Autonomic Functions Coordination By The Example

P. Peloso, L. Ciavaglia

ANIMA WG, IETF 96th
Objectives

- Show feasibility of coordination between ASAs
- Identify required elements for a coordination function
- Derive requirements for ASA and ANI
Explaining the use case

Network composed of:
- edge and core routers
- 3 clients on the edges
- 2 types of traffic for each client (premium/best-effort)
- 6 LSP to cope with each traffic
Explaining the use case

ASA sets

2 ASAs running on router A

- Load Balancing ASA
  Balances client traffic between network interfaces
- Risk Aware Routing ASA
  Avoids paths at risk for premium traffic
Deployable over edge router

When deployed

- Monitors network interfaces loads
- Modifies LSPs routes to balance load
Deployable over edge router

When deployed

- Monitors network interfaces risks of failure
- Modifies Premium LSPs routes to reduce their failure exposition
ASA plugging to ANI and device

Current ANIMA picture
- Excerpt from Ref Model
- ASA use GRASP signaling in-between them
- ASA monitor the node and modify its state directly using either NetConf, OpenFlow, SNMP, CLI…
Deployment of ASAs onto network equipment

Deployment means:
- The process during which the ASA “gets in touch” with the device(s) it controls

During deployment:
- ASA should establish session with the device(s) (credentials?)
- ASA should collect from the device some setting info (e.g. nbr of interfaces and interfaces id)
- Use these collected info to compute its Instance Manifest
  - Compute the actual metrics
  - Compute the actual parameters
- Share its Instance Manifest within the ANI
  - Either a selective sharing or broadcast sharing
Deployed over router A:

Monitors network interfaces loads
Modifies LSPs routes to balance load

<InstanceManifest>
  <ASA_Class_ID>
    <Name>LoadBalancing</Name>
    <Provider>AnimaCorp</Provider>
    <Version>1.0.0</Version>
  </ASA_Class_ID>
  <ASA_Instance_ID>3567456</ASA_Instance_ID>
  <AcquiredInputs>
    <InfoSpec>
      <name>InterfaceLoad</name>
      <contentType>Numeric</contentType>
      <context>{RtrA:IfcAB, RtrA:IfcAB}</context>
    </InfoSpec>
  </AcquiredInputs>
  <PossibleActions>
    <ActionSpec>
      <name>LSPPath</name>
      <contentType>ERO</contentType>
      <context>{LSP_{1P}, LSP_{1BE}, LSP_{2P}, LSP_{2BE}, LSP_{3P}, LSP_{3BE}}</context>
    </ActionSpec>
  </PossibleActions>
</InstanceManifest>
2\textsuperscript{nd} ASA – Risk Aware ASA

Deployed over router A:

Monitors network interfaces risks of failure

Modifies Premium LSPs routes to reduce their failure exposition

<InstanceManifest>
  <ASA_Class_ID>
    <Name>RiskAware</Name>
    <Provider>AnimaCorp</Provider>
    <Version>1.0.0</Version>
  </ASA_Class_ID>
  <ASA_Instance_ID>7167456</ASA_Instance_ID>
  <AcquiredInputs>
    <InfoSpec>
      <name>InterfaceRisk</name>
      <contentType>Probability</contentType>
      <context>{RtrA:IfcAB, RtrA:IfcAB}</context>
    </InfoSpec>
  </AcquiredInputs>
  <PossibleActions>
    <ActionSpec>
      <name>LSPPath</name>
      <contentType>ERO</contentType>
      <context>{LSP\textsubscript{1P}, LSP\textsubscript{2P}, LSP\textsubscript{3P}}</context>
    </ActionSpec>
  </PossibleActions>
</InstanceManifest>
And now – ASA Execution

CHAOS?

ASA LB_A

ASA RA_A
Conflict Identification

- **Role:**
  Identify potential conflicts

- **Inputs:**
  Instance Manifests of All ASAs in the ANI

- **Outputs:**
  Groups of ASAs (or Autonomic Functions) that may be conflicting

- **Situated:**
  Either as a part of the ANI
  Or sitting on top of ANI
  (Can be distributed)
Autonomic Functions Coordination

Situated Conflict Identification entity

Autonomic Function A

Autonomic Function B

ANI

Node 1

Node 2

Node 3

Node N
Conflict identification

1. Aggregate the graphs provided by each ASA Instance Manifest
2. Identify loops:
   - Loops formed by different ASAs
   - Loops sharing edges
3. Return list of ASAs implied in conflicting loops

<PotentialConflict>
<ConflictID>1</ConflictID>
<ConflictingASAs>
{7167456, 3567456}
</ConflictingASAs>
</PotentialConflict>
And now – ASA Execution

CHAOS?

ASA LB_A  ASA RA_A
Between ASA Coordination

- **Role:**
  Address potential conflict between ASAs

- **Inputs:**
  Potential Conflict description from Conflict Identification

- **Outputs:**
  Control commands to ASAs

- **Situated:**
  Either as a part of the ANI
  Or sitting on top of ANI
  (Can be distributed)
Situated Coordination entities
Between ASA coordination

1. Receives a Potential Conflict description
2. Identify applicable algorithm:
   • Depending on available algorithms
   • Depending on control capacities disclosed by ASAs Instance Manifest (If applicable, otherwise supposed no more than start/stop)
3. Instantiate an algorithm process in charge of the ASA group
4. If applicable set algorithm parameters
   • Depending on ASA priorities (from Intents)
   • Depending on ASAs features additionally disclosed by Instance Manifest
5. Run the algorithm process, which sends control commands to ASA (e.g. a random token determines which ASA can execute)
Time separation method

Allow the least impacting ASA to converge in-between two iterations of the most impacting one

Load Balancing

Risk Aware ASA

time
Conclusion

Conflict resolution based on common coordination components is feasible
- Applicable to ASA complying to set of requirements (manifests)

- Showing the step-by-step process

- Showing the information to be conveyed

- Showing possible formats (XML based, but can be TLV based)

- Providing a basic method to achieve the process (multiple algorithms possible)
Conclusion

ASAs must follow a defined process

Requirements to ASA

ASA Life-cycle

Know what ASA “does” to the network

Instance Manifest

Control when/how ASA runs

Start/Stop
Conclusion

ASA life-cycle

Undeployed

Installed

Un-Instantiating

Instatiated

Instantiating

Operational

Undeployed

Receives a Mandate

Advertizes its Manifest

Receives a Set-up cmd

Receives a Set-down cmd

Receives revoke Mandate

Advertizes blank Manifest
Augmenting ANI with ASA Life Cycle Management