Behave	S. Perreault, Ed.
Internet-Draft	Viagénie
Intended status: Standards Track	J. Rosenberg
Expires: September 5, 2009	Cisco Systems
	March 04, 2009

TOC

Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations draft-ietf-behave-turn-tcp-02.txt

Abstract

This specification defines an extension of Traversal Using Relays around NAT (TURN), a relay protocol for NAT traversal, to allows a TURN client to request TCP allocations, and defines new requests and indications for the TURN server to open and accept TCP connections with the client's peers. TURN and this extension both purposefully restrict

the ways in which the relayed address can be used. In particular, it prevents users from running general purpose servers from ports obtained from the STUN server.

Table of Contents

---

1. Introduction

TOC

Traversal Using Relays around NAT (TURN) [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” November 2008.) is an extension to the Session Traversal Utilities for NAT [RFC5389] (Rosenberg, J., Mahy, R., Matthews, P., and

D. Wing, “Session Traversal Utilities for NAT (STUN),” October 2008.) protocol. TURN allows for clients to communicate with a TURN server, and ask it to allocate ports on one of its host interfaces, and then relay traffic between that port and the client itself. TURN, when used in concert with STUN and Interactive Connectivity Establishment (ICE) [I-D.ietf-mmusic-ice] (Rosenberg, J., “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols,” October 2007.) form a solution for NAT traversal for UDP-based media sessions.

However, TURN itself does not provide a way for a client to allocate a TCP-based port on a TURN server. Such an allocation is needed for cases where a TCP-based session is desired with a peer, and NATs prevent a direct TCP connection. Examples include application sharing between desktop softphones, or transmission of pictures during a voice communications session.

This document defines an extension to TURN which allows a client to obtain a TCP allocation. It also allows the client to initiate connections from that allocation to peers, and accept connection requests from peers made towards that allocation.

2. Conventions

TOC

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 [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

3. Overview of Operation

TOC

```
                                                        +--------+
                                                        |        |
                                                        | Peer1  |
                                                     /  |        |
                                                    /   |        |
                                                   /    +--------+
                                                  /
                                                 /
                                                / Peer Data 1
                                               /
        +--------+  Control      +--------+  /
        |        | -------------- |        | /
        | Client | Client Data 1  | TURN   |
        |        | -------------- | Server | \
        |        | -------------- |        |  \
        +--------+ Client Data 2  +--------+   \
                                                \
                                                 \
                                                  \     +--------+
                                                   \    |        |
                                        Peer Data 2 \   | Peer2  |
                                                     \  |        |
                                                        |        |
                                                        +--------+
```

Figure 1: TURN TCP Model

The overall model for TURN-TCP is shown in Figure 1 (TURN TCP Model). The client will have two different types of connections to its TURN server. For each allocated port, it will have a single control connection. Control connections are used to obtain allocations and open up new connections. Furthermore, for each connection to a peer, the client will have a single connection to its TURN server. These connections are called data connections. Consequently, there is a data connection from the client to its TURN server (the client data connection) and one from the TURN server to a peer (the peer data connection). Actual application data is sent on these connections. Indeed, after an initial TURN message which binds the client data connection to a peer data connection, only application data can be sent - no TURN messaging. This is in contrast to the control connection, which only allows TURN messages and not application data.
To obtain a TCP-based allocation, a client must have a TCP or TLS connection to its TURN server. Using that connection, it sends an Allocate request. That request contains a REQUESTED-TRANSPORT attribute, which indicates a TCP-based allocation is desired. A server

which supports this extension will allocate a TCP port and begin listening for connection requests on that port. It then returns the allocated port to the client in the resposne to the Allocate request. The connection on which the Allocate request was sent is the control connection.

If a client wishes to establish a TCP connection to a peer from that allocated address, it issues a Connect request to the TURN server over the control connection. That request contains a XOR-PEER-ADDRESS attribute identifying the peer IP address and port to which a connection is to be made. The TURN server attempts to open the TCP connection, and assuming it succeeds, then responds to the Connect request with a success response. The server also creates a connection identifier associated with this connection, and passes that connection identifier back to the client in the success response.

