RSVP Extensions for Policy Control

Shai Herzog,
IBM T. J. Watson Research Center
herzog@watson.ibm.com

draft-ietf-rsvp-policy-ext-01.{txt,ps}

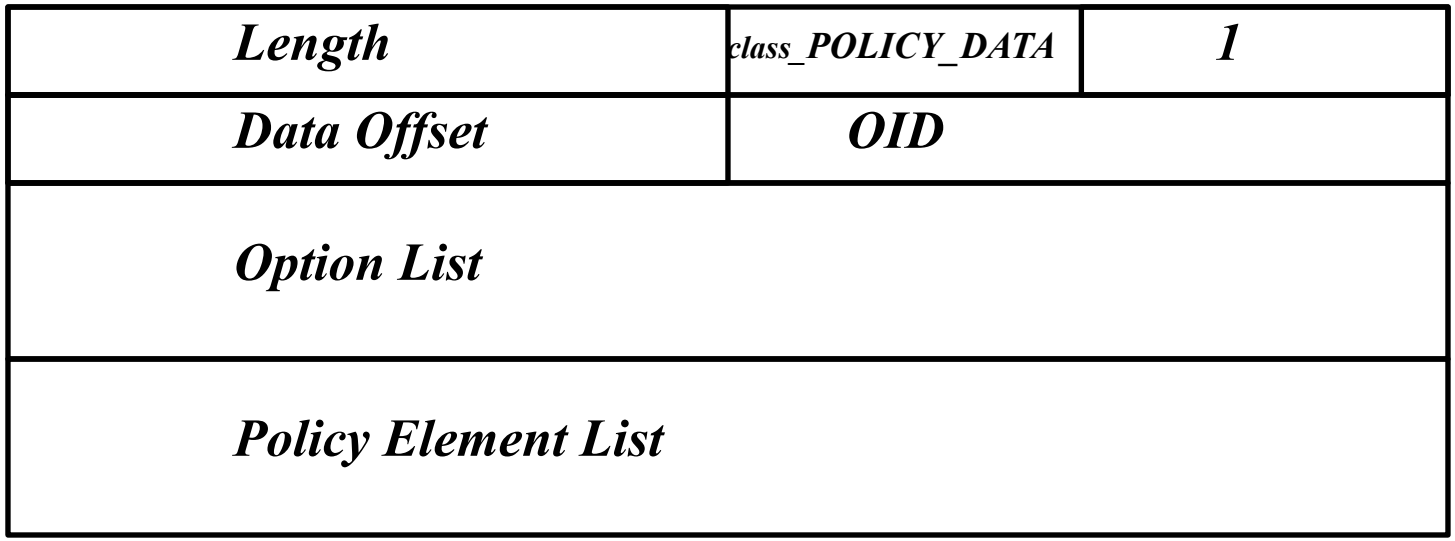
General Policy Control Characteristics

- **Cannot assume global info. or agreements**
  - Bilateral agreements (even with non-neighbors?)
  - Distributed responsibility
  - Good scaling properties
- **Policies must be controlled/configured locally**
  - However, it is important to have:
    - Inter-operability
      - Multi-vendor environments
      - Between providers
    - Some globally adequate/consistent policies

RSVP Extensions

- **Purpose:**
  - Initial vehicle for experimentation and development
  - Inter-operability in multi-vendor environments
  - Consistent, comprehensive, and flexible p.c.
- **What do we care about?**
  - Data formats
  - RSVP/PC interface
    - Define required functionality, not form
  - Router/Policy Server interface
    - Define required functionality.
  - Common semantics/processing rules
    - Security, errors, default handling, fragmentation, etc.

POLICY_DATA Object



- **RSVP Style Options:**
  - FILTER_SPEC, RSVP_HOP, INTEGRITY
- **Policy Options:**
  - Fragmentation, NoChange, FilterList

Security Model

- **Same model as RSVP:**
  - Hop-by-Hop
  - INTEGRITY object protection
- **Policy hops (vs. RSVP hops)**
  - Policy may be performed at edges
  - In-cloud RSVP nodes are not trusted
  - Key exchange only between policy nodes

Policy Options

Fragmentation

Length	*0*	*1*
Variable Length		

NoChange

Length	*0*	*2*
Reserved (0)	*Previous OID*	

Filter List

Length	*0*	*3*
Counter	*Hash type*	*Reserved (0)*
FILTER_SPEC List 32 bit hash/CRC		

