Port Mapping Between Ucast/Mcast RTP Sessions

`draft-begen-avt-ports-for-ucast-mcast-rtp-01`

IETF 76 – November 2009

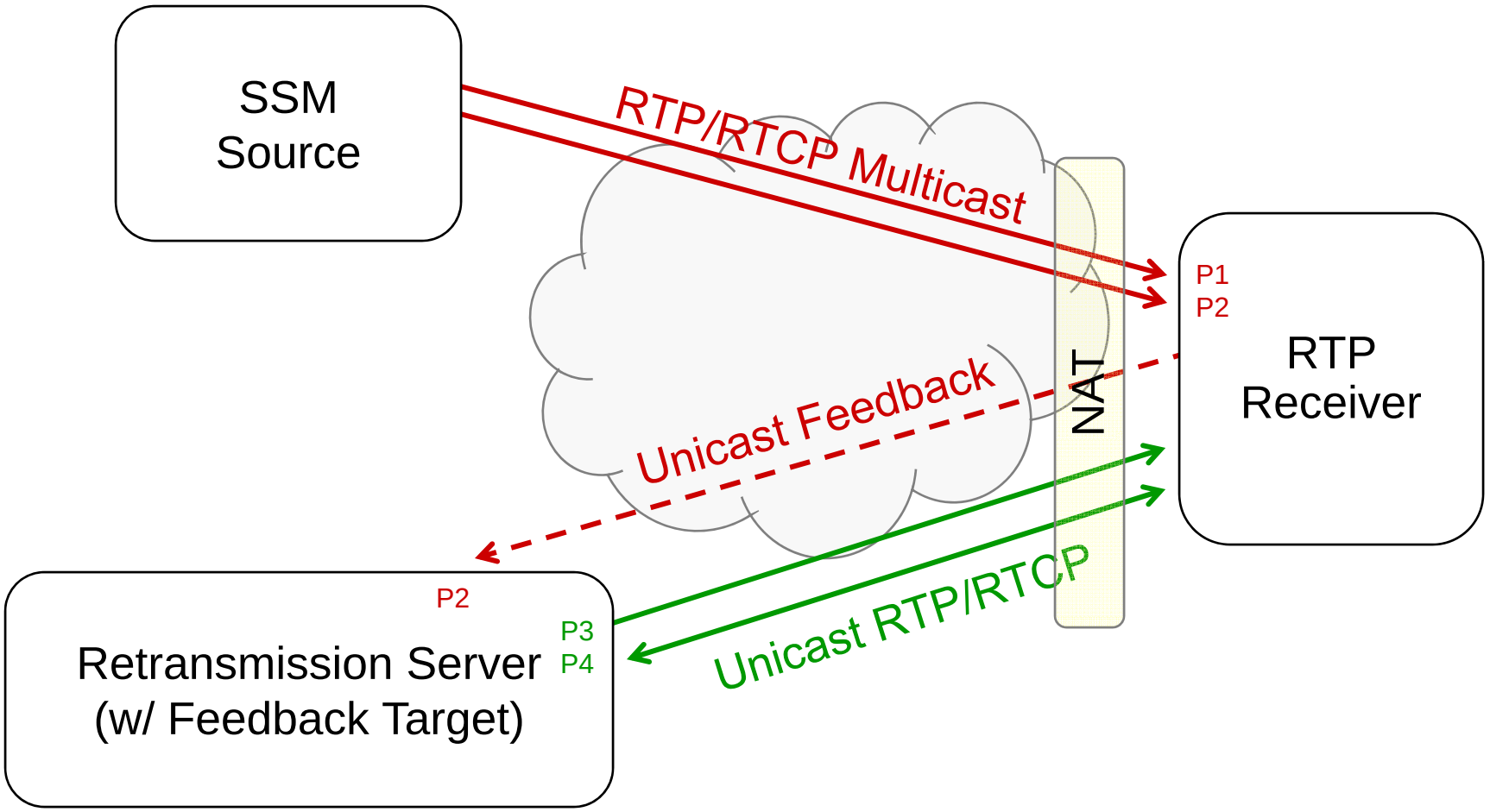
Ali C. Begen and Bill Ver Steeg

{abegen, billvs}@cisco.com

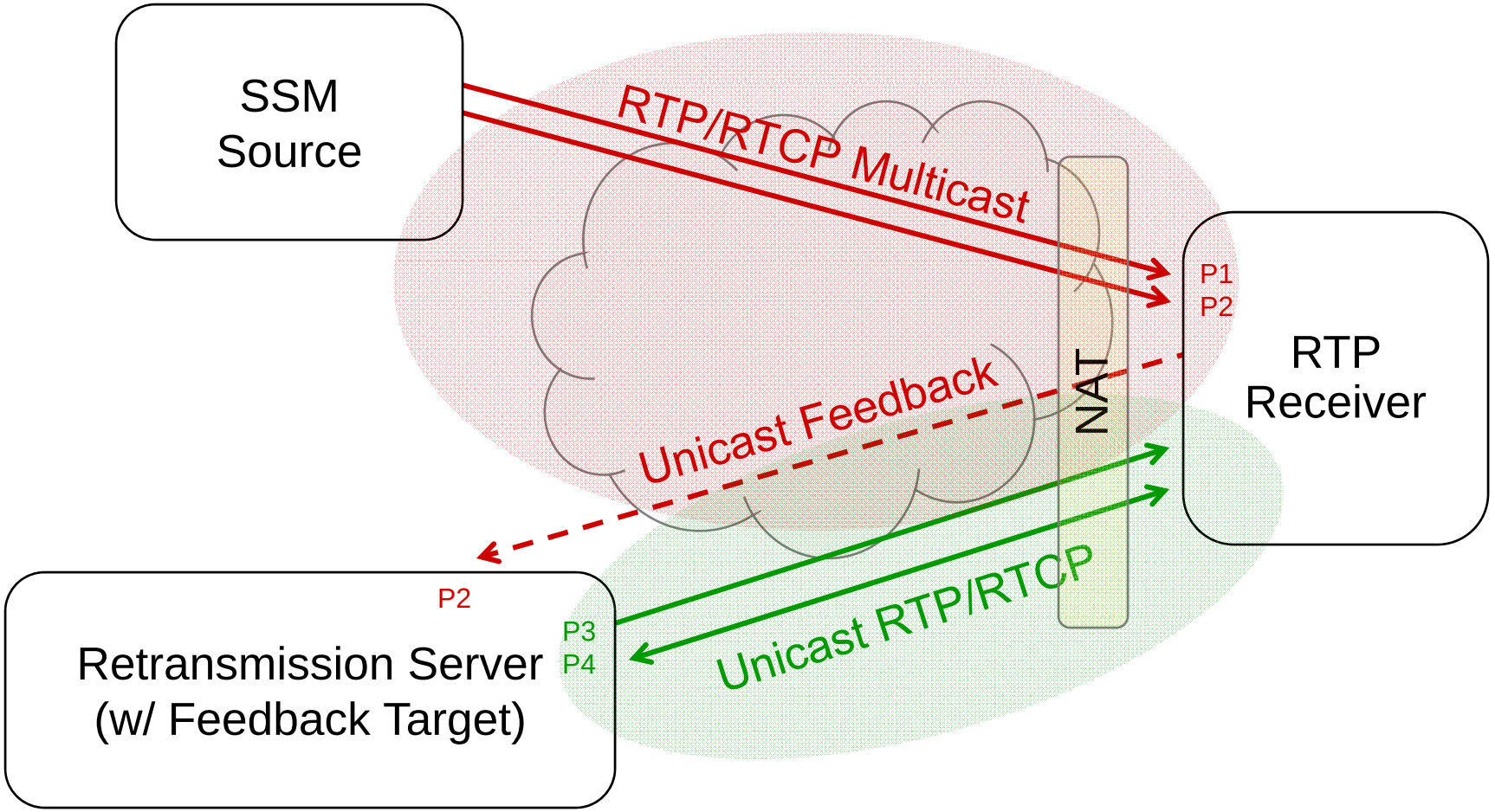
Introduction

- When an RTP application mixes an SSM session with unicast session, issues with port selection may arise
  - In multicast, ports are defined declaratively
