INTERNET DRAFT
Expires May 2001

A. C. Snoeren
MIT LCS
H. Balakrishnan
MIT LCS
November 2000

TCP Connection Migration
<draft-snoeren-tcp-migrate-00.txt>

Status of this Memo

Abstract

This document describes a set of TCP options to support the migration of an active TCP connection across IP addresses and TCP port numbers. Using this option, a TCP peer can open a migrateable connection, transfer one or more bytes on it, and continue the connection from another IP address/TCP port pair in an application-transparent fashion. The set of addresses or ports from where a connection might be continued need not be known in advance. Security against connection hijacking is achieved using a secret cryptographic cookie negotiated through an Elliptic Curve Diffie-Hellman [ANSI-X962] exchange during connection establishment. The initiation of migration can be done either by one of the communicating peers, or by a trusted third-party that presents the negotiated cryptographic

cookie.

TCP connection migration has several applications, including preserving TCP communication across host mobility in the Internet, and inter-host migration of connections for fault-tolerance or load-balancing. Furthermore, unlike previous approaches such as ETCP [Huitema95], the Migrate options require no modification to the TCP header, packet format, or semantics, and can be initiated by a trusted node different from the communicating peers.

Introduction

A TCP connection [RFC-793] is associated with a particular IP address and TCP port pair. When one of the communicating peers changes its IP address, all TCP on-going connections are terminated. This document describes a set of TCP options to migrate connections from one IP address to another, thereby permitting on-the-fly re-addressing of end-hosts. In-progress connections may also change port numbers, enabling them to function correctly across legacy Network Address Translators (NATs) and Port Address Translators (PATs) [RFC-2663].

Although TCP connections today are bound to particular IP addresses, most TCP applications communicate with end-hosts or end-services. An end-host (e.g., a desktop or mobile computer) may have multiple network interfaces on it, or have one interface whose address changes due to mobility, while an end-service (e.g., a Web server) may be replicated on several different computers on the Internet. Most TCP applications open a connection to a name, which gets resolved by the Domain Name System (DNS) to an IP address. If the IP address associated with the name were to change for any reason, connections in progress cannot continue. The TCP migration options are motivated by the desire for a TCP that can work across changes in name-to-address mappings, e.g., due to host mobility, or server-to-server migration of one end-point of a connection for fail-over or load-balancing reasons.

The Migrate options allow hosts to divert established TCP connections to a new IP address/TCP port pair by referencing them with a cryptographic cookie, which protects against connection hijacking by malicious parties. This cookie is established using an Elliptic Curve Diffie-Hellman key exchange during the establishment of the initial connection [ANSI-X962]. The security provided by this cryptographic cookie also allows a trusted host other than the communicating peers to take over a connection. Any host with knowledge of the cookie can request migration of the connection, allowing servers in a cluster to continue servicing a client of a failed server.

Document Conventions

The terms "migrating host" and "mobile host" are used to refer to the TCP peer that is being re-addressed. The connection, however, is completely symmetric. Either (or both) peer(s) may move after the connection is established, but not at the same time. During a migration by the migrating host, the peer TCP host is referred to as the fixed host.

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].

Background

Mobile IP [Perkins] supports re-addressing and could be used in this context, but has two major drawbacks. First, all traffic to a mobile host must (at least initially) travel via the home agent. The necessity of a home agent can be a significant burden, especially in ad-hoc networks of devices. Additionally, depending on the network topology, packets routed through the home agent may travel a significantly longer path than necessary.

Secondly, packets sent by the mobile host use the mobile's home address as the source address. It is often the case that this source address does not topologically belong to the subnet that the mobile is currently attached to. Current firewalls and secure routers often block such traffic to prevent IP spoofing. As an alternative, packets may be "reverse tunnelled" back to the home agent before being sent to the destination using the host's home IP address. Unfortunately, this imposes the "trangle routing" penalty described above on packets leaving the mobile host as well.

More importantly, however, Mobile IP requires that the original (home) address continue to be allocated to the mobile host. In cases when the available address space is small, such a restriction may make re-addressing prohibitive. One would like the mobile node's previous address to be immediately reusable by another host.

The widespread adoption of the incrementally-deployable migration options described in this document will obviate the need for network-layer approaches like Mobile IP to handle host mobility for TCP applications. Furthermore, many mobile computers use multiple network interfaces; selecting between them based on the available connectivity and network performance on a per-connection basis is supported by connection migration.

