Behavior Engineering for Hindrance Avoidance   Xin. Chen
Internet-Draft   Huawei
Intended status: Standards Track   May 13, 2013
Expires: November 14, 2013

Traversal Using Relays around NAT (TURN) Extensions for Websocket Allocations
draft-chenxin-behave-turn-websocket-00

Abstract

This document defines an extension to TURN that allows it to run over a Websocket [RFC6455] channel. This will allow a client in a restrictive network to exchange and relay media or data over the websocket.

Status of This Memo

Copyright Notice

1. Introduction

Traversal Using Relays around NAT (TURN)[RFC5766], which assigns a transport address allocation for clients and relays data between the address and the clients, is an extension to the Session Traversal Utilities for NAT [RFC5389] protocol. TURN is used for NAT traversal in some complicated types of NAT network by UDP-based media sessions [RFC5766] or TCP-based media sessions [RFC6062]. It is also used with Interactive Connectivity Establishment (ICE) [RFC5245].

In some particularly restrictive networks, e.g. a web proxy or firewall that only allows HTTP traffic to pass through, TURN UDP-based media sessions and TURN TCP-based media sessions do not work. These types of network often appear in the prison, the hotel, the airport and other places that need limit the access of network. RTCWeb provides direct interactive rich communication between two peers’ web-browser, which has the similar requirement to allow two peers to use fallback communication in the http-only network [I-D.ietf-rtcweb-use-cases-and-requirements](F37).

Another use case of Turn over websocket is that the turn server could be a generic relay for some protocols over websocket, such as BFCP[RFC4975] and MSRP[I-D.ietf-bfcpbis-rfc4582bis]. There are drafts to support BFCP [I-D.pascual-dispatch-bfcp-websocket]and MSRP[I-D.pd-msrp-websocket] transported in the websocket, however in these drafts a special server is needed as the intermediary to receive the protocol data from the client. In the Peer to Peer scene, BFCP and MSRP message is required be exchange by two peers, and additional server is a little complicated to deploy in such scene. The turn over websocket could be used as generic relay server, which could relay these protocol data between two peers.

This document defines an extension to TURN that allows it to run over a Websocket [RFC6455] channel. This will allow a client in a restrictive network to exchange media over a websocket. This is useful to solve the http fallback problem and is easy to be implemented in the existing rtcweb framework. The connection between the server and the peer is still UDP as [RFC5766] or TCP , TLS over TCP as [RFC6062].

TURN client to TURN server	TURN server to peer
Websocket over TCP	UDP TCP TLS over TCP

2. Terminology

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].

3. Deployment Topology

As mentioned in [I-D.miniero-rtcweb-http-fallback], within the context of real-time multimedia communications and considering a scenario that involves two peers,an HTTP fallback mechanism may fall in basically three different network topologies:

3.1. A topology whereas only one of the involved peers needs HTTP fallback for communication

```
                            +-----------+
                            | TURN &    |
             +--------------+ websocket +----------------+
             |  websocket   | server    |                 \ UDP/TCP
             |              +-----------+                  \
             |                                              \
             |                                               |
         +---+---+                                       +---+---+
         | Alice |                                       |  Bob  |
         +-------+                                       +---+---+
```

Figure 1

In Figure 1, only one involved peer(Alice) is in the restrained network, so Alice need use websocket connection to traverse the firewall and proxy. The situation for Bob is better, he could connect to TURN server by UDP or TCP using existing mechanism.

When Alice wants to communicate with Bob, she need request a UDP or TCP allocation in websocket server for Bob, which is transferred in the websocket channel. The websocket server will receive the request and handle it like a TURN server. The procedure of TURN message is the same as TURN UDP and TURN TCP, and the websocket server will also allocate a UDP or TCP relay address for Bob. The application data from Alice and Bob will be packaged and relayed to each other by the websocket server.

3.2. A topology whereas both the involved peers need HTTP fallback for communication, using two different intermediaries

```
                      +-----------+         +-----------+
                      | TURN &    |         | TURN &    |
         +------------+ websocket +---------+ websocket +--------- +
         | websocket  | server    | UDP/TCP | server    |websocket |
         |            +-----------+         +-----------+          |
         |                                                         |
         |                                                         |
     +---+---+                                                 +---+---+
     | Alice |                                                 |  Bob  |
     +-------+                                                 +---+---+
```

Figure 2

In Figure 2, both Alice and Bob are i restrictive networks, so both need a fallback mechanism. One complex scenario is that Alice and Bob each have their own websocket server. In this scenario, Alice and Bob need to request the Turn allocation in their own websocket server using a websocket connection. The procedure of TURN message is the same as TURN UDP and TURN TCP. The websocket server should relay the application data to each other by UDP ,TCP or other existing ways. It is better that Alice and Bob allocate the same type of transport address, so their own websocket server could connect to each other by this address directly.

