Behave
Internet-Draft
Intended status: Standards Track
Expires: May 7, 2009

J. Rosenberg
Cisco Systems
R. Mahy
Unaffiliated

November 3, 2008

Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations

draft-ietf-behave-turn-tcp-01.txt

Status of this Memo

Copyright Notice

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

Traversal Using Relays around NAT (TURN) [2] is an extension to the Session Traversal Utilities for NAT [1] 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) [4] 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. Overview of Operation

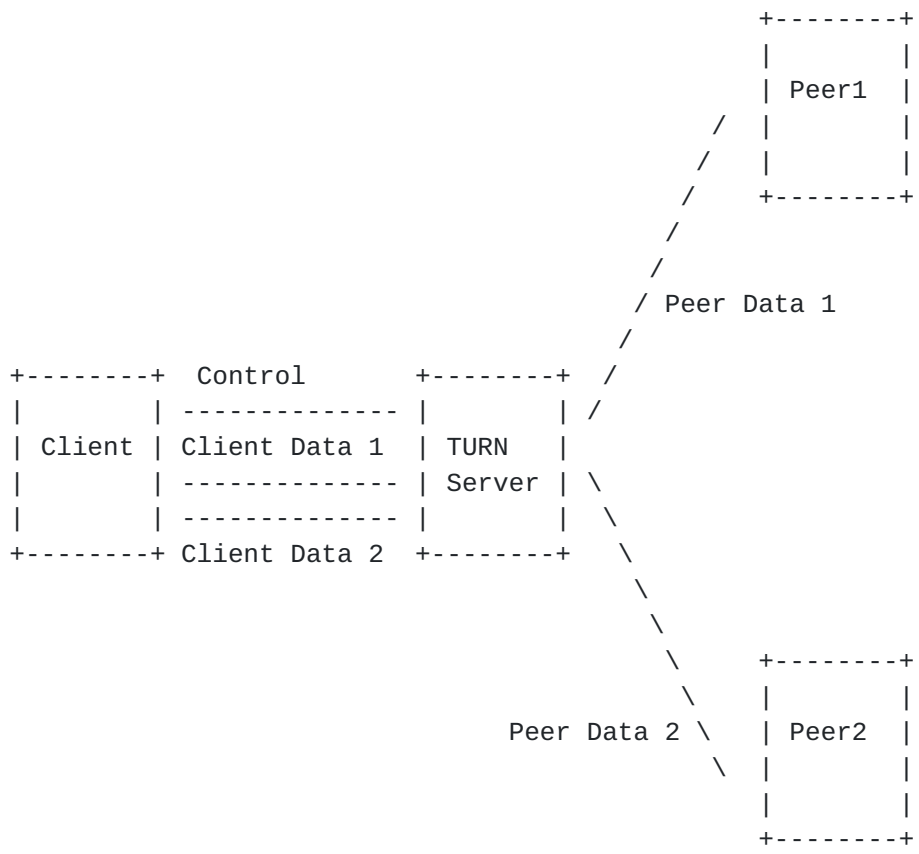

Figure 1: TURN TCP Model

The overall model for TURN-TCP is shown in Figure 1. 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, when a peer attempts to open a TCP towards the allocated port, if there is no permission in place for that peer, the connection attempt is discarded. 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 accept the connection from the peer and begin relaying data back and forth.

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.

3. Client Processing

3.1. Creating an Allocation

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 [2]. 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 TCP 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.

3.2. Refreshing an Allocation

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

3.3. 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.

3.4. Receiving a Connection

3.5. Sending and Receiving Data

3.6. Data Connection Maintenance

4. TURN Server Behavior

4.1. Receiving a TCP Allocate Request

4.2. Receiving a Connect Request

4.3. Receiving a TCP Connection on an Allocated Port

4.4. Receiving a ConnectionBind Request

4.5. Connection Maintenance

5. IANA Considerations

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.

5.1. New STUN Methods

0x007 : Connect
0x008 : ConnectionBind
0x009 : ConnectionAttempt

5.2. New STUN Attributes

0xTBD : ConnectionID

5.3. New STUN response codes

446 Connection Already Exists
XXX Connection Timeout or Failure

6. Security Considerations

TBD

7. IANA Considerations

TBD

8. IAB Considerations

TBD.

9. Acknowledgements

The authors would like to thank Marc Petit-Huguenin for his comments and suggestions.

10. References

10.1. Normative References

[1] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[2] 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-11 (work in progress), October 2008.

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

10.2. Informative References

[4] 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.

Authors' Addresses

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

Rohan Mahy
Unaffiliated

Email: rohan@ekabal.com

Full Copyright Statement

Intellectual Property

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).