  - In unicast, receivers may want to choose their own ports
- E.g., in SSM distribution:
  - RTP Receiver – NACK/RAMS-R → Feedback Target (Multicast RTP session)
  - Ret. Server – Ret. Packets → RTP Receiver (Unicast RTP session)
- SDP is sent via session announcement
- There is no offer/answer phase

SSM Distribution w/ Unicast Retransmissions



SSM Distribution w/ Unicast Retransmissions

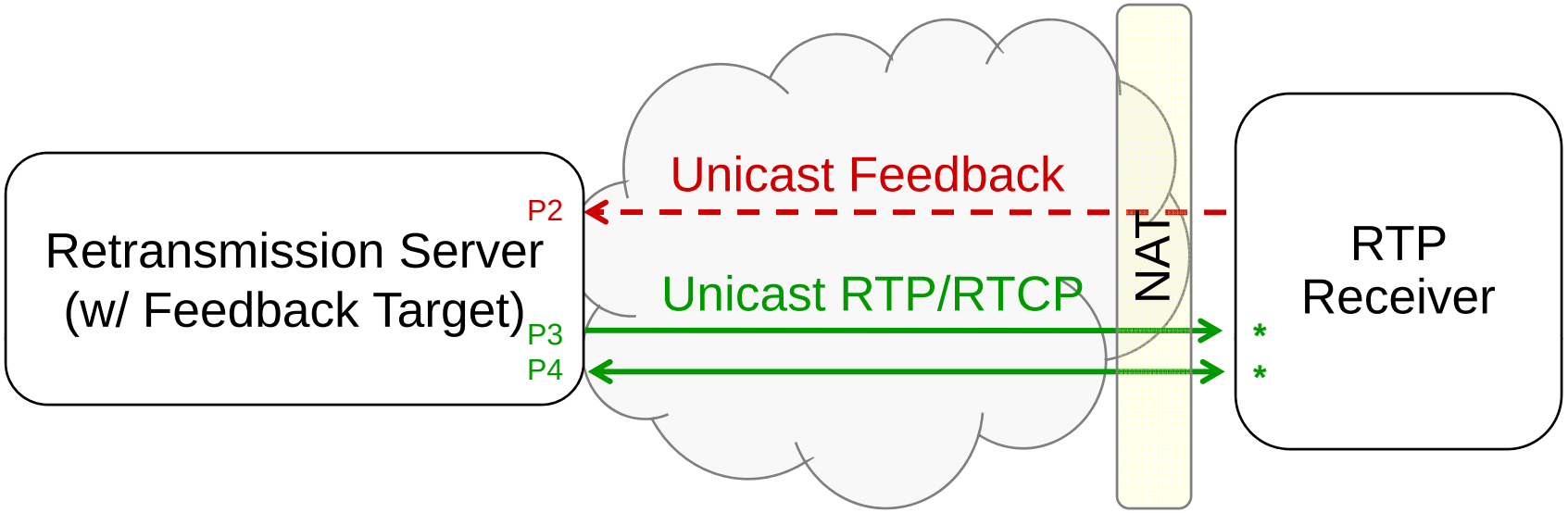


SSM Distribution w/ Unicast Retransmissions

```
a=group:FID 1 2
m=video 41000 RTP/AVPF 98
i=Primary Multicast Stream
c=IN IP4 233.252.0.2/255
a=source-filter: incl IN IP4 233.252.0.2 192.0.2.2
a=rtpmap:98 MP2T/90000
a=rtcp:41001 IN IP4 192.0.2.1
a=rtcp-fb:98 nack
a=mid:1
m=video 41002 RTP/AVPF 99
i=Unicast Retransmission Stream
c=IN IP4 192.0.2.1
a=rtpmap:99 rtx/90000
a=rtcp:41003
a=fmtp:99 apt=98; rtx-time=5000
a=mid:2
```

