Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 14, 2011

B. Mongazon-Cazavet
Alcatel-Lucent Bell Labs

October 11, 2010

TCP Rehash
draft-mongazon-tcpm-tcp-rehash-00

Abstract

The present specification describes a light extension of the TCP protocol [RFC793] that allows a TCP transport connection to be maintained operational whenever the underlying IP network address of its end-points changes. This includes situations where a single-interface host changes its IP address or where a multiple-interface host diverts a connection from one interface to another. The mechanism used to maintain a transport connection in such situations is based on the capability of both end-points to "Rehash" a TCP connection (rebuild its lookup key) with a new IP address in a fast and reliable manner.

Requirements Language

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

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 April 14, 2011.

Copyright Notice

Table of Contents

1. Introduction

TCP Rehash is designed as a light mechanism to provide a "fair-enough" (could say "low-cost") connection mobility and migration in areas where mobility protocols such as [RFC3344] and [RFC3775] are not desirable or applicable. TCP Rehash is currently defined for IPv4 version but could also apply to IPv6 version with minor adaptation. TCP Rehash allows connected hosts to notify each other, on a connection basis, of their IP address changes through time in order to keep the connections updated with their current underlying IP address. A TCP connection between two hosts is generally maintained in each host using a lookup key that depends on the source and destination IP addresses of its end-points. Whenever a host decides to change it's current IP address for a connection, it sends to the other host an order to "rehash" the connection with the new IP address prior to pursue activity. TCP Rehash specifies how such an order needs to be provided through the TCP protocol to achieve connection mobility and migration in a safe, fast and reliable manner.

TCP Rehash is designed as a light and backward compatible TCP extension but does not necessarily apply to all underlying IP networks. In particular, the protocol extension is not expected to work properly when the IP network encompasses NAT and firewall devices. Future revisions of the present document might solve such a limitation. In the present document, the term "mobility" shall be understood for both connection mobility and connection migration. From the TCP Rehash perspective there is no difference between both aspects.

2. Comparison with others mobility protocols

To the contrary of Mobile IP protocols family, TCP Rehash does not require mobility agents such as FA and HA to be present in the network to provide session continuity through IP address preservation when hosts change their address. In addition, execution of TCP Rehash is expected to be more simple and efficient than execution of MIP protocols. This includes the following major differences:

- No need for hosts to register/identify/authenticate toward mobility agents
- Faster handover time due to the reduced number of signaling messages

TCP Rehash is a host-to-host mobility protocol that might compete with numerous similar protocols such as [RFC5201], [I-TCP], [M-TCP],

[TCP-MIGRATE] and [MPTCP] to name a few. However, TCP Rehash claims to be more limited, simple and efficient and it would not compete as a general host mobility solution. Moreover TCP Rehash has been implemented in Linux Kernel with less than 2000 lines of source code which make it attractive for research and innovation activities.

3. Applicability

TCP Rehash is intended to meet a new family of applications that require simple transport mobility without the need to implement application-level mobility mechanisms such as the one currently provided by SIP based (VOIP) and P2P software. In particular, TCP Rehash claims to be invisible to applications that use the regular socket API [SOCKET].

TCP Rehash is intended to be used in plain (or flat) IP networks that do require light and agile mobility support.

4. Usage restriction

Simultaneous mobility / migration of a transport connection cannot be guaranteed. Due to TCP Rehash design, it is not possible to move the same connection on both peers exactly at the same time. However, the probability for such an event to occur is very minimal and should be considered as a minor restriction.

Transport connection continuity cannot be guaranteed when Firewall / NAT devices are found on the path between TCP Rehash hosts.

5. Communication model

Regular hosts establish peer TCP/IP connections and exchange byte streams of application data. TCP Rehash performs signaling on a peer connection basis. The mobility signaling is handled by hosts at networking stack level (inside TCP code) and based on the exchange of specific TCP options called TCP Rehash options. Such options are added to the current TCP flow, so TCP Rehash signaling shall be considered as an "in-band" mobility signaling. TCP Rehash options can be carried in almost any TCP protocol packet (DATA, ACK, SYN...) possibly together with other TCP options (SACK, TS). TCP Rehash options are of very limited size to avoid oversizing of the whole TCP Options area. TCP Rehash options should be added in a "in-band" manner and shall not break the regular TCP logic.

