



# Reducing RSVP Refresh Overhead using State Compression

Lan Wang, Andreas Terzis, Lixia Zhang

4/29/99

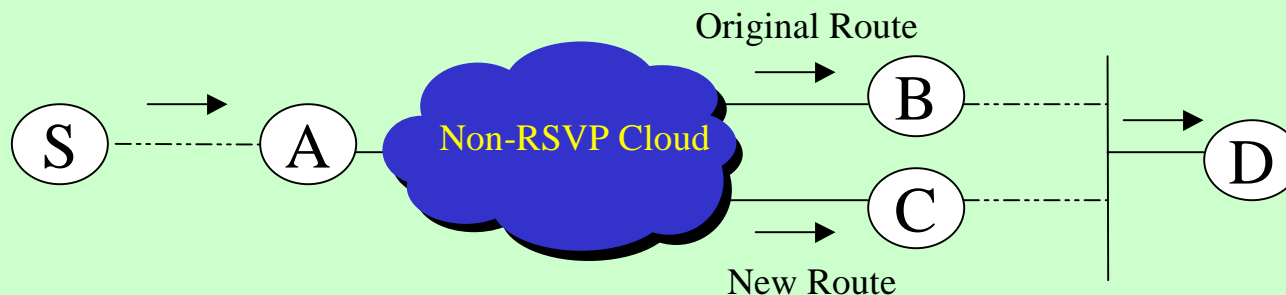


# Goals

- Reduce the refresh overhead of RSVP
  - one refresh message per RSVP state => one refresh message per RSVP neighbor
- Minimize re-synchronization delay
- Keep the soft-state nature of RSVP
  - Keep RSVP state consistent rather than reliably deliver all messages

# Constraints

- Need to group all RSVP sessions on a per neighbor basis
- Rely on route change notification
- Non-RSVP Cloud





# Our Approach

- Digest
  - a set of MD5 signatures
  - a compressed version of the RSVP state shared between two nodes
- Digests sent periodically to neighbors
  - refresh RSVP state
  - discover inconsistency

## Our Approach (cont.)

- Raw RSVP messages sent only when
  - state changes: e.g. Tspec or Rspec changes, route changes
  - re-synchronizing between two nodes
- An option to request ACK
  - ensure delivery of critical RSVP messages
  - discover compression-capable neighbors
  - keep track of synchronization point



# Digest Computation

- Objects included in computation

| RSVP Structure | Objects to Include                                 |
|----------------|--|
| Session        | Session Object                                     |
| PSB            | Sender Template, Sender Tspec, ADSPEC, Policy Data |
| RSB            | Filterspec, Flowspec, Style, Policy Data           |

