tackle this problem by implementing the Slow Start algorithm and Congestion Avoidance algorithm to avoid overloading the routers between the users. The STP instance using version C protocol shall be able to make assumptions about the network and react on it by reducing or increasing the amount of data it sends.

Negative Acknowledgement(NACK) In the basic version of STP, the two hosts can only exchange information about which packets they have correctly received, but not what they have not received. It could lead to scenarios where only one packet is missing, but all packets since are retransmitted. Version C tries to solve this problem by introducing negative acknowledgements (NACKs), which allows telling the other host which packet a STP instance is missing and needs to be retransmitted.

3.2 Interaction with other layers

STP runs in the application layer and provides the functionalities to connect, send and receive data to the application. The data given from the application or given to the application is seen as a byte stream stored in a buffer.

To send data STP encapsulates the data from the byte stream in the buffer as payload into a STP packet with its own header. Afterwards it calls for UDP to send the data using UDP datagrams.

To receive data, STP receives a packet from the lower layer and processes the STP header. In case there no errors are found and the receiver is correct, then the content of the the payload is moved into a receive buffer. The receive buffer is readable accessible to the application in the upper layer.
Details to these operations are explained in section 4.

3.3 Purpose and practical aspects

STP draws a lot of inspiration from TCP, since it is supposed to be able to perform the functions TCP. Although the protocol is reliable and relatively lightweight compared to TCP and with less overhead, it is not intended as a practical and commercial replacement for the well established and widespread TCP protocol.
STP is an experimental protocol to be used within a controlled environment and its main purpose is an educational tool to be used by students.

4 Functional Specification

4.1 Header Format

STP packets are sent as User Datagram Protocol (UDP) packets. The UDP header has four control information fields including the source and destination port, the length of the UDP packet, and a checksum over its header and data. A STP header follows the UDP header, providing information specific to the STP protocol.