However, in the case where there is no TCP traffic to carry a TCP

Rehash Option, a duplicate ACK shall be generated and sent on the connection to carry the Rehash option. A duplicate ACK might be generated at host convenience for example after a period of inactivity time. Whenever TCP packets carrying a TCP Rehash option needs retransmission, the TCP Rehash option shall go again together with the retransmitted packet without breaking the TCP ordering logic.

In order to make the TCP Rehash protocol generic and extensible, a single TCP option format is used. The format includes a control header and common fields to hold IPv4 addresses and the authentication token.

6. Protocol operation

TCP Rehash protocol specifies two operation flow charts:

1. REHASH-INIT exchange

2. REHASH-ADDR exchange

6.1. REHASH-INIT exchange

A TCP Rehash host (Host A) initializes a rehash-capable connection towards another TCP Rehash host (Host B) by performing the init exchange. The init exchange is done on an individual connection basis preferably at connection establishment time (SYN, SYN-ACK). A TCP Rehash init can however be performed at any time during connection lifetime. The init exchange might be initiated by any side of the connection.

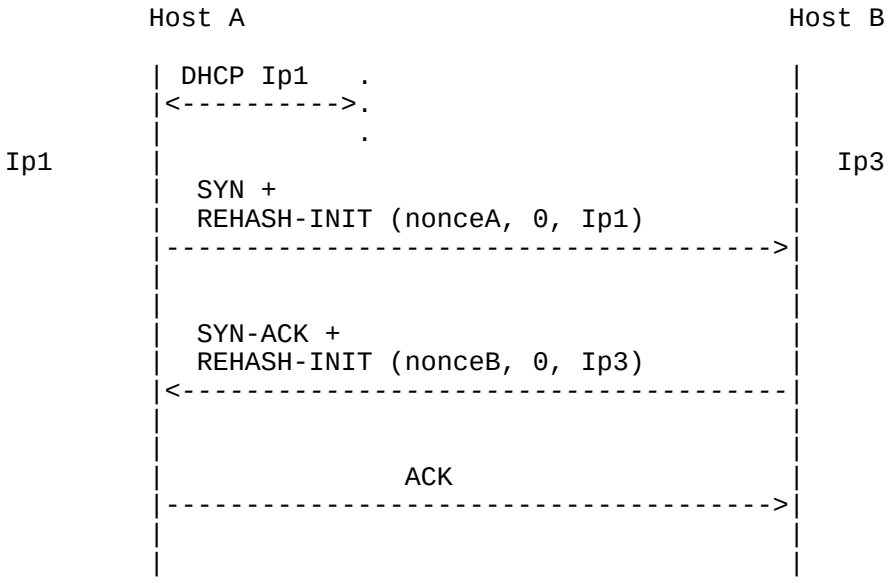
Host A sends a REHASH-INIT option to host B that sends a REHASH-INIT option in reply. REHASH-INIT options are not numbered nor correlated. If host B does not support TCP Rehash, no REHASH-INIT is received in reply. In such a case, host A might still issue a REHASH-INIT option in subsequent TCP messages but will not mark the connection with rehash capability until a REHASH-INIT is received. The reissue process might be stopped after some time. TCP connection that are not marked with Rehash capability proceed to regular TCP processing. In this case, connection mobility will not occur. REHASH-INIT options might cross each other if issued by both peers at the same time.

When exchanging REHASH-INIT options, each peer provides to the other peer a random nonce value for authentication purpose. Each peer shall store its own nonce and the peer's nonce in the connection context. The peer's nonce shall be provided later during REHASH-ADDR

exchange initiated by a host. Peers are expected to generate a separate random nonce on a per connection basis to increase security of protocol operation. In the present specification nonce are valid for the whole lifetime of the connection, possibly including mobility of the connection. Nonce computation might typically be performed using random number generation.