Policy Control (PC) Interface

- **Minimal changes to the RSVP spec:**
  - Use only existing RSVP messages and signaling
  - Push the “smarts” to the PC module
    - Flow identification (Filter Spec lists)
    - Security (authentication)
    - Fragmentation
    - Required action and error signaling are decided by PC
- **Synchronous with RSVP messages**
  - No additional timers, asynch. events, etc.
- **Handle multiple P.D. Objects**
  - Simple and extendible P.D. object format

Main Requirements from *Any* PC Interface

- **Allow exchange of policy information**
  - Receive, process and send policy objects
  - PC_InPolicy() and PC_OutPolicy()
- **Allow Path/Resv status checks**
  - PC_AuthCheck()
- **Maintain synchronization with RSVP**
  - PC_Branch() and PC_Close()
- **Initialize**
  - PC_Init()

Example/Prototype of PC Services

- **PC_InPolicy(**session, lih, rsvp_hop,msg_type,in_objs, rhandle, rflow_spec, timeout**)**
  - Process a set of incoming objects
  - AuthCheck for ***lih***.
  - May accept, reject, or allow preemption
  - Instructs RSVP on Error handling
- **PC_OutPolicy(**session, filter_list, lih, rsvp_hop, msg_type, out_objs, max_pd, avail_pd**)**
  - Assemble a list of outgoing objects destined to rsvp_hop
  - Attempt / fragment to comply with max_pd and avail_pd

PC Services (Cont.)

- **PC_AuthCheck(session, filter_list, lih, msg_type, rhandle, rflow_spec)**
  - Check the status of a reservation
  - If message arrived from
    - Downstream: check Path authorization
    - Upstream: check Resv authorization
- **PC_Branch(session, filter_list, rsvp_hop, op_type)**
  - Synchronize branch state with RSVP
    - Blockade state or purge state
- **PC_Close(session, filter_list)**
  - Close a policy control session
- **PC_Init(void)**
  - Initialize the PC module

PC Success Codes:

- **Function return code, PC_errno, PC_flags**
- **PC_Flags instruct RSVP on immediate action:**
  - PC_RC_ModState: Modified policy state; force a refresh
  - PC_RC_SendErr: Send Error (PathErr or ResvErr)
  - PC_RC_Respond: Send a response/reverse message
  - PC_RC_Cancel: Reject the reservation (or path)
  - PC_RC_Preempt: Accept but allow later preemption, if needed

Error Signaling Sequence

- **PC_AuthCheck() or PC_InPolicy() fail**
  - PC_RC_SendErr is set
- **RSVP performs standard error handling**
  - Generate PathErr or ResvErr
- **RSVP queries the PC for outgoing objects**
- **The PC provides a set of outgoing objects**
- **RSVP sends the error + objects**
  - If no objects are given, may suppress the err message
- **PathErr and ResvErr processing unchanged**

Default Handling of P.D. Objects

- **Non policy nodes, or non-recognized policies**
- **Forward P.D. (or policy elements) as-is**
  - Use same message type
  - Concatenate in merging nodes
  - If Concatenation creates large P.D. lists:
    - Syntactic fragmentation, or
    - Leave it to RSVP fragmentation