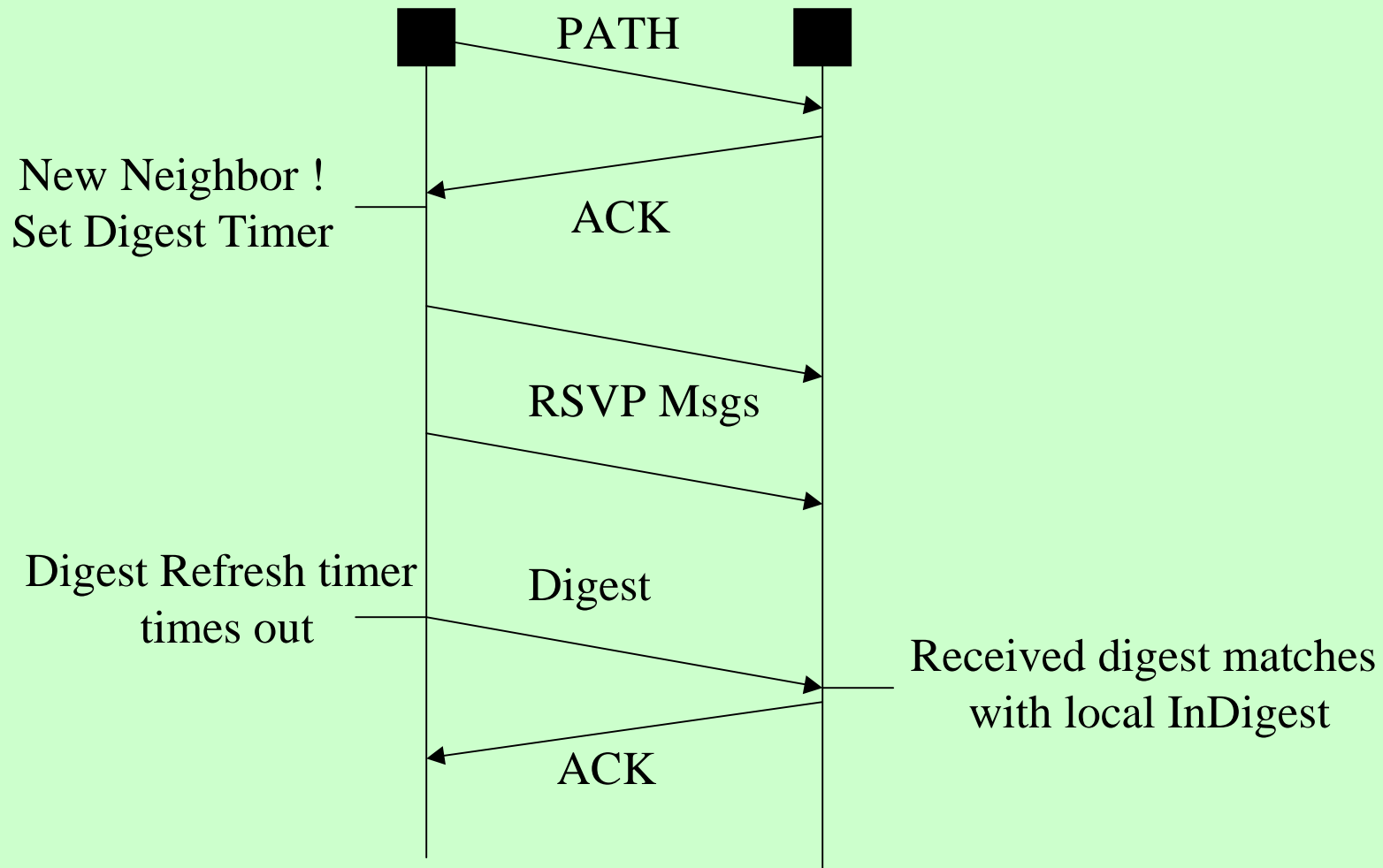


# Neighbor Discovery

- Request ACK message for raw RSVP messages of a new state
- Three cases:
  - Receive ACK message: neighbor is compression-capable
  - Receive PathErr or ResvErr message: neighbor is compression-incapable
  - No response: try again

