Fragmentation issues in IPv4/IPv6 translation

marcelo bagnulo

behave WG – IETF76

Incoming IPv4 packet - Full pkt, DF=1

- Resulting IPv6 packet does NOT include the fragment header (PMUTD should work e2e)
- Issue: If the packet is too big for the outgoing MTU in IPv6, should the translator generate and ICMP error Packet too big?
- Proposed solution: both stateless and stateful translator MUST behave as a router on this
  - I.e. Send packet too big errors

Incoming IPv4 packet - Full pkt, DF=0

- As currently defined: Resulting IPv6 packets MUST fit in 1280 bytes, so add the fragment header if needed. The fields of the fragment header are copied from the IPv4 fields.
- Issue: Would this result in significant amount of fragmentation?
- Should the translator perform PMUTD for IPv6 destinations?
- This affects both stateless and stateful,
- should we use the same approach to both?
- My reading so far is that we should send 1280 byte long packets (i.e. as currently defined in stateless)

Incoming IPv4 packet – Fragment (I)

- Fragment: Add fragment header, copy fields from IPv4 header, make sure that the packet does not exceeds 1280 bytes.
- Issue: how to determine the session for fragments? We need the TCP/UDP header (which is not available in fragments other than the first one)
  - Opt 1: reassembly
  - Opt 2: keep state.
- Specific to the stateful case

Incoming IPv4 packet – Fragment (II)

Opt 1: reassemble

- Fragments arrive to the translator
- Translator reassembles the packets
- Processes the packet once the full packet is reassembled.
- Pros: simple
- Cons: requires memory for storing fragments, which needs to be limited to avoid DoS attacks against the translator

Incoming IPv4 packet – Fragment (III)

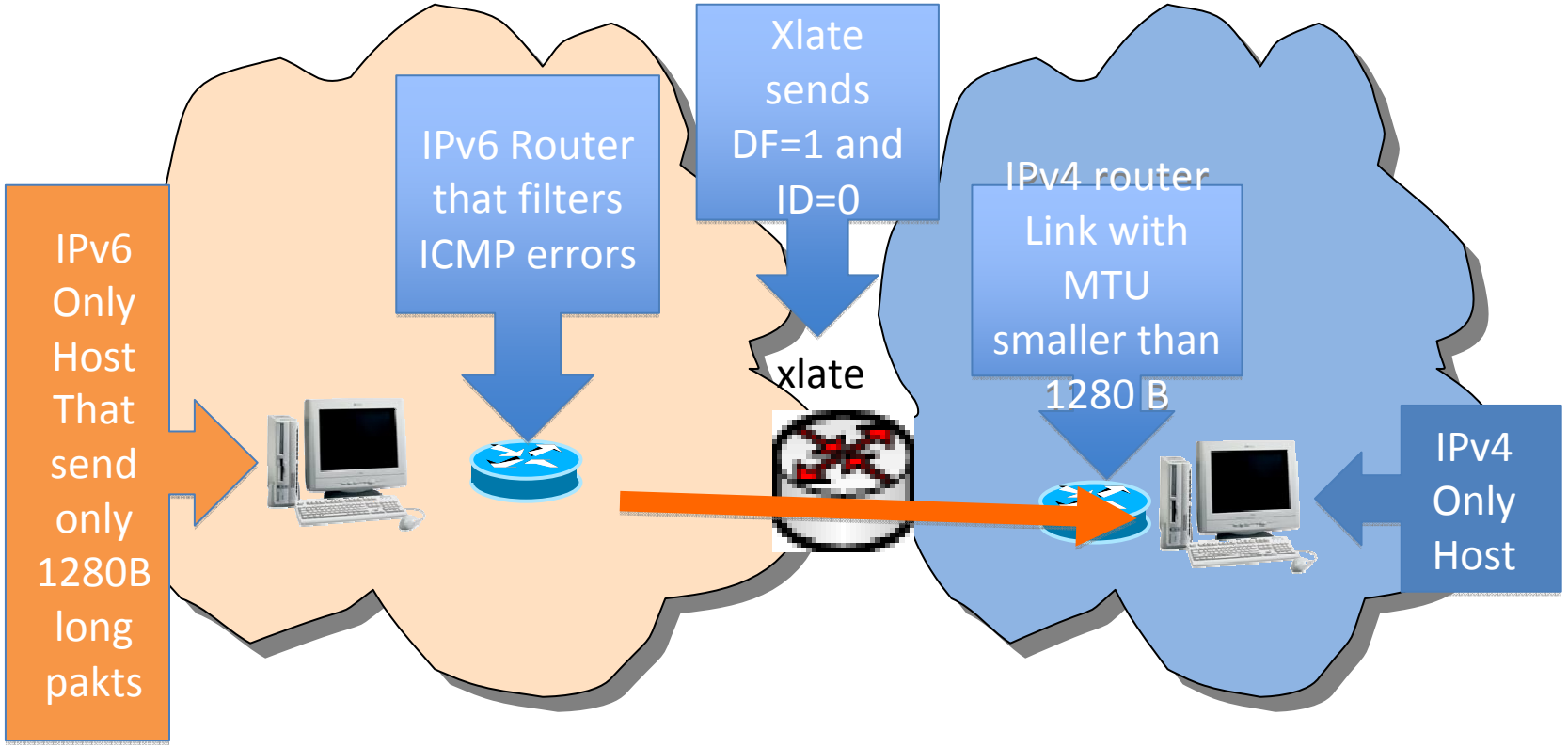
Opt 2: Keep state

- Store state that allows to translate packets containing fragments other than the first one. The state needed is the ID value.
- The state is learned from the packet containing the first segment.
  - If packets that contain fragments that are not the first arrive and the ID does not match to any of the stored ID values for the src and dst addreses, then store the packets.
  - If packets that contain fragments that are not the first arrive and the ID does match to any of the stored ID values for the src and dst addresses, then use the corresponding session to translate the packet.
  - If packet that contain the first fragment arrive, then store the ID value in the session state and forward the packet.
- Pro: requires less memory to store packets
- Con: more complex and needs to deal with attacks like the ones defined in RFC3128 and RFC1858
- So, what shall we do?

Incoming IPv6 packet – No frag Hdr

- Stateless draft defines: IPv4 pkt Id set to all zeros and DF set to 1.
- Issue #1: Some moddle boxes fragment even if DF is set to 1
  - Do we care about boxes that don't follow the spec?
- Issue #2: blackhole
  - Next slide

Incoming IPv6 packet – No frag Hdr



IPv6 host sends IPv6 pakt of 1280 B
Xlate sends IPv4 pkt with DF=1 and ID=0
IPv4 router with MTU<1280 discard IPv4 pkt and sends ICMP pkt too big error
IPv6 router discards ICMP error
Black hole!

Incoming IPv6 packet – No frag Hdr

- Alternative approach
- Set the DF=0 and use ID other than 0
  - At least for packets of 1280B or smaller
- Need to keep a counter for the ID number
  - Simple counter per source address is enough?
- Question:
  - Should the stateless and the stateful have the same behaviour?
    - If no, which behaviour should each one have?
    - If yes, which should be the common behaviour?

Incoming IPv6 packet – Frag Hdr

- Current behaviour: copy fields from the fragment header to the IPv4 header fragmentation fields
- Higher probabilty clash if we decide to use a non zero value for the IPv6 packets wihtout fragment header.
  - We can split the ID space in two
    - With MSb set- for IPv6 pkts with frag header
    - With MSb reset- for IPv6 pkts wihtou frag header
  - Is it worth it?