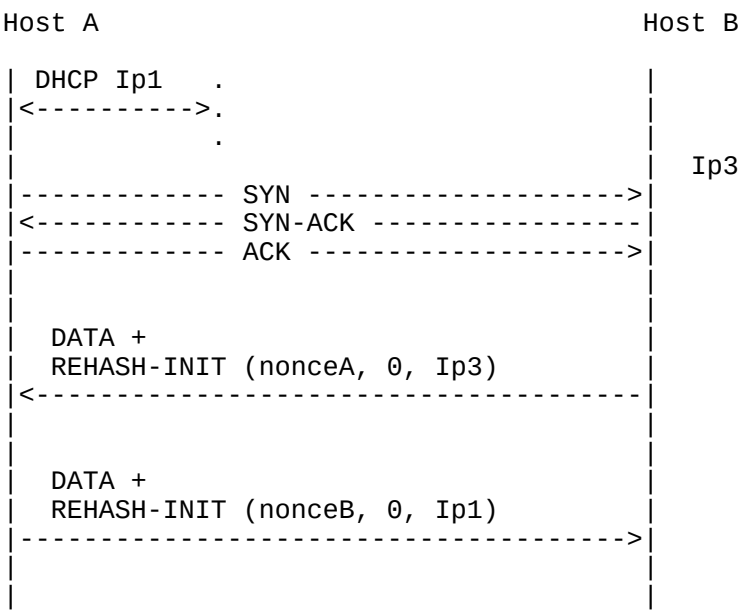
In addition to the nonce, the REHASH-INIT option carries the previous (old) and current (new) IP addresses of the option sender. The previous (old) IP address shall be set to 0 during init exchange. The current IP address shall be set to the current source IP address of the connection as resolved by the TCP/IP stack.

The following diagram shows REHASH-INIT exchange synchronized with the TCP connection establishment. Note that the REHASH-INIT exchange is a two-way handshake while connection establishment is a three-way-handshake. Should the three-way handshake fail, the need for connection rehash disappears with the connection since only established connections can be "rehashed". Should the connection establishment phase need retransmission(s) of SYN, SYN-ACK or ACK packets, REHASH-INIT options should be sent together again with retransmitted packets.



The following diagram shows REHASH-INIT exchange performed by Host B after connection establishment. Should DATA or ACK packets need retransmission, REHASH-INIT options should be sent together with

retransmitted packets.



6.2. REHASH-ADDR exchange

When host A changes the underlying IP address of a connection (mobility or migration), it shall send a REHASH-ADDR option onto the connection providing it has previously performed a valid init exchange on the connection. Host A shall provide the peer nonce of the connection together with the REHASH-ADDR to authenticate the command from the peer perspective. It shall also provide its previous (old) and current (new) IP address in the option. Prior to issue the REHASH-ADDR option, the initiator of connection movement must perform locally the connection rehash and ensure that the REHASH-ADDR option is mandatory carried in the first outgoing packet flowing from the new IP address.

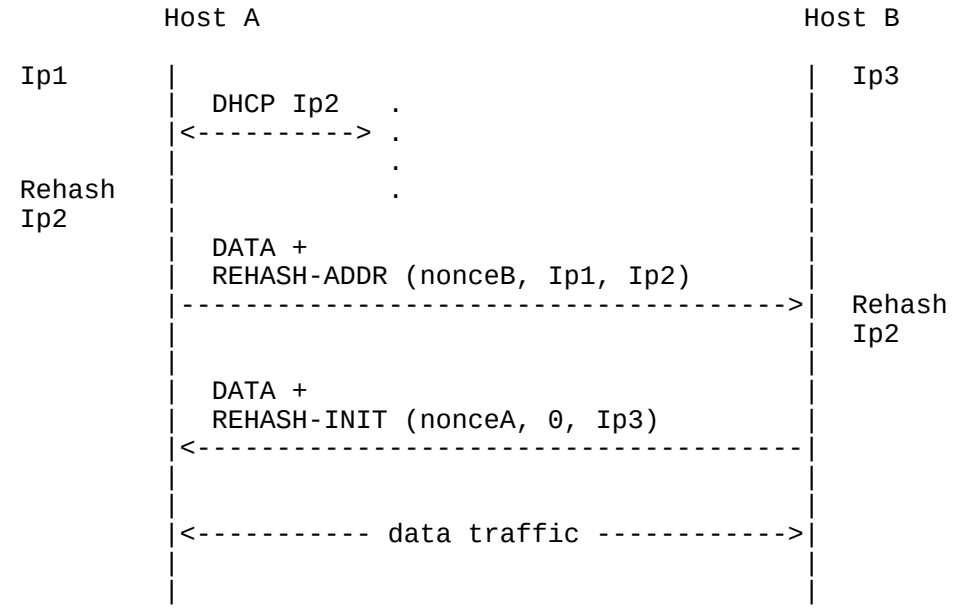
A regular TCP implementation would reject (with a RST) a packet received with the REHASH-ADDR option since the hash computed from the new source address does not match an existing connection. As an exception case to regular TCP processing, TCP Rehash shall trap such a condition and evaluate the presence of a REHASH-ADDR option in the packet. If present, TCP Rehash shall attempt to lookup the connection giving the old IP address provided by the peer. If the connection is found and the nonce provided in REHASH-ADDR option matches the nonce locally stored, TCP Rehash must rehash locally the

