The Flexible Transmission Control Protocol (FTCP)
draft-rosenau-flexible-tcp-00

Abstract

This document describes a protocol that may be used as replacement for the Transmission Control Protocol (TCP).

The protocol allows changing the lower-layer address (such as IP address) or even a change of the lower-layer protocol (e.g. from IPv4 to IPv6) in the middle of the connection.

Use cases are private peer-to-peer applications (such as computer games) or mobile communication.

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 May 19, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect
1. Introduction

Traditionally for connection-oriented data exchange the Transmission Control Protocol (TCP) [RFC0793] is used. This protocol relies on the fact that the IP addresses of the two hosts involved in a connection do not change during the connection.

Many private internet connections (such as DSL) are "hung up" once in 24 hours and the IPv4 address of the host changes. This was done because the IPv4 address shortage was easier to handle this way by the internet provider.

Using IPv6 it would theoretically be possible to assign a fixed IPv6 address to each customer however many internet providers change the IPv6 addresses even multiple times in an hour because of security and/or privacy reasons.

As a result it is not possible to establish a TCP-based connection for more than 24 hours.

The protocol described in this document can be used as a replacement for the Transmission Control Protocol (TCP) which allows a change of the lower-layer address (e.g. IP address) or even the lower-layer protocol (e.g. from IPv4 to IPv6) while the connection is active.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119] and indicate requirement levels for compliant implementations.

3. Functional specification

3.1. Datagram format

The Flexible Transmission Control Protocol may use many different protocols as lower-layer protocol. Examples are: IPv4, IPv6, raw Ethernet frames, encapsulated into UDP packets...

A Flexible Transmission Control Protocol datagram has the following form (not containing the lower-layer header):
Source Connection: 32 bits
This field describes the sending host’s identifier of the connection.
Using traditional TCP a connection is identified using source address, destination address, source port and destination port.
All these four fields are required to uniquely identify a connection.
Using the Flexible Transmission Control Protocol however the "source connection" field uniquely identifies the connection on the source host and the "destination connection" field uniquely identifies the connection on the destination host. For security reasons the receiving host SHOULD check if the "source connection" field matches the "destination connection" field.

Destination Connection: 32 bits
This field describes the receiving host’s identifier of the connection.

Sequence Number: 32 bits
The sequence number of the first data octet in this segment. The calculation of the sequence number differs from the calculation in traditional TCP.

Acknowledgement Number: 32 bits
The sequence number the sender is expecting to receive.
Type: 4 bits
This field contains the type of the datagram:
0 - Acknowledge
1 - Data
2 - Connection request
3 - Connection response
4 - Connection reject
5 - Disconnection request
6 - Disconnection response
7 - Type reject
8 - Options (optional)
9 - Hold request (optional)
10 - Hold response (optional)
11 - Continue (optional)
12-15 - Reserved

Reserved: 12 bits
Reserved for future use. MUST be zero.

Window: 16 bits
Size of the "data" field measured in octets.

Checksum: 16 bits
The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the datagram NOT including the lower-level headers nor the padding.
If the datagram contains an odd number of octets ("window" is odd) a zero octet is appended to the "data" field for the checksum calculation.
The "checksum" field itself is set to zero for the checksum calculation.
Note: Unlike traditional TCP the IP addresses are NOT used for checksum calculation (traditional TCP uses a "pseudo-header").

Data: variable
Content depends on the "state" field.

Padding: variable
Additional bytes used if the lower-layer protocol requires the datagram to have a certain size. The sender SHOULD use zero bytes; the receiver MUST ignore the bytes. The padding is NOT part of the Flexible TCP datagram but it is seen as part of the lower-layer protocol.
3.2. Establishing a connection

To establish a connection a host sends a datagram of the type "connection request" (type=2) to the listening host.

The sequence number, acknowledgement number and destination connection fields MUST be zero; the source connection field is set to a value chosen by the host.

Unlike traditional TCP there is no "destination port" which is used to identify the service (such as port number 80 to identify HTTP).

Instead the "data" field identifies the service:

An empty data field ("window"=0) is used if the lower-layer protocol already specifies the service to be used. An example would be an application that uses FTCP-over-UDP; in this case the lower-layer protocol (the UDP destination port number) is used to route the incoming datagram to the correct application.

If the lower-layer protocol is not sufficient to route incoming datagrams to the correct application (for example if FTCP is really used as replacement for TCP on operating system level) the "data" field contains the name of the service to be connected to. The "window" field is the length of the name in characters. No padding or termination (such as NUL-termination) characters are used in the "data" field.

"Standard" services (as defined in "/etc/services" files) are identified by lower-case letters (such as "http" for HTTP or "ftp" for FTP).

Vendor-defined services are specified by a "reverse-domain-name" scheme: The owner of the domain "hello.example.com" might define a vendor-specific service "com.example.hello.world".

When the connection is accepted the listening host answers with a "connection response" datagram (type=3). The acknowledgement number and the sequence number are zero; the source connection is the connection identifier chosen by the listening host and the destination connection field is the value of the source connection field. The "data" field is empty ("window"=0).