In order to actually send data on the new connection or otherwise utilize it in any way, the client establishes a new TCP connection to its TURN server. Once established, it issues a ConnectionBind request to the server. That request echoes back the connection identifier to the TURN server. The TURN server uses it to correlate the two connections. As a consequence, the TCP connection to the peer is associated with a TCP connection to the client 1-to-1. The two connections are now data connections. At this point, if the server receives data from the peer, it forwards that data towards the client, without any kind of encapsulation. Any data received by the TURN server from the client over the client data connection are forwarded to the peer, again without encapsulation or framing of any kind. Once a connection has been bound using the ConnectionBind request, TURN processing is no longer permitted on the connection.

In a similar way, if a client wishes to receive TCP connections to the allocated address, it issues a Listen request to the TURN server over the control connection. The server then starts listening and accepting incoming connections.

Once a new connection is accepted, the server checks if there is a permission in place for that peer. If there is none, the connection is closed. Permissions are created with the CreatePermission request sent over the control connection, just as for UDP TURN. If there is a permission in place, the TURN server sends, to the client, a ConnectionAttempt Indication over the control connection. That indication contains a connection identifier. Once again, the client initiates a separate TCP connection to its TURN server, and over that connection, issues a ConnectionBind request. Once received, the TURN server will begin relaying data back and forth. The server closes the peer data connection if no ConnectionBind request is received after a timeout.

If the client closes a client data connection, the corresponding peer data connection is closed. If the peer closes a peer data connection, the corresponding client data connection is closed. In this way, the status of the connection is directly known to the client.

The TURN server will relay the data between the client and peer data connections, utilizing an internal buffer. However, back pressure is used in order to achieve end-to-end flow control. If the buffer from client to peer fills up, the TURN server ceases to read off the client data connection, which causes TCP backpressure through the OS towards the client.

4. Client Processing

TOC

4.1. Creating an Allocation

TOC

To create a TCP allocation, a client MUST initiate a new TCP or TLS connection to its TURN server, identical to the TCP or TLS procedures defined in [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” November 2008.). TCP allocations cannot be obtained using a UDP association between client and server.
Once set up, a client MUST send a TURN Allocate request. That request MUST contain a REQUESTED-TRANSPORT attribute whose value is 6, corresponding to TCP.
The request MUST NOT include a DONT-FRAGMENT, RESERVATION-TOKEN or EVEN-PORT attribute. The corresponding features are specific to UDP based capabilities and are not utilized by TURN-TCP. However, a LIFETIME attribute MAY be included, with semantics identical to the UDP case.
The procedures for authentication of the Allocate request and processing of success and failure responses are identical to those for TCP.
Once a success response is received, the TCP connection to the TURN server is called the control connection for that allocation.

4.2. Refreshing an Allocation

TOC

The procedures for refreshing an allocation are identical to those for UDP. Note that the Refresh MUST be sent on the control connection.

TOC

4.3. Initiating a Connection

To initiate a TCP connection to a peer, a client MUST send a Connect request over the control channel for the desired allocation. This request MUST NOT be sent until an Allocate request has been completed successfully over that connection. The Connect request MUST include a XOR-PEER-ADDRESS attribute containing the IP address and port of the peer to which a connection is desired.
If the connection is successfully established, the client will receive a success response. That response will contain a CONNECTION-ID attribute. The client MUST initiate a new TCP connection to the server, utilizing the same destination IP address and port on which the control connection was established to. This connection MUST be made using a different local IP address and port. Once established, the client MUST send a ConnectionBind request. That request MUST include the CONNECTION-ID attribute, mirrored from the Connect Success response. When a response to the ConnectionBind request is recevied, if it is a success, the TCP connection on which it was sent is called the client data connection corresponding to the peer.
If the result of the Connect request was a Error Response, and the response code was XXX, it means that the TURN server was unable to connect to the peer. The client MAY retry, but MUST wait at least 10 seconds.
Once a Connect success response has been received, further Connect or Listen requests for the same allocation MUST NOT be sent.