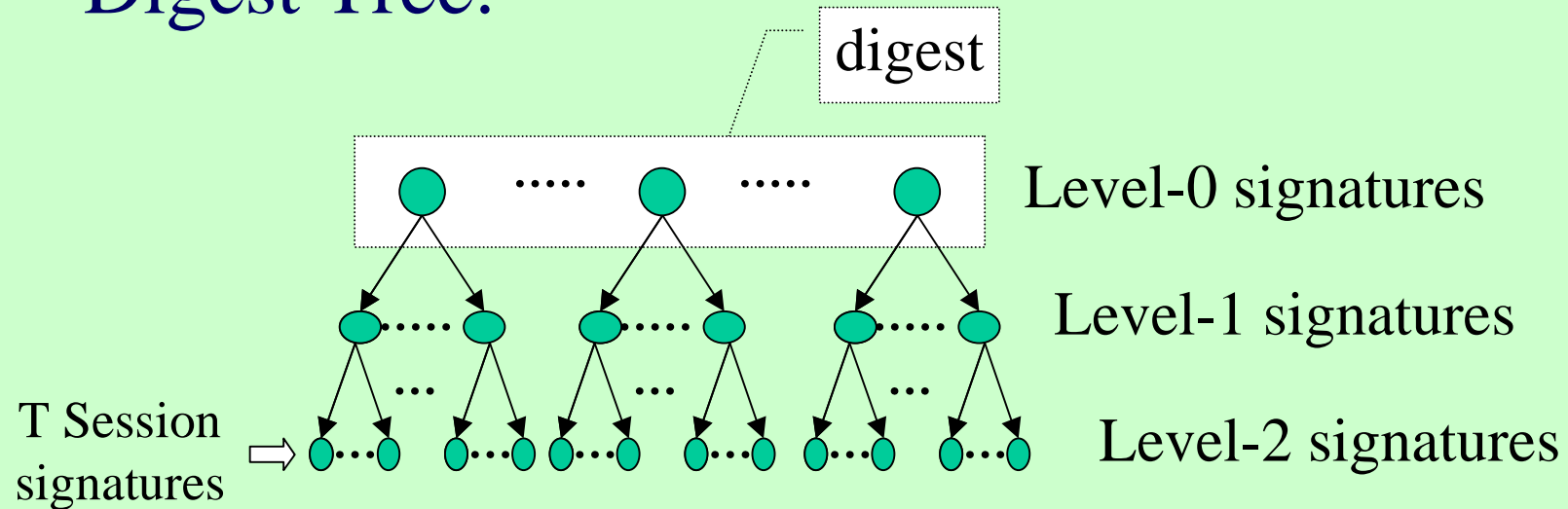


# Normal Case Example



# Digest Computation

- Digest Tree:



- $N$ : number of signatures contained in a digest
- $T$  sessions,  $\log_N T$  levels
- $S(i,j) = \text{MD5}(S(i+1, j*N), \dots, S(i+1, j*N+N-1))$



# New Messages

## – ACK Message (Msg Type 15)

<ACK Message> ::= <Common Header> [ <INTEGRITY> ]  
<TIMESTAMP> [ <TIMESTAMP>... ]

- Acknowledges the receipt of message(s) carrying  
<TIMESTAMP> [, <TIMESTAMP>]

## – Digest Message (Msg Type 14)

<Digest Message> ::= <Common Header>  
[ <INTEGRITY> ] <TIMESTAMP> <DIGEST> [ <TIME\_VALUES> ]

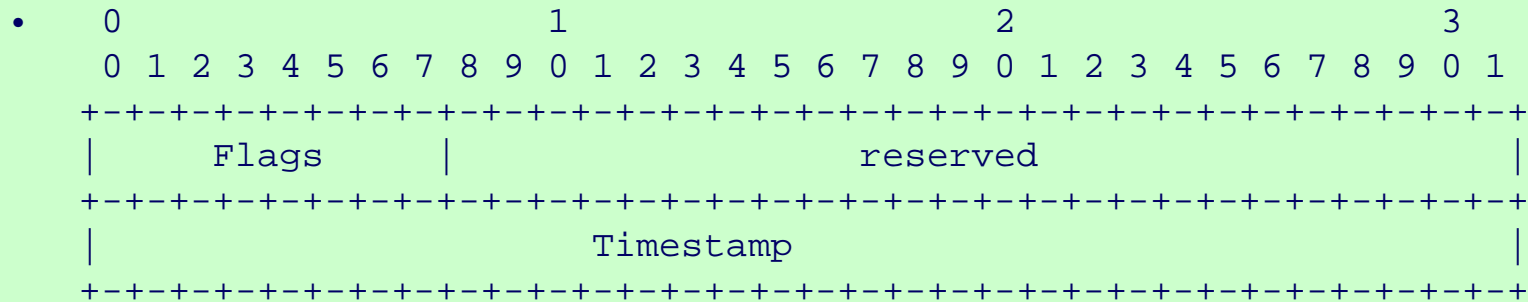
## – DigestErr Message (Msg Type 16)