connection with the new IP address provided in the option and proceed to regular TCP packet processing.

Should the nonce carried in the REHASH-ADDR be invalid (does not match the own nonce sent to the peer during init exchange), the receiving host drops the packet silently. If the REHASH-ADDR has been successfully processed, the receiving host sends back a REHASH-INIT option to acknowledge the rehash process. The sender of a REHASH-ADDR option shall continue to issue a REHASH-ADDR option in outgoing packets until a REHASH-INIT is received.

Should a packet carrying the REHASH-ADDR option be retransmitted, the option shall be sent again in the retransmitted packet. Based on implementation experience, hosts might allow for a maximum number of REHASH-ADDR retries. A connection that fails to rehash shall be closed by the host that initiated the rehash process.

The following diagram shows a successful REHASH-ADDR cycle initiated by Host A. Host A uses Ip1 for its current connection and changes to Ip2 through a DHCP cycle. The old IP address is Ip1, the new IP address is Ip2.



Note that TCP packets might be in-flight while a host initiates a

REHASH-ADDR exchange. As a consequence, in-flight packets received after REHASH-ADDR option has been sent by the initiator do not match an active connection. A regular TCP implementation would reject such packets (with a RST) possibly closing the connection being migrated. As an exception case to regular TCP processing, TCP Rehash shall detect that such packets belong to a connection whose rehash is in progress and drop them silently. This can be achieved by storing the old IP address in the connection context. Any dropped "in-flight" packet will be retransmitted using regular TCP mechanisms when rehash completes.

Also note that following a successful REHASH-ADDR cycle, the connection should enter the slow start mode.

7. TCP Rehash Option format

TCP Rehash uses a single TCP Option called TCP Rehash option.

Until the option is assigned an official value by IANA, the TCP Rehash option code is set to 253 (0xFD). The TCP Rehash Option length is set to 16 (0x10) bytes including 2 bytes for Option code and Option value. The TCP Rehash Option length bytes-aligned is set to 16 (0x10) bytes.

The TCP Rehash option is used to carry the following logical messages:

a. REHASH-INIT

b. REHASH-ADDR

REHASH-INIT is used to both initialize a connection between peer hosts for TCP Rehash processing and to acknowledge a REHASH-ADDR after successful rehash.

REHASH-ADDR is used to change an extremity of a connection between peer hosts that has been previously initialized for rehash processing.

REHASH-INIT and REHASH-ADDR use a common TCP option format as specified below.

TCP Rehash Option format

Byte 0-1 Control Header

0x0001 REHASH-INIT
0x0010 REHASH-ADDR

Other values are undefined and reserved for future use.

Byte 2-5 Authenticator

Nonce value.

The nonce value shall be initialized by the sender of REHASH-INIT and stored in its connection context (own nonce). The nonce value shall be stored by the receiver of REHASH-INIT in its connection context (peer nonce).

The nonce value shall be set to the peer nonce value stored in the connection context by the sender of REHASH-ADDR and compared to the own nonce value stored in the connection context by the receiver of REHASH-ADDR.

