GeneRic Autonomic Signaling Protocol
IETF109 RTGWG
An overview

Toerless Eckert, Brian Carpenter
November 2020

draft-ietf-anima-grasp (RFC Editor)
draft-ietf-anima-grasp-api (IESG agenda)
TCP/IP and automation... great start, but then...

- Dark ages (PSTN): hierarchically centralized management and control of network functions including traffic path selection
- 1969 ARPANET – distributed, self-optimizing, self-healing traffic path selection (‘routing’).
- 1990's TCP/IP routers/networks starting to grow into their role of “nerd knob heaven”. Nothing beyond 1960th design components is self-{building,healing,optimizing}
- 2010’ SDN for TCP/IP evolves. SS7 rises from the Ashes.

- 1980’s Device-level plug&play networks (e.g.: appletalk). Distributed routing, Auto-addressing, auto-naming, ...
- 2000’s notion of autonomous networks becomes more formalized (IBM et. all).

- How could we bring more distributed automation / self-x to TCP/IP ?
  - IRTF NMRG -> IETG ANIMA WG
Pragmatic automation

• Routers of many vendors had “scripting” for automation often since 2000'th
  • First language: TCL – vendors C, H, J, probably more. Now many more languages
  • Widely used (before SDN) to provide device-local automation:
    • Script uses router internal CLI to poll/monitor some router behavior and trigger actions (CLI configs, logging, ... from it)

• Scripting ~=
  • Runs on the router itself
  • Can be built not only vendor (e.g.: also customer)
  • Can have its own lifecycle independent of “router OS”
Pragmatic automation – limitations

• How to build LAN or network wide coordinated automation (scripts) ?
  • ANIMA calls them “Autonomic Service Agents” (ASA)

• Examples: automate setup & securing routing protocols:
  • Negotiate session key with possible peers, auto-configure routing protocols security.
  • Auto-configure IPsec SA for routing protocol packet
    ... when routing protocol does not support security itself (e.g.: PIM)
  • L2: auto-configuring MacSec.
  • Non-security automation:
    • Distributed elect “most central router in network” and configure it as multicast RP (multicast server)...
  • Negotiate various service parameters (QoS – class weights)
  • Operational scripts, ...

• Gee... Communications is very hard to script...
  ... And repetitiously requires common components
What common components would we need for such distributed automation?

• (Routing) protocol / use-case independent:
  • Mutual keying material to authenticate peers, Confidentiality for communication between peers
    • Even if you do not like security: When everything is meant to run automatic, you removed the implicit security stemming from the human operator – security is a big MUST!
  • Common, easily extensible negotiation protocol
  • Peer discovery mechanism – link-local and network wide

• For communication between non-L2 adjacent candidate peers:
  • Routing protocol independent L3 reachability
    • *routing protocol may first need to be auto-configured*
GRASP step by step

- Preconditions for peer-to-peer communication:
  - Assume one peer knows another peer’s IP address
    - And has network layer connectivity to it
  - Assume both peers have mutually trusted keying material

- **Use TLS connections** – authentication, confidentiality

- What else is needed? Learn from application/”web” layer protocols!!!
  - Limited set of reused ‘common’ protocols, quite successful, our inspirations:

- JSON (‘data encoding’/ ‘presentation layer’)
  - Native from Javascript: **easy and flexible use of arbitrary data structures**

- REST via HTTP/URLs (‘communication primitives’ / ‘session layer’)
  - Few communication primitives – everything else left up to ‘application’ definitions
GRASP step by step

• So.. Why not just use what app layer does (HTTP, JSON, URLs) ?

• Want more compact encoding that supports binary
  • But without “ASCII-Art” one-off protocol / application specification

• GRASP uses **CBOR (RFC7049) for application payload encoding**
  • CBOR =~ binary encoding of ‘almost’ JSON data structures (support text & binary data)
  • CDDL = schema description for CBOR =~ formal language replacement for ASCII art specs.

• HTTP also has too much overhead (text format), REST via URLs too.