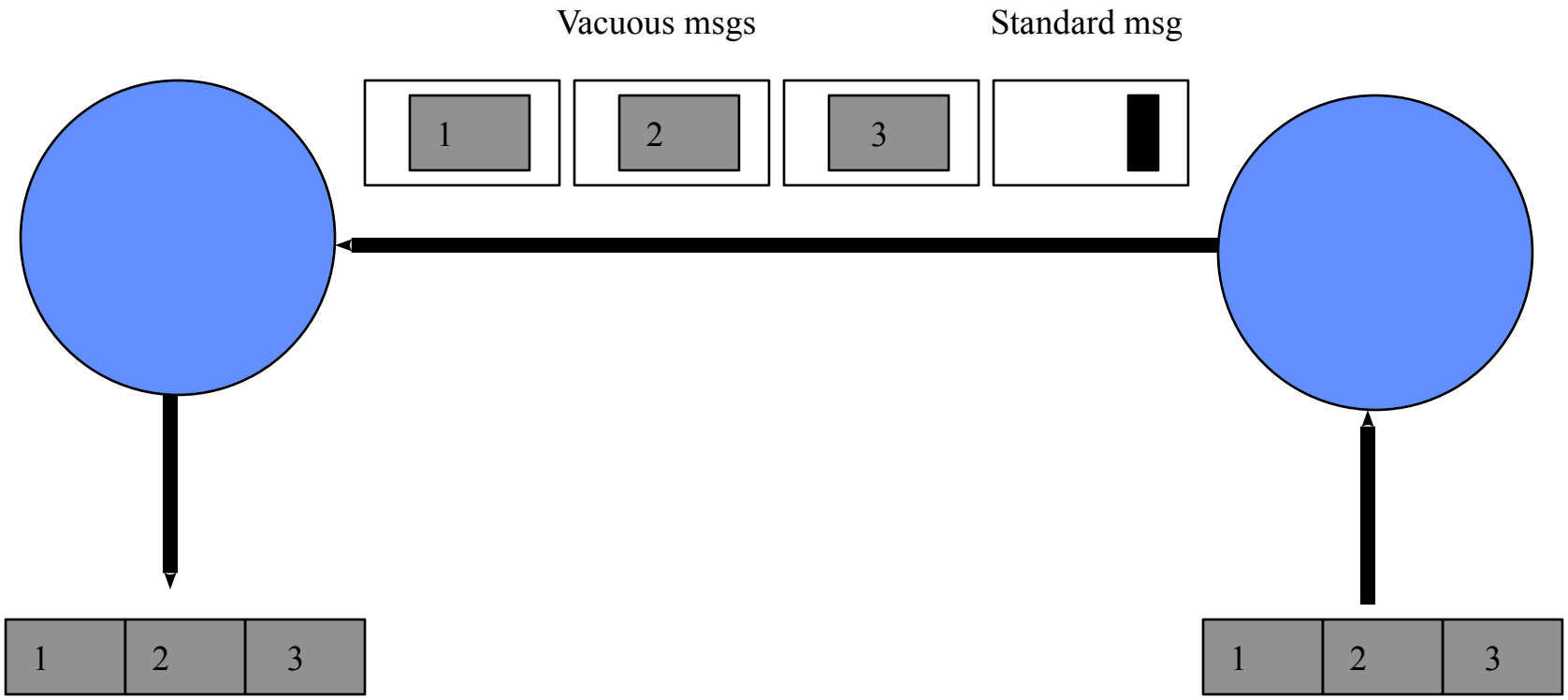
Router/Policy Server Interface

- **P.C.s on routers may query policy servers**
  - Routers may not be able to handle complex policies
  - Policy reply from server may be delayed:
    - P.C. should not approve until server reply is received
    - RSVP should not block
- **Interface definition**
  - Describes generic functionality
  - Basic inter-operability across multi-vendor env.
    - Router and Server may belong to different vendors
    - Routers from various vendors could interface with a single policy server

Syntactic Fragmentation of P.D. Objs

- **Large P.D. -> Large RSVP msg -> Frag. loss**
- **Objectives:**
  - Isolate RSVP from this risk
  - Push P.D. fragmentation to P.C. module
    - Save the added complexity from RSVP
    - P.C. module has greater semantic knowledge
  - Allow local flexibility in fragmentation schemes
    - Policy Hop-by-Hop fragmentation and reassembly anyway
    - Semantic vs. IP style
- **Approach: Syntactic Fragmentation**
  - RSVP is aware of the syntax but not the semantics of P.D. Fragmentation

Fragmentation Example



Syntactic Frag.: building blocks (1)

P.D. Fragments

- All fragments of a PD object have the same OID
  - Like IPv6 Frag. ID
  - OID selection is the responsibility of the sending node
- Fragmentation: PD -> PD_1,... ,PD_n, PDE
- PD_i Carry the Fragment Option
- PDE: small/token object embedded in std. msg
  - Token: 64 bit object, made only with the header and OID

Sending Fragments

- If PC_OutPolicy() produces fragments:
  - Send all fragments first in vacuous messages
- Embed token & non-fragments in standard RSVP msg
- Send out standard messages

Syntactic Frag.: building blocks (2)

- **Vacuous RSVP messages**
  - RSVP messages with discardable RSVP state:
    - Only the state required to route the message
    - Only a duplicate of state delivered by other RSVP msgs.
- **Receiving Fragments**
  - P.D. objects marked as fragments:
    - Are handed over to the PC module regardless of error conditions
    - P.C. Success codes are ignored.
  - Reassembly (and checks): only for non-fragments.

API Considerations

- **P.D. parameters:**
  - Applications provide fully built P.D.s
  - Applications provide guidelines, API library builds
  - Hybrid: P.D.
    - Mainly built by applications
    - Some options are added by API library (like integrity, rsvp_hop, etc.)
- **Per application state**
  - Allow signaling/counting of individual applications
  - Virtual “Prev/Next hop” from local node.