Computation of the nonce value is implementation defined.

Byte 6-9 Old (previous) IP address

IPv4 address.
Set to 0 in REHASH-INIT.
Set to old (previous) IP address in REHASH-ADDR.

Byte 10-13 New (current) IP address

IPv4 address.
Set to current IP address in REHASH-INIT.
Set to new IP address in REHASH-ADDR.

8. TCP implementation aspects

The present specification is an optional extension to a regular TCP stack implementation. The extension might be activated for the whole TCP stack or on an individual connection basis. This choice is left to implementors. The TCP Rehash implementation shall not impact

applications that use TCP services. In order to do so, the present specification provides the following recommendations:

- Performing the Rehash init exchange in sync with connection establishment is preferable.
- If an initiated rehash fails, drop the connection locally.
- When rehash is initiated, look for local connections in listening state that are bound explicitly to the old IP address and also rehash such listening connections with the new IP address. Local connections in listening state that are bound to the "any" address might not need rehash.
- be fully-compatible with peers that do not implement TCP Rehash.

The present specification might be added some specific socket extensions to:

- request the usage of TCP Rehash on a connection basis and provide nonce and configuration values
- be informed of TCP rehash activity (new IP address, rehash success/failure...)

9. IANA Considerations

TCP Rehash requires a specific TCP option number to be allocated by IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

10. Firewall and Network Address Translation considerations

In the current release of the protocol, TCP Rehash is not guaranteed to operate when firewall and NAT devices are present in the path between hosts. Further release of the protocol might provide extension to remove such a restriction.

11. Security Considerations

TCP Rehash can be used in 3 different network security contexts:

- Secure network (case 1)
- Insecure network but IPsec or another form of secure tunnel in use (case 2)
- Insecure network without tunnels (case 3)

In case 1, hosts implementing TCP Rehash are fully trusted each other and do not need to authenticate or encrypt data exchanges. The usage of nonce in TCP Rehash options is sufficient to operate the protocol properly.

In case 2, hosts implementing TCP Rehash are not trusted each other. The TCP Rehash protocol can be used over IPsec tunnels (or another form of secure tunnels) established between hosts themselves.

In case 3, IPsec (or another form of secure tunnel) is not possible or desirable, the TCP Rehash protocol shall not be used.

In cases 1 and 2, man-in-the-middle attacks are not possible given the assumptions.

TCP Rehash is expected not worse than [MPTCP-THREAT]concerning connection hijacking.

12. Acknowledgements

The author would like to thank his colleagues from Alcatel-Lucent Bell Labs: Paul Polakos, Peter Bosch, Sape Mullender, Georg Hampel, Telemaco Melia, Davide Cherubini, Noah Evans, Johan Moreels, Rouzbeh Razavi, Jim McKie and Michael Scharf for their support and help.

13. References

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

13.2. Informative References

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, June 1999.

[RFC3344] Perkins, C., "IP Mobility Support for IPv4", RFC 3344, August 2002.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC5201] Moskowitz, R., Nikander, P., Jokela, P., and T. Henderson, "Host Identity Protocol", RFC 5201, April 2008.

13.3. References

[I-TCP] Bakre, A., "I-TCP: Indirect TCP for Mobile Hosts", October 1994.

[M-TCP] Brown, K., "M-TCP: TCP for Mobile Cellular Networks", July 1997.

[MPTCP] Eardley, P., "Multi path TCP Working Group", January 2009.

[MPTCP-THREAT] Bagnulo, M., "Threat Analysis for Multi-addressed/ Multi-path TCP (draft-ietf-mptcp-threat)", September 2010.

[RFC793] DARPA, "RFC793 : Transmission Control Protocol", September 1981.

[SOCKET] IEEE, "POSIX.1g - IEEE Std 1003.1g-2000", May 2000.

[TCP-MIGRATE] Snoeren, A., "TCP Connection Migration", November 2000.

Author's Address

Bruno Mongazon-Cazavet
Alcatel-Lucent Bell Labs
Centre de Villarceaux - Route de Nozay
Nozay, 91620
France

Email: bruno.mongazon-cazavet@alcatel-lucent.com