Parameter	Explanation
S=192.0.2.2	Address of the distribution source
G=233.252.0.2	Destination address where the primary multicast stream is sent to
P1=41000	Destination (RTP) port where the primary multicast stream is sent to
P2=41001	RTCP port on RS and clients for the primary multicast session
RS=192.0.2.1	Address of the retransmission server
P3=41002	RTP port on RS for the unicast session
P4=41003	RTCP port on RS for the unicast session

SSM Distribution w/ Unicast Retransmissions



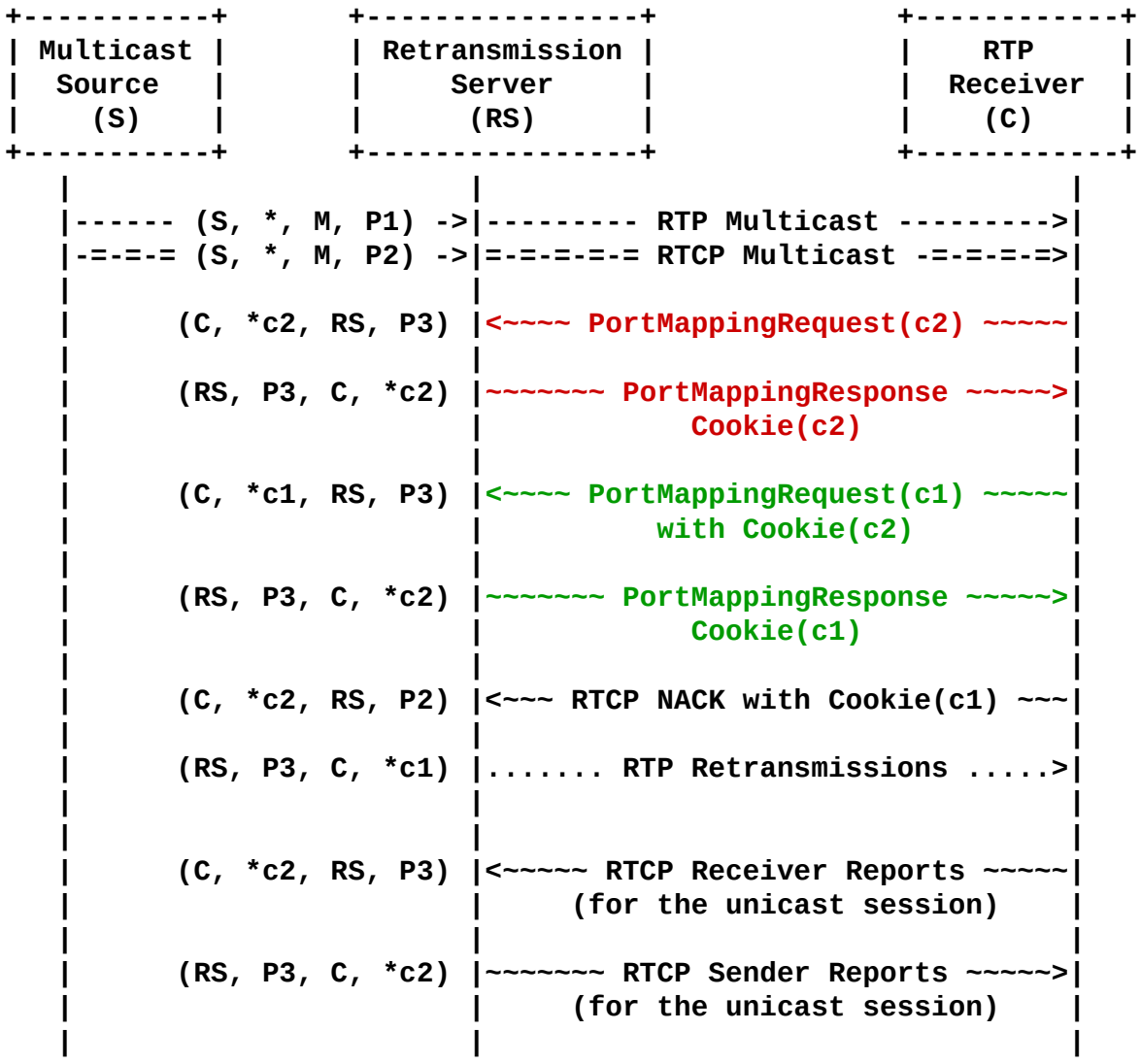
- RTP receiver may set up the NAT by sending packets to ports P3 and P4 from its desired RTP and RTCP ports, respectively
- But, how does the server correlate those messages with the unicast feedback on the multicast session?