The TCP migrate option avoids these drawbacks, but functions only for

TCP. UDP-based protocols typically fall into two classes: short transactions and longer streams. Transactional protocols, such as DNS and Remote Procedure Call (RPC) protocols are typically mostly stateless and involve at most a small number of packets. Although the server in such protocols might move, locating it is a matter of determining the most current name-to-address binding, which can be done using dynamic DNS [RFC-2136]. It is, of course, possible for the requesting host to have moved concurrently, but in a network where the probability of this occurring is significantly smaller than the probability of multiple packet losses (as is typically expected in today's Internet), an application-level retry (consisting of a DNS lookup) of the form used for loss recovery should suffice.

UDP-based unicast streaming applications, on the other hand, tend to make use of an application-level protocol like RTP [RFC-1889] to implement a connection-like abstraction. Although this document does not describe any details, we expect that a connection migration scheme for such protocols to be easy to develop, similar to the one for TCP proposed herein. Alternatively, the Transport Area Working Group has recently proposed the Stream Control Transport Protocol (SCTP) [SCTP], which provides similar "multi-homing" functionality, although the semantics provided by the current draft are not as amenable to address changes.

Extending the lives of TCP connections across address changes is not a new idea. A proposal based on a "context identifier" (essentially equivalent to our cryptographic cookie) was proposed by Huitema [Huitema95] and Steve Deering. However, these identifiers were passed in the clear, so any eavesdropper could hijack the connection quite easily.

Huitema's "Multi-homed TCP" allowed multiple concurrent addresses on both ends of the connection, but required a new protocol, extended TCP (ETCP) to be deployed on the end hosts [Huitema95]. While our proposal limits the connection to one active address per peer at a time, it requires significantly less modification to the TCP stack, and is compatible with many protocol-based firewalls, filters, and such "middle boxes" that are a reality of today's Internet infrastructure.

Alternative techniques, aimed at separating routing identifiers (e.g., IP addresses) from endpoint identification (e.g., permanent endpoint identifiers, often called EIDs), have been proposed. We believe that such proposals are worthy of research, but that they are likely to incur greater deployment costs. The approach proposed in this document is rather different, since it does not require additional level of global addressing or indirection, but only a (normally pre-existing) DNS entry for endpoint identification.

Furthermore, any technique based upon "permanent identifiers" only pushes the problem one level higher, and awaits the time when those too need to be re-assigned.

Basic Approach

The Migrate Options facilitate rebinding TCP connections to a new address/port pair that need not be known a priori. This differs for the "multi-homing" capability found in SCTP [SCTP], which requires the set of approved addresses for the connection to be negotiated in advance. Further, standard TCP connection semantics are preserved for transparent operation in the presence of legacy network infrastructure such as NATs or PEPs. Finally, this is achieved with signalling sent only on connection establishment--individual data segments have no additional overhead.

Fundamentally, the Migrate options allows corresponding hosts to synchronize two separate TCP connections such that the context is identical. In particular, establishment of the second connection terminates the first, and transfers the original TCB to the second connection, except for the IP address/port binding, and congestion control state. This allows TCP (and any of its attending options) to function properly across both connections, regardless of its current state.

By establishing a completely separate connection to continue communication, the correspondent hosts may conspire to support connection migration without the network's knowledge or consent. Both connections appear to be standard TCP connections (except for the new options passed in the SYN packets), so stateful firewalls, NATs, PATs, and the like will not interfere with proper operation.

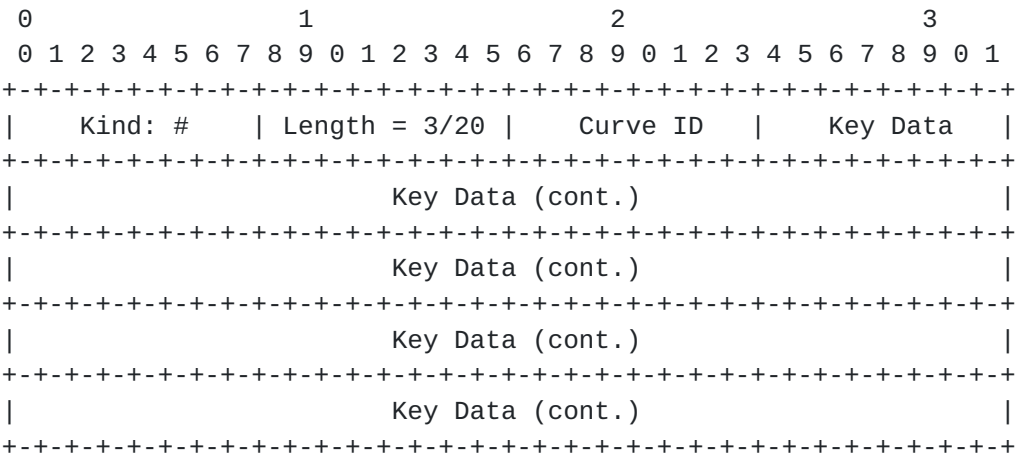
The migration process consists of two parts: an initial key exchange conducted during the three-way handshake, and a migration request. In order for a connection to by migrateable, the correspondent hosts must first exchange Migrate-Permitted Options during the SYN handshake. This negotiates a shared connection token, which is then used to identify the connection in later Migrate Requests (as opposed to the IP address/port pair, which may change).