If the connection is rejected a "connection reject" datagram (type=4) is sent. In this case the source connection field SHALL also be zero.
3.3. Data transmission

To transmit data a "data" packet (type=1) is sent. "data" contains the actual data to be transmitted.

An FTCP datagram does not have an "options" field (as present in the TCP datagram). Instead the special datagram type "options" (type=8) is used. In this datagram the "data" field contains TCP options as described in RFC 793 [RFC0793].

The sequence number and the acknowledgement number are incremented by "data" and "options" datagrams. The numbers are incremented by the number of octets in the "data" field (the value of the "window" field). Both fields use an ones-complement sum, so the next value after (2^32-1) is 1, not 0. Note that in traditional TCP the length of the "options" field does not affect these numbers while in Flexible TCP it does.

An empty datagram is sent if it is only desired to acknowledge the reception of a datagram or not to cause a connection time-out. These datagrams have the type 0 ("acknowledge packet").

If a packet of types 8-15 is received which is not supported by the receiver the receiver answers with a "type reject" packet (type=7). The "data" field is 1 byte long ("window"=1) and contains the value of the "Type" field which was not understood by the receiver. Neither acknowledgement number nor sequence number are modified in this case (independent of the "window" field of the packet type not supported and although the "type reject" packet has a "data" length of 1 octet).

3.4. Regular disconnection

If one host wishes to hang up (disconnect) the connection it sends a "disconnection request" packet (type=5). The sequence number and acknowledgement number fields SHALL be set correctly but the receiver SHALL ignore these fields. The data of this packet is empty ("window"=0).

The receiver answers with a "disconnection response" packet (type=6; "window"=0).

The sender of "disconnection response" SHALL ignore all further packets belonging to this connection but "disconnection request" after sending this packet. It SHOULD be prepared to receive an additional "disconnection request" packet if the "disconnection response" packet was lost. In this case the "disconnection response" packet is repeated.
Once the sender of the "disconnection request" packet received the "disconnection response" it SHALL ignore all further packets belonging to this connection.

4. Address changes

The main reason for using the Flexible Transmission Control Protocol instead of traditional TCP is that the lower-layer address of a host is changing while the connection is active.

To allow such address changes a host will check if the sequence number, acknowledgement number and source connection fields of a datagram received are matching to the destination connection field. However the host does NOT check if the lower-layer address (such as the IP address) is matching. Instead the host SHALL assume that the lower-layer address (or even protocol) of the other host has changed if the address is not matching this host’s idea of the other host’s lower-layer address.

Example:

- Host A supports two lower-layer protocols for the Flexible TCP protocol: IPv6 (Flexible TCP is the "next header" protocol of IPv6) and UDP/IPv4 (Flexible TCP frames are packed into UDP packets).

- There is an active Flexible TCP connection between host A and host B

- Host B has the IPv4 address 192.168.0.1, UDP port 1234

- Host A receives a Flexible TCP packet from address 192.168.0.2, UDP source port 4321

- The destination connection and source connection fields match the connection between hosts A and B. The source connection, sequence number and acknowledgement number fields are plausible. (The check of the sequence number and acknowledgement number is necessary to detect old packets that are not longer valid.)

- Now host A assumes that host B will use UDP port 4321 instead of 1234 for this Flexible TCP connection.

- Host A receives a Flexible TCP packet from address 192.168.2.3, UDP source port 5678. The fields (source connection, destination connection, sequence number, acknowledgement number) match the active connection.
Host A assumes that the IPv4 address of host B has changed from 192.168.0.1 to 192.168.2.3. Note that traditional TCP connection would assume that the datagram belongs to another connection in this case.

Host A receives a packet from IPv6 address FC00::2 while all the fields are matching the active connection.

Host A assumes that host B is now using IPv6 instead of UDP/IPv4 for this connection.

Both hosts MUST be prepared for an address change of the other host.

Support for lower-layer protocol changes (such as from UDP/IPv4 to IPv6 as shown in the example) is optional.

5. "Hold" feature

The "hold" feature may be used to remove a host from the network without hanging up the FTCP connection. Mobile devices might be a possible use case. Virtual private networks may be another use case.

One host sends a "hold request" packet (type=9; "window"=0).

If the other host supports the "hold" feature it answers with "hold response" (type=10; "window"=0). (Otherwise a "type reject" packet is sent.)

After having received the "hold request" or "hold response" packet the connection is in "hold" state. In "hold" state there is no connection time-out and no packets are sent. All packets but "hold request" and "continue" are ignored. "Hold request" packets are not ignored because the "hold response" packet may have been lost so the other host may send additional "hold request" packets to request a (additional) "hold response" packet from this host.

To exit hold state the higher-level application of one of the two hosts must explicitly request leaving "hold" state. Typically this will be done by a human user interaction. A "continue" packet (type=11) is sent to the other host. The other host acknowledges this "continue" packet using an "acknowledge" packet (type=0).

6. References
6.1. Normative References


6.2. Informational References


Author’s Address

Martin D. J. Rosenau

Email: martin@rosenau-ka.de