4.4. Receiving a Connection

TOC

To indicate its willingness to receive incoming TCP connections from peers, a client MUST send a Listen request over the control channel for the desired allocation. This request MUST NOT be sent until an Allocate request has been completed successfully over that connection.
If a success response is received, the client will start receiving a ConnectionAttempt indication each time a peer attemps a new connection to the allocated address. This indication will contain a CONNECTION-ID and a XOR-PEER-ADDRESS attributes. If the client wishes to accept this connection, it MUST initiate a new TCP connection to the server, utilizing the same destination IP address and port on which the control connection was established to. This connection MUST be made using a different local IP address and port. Once established, the client MUST send a ConnectionBind request. That request MUST include the CONNECTION-ID attribute, mirrored from the ConnectionAttempt indication. When a response to the ConnectionBind request is received, if it is a success, the TCP connection on which it was sent is called the client data connection corresponding to the peer.

Once a Listen success response has been received, further Connect or Listen requests for the same allocation MUST NOT be sent.

4.5. Sending and Receiving Data

TOC

Once a client data connection is established, data sent on it by the client will be relayed as-is to the peer by the server. Similarly, data sent by the peer to the server will be relayed as-is to the client over the data connection. Data on a data connection MUST NOT be interpreted as STUN messages.

4.6. Data Connection Maintenance

TOC

The client MUST refresh the allocation corresponding to a data connection, using the Refresh request as defined in [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” November 2008.), for as long as it wants to keep the data connection alive.
When the client wishes to terminate its relayed connection to the peer, it MUST close the data connection to the server. If the data connection was created as a result of a Connect request, the allocation cannot be used for any other purposes and the client SHOULD explicitly delete it by sending a Refresh request with a LIFETIME attribute of value 0, as indicated in [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” November 2008.). If the data connection was created as a result of a Listen request, the allocation is still valid and further ConnectionAttempt indications may be received.

5. TURN Server Behavior

TOC

5.1. Receiving a TCP Allocate Request

TOC

The process is similar to that defined in [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, “Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for

NAT (STUN)," November 2008.), Section 6.2, with the following exceptions:

1. If the REQUESTED-TRANSPORT attribute is included and specifies a protocol other than UDP or TCP, the server MUST reject the request with a 442 (Unsupported Transport Protocol) error. (If the value is UDP, the server MUST continue with the procedures of [I-D.ietf-behave-turn] (Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," November 2008.) instead of this document.)

2. If the client connection transport is not TCP or TLS, the server MUST reject the request with a 400 (Bad Request) error.

3. If the request contains the DONT-FRAGMENT, EVEN-PORT, or RESERVATION-TOKEN attribute, the server MUST reject the request with a 400 (Bad Request) error.

4. A TCP relayed transport address MUST be allocated instead of a UDP one.

5. The RESERVATION-TOKEN attribute MUST NOT be present in the success response.

Until a Listen request for this allocation is successfully processed, the server MUST deny any connection attempts received on the relayed transport address.

5.2. Receiving a Connect Request

When the server receives a Connect request, it processes as follows.
If the request is received on a control connection for which no allocation exists, the server MUST return a 437 (Allocation Mismatch) error.
If the server has already successfully processed a Connect or Listen request for this allocation, it MUST return a 446 (Connection Already Exists) error.
If the request does not contain a XOR-PEER-ADDRESS attribute, or if such attribute is invalid, the server MUST return a 400 (Bad Request) error.
Otherwise, the server MUST initiate an outgoing TCP connection. The local endpoint is the relayed transport address associated with the allocation. The remote endpoint is the one indicated by the XOR-PEER-ADDRESS attribute. If the connection attempt fails or times out, the server MUST return a XXX (Connection Timeout or Failure) error.

If the connection is successful, it is now called a peer data connection. The server MUST buffer any data received from the peer. Data MUST NOT be lost. It is up to the server to adjust its advertised TCP receive window should the buffer size become a limiting factor.
The server MUST include the CONNECTION-ID attribute in the Connect success response. The attribute's value MUST uniquely identify the peer data connection.

5.3. Receiving a Listen Request

TOC

When a server receives a Listen request, it processes as follows.
If the request is received on a control connection for which no allocation exists, the server MUST return a 437 (Allocation Mismatch) error.
If the server has already successfully processed a Connect or Listen request for this allocation, it MUST return a 446 (Connection Already Exists) error.
Otherwise, the server MUST start accepting incoming TCP connections on the relayed transport address. Refer to Section 5.4 (Receiving a TCP Connection on an Allocated Port) for details.
The server replies with a Listen success or failure response depending on the success of this.