At any point after the initial SYN handshake (while the connection is in either ESTABLISHED or FIN-WAIT) either host may attempt to establish a second connection (from a new address/port pair) containing a Migrate Request option. This Migrate Request option contains the token referencing the previously established connection. If the request meets various security requirements described below, the correspondent host terminates the first connection (without

emitting any further packets) and instantiates the second connection with a TCB identical to the first. The new connection is associated with the same socket (application) as the first. In practice, this is often implemented by simply modifying the TCB of the first connection rather than allocating an entirely new one.

Migrate-Permitted Option

This variable-length option SHOULD be sent in SYN segments by a TCP that has been extended to support the Migrate options. It MUST NOT be sent on non-SYN segments.



TCP Migrate-Permitted Option Format

The Migrate-Permitted option has two varieties. It may either specify the non-secured variant of the Migrate Options (NOT RECOMMENDED) or the secure version. The non-secure version has length 3, and contains a zero in the Curve ID field.

The secured version has length 20 and contains two fields, a one byte Curve ID field and 17 bytes of Key data. The Curve ID contains one of a predefined set of values to select a particular set of Elliptic Curve parameters. The Key Data contains the most significant 17 bytes of the 25 bytes of Key Data used to compute the secret connection token.

In order to support key lengths up to 200 bits, the remaining 8 bytes of Key Data are transmitted in the Timestamp option [RFC-1323], which MUST be included on any SYN packet containing a non-secure variant of the Migrate-Permitted Option. The least significant 4 bytes are transmitted in the Time Stamp Echo Reply (TSecr) field, and the remaining 4 bytes are transmitted in the Time Stamp Value (TSval) field.

Migrate Option

The Migrate option is used to request the migration of a currently open TCP connection to a new address. A variable-length option, it may only be sent in a SYN segment to a host with which a previously-established connection already exists (in the ESTABLISHED or FIN-WAIT states), over which the Migrate-Permitted option has been negotiated. It MUST NOT be sent in non-SYN segments.

TCP Migrate Option Format

A 19-byte option, the Migrate Option includes four fields. There are two 64-bit fields in a Migrate option: a Token, and a Request. In addition, there is a 7-bit sequence number field, ReqNo, which must be monotonically increasing with each new migrate request issued by an end host for a connection, and a 1 bit Preload flag field. The Preload flag SHOULD be zero, unless the Migrate Option is sent by a host other than one of the original two corresponding hosts.

Migrate-Permitted Processing

A host wishing to establish a migrateable connection MUST include a Migrate-Permitted option in the initial SYN segment. For the insecure variant, the length MUST be 3, and the Curve ID field set to zero. In the secure variant, the Curve ID must contain a defined value specifying a particular set of Elliptic Curve domain parameters. In particular, each set of domain parameters contains a finite field, F, a generating point P, and the order of P, n. The Key Data field MUST contain a randomly generated public key, k, generated over the field F by selecting a random number X in the range [1,n-1] and computing

$$k = X * P,$$

where the * operation is the scalar multiplication operation over the field F. The securing of the connection hinges on the secrecy of the negotiated key, hence X should be randomly generated and stored in the control block for each new connection. Any retransmissions of the SYN packet MUST contain the same values in the Migrate-Permitted Option.

Upon receipt of an initial SYN (without an ACK) with a Migrate-Permitted Option, a compliant TCP MUST include a Migrate-Permitted Option in its SYN/ACK segment or send a RST. If a SYN/ACK is sent, the length and Curve ID MUST be identical to those in the received SYN segment. If the secure variant is transmitted, it must be transmitted along with a Timestamp option, and contain a random public key selected as described above.

If the implementation does not support the domain parameters corresponding to the Curve ID in the received option, it MUST NOT include a secure Migrate-Permitted option in its SYN/ACK.