<DigestErr Message> ::= <Common Header>  
[ <INTEGRITY> ] <TIMESTAMP> <DIGEST>

- Negative ack, contains digest computed at receiver



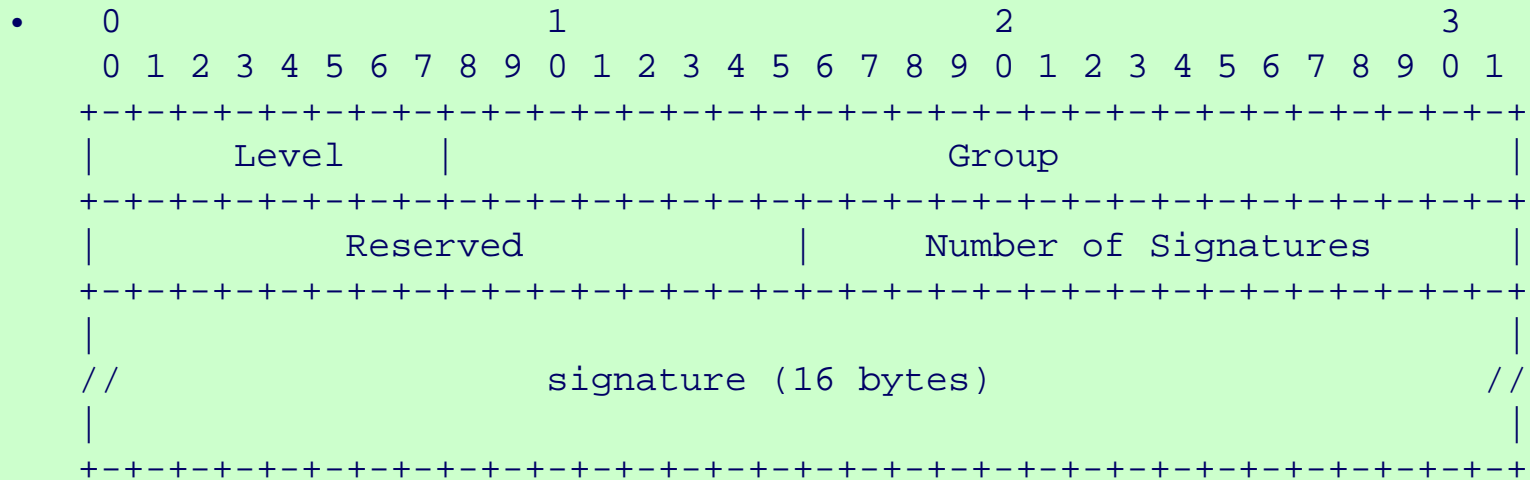
# TIMESTAMP Object



- Class in 0bbbbbbb range
- Flags: 0x80 = ACK\_Requested flag
- Timestamp:
  - 32 middle bits of NTP time
  - When combined with the message generator's IP address, uniquely identifies a message.



# DIGEST Object



- Class in 10bbbbbb range
- Level: level of signatures in digest tree
- Group: Block of signatures inside level
- Number of Signatures: # of signatures in object
- signature: an MD5-checksum