STP Header Format, note that one tick mark represents one bit position

```
 1               2               3               4
 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Sequence Nr                  |  Ack Nr       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Acknowledgement Nr       |        Window size            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               |S|A|F|R|       |               |
|          Window size          |Y|C|I|S|Version| Option length |
|                               |N|K|N|T|       |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Options                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: STP Version C Header Format

4.1.1 Header fields

Following header fields must be used as specified in the STP standard: Sequence Nr, Acknowledgement Nr, Window size, SYN, ACK, FIN, RST, Version.
In version C, the reserved bits in the basic version are used as Option length field and the header is expended with extra option fields. The specification of Option length and Options are given below:

Option length: 8 bits Provides the length of option field in 32-bits words. This indicates where the payload of this packet begins as the length of the header in 32-bits words is calculated as 3 + Option length. This field must be filled in every STP packet. Since 8 bits are used for Option length, any integer number between 0 and $2^8 - 1$ can be written in this field. A STP packet must not, at any time, use more than $2^8 - 1$ option fields. It is, however, recommended to keep the number of options between 0 and 10 in the data transmission phase.

Option: variable length Options are extra information exchanged for various extensions of STP protocol. Options may occupy space at the end of the STP header and are a multiple 32 bits in length. The number of 32-bit words used for options must be the same as indicated in the "Option len" field, if "Option len" is set to 0, no option is used in this packet. The first byte of a 32-bit word must indicate the type of the option. And the following 24 bits shall be interpreted as described in the option's specification. If an option does not need all 24 bits, the unused bits must be filled with 0. Please note that options are not inter-operable between different groups, so these information are only evaluated and used when the two partners use protocol from the same group

Currently defined options include: (Types are given in hexadecimal numbers)

Type	Option
0x01	Data Compression
0x02	Congestion Control
0x03	NACK

The specifications for the options are given in section 4.6

4.2 Control bits

The specifications for control bits are given in the standard of the base version of STP.

4.3 Sequence Number

The specification of sequence number is given in the standard of the basic version of STP.

4.4 Connection

In order to establish or terminate a connection multiple steps are necessary, explained in the next section.

4.4.1 Connection establishment

Option Slot: an option slot is defined as a 32-bits word after the header that is used to exchange information about a certain option.

The connection establishment shall be performed as described in the standard of the basic version of STP with few modifications. If a Version C STP instance A wants to initialize a connection, following

actions must be performed: the first SYN-message from STP instance A must provide a list of extensions, using one option slot for each option it wishes to use in this connection and the "Option length" field filled as specified in section 4.1. The option slots must be filled as specified in section 4.6.
If the STP instance B receiving this message also uses Version C of STP, it must check the option slots. If it also supports and wishes to use any of the options, it must also use one option slot for each option it supports to indicate it. The messages contained in this message is known to both instances and must be used in this connection.
If the receiving instance B does not use Version C, it must, as described in the standard of basic option, indicate it in the Version field.
Upon receiving the SYN-ACK-message from B, instance A must check the Version field to see if the other instance also uses version C. If the basic version is in the Version field. No further operations are needed since no options can be used. Otherwise it must check which options are used in this connection.
No modifications are made to the last SYN-ACK message from A to B.

1) A ----> B; SYN, Seq Nr: ISN of A, Ack Nr: Do not care, Options A wishes to use.
2) A <---- B; SYN, ACK, Seq Nr: ISN of B, Ack Nr: ISN of A, Options used in this connection.
3) A ----> B; SYN, ACK, Seq Nr: (ISN of A) + 1, Ack Nr: ISN of B.

4.4.2 Connection termination

The termination of connection is specified in the standard of the basic version.

4.5 Data transmission

Data transmission of Version C shall be done as specified in the standard of the basic version with following differences.

4.5.1 Acknowledgement of a sequence number

A STP instance does not have to acknowledge every packet it receives, the acknowledgement is sent with either the next payload packet by filling it's "Ack Nr" field or heartbeat packet as specified in section 4.5.2.

4.5.2 Heartbeat packets

In order to prevent a deadlock caused by flow control, a STP instance must start a timer after the connection has been established. After sending any packet the STP instance shall reset the timer. If the timer runs out, the instance must send a Heartbeat packet. All header fields must be set as specified before and this packet may not contain any payload.

4.5.3 Congestion Control

In order to implement congestion control, the following variables are introduced: cwnd, threshold, CDT. This has the goal of avoiding an overload of the network.

CONGESTION WINDOW (cwnd) - The congestion window is a STP variable that limits the amount of data that STP can send. The starting value is arbitrary and depends on the implementation and circumstances, and is increased every time a package is acknowledged. This increase in size of cwnd continues until threshold value is reached, at which point the size of cwnd continues to increase in a linear fashion. The sequence number of a data packet MUST NOT be higher than the sum of the last received acknowledgement number and the cwnd size. STP can only send as much data as allowed by cwnd to avoid overloading the network.

THRESHOLD - The threshold is a STP variable that marks the point where STP switches from Slowstart mode to congestion avoidance mode. Its value MUST be a number of full data segments. A higher value of threshold is preferred, as it allows STP data rate to rise, the ultimate value threshold depends on the implementation. A sensible recommendation would be half of the initial receiver’s window size. Under this recommendation the data rate achieved immediately after reaching the threshold value is still smaller than the initial window size, this also helps to not overload the receiver.

CONGESTION DETECTOR TIMER (CDT) - The Congestion Detector Timer is a timer set to a specific starting value, depending on the implementation, that will start when the data transmission begins. After an ACK is received CDT will be reset to its starting value. When the timer runs out before an ACK is received, then a congestion has been detected.

