Towards PubSub and Storage integration in ANIMA

ANIMA WG, IETF 99, Prague

Xun Xiao, Zoran Despotovic, Ramin Khalili, Artur Hecker

Information Distribution in ANIMA

- *draft-ietf-anima-reference-model-04*
  - §4.7, p12: Defines Information Distribution as unchartered item, function of ACP
    - Defines flooding as the implementation choice
    - Does not describe details of the implementation
  - §7.3, p18: Aggregated reporting
- *draft-liu-anima-grasp-distribution-04*
  - An implementation candidate using GRASP for information distribution:
    - GRASP Flooding for the whole domain
    - GRASP synchronization for the peer-to-peer distribution
    - Selective Flooding: containing criteria with flooding messages
    - Conflict Handling: with timestamps or version info.
  - Three scenarios:
    - Whole domain, selective domain, and incremental for newly joining node

The “why’s”
Use cases for non-time critical info distribution

- Configuration
  - Autonomic does not preclude management, see current ref. model (and RFC7575)
- Intents (by definition, require distribution to the whole AD)
  - Some intents might “trigger” later: need to store the rule & objects
    - (for special ASAs) “Switch off the AC, once the temperature reaches 23°C”
    - (For ACP) Aggregated reporting, information distribution
- Autonomic applications != no central/unique points
  - Require consensus, collaboration, negotiation, agreement on values, actions
- Autonomic computing: avoid repetitive development
  - Developers / apps would highly profit from a ANIMA-provided means for:
    - Scalable and efficient message distribution (e.g. only to some ASA types)
    - Universal storage means

Flooding vs. distribution

- **Flooding seems to be understood as “distribution to all recipients”**
  - Note that typical definition of flooding is different

vs.

  - E.g. Wikipedia: *"Flooding is a simple computer network routing algorithm, in which every incoming packet is sent through every outgoing link except the one it arrived on“*
- **Flooding as implementation** (= “unconstrained broadcast”)
  - Simple and working solution, *iff*
    - The network is small in scale
    - The network is sparse
    - The network is a tree / guaranteed loop free / does not have multiple paths
  - Note: ANIMA ACP does not fulfill any of these
    - Uses routing, does not constrain scale, does not constrain connectedness, etc.
  - In other networks, flooding exhibits an explosive growth and does not scale
- There are better implementations than the latter achieving the first

PubSub

- An accepted popular model for async communications
  - Decouples pools of subscribers and publishers
    - Publishers do not need to know about subscribers and vs.
    - Provides more flexibility in distribution/interest sets and much higher system scalability
  - Usually implemented as a middleware, can be distributed or centralized
    - OMG DDS, MMQT, XMMP, PubSub
  - In principle, nothing else but application-layer multicast
- Suits nicely the autonomic paradigm
- Can achieve more precise distribution than flooding
- (Usually) Requires storage in its implementation
  - To hold the so-called “backlog” (error handling, etc)

Storage

- Closed loop support for storage can be added
  - With general calls (e.g. *put()* and *get()*)
- All nodes start from the same state, run a procedure and end up with network-wide storage
  - Does not mean that all nodes have to support storage (not a MUST)
    - Instead, the local API call would hide the complexity of *how* it is implemented
  - Proposal: the node whose API is invoked MUST do one of the following:
    1. It MAY store the data object locally at the AN
    2. It MAY use GRASP to find storage-capable nodes
    3. It MAY use the distributed storage to locate the URL of the node that stores
    4. It MAY use an algorithmic means to map the data object to a suitable AN
    5. It MAY report an error (“storage not available”), e.g. during network convergence or while no storage nodes are available.

Integration in ANIMA: Possible solutions

- Dedicated ASA
  - Define an (Optional? Mandatory?) ASA to implement and support storage
  - Advantage: probably easier as less standard-costly
  - Problem: likely less interoperable
- Integration in ANI
  - Storage and PubSub as part of ACP functionality
  - Advantage: understands storage as a fundamental support for autonomic apps (just like routing)
    - PubSub can be easily implemented on top of storage
  - Disadvantage: probably needs rechartering, protocol extensions, additional protocols, etc.

Who needs which API

(Non time critical use cases)	PubSub	Storage
Configuration	X	
Intent distribution	X	
Internal ANIMA implementation	X	X
Autonomic computing	X	X