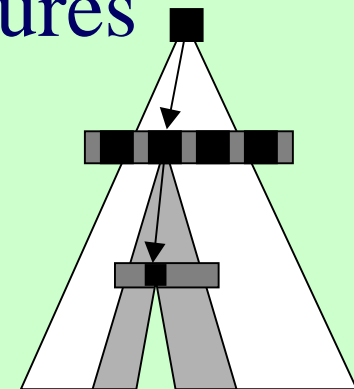


# Recovery Procedure

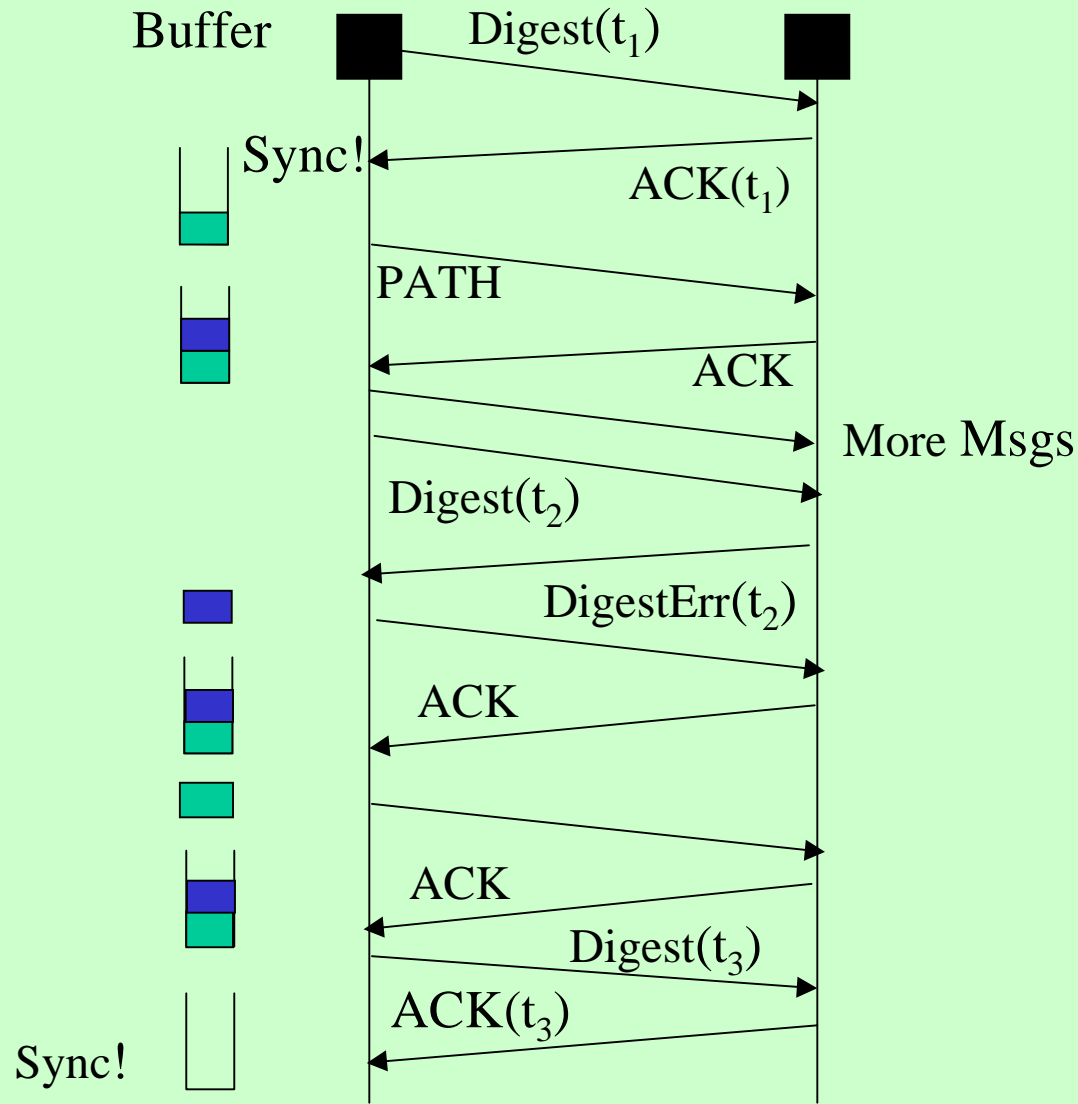
- Each node keeps a buffer of messages sent between digests
  - purges them once it receives ACK for digest
  - re-sends them if it receives a DigestErr, with ACK\_Requested flag turned on
  - then resends digest, if ACK then sync'ed else the next procedure is followed