• **GRASP itself built solely with CBOR**
  • Few primitives: For P2P: Request Negotiate/Sync, Negotiate,/End , Synchronize
Simple “Synchronization” request/reply example (from GRASP section D.3):

CBOR encoded messages via a GRASP TLS connection:

**Initiator to Responder:**

[M_REQ_SYN, 4038926, ["EX2", F_SYNCH_bits, 5, 0]]

On the wire encoded: h'83041a003da10e8463455832050500'

**Responders reply to initiator:**

[M_SYNCH, 4038926, ["EX2", F_SYNCH_bits, 5, ["Example 2 value=", 200]]]

On the wire encoded: h'83081a003da10e8463455832050582704578616d706c6520322076616c75653d18c8'
GRASP step by step

• Discovering GRASP peers (for a specific ‘objective’)

• GRASP can use L2 multicast to announce or request objectives
  • Specified / standardized with IPv6 link-local multicast

• GRASP can do L3 domain wide multicast announce or request objectives
  • Not requiring any L3 connectivity (unicast or multicast routing)
  • Instead relying on GRASP per-L3-hop GRASP message propagation
    • GRASP forwarding agent:
      Flooding of messages with loop detection/breaking (per-message unique identifiers).

• GRASP discovery communication primitives: Flood, Discover
How to use / deploy GRASP – many options

- Minimum: With just peer-to-peer unicast and L2 multicast
  - Dependency: Keying material for TLS authentication / confidentiality
  - This is an IETF standardization requirement (security). Of course GRASP could equally run solely over TCP
  - Requires just pre-existing L3 unicast reachability between only L3 reachable peers
    - Aka: not sufficient to use GRASP to e.g.: autoconfigure routing protocols to establish L3 routing

- Add GRASP forwarding agent:
  - Adds ability to do GRASP discovery across L3

- Add ANIMA “Autonomic Control Plane” (ACP)
  - ACP: Hop-by-hop automatically built “virtual out-of-band-management network (VRF)”
  - Comes with GRASP forwarding agent and automatic L3 unicast connectivity (prior to any routing config)

- Add ANIMA Bootstrap Remote Key Infra (BRSKI)
  - Depends on ACP
  - Provides “zero-touch” bootstrap of keying material for all nodes/routers in a domain.
GRASP Prototype

- A Python 3 implementation of GRASP as a module `grasp.py`
- About 2400 lines of code
- A test suite to exercise as many code paths as possible
- Various toy ASAs to test "real" operation across the network
  - bank/client negotiation
  - model of secure bootstrap process
  - model of IPv6 prefix management
  - bulk transfer using GRASP
- Some documentation
Summary, outlook, Questions?

• GRASP ready to use
  • ANIMA WG now working on API, ASA usage guidelines and some key ASA functions
  • Some existing ASA examples (reference one: draft-ietf-anima-prefix-management)

• If you want to automate services, think about defining this as ASA with GRASP
  • ANIMA WG happy to help
    (and dependent on which WG has best use-case expertise also be home for ASA docs)

Self driving
Networks

without an SDN
Controller called “Mom”
Backup Slides
Autonomic Network / ASA
Reference Model – High Level View

Autonomic Networking Infrastructure:
GRASP, Bootstrap, ACP, Naming, Addressing, Discovery

Network with autonomic functions

ASAs deployed as needed
Base infra: Every node must support

ASA
ASA
ASA
ASA
ASA

ASA
ASA
ASA
ASA
ASA

ASA
ASA
ASA
ASA
ASA

ASA
ASA
ASA
ASA
ASA

ASA
ASA
ASA
ASA
ASA

Domain ID
Pre-set ID
Domain ID
Pre-set ID
Domain ID
Pre-set ID
Domain ID
Pre-set ID
Domain ID
Pre-set ID

* imminent draft

draft-ietf-anima-asa-guidelines *
draft-ietf-anima-bootstrapping-keyinfra
draft-ietf-anima-grasp-api
draft-ietf-anima-autonomic-control-plane
draft-ietf-anima-bootstrapping-keyinfra

November 2020
GRASP overview
References...

• RFC 7575
• RFC 7576
• https://datatracker.ietf.org/wg/anima/documents/
• https://github.com/becarpenter/graspy