Any necessary retransmissions of SYN or SYN/ACKs MUST include identical values for the Migrate-Permitted option, if present. A compliant host receiving a SYN/ACK with the Migrate-Permitted option set MUST compare the Curve ID. If they differ, a RST MUST be sent and the connection aborted as in [RFC-793].

If both acceptable, secure-variant Migrate-Permitted Options are received by both hosts, they MUST compute a shared secret key, K, using their random value, X, and received public key, k,

$$K = k * X,$$

where * is the same operation as before. If the indicated domain parameters result in a key of length less than 200 bits, the value MUST be left padded with zeros to 200 bits. If the unsecured variant is negotiated, K MUST be set to all zeros. The secret key is then used to compute a connection validation token T, by computing the SHA-1 hash of the initial sequence numbers of the SYN (Ni) and SYN/ACK (Na) packets as:

$$T = \text{SHA-1}(Ni, Na, K)$$

SHA-1 produces a 160-bit hash, of which only the 64 most significant bits are retained, and stored in the connections TCB.

Timestamp Option Processing

As stated above, the TSecr and TSval fields of a Timestamp option sent in the same SYN segment as a Migrate-Permitted option MUST contain the least significant 8 bytes of the Migrate-Permitted Key

Data field. Hence these values MUST NOT be used in either of the algorithms specified in [RFC-1323]. We note that both Protection Against Wrapped Sequence Numbers (PAWS) and the Round-Trip Time Measurement (RTTM) are performed only on Synchronized connections, hence a compliant TCP stack should not operate on these values in a SYN segment in any event.

The ramifications of the Migrate-Permitted option's interaction with the Timestamp option are two-fold. First, a Migrate-compliant host MOST NOT echo the value contained in the TSval from the SYN segment in the Timestamp option of its SYN/ACK. Instead, it should fill both the TSval and TSecr fields of the Timestamp option as described above. Secondly, a host receiving a SYN/ACK with the Migrate-Permitted option MUST NOT perform either a PAWS check or a RTTM calculation on the values in the Timestamp option. It should, however, echo the value in the TSval back in the TSecr field of its next segment, as specified in [RFC-1323]. After the three-way handshake, timestamp processing SHOULD proceed as specified in [RFC-1323].

Before Migration

If a connection becomes aware that its address/port pair is about to change, the migrating host SHOULD pause transmission of all buffered TCP traffic, and MAY generate a duplicate ACK advertising a receive window size of zero to suppress further data transmissions from the fixed host.

Requesting Migration

If the host requesting migration is not the original host, it MUST preload the TCB with the appropriate state extracted from the TCB at the previous host. This data can be stale (the current sequence space information, for example), but must be from the original connection.

In any event, before resuming communication, the congestion-related state of the TCP connection must be reset to initial slow-start conditions in order to re-discover the properties of the new path in both directions.

After the host has determined its new address, a SYN segment SHOULD be generated including a Migrate Option, using the sequence number of the most recently acknowledged byte. The Token field should be filled with the Token previously computed and stored in the TCB. The ReqNo field MUST contain a monotonically increasing number in the range [0-127], with standard sequence number wraparound. The Request field MUST contain the the following computed value:

R = SHA-1(Ni, Na, K, S, ReqNo)

where Ni and Na are the initial sequence numbers for the connection as before, K is the secret key, ReqNo is the ReqNo field from this option, and S is the sequence number of this SYN segment. If the connection had previously negotiated the non-secured variant, K should simply be all zeros.

This SYN segment MUST NOT contain any options except the Migrate option. TCP MUST then move the connection to the SYN-SENT state.

Upon receipt of a SYN/ACK, the connection then resumes as before, with all previously negotiated options enabled. The sequence number in the ACK field of the SYN/ACK should be treated as an acknowledgement of data transmitted with that sequence number, and not of the SYN segment (the SYN segment need not be explicitly acknowledged). Further, if the Preload flag was set (meaning this host was not an original end point), it should examine the sequence number of the SYN/ACK and consider that the most recently acknowledged segment.

Migrate Option Processing

Upon receipt of a SYN packet with the Migrate Option set, a TCP stack that supports migration MUST attempt to locate the connection associated with the provided Token. If none is found, a RST MUST be generated and the SYN ignored. Otherwise, the ReqNo should be compared to the most recently received ReqNo on this connection. If it is not larger, this SYN packet MUST be treated as a duplicated and handled accordingly. If it is greater, the host should generate a request as described above and compared it to the received packet. If they do not match, the host MUST generate a RST segment and continue.