  What if messages arrive out of order?
- In the RAMS context, initial setup delay is not desirable
- Port muxing helps but does not avoid the problem

Requirements for Solution

- Design a scalable and distributable system
- Use atomic, client-driven transactions in order to limit the amount of state information maintained by the server
- Use idempotent transactions to limit the impact of lost messages
  - The state of the system only depends on the last successfully received message
- Do not try to correlate information from messages that do not fate-share
- Do not introduce new vectors for attacks
- Do not carry transport addresses explicitly at the application layer
- Do not have any IPv4/IPv6 dependencies
  - Use opaque address information – a cookie
  - Cookies are not meant to be understood by clients or other ALG-like devices
- Be NAT-tolerant

Proposed Methodology



Proposal – Request Phase

- Client ascertains RS, P3 and P4 from the SDP
- Client determines its port numbers – *c1 and *c2
- Client sends separate PortMappingRequest messages from ports *c1 and *c2 to server ports P3 and P4, respectively
- Receiving an RTCP packet on its RTP port requires server to support muxing

  Server must support muxing on port P3

  → There is no need to specify port P4 in the SDP
- Server derives client address (C) and ports *c1 and *c2

Proposal – Response Phase

- For each PortMappingRequest message, server generates a cookie that conveys the addressing information using a reversible transform
- If client DOES support muxing on port *c1
  - A single request and cookie via a PortMappingResponse message is sufficient
  - There is no need for port *c2
- If client DOES NOT support muxing on port *c1
  - Both PortMappingResponse messages MUST be sent to port *c2
  - PortMappingResponse messages must then indicate which port the cookie is for

Editor's note: This requires client to include the cookie for port *c2 when requesting the cookie for port *c1, which introduces delay and dependency

Proposal – Subsequent Messages

- Assume that client chooses two distinct port w/o muxing
- If an RTCP message will trigger server to send

  RTP traffic only, the RTCP packet has to include Cookie(c1)

  RTP and RTCP traffic, the RTCP packet has to include Cookie(c1) and Cookie(c2)
- If no transmission will be triggered (e.g., receiver reports), no need for cookies
- Each distinct 3-tuple (RS, P3, *c1/*c2) MUST have its own cookie

Keep Server/Client Ports Unchanged across Sessions?

- Background
  - FTAp = Feedback target with a specific address and port
  - SSM sessions are identified by (S, G)
    - An SSM session can have only one FTAp
    - Different SSM sessions may share the same FTAp
  - All RTP streams sharing an FTAp must have a unique SSRC value
- Should we keep server/client ports unchanged across sessions?
  - No setup delay (Makes NAT traversal easier)
  - This requires strict SSRC management across all multicast RTP streams
  - What is WG's input on this?

Input on RFC 4588

- RFC 4588 says:

  In the case of session-multiplexing, the same SSRC value MUST be used for the original stream and the retransmission stream

- This requires the server to use different RTCP ports
- What if the primary multicast and retransmission streams use different SSRCs?

  Can the server use the same RTCP port for both sessions?