5.4. Receiving a TCP Connection on an Allocated Port

TOC

When a server receives an incoming TCP connection on a relayed transport, it processes as follows.
If no Listen request has been successfully processed for this allocation, the server MUST reject the connection. The means (silently ignoring it, replying with a TCP reset, etc.) are not specified.
Otherwise, the server MUST accept the connection. If it is not successful, nothing is sent to the client over the control connection.
If the connection is successfully accepted, it is now called a peer data connection. The server MUST buffer any data received from the peer. Data MUST NOT be lost. It is up to the server to adjust its advertised TCP receive window should the buffer size become a limiting factor.
The server then sends a ConnectionAttempt indication to the client over the control connection. The indication MUST include a XOR-PEER-ADDRESS attribute containing the peer's address, as well as a CONNECTION-ID attribute uniquely identifying the peer data connection.
If no ConnectionBind request associated with this peer data connection is received after 30 seconds, the peer data connection MUST be closed.

5.5. Receiving a ConnectionBind Request

TOC

When a server receives a ConnectionBind request, it processes as follows.
If the client connection transport is not TCP or TLS, the server MUST return a 400 (Bad Request) error.
If the request does not contain the CONNECTION-ID attribute, the server MUST return a 400 (Bad Request) error.
Otherwise, the client connection is now called a client data connection. Data received on it MUST be sent as-is to the associated peer data connection.
Data received on the associated peer data connection MUST be sent as-is on this client data connection. This includes data that was received after the associated Connect or Listen request was successfully processed and before this ConnectionBind request was received.
Data received on a client or peer data connection MUST NOT be interpreted as a STUN message.

5.6. Data Connection Maintenance

TOC

If the allocation associated with a data connection expires, the data connection MUST be closed.
When a client (resp. peer) data connection is closed or times out, the server MUST close the corresponding peer (resp. client) data connection.

6. IANA Considerations

TOC

This specification defines several new STUN methods, STUN attributes, and STUN response codes. This section directs IANA to add these new protocol elements to the IANA registry of STUN protocol elements.

6.1. New STUN Methods

TOC

This section lists the codepoints for the new STUN methods defined in this specification. See elsewhere in this document for the semantics of these new methods.

0x007	:	Connect	(only request/response semantics defined)
0x008	:	Listen	(only request/response semantics defined)
0x009	:	ConnectionBind	(only request/response semantics defined)
0x010	:	ConnectionAttempt	(only indication semantics defined)

6.2. New STUN Attributes

TOC

This STUN extension defines the following new attributes:

0xTBD	:	CONNECTION-ID

6.2.1. CONNECTION-ID

TOC

The CONNECTION-ID attributes uniquely identifies a peer data connection. It is a 32-bit unsigned integral value.

6.2.2. New STUN response codes

TOC

446	Connection Already Exists
XXX	Connection Timeout or Failure

6.3. Security Considerations

TOC

TBD

6.4. IANA Considerations

TOC

TBD

TOC

6.5. IAB Considerations

TBD.

6.6. Acknowledgements

TOC

Thanks to Rohan Mahy and Philip Matthews for their initial work on getting this document started.
The authors would also like to thank Marc Petit-Huguenin for his comments and suggestions.

7. References

TOC

7.1. Normative References

TOC

[RFC5389]	Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)," RFC 5389, October 2008 (TXT).
[I-D.ietf-behave-turn]	Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)," draft-ietf-behave-turn-12 (work in progress), November 2008 (TXT).
[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).

7.2. Informative References

TOC

[I-D.ietf-mmusic-ice]	Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols," draft-ietf-mmusic-ice-19 (work in progress), October 2007 (TXT).

Authors' Addresses

TOC

	Simon Perreault (editor)

	Viagénie
	2600 boul. Laurier, suite 625
	Québec, QC G1V 4W1
	Canada
Phone:	+1 418 656 9254
Email:	simon.perreault@viagenie.ca
URI:	http://www.viagenie.ca
	Jonathan Rosenberg
	Cisco Systems
	600 Lanidex Plaza
	Parsippany, NJ 07054
	US
Phone:	+1 973 952-5000
Email:	jdrosen@cisco.com
URI:	http://www.jdrosen.net