3.3. A topology whereas both the involved peers need HTTP fallback for RTP, using the same intermediary

```
                                +-----------+
                                | TURN &    |
         +----------------------+ websocket +--------------------+
         |  websocket           | server    |         websocket  |
         |                      +-----------+                    |
         |                                                       |
         |                                                       |
     +---+---+                                               +---+---+
     | Alice |                                               |  Bob  |
     +-------+                                               +---+---+
```

Figure 3

In Figure 3, Alice and Bob are both also in the restrictive network as Figure 2 with a slight difference that Alice and Bob share the same websocket server. In this scenario, Alice and Bob need exchange TURN messages as TURN UDP and TCP using websocket connection. The websocket will assign the allocation for both sides and decide how to bind the two allocations together and bridge the TURN application data, which is entirely implementation specific.

4. Allocation

Because the UDP and TCP allocation for the peer can satisfy the different scenarios, we just need to extend the connection between the client and the server - websocket connection. Considering websocket connection is based on TCP transport connection, we should add a flag in server's allocation to distinguish these two connections: 'app-connection'.

In the application of TURN UDP and TURN TCP, the 5-tuple (client's IP address, client's port, server IP address, server port, transport protocol) is used for stating the connection between the client and the server. However the 5-tuple is hard to state the websocket connection which is an application-level connection. The app-connection is used for stating the application level connection, whose value is 'ws' or'wss' when the client uses websocket or secure websocket to connect TURN server.

5. Operation

The operation of the client, server and peer is the same as TURN UDP and TURN TCP with a difference of the new connection channel - websocket.

6. Multiplexing

In restrictive networks, the port resource is limited, so multiplexing the TURN connections on a single port is necessary. A method for websocket multiplexing is proposed in [I-D.ietf-hybi-websocket-multiplexing], which describes a multiplexing extension for the WebSocket Protocol. With this extension, one TCP connection can provide multiple virtual WebSocket connections by encapsulating frames tagged with a channel ID.

This method could be used in conveying TURN data. One control websocket channel is needed to convey the TURN control message. When allocation request is successful, a new websocket connection should be established for transferring TURN application data. The channel ID for the new websocket connection should be saved in the allocation in server, so that all data in this channel will be bound to this specific allocation.

7. IANA Considerations

TBS

8. Security Considerations

TBS

9. Acknowledgements

Paul Kyzivat helped with the formatting of this draft.

10. References

10.1. Normative References

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6062] Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.

10.2. Informative References

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

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[I-D.ietf-bfcpbis-rfc4582bis]
Camarillo, G., Drage, K., Kristensen, T., Ott, J., and C. Eckel, "The Binary Floor Control Protocol (BFCP)", draft-ietf-bfcpbis-rfc4582bis-09 (work in progress), April 2013.

[I-D.pascual-dispatch-bfcp-websocket]
Pascual, V., Welbourn, R., and S. Cazeaux, "The WebSocket Protocol as a Transport for the Binary Floor Control Protocol (BFCP)", draft-pascual-dispatch-bfcp-websocket-00 (work in progress), February 2013.

[I-D.pd-msrp-websocket]

Dunkley, P. and G. Llewellyn, "The WebSocket Protocol as a Transport for the Message Session Relay Protocol (MSRP)", draft-pd-msrp-websocket-02 (work in progress), November 2012.

[I-D.ietf-rtcweb-use-cases-and-requirements]
Holmberg, C., Hakansson, S., and G. Eriksson, "Web Real-Time Communication Use-cases and Requirements", draft-ietf-rtcweb-use-cases-and-requirements-10 (work in progress), December 2012.

[I-D.miniero-rtcweb-http-fallback]
Miniero, L., "HTTP Fallback for RTP Media Streams", draft-miniero-rtcweb-http-fallback-00 (work in progress), August 2012.

[I-D.ietf-hybi-websocket-multiplexing]
Tamplin, J. and T. Yoshino, "A Multiplexing Extension for WebSockets", draft-ietf-hybi-websocket-multiplexing-09 (work in progress), November 2012.

Author’s Address

Xin Chen
Huawei

Email: hangzhou.chenxin@huawei.com