STP has two modes of operation to avoid congestion, namely the Slowstart mode and congestion avoidance mode. Both work similar to the TCP Slowstart and TCP congestion avoidance mode.

SLOWSTART - All connections begin in Slowstart mode. This way STP data rate starts at a lower rate to make sure it does not trigger a network congestion. During the Slowstart mode, the **cwnd** increases with every acknowledged packet by 1. This way **cwnd** increases with every data burst by a factor of 2. When **cwnd** exceeds the **threshold** value STP enters congestion avoidance mode. If a congestion is detected during Slowstart mode, then **cwnd** is reset back to the initial value and **threshold** is set to the half of the current data rate. This way STP can slowly reach a safe data rate, but still be able to handle network congestion in case it occurs.

CONGESTION AVOIDANCE - When in congestion avoidance mode, the **cwnd** increases with every data burst by 1. This way STP still slowly increases data rate but at a much slower rate. The maximum value that **cwnd** can reach is the receiver’s window size. In case a congestion is detected, **cwnd** is reset back to the initial value and **threshold** is set to the half of the current data rate and Slowstart mode is initiated with these variables.

The initial SYN/ACK messages send when establishing a connection MUST NOT increase the cwnd. The Congestion Control starts when the main data transfer begins.

4.6 Options

A option is an extension that is supported by Version C of STP, that requires two instances to cooperate. An option slot is defined as a 32-bits word after the header that is used to exchange information about a certain option. The first byte of a option slot shall always be used to identify the type of an option, and the other 3 bytes are used to carry information. If an option does not require 3 bytes, the bits that are not needed must be filled with 0. The "Option length" must be the same as the number of option slots used in this packet.
If a STP instance wants to use various extensions of TCP, the following actions must be done at the connection establishment: For each extension the client wishes to use, it must use an option slot in the initial SYN message with the type of that option filled in the first byte. The other 3 bytes are to be filled as specified in the subsections. If nothing is specified in the corresponding subsection, they are to be filled with 0.
The server shall check the group number first. If the server uses another protocol as the client, no extensions shall be used in this session and this is indicated by answering with a SYN,ACK message with no option slot used. If the server uses protocol from the same group, it now answers with every extension it supports and these will be used in this session. The SYN-ACK message from the client must contain the same Options as the SYN-ACK message from the server to verify the options used in the connection, if the server recognises a error, it must terminate the connection as specified in 4.4.2.
currently defined options include: (Types are given in hexadecimal numbers in big endian order)

Type	Option
0x01	Data Compression
0x03	NACK

4.6.1 Data Compression

In order to use Data Compression, this must send an option slot with its first byte filled with 0x01, and it must send a list of supported

compression methods by filling the other bytes with the codeword of algorithms that it supports (see section 4.4.1).
Example: If a STP instance wishes to use Data compression and supports both Huffman Encoding and Lempel-Ziv-Welch Encoding, the option slot for it may look like this:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     0x01      |     0x02      |     0x05      |     0x00      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The other instance then responds with the number corresponding to the method they wish to use. A compression method must be chosen from the following two options by adding the number corresponding to the method in the options part of the header as defined here (One byte for each Algorithm):
- 0x02 Huffman Encoding
- 0x05 Lempel-Ziv-Welch Encoding

If the flag for use of compression is not set in the connect function, or if none of the compression methods are supported by both the client and the server, no compression will take place.
   We wait until we have 1 Kilobyte of payload data, or until 500 milliseconds have elapsed since the first byte of data is received from the application. This timeout is to prevent long waits when there is no more data to be transmitted.
   Once the package is complete (through being filled or a timeout), compression takes place on the payload portion of the package. The compression is indicated by setting the compression flag to the method used. If no compression takes place, the flag is set to 0x00.
   The flag for compression is set as the first bit after RST bit in header.

4.6.2 NACK

In the initialisation phase, if the client wishes to use NACK Option, it must use an option slot with 0x03 as its first byte and the rest 24 bits shall be filled with 0 as specified before. Upon receiving a packet with the SYN bit set and a NACK option slot, the receiver knows that this is not a message to deliver a negative acknowledgement. If it is the server, it must, if it supports NACK, send an option slot where the first byte is filled with 0x03, the other bits filled with 0. And the NACK Option will be used in this connection. If the client receives a packet with SYN bit set and a NACK option slot, it must use NACK option in this connection.
In the communication phase, if a STP instance receives a packet it is not expecting and the packet has not yet been received. It shall keep track of packets it is still missing after receiving every packet. Upon sending the next packet, if it is still missing packets, it shall use one option slot to indicate that.
The last correctly received sequence number is given implicitly through the acknowledgement number field in the STP header, this instance now use a bit-map to indicate packets that its still missing. If X is the sequence number acknowledged in this packet, bit 8+Y in this option slot indicates whether packet X+Y is received by the STP instance or not. If packet X+Y has been received and buffered, the 8+Y bit in the option must be set to 1, otherwise it must be set to 0 to indicate that the packet is missing. Since there are 32 bits

in an option slot, a total of 24 packets can be represented in an option slot. Hence no more than 24 packets can be displayed, the NACK of further packets will be held back until we receive some of the missing packets.
Upon receiving a packet with NACK options used, once the header without option slots has been processed, the receiver must calculate which packets are still missing. For every sequence number between SND.UNACK and $min(SND.UNACK + 24, SND.NXT)$, it must check the corresponding bit in the bit map and immediately retransmit packets that has this sequence number.

4.7 Interface

The Interfaces of this protocol is specified in the standard of the basic version of STP.