## Recovery Procedure (cont.)

- DigestErr contains set of top level signatures computed at the receiving side
  - Sender finds which of the N signatures differ
- Sends new Digest message(s) of next lower level rooted at mismatched signatures
  - Follow same procedure until the lowest level is reached
  - Refresh all N sessions



# Recovery Example



Sync!  
4/29/99

# Digest Tree Costs

- Cost of MD5 is linear on the size of the message
- Digest Tree requires  $O(T)$  space
- Change session parameter costs  $O(\log_N T)$
- Session Insertion/Deletion
  - BB (red-black) trees support  $O(\log_2 T)$  insertion/deletion time
  - B trees support  $O(N \log_N T)$  insertion/deletion time



# Features

- Efficient state re-synchronization
- Allow individual nodes to choose original RSVP refreshes or the refresh reduction
- Backward compatibility with the current RSVP specification
- Incremental digest computation when part of the session(s) changes state

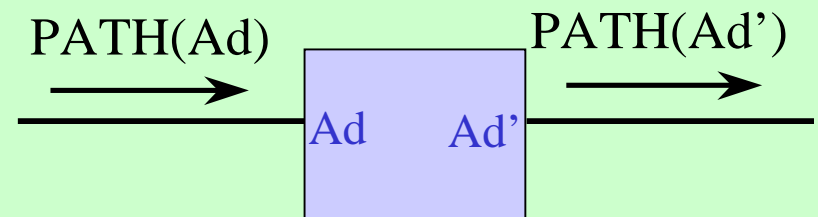


# Summary of Changes Required

- Protocol
  - 2 new objects (TIMESTAMP, DIGEST)
  - 3 new message types (Digest, ACK, DigestErr)
- State
  - Neighbor Data Structure
  - POLICY & ADSPEC
    - These objects can change locally
    - Current RSVP spec keeps object received

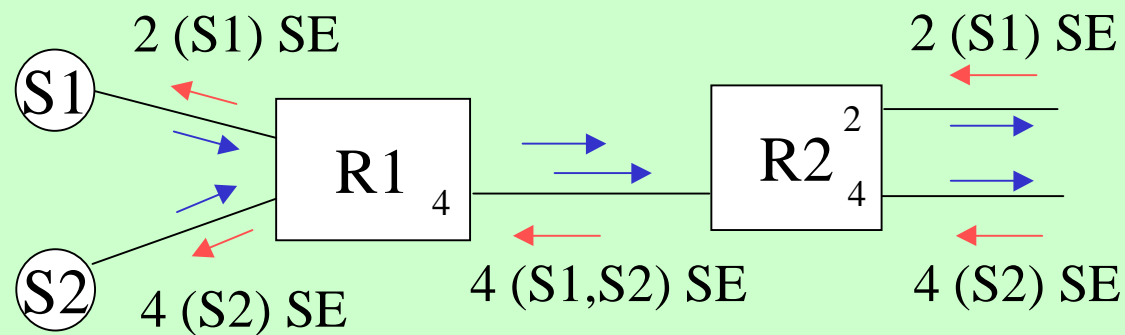
# ADSPEC & POLICY\_DATA

- Need to keep copy of the forwarded object to calculate digest for that neighbor
- Digest refreshes trigger updates for these objects
- How to detect changes
  - Assume always new
  - Change interface
  - Perform binary comparison



# Open Issues

- Multicast
- Many-to-one Unicast



→ PATH message;    → RESV message  
 S1: sender, Tspec = 2;    S2: sender, Tspec = 4  
 2 (S1) SE: Rspec = 2, Filterspec = S1, Style = SE