Framework for IPv4/IPv6 Multicast Translation

draft-venaas-behave-v4v6mc-framework-03.txt

Overview

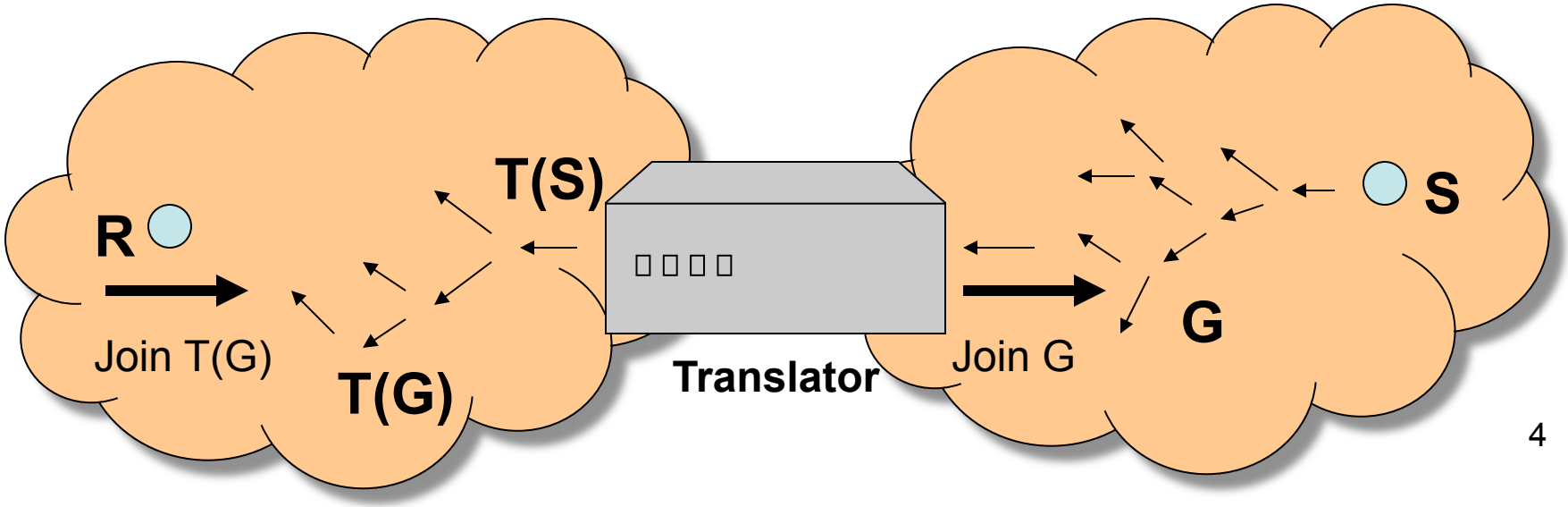
- Behave did a unicast translation framework, RFC 6144
- This draft started out as a multicast version of that
- Scenarios talk about Internet which is not so common for multicast
  - Think of it as a large network or a set of networks
  - The important thing is scale, and whether you have administrative control

Scenarios

1. **An IPv6 network receiving multicast from IPv4 Internet**
2. IPv6 Internet receiving multicast from an IPv4 network
3. **An IPv4 network receiving multicast from IPv6 Internet**
4. IPv4 Internet receiving multicast from an IPv6 network
5. An IPv6 network receiving multicast from an IPv4 network
6. An IPv4 network receiving multicast from an IPv6 network

Multicast and translation

- When doing multicast translation, you have a source S sending to a group G
- A receiver of translated content would join T(G) and receive from T(S), where T is some translation/mapping function
- How is T defined, stateful/stateless? Well-known?
- How does the receiver know T(G)?
- For SSM the receiver would need to know T(S) as well



IPv6 network receiving multicast from IPv4 Internet

- Not so hard since IPv4 address space can be embedded into IPv6
  - E.g. T(224.1.2.3) = ff1e::ffff:224.1.2.3
  - May need to accommodate for SSM and scopes
  - T(232.1.2.3) = ff3e::ffff:232.1.2.3
  - T(239.1.2.3) = ff35::ffff:239.1.2.3
- R wants to receive G and joins T(G)
- Note that this may also allow IPv4 to receive from IPv6, if IPv6 sources send do T(G), it can be translated back to G.

IPv6 network receiving multicast from IPv4 Internet

- How does R know T(G)?
- If receiver is unaware of translation
  - Content provider could provide SDP with both T and T(G)
  - Assumes content provider knows translation may take place and the function T
  - SDP data may be translated by an ALG
- If receiver is aware of translation
  - Application or stack on R knows T
  - If only IPv4 address in SDP, app/stack can apply T and join T(G)
  - How to know/learn T? Well-known prefix?

Well-known multicast prefix(es)?

- Well-known multicast prefixes could be hardcoded in apps/stacks so that they know T() and join translated groups when needed
- If not well-known, there can be different prefixes for different translators
  - May choose which translator is used
- For trees to pass through the translator, it may need to be an IPv6 Rendezvous Point. In that case embedded-RP might be useful
  - Embedded-RP encodes the unicast address of the RP in the group address. Hence well-known multicast prefix is hard, unless also well-known unicast address (anycast)

IPv4 network receiving multicast from IPv6 Internet

- We cannot use a simple embedding and stateless translation
- How does translator get a mapping so it can translate an IPv4-join into IPv6?
  - And translate data from IPv6 to IPv4
- Might use some ALG to translate e.g. SDP as it passes through the translator
- Or new signaling mechanisms between receiver and translator
- An administrator might add static mappings and inform users, or create new SDP, with groups to use

New signaling mechanisms

- We may need new signaling mechanisms for an IPv4 host or network to be able to receive arbitrary IPv6 groups
- A translation aware application or stack could send a query to the translator saying:
  - I want to receive G, which T(G) should I join?
  - The translator can have a pool of IPv4 addresses and allocate them as needed

Summary

- Solutions depend on whether either content provider or receiver is aware of translation taking place, and the mapping function
- For IPv6 receiving from IPv4, stateless translation is simple
  - Well-known prefix?
- For IPv4 receiving from IPv6, stateful translation may be needed
  - How to create the state/mapping?