draft-denis-udp-transport-00

IETF#73 Minneapolis
Transport area open meeting

Rémi Denis-Courmont
Nokia Devices R&D
Maemo Sofware

Motivation

- Plain ICE supports only UDP
- It also has return routability
- Need TCP semantics for certain medias:
  - Congestion control
  - Packet loss recovery
- ICE-TCP adds TCP support

TCP “simultaneous open” issues

- It is a corner case of TCP, poorly supported:
  - e.g. not possible on Windows (until Vista)
  - Clumsy usage of the socket API
- NATs often assume
  SYN outbound then SYN/ACK inbound
- All firewalls reject inbound SYN,
  even if “solicited”
- UDP hole punching better with NAT2NAT

Proposal

- Rather than re-invent transport protocol, use almost normal TCP on top of UDP.
- UDP provides the port numbers and checksum
- Extra header provides:
  - sequence numbers
  - flags

Advantages

- Same overhead as normal TCP:
  20 bytes transport header
- Support multiplexing with STUN so that it can run on top of ICE out-of-the box
- Can be implemented in the TCP/IP stack
- Can also be implemented by the application on top of normal UDP sockets
- Much better success rate than TCP-SO,
  much less likely to need media relays (TURN)

Disadvantages

- Ugly
- Will not “benefit” from MSS clamping; RFC4821 (app-layer MTU) is required
- Not backward compatible in any way; both sides need to implement it
- TCP urgent data not supported (alternative: higher overhead, different MSS)

Other transports?

- SCTP proposed too - Needs checking!!!
- UDP-Lite would make no sense here
- DCCP not supported, but there is
  `draft-phelan-dccp-natencap`
- UDP Length needed:
  - UDP socket API compatibility
  - At least Linux **does** check that field!
- General case: tricky due to ports and checksum

Way forward?

- Bring it to TSVWG? Keep it here? Drop it?

- Security considerations
- API considerations??

- Further work would be needed in MMUSIC wg (ICE-TCP candidate type)

Questions?