Preventing Fragmentation in Client Initiated Connection

Marc Petit-Huguenin

draft-petithuguenin-sip-outbound-fragmentation-02
Hybrid UDP-TCP Transport

UAC/Proxy

TU Transaction Layers

Transport Layer

DNS

UAS/Proxy

Query NAPTR

Answer SIP+D2U

Request UDP

Request TCP
Hybrid UDP-TCP Transport

• If a server listens on UDP, it must also listen on TCP

  Section 18.2.1: “For any port and interface that a server listens on for UDP, [the server] MUST listen on the same port and interface for TCP.”

• A client must switch from UDP to TCP if the message is larger than the MTU

  Section 18.1.1: “If a request is within 200 bytes of the path MTU [...] the request MUST be sent using a RFC 2914 congestion controlled transport protocol, such as TCP.”

• Not defined, but a Hybrid DTLS-TLS Transport should work the same way by using session resumption.
Response Fragmentation

UA/Proxy

UA/Proxy

Transport Layer

TU Transaction Layers

REGISTER UDP

Request

Response

Fragment

Fragment
Response Fragmentation

• This is a different problem that will not be discussed today.

• See the following drafts:
  draft-gurbani-sip-large-udp-response
  draft-petithuguenin-sip-fragmentation-responses
Problem with NAT

- UA
- DNS
- NAT
- UA/Proxy
  - Transport Layer
  - TU Transaction Layers

Transactions:
- Query NAPTR
- Answer SIP+D2U
- REGISTER UDP
- 200 R UDP
- INVITE TCP
- INVITE UDP
Problem with NAT

- The UA inside the NAT will listen on an UDP port and a TCP port.
- The registration will create an UDP binding in the NAT.
- The TCP connection in the other direction will be blocked by the NAT and will never reach the UA inside the NAT.
Why UDP: Performances

• Only few SIP messages needs TCP:
  – INVITE/ACK/UPDATE/200 with SDP and/or History-Info.
  – NOTIFY with full notification
  – MESSAGE

• Other SIP messages can use UDP:
  – INVITE/ACK/UPDATE/200 without SDP
  – BYE/CANCEL/SUBSCRIBE/PRACK
  – NOTIFY with partial notification
Why UDP: Direct Connection for Subsequent Requests
Why UDP: Direct Connection for Subsequent Requests

● If the proxies does not Record-Route and UDP is used, the subsequent requests can be sent directly from UA to UA in most of the cases.

● If TCP is used, at least one relay is needed on the public Internet.
Solution 1: Extend Outbound

UA

NAT

REGISTER (UDP)

200 R (UDP)

STUN ForceTCP (UDP)

STUN GetToken (TCP)

INVITE (TCP)

INVITE (UDP)

Transport Layer

UA/Proxy

TU Transaction Layers
Solution 1: Extend Outbound

- Solution described in the draft.
- Use the existing UDP flow to send a STUN message to the UA.
- The UA opens a TCP connection to the same port than used by the UDP flow.
- The server uses the new TCP connection to send the large SIP message.
Solution 2: Extends STUN Relay

- **UA**: User Agent
- **NAT**: Network Address Translator
- **STUN Relay**: STUN Relay Server
- **UA/Proxy**: User Agent/Proxy
- **Transport Layer**: Transport Layer
- **TU Transaction Layers**: Transaction Layer

**Message Flows**:
- **Allocate+** from UA to STUN Relay
- **REGISTER UDP** from STUN Relay to UA/Proxy
- **Set Active Destination** from UA/Proxy to STUN Relay
- **ForceTCP** from STUN Relay to UA/Proxy
- **SetToken** from UA/Proxy to STUN Relay
- **INVITE TCP** from UA/Proxy to UA
- **INVITE UDP** from UA to UA/Proxy
- **INVITE TCP** from UA/Proxy to UA
Solution 2: Extends STUN Relay

- The UA sends an Allocate Request over UDP to a STUN Relay, with an extension signaling that the relay should listen for UDP and TCP on the same port.

- When the STUN Relay receives a connection on the TCP port, it sends a ForceTCP message to the UA over UDP.

- The UA opens a TCP connection to the STUN Relay, that can be used to relay the data.
Questions

- Do we agree on the problem?
- Is it the right WG for this work?