Only if the Request matches should the host update its stored value of the ReqNo and change the address and port in the TCB to match that of the received IP datagram. After altering the TCB, a SYN/ACK packet should be generated using the sequence number of the last successfully ACKNOWLEDGED byte of data, and the ACK field set to acknowledge that last data packet on the previous connection (the sequence number in the SYN packet is ignored), and the connection placed in the SYN-RCVD state. Further, the congestion-related state of the connection MUST be reset to initial values. Upon receipt of an ACK, the connection continues as before.

If the received Migrate Option has the Preload flag set, the connection MUST flush any previously-received out-of-order packets and corresponding SACK blocks.

SYN Processing

SYN packets may now correctly arrive on a bound port not in the LISTEN state. They should be processed only if they contain the Migrate Option as specified above. Otherwise, they should be treated as specified in [RFC-793].

RST Processing

Migration requires additional care when handling RST packets, as an unfortunate circumstance may arise when the address used to establish a migrateable connection has been reassigned, and transmitted packets reach a new host with the same address before the old host has had a chance to properly migrate the connection.

In this case, a RST packet will be generated by the new host. [RFC-793] specifies that connections should be immediately closed upon receipt of a RST when in the ESTABLISHED state. If, however, the connection had negotiated the Migrate-Permitted option, it MUST NOT be immediately closed. Instead, it should be placed in a new MIGRATE-WAIT state.

MIGRATE-WAIT State

TCP connections in the MIGRATE-WAIT state SHOULD NOT emit any segments, instead, they MUST wait until either a SYN packet with the Migrate option is received, which will resume the connection, or the appropriate timeout (identical to the TIME-WAIT state) has occurred, in which case the connection is closed.

Any packets received while in the MIGRATE-WAIT state should be processed as in the ESTABLISHED state, except that no ACKs should be generated. Further, any RST packets received in the MIGRATE-WAIT state MUST be ignored.

Security Considerations

The ability to blindly hijack connections is dangerous. It is therefore of paramount importance that the cookie is cryptographically secure, such that it is highly unlikely an attacker can guess the appropriate cookie to hijack a connection. The cryptographic security of a cookie generated using the Elliptic Curve Diffie-Hellman technique described in this document is fundamentally based on the quality of the random number generator used.

Firewalls

Many currently-deployed firewalls enforce restrictions on the directions in which TCP connections can be established. Most do so by monitoring network traffic for TCP SYN packets, and blocking those traveling in the wrong direction. The Migrate Option requires sending a SYN packet from the mobile host to the fixed host. Depending on the direction the connection was originally established, this may or may not be permitted by the firewall.

IPsec Interactions

Note that Security Associations are established on a address basis. When a TCP connection with an associated SA is migrated, a new SA MUST be established with the new destination address before communications are resumed. If the establishment of a this new SA would conflict with existing policy, the connection MUST be closed.

Denial of Service

SYN packets carrying invalid Tokens are rejected immediately, so no additional resources are wasted processing invalid packets. Hence the TCP migrate option introduces no additional security concerns in addition to standard TCP, modulo the cryptographic security of the cookie used to migrate connections.

Acknowledgments

We thank Allison Mankin for encouraging us to document the Migrate options in this form and Dave Andersen for his comments on earlier versions of this document.

Authors' Addresses:

Alex C. Snoeren
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
USA
Phone: +1 617 452-2820
Email: snoeren@lcs.mit.edu
Web: http://www.mit.edu/~snoeren/

Hari Balakrishnan
MIT Laboratory for Computer Science
200 Technology Square
Cambridge, MA 02139
USA

Phone: +1 617 253-8713
Email: hari@lcs.mit.edu
Web: http://nms.lcs.mit.edu/~hari/

References

[ANSI-X962] American National Standards Institute, "Public Key Cryptography for the financial security industry: The Elliptic Curve Digital Signature Algorithm. ANSI X9.62-1998, January 1999.

[Huitema95] Huitema, C., "Multi-homed TCP", Internet Draft, May 1995. Now expired.

[Perkins] Perkins, C., "IP Mobility Support for IPv4, revised", Internet Draft, October 1999.

[RFC-793] Postel, J., "Transmission Control Protocol", STD 5, RFC 793, September 1981.

[RFC-1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 5, RFC 1122, August 1989.

[RFC-1323] Jacobson, V., Braden, R., and Borman, D., "TCP Extensions for High Performance", RFC 1323, May 1992.

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

[RFC-2136] Vixie, P., et al., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC-2401] Kent, S., and Atkinson, R., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC-2663] Srisuresh, P., and Holdrege, M., "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[SCTP] Stewart, R., et al., "Stream Control Transmission Protocol", Internet Draft, draft-ietf-sigran-sctp-13.txt, July 2000.