constrained ACP (cACP)
Initial thoughts

IANIMA WG, IETF119 Brisbane, v1.0

ENODRAFTYET

Toerless Eckert, Futurewei USA (tte@cs.fau.de)

ACP Reminder / Challenges

ACP: RFC8994

- Automatic, hop-by-hop (IPsec, BRSKI certificate) secured IPv6 connectivitiy across domain
- Line-rate (HW) forwarding possible, 20,000 or more sized networks supported (RPL routing)
- Enables secure connectivity for ANY (IPv6 capable) management/control plane traffic
  - SNMP, Netconf, SSH, HTTP(S), DNS, NTP, Diameter/Radius/…

Challenges:

- Vendor implementation may encounter various challenges based on their router/switch SW-infra.
- Plan for Linux open source reference implementation, but also mplementation complexities.
- Need separate routing table “VRF” (TBD details if linux has everthing)
- Need IPsec kernel support for IPsec tunnels across link-local-scope-addresses (requires kernel feature)
  - May propose a workaround for this (e.g.: use ACP ULA derived addresses for ACP secure channels)
- Complexities of RPL ?!

Lightweight ACP ?

Vision ?!

ACP code that can be implemented purely as application code without / minimal OS dependencies

Lightweight ACP == constrained ACP

Forwarding across an application-level only ACP will be CPU only. Performance limited

Gigabit Ethernet switches may have CPU throughput <= 1 Mbps

A controller may have configs of eg.: 1MByte of data – and want to send to 100 nodes

1MByte each to 100 receivers over 1 Mbps = 800 seconds CPU on this switch full (spanning tree will fail)

Need to manage/limit CPU consumption

Equal/Worse: remote router may not operate correctly and need Firmware download via ACP. Single Node easily 100MByte (big router). Download crash diagnostic from router etc. Pp..

Practical use-case example: New installed firmware needs new license to run existing features. Need to load other special/downgrade …

But: Having SOME ACP option much better than not to have anything! And drive to remote location to fix.

But let’s call it what it only can be: “constrained” ACP

constrained ACP (cACP) functionality ?

cACP may resort for end-to-end traffic to (HW accelerated) Data-Plane (automatically).

Network devices may not have connectivity in Enterprises. Think user-segmentation. But with many segmentation designs, “central nodes” (controller etc.) are reachable from all segments.Vision ?!

What is most generic application-level hop-by-hop forwarding we can do towards any remote cACP network device ?

GRASP – part of original design (during WG forming), but eliminated from GRASP in favor of IPv6

GRASP alone would not allow to transport core management protocols: netconf, ssh, radius/diameter, …

UDP – with a slim new “routing header” (similar to our cBRSKI proxy header)

Limitations: when packet is decapsulated on destination, UDP receiver App MUST NOT care about sender address/port – this will be local proxy function, not original sender.

Aka: only “proxy capable” management connections possible. Ideally explicit Sender/Receiver app support

TCP – difficult to do hop-by-hop proxying.

HTTP(s) – yes (because destination in HTTP header). But Netconf does not use HTTP.
Would force to use RESTCONF

We could start with

http(s) hop-by-hop proxying

UDP proxying with header (and shim library in applications)

Both is similar to MASC except for multipole hops, application routing etc.. But likely can use MASC knowledge.

constrained ACP (cACP) further thoughts/questions

Hop-by-hop secure channels – mutual trust with cACP certificates

- dTLS for UDP traffic
  - Ultimately run QUIC over this UDP/dTLS connectivity
- TLS for HTTP/HTTPs traffic

Routing protocol

- We do not need to support 20,000 nodes: Lets assume individual cACP “subdomains” e.g.: Enterprise site is 100 nodes. And connects via Internet to controllers (with cACP).
- What would be most lightweight/recommended routing protocol ? (What does e.g.: Thread use ?)
  - How about selecting based on lines-of-code open source Linux code to pick for reference ?

Addressing

- IMHO: will avoid unnecessary re-inventing of many details if we just stay with IPv6 addressing.
  - Like RFC8994 ACP this does not mean that rest of the network needs to support IPv6.
  - Aka: simply include ingress/egress IPv4/IPv6 proxying into our forwarding.
  - Ethernet addressing: Does only add new complexity. And every node of interest already has IPv4 and/or IPv6 host stack.
  - IPv6 cACP address can solely be inside of cACP application. No need for operating system to know about it.

cACP GRASP:

- No changes ?

Summary: Operator view

Limited size application acting as a secure, hop-by-hop proxy for management traffic

- Primarily for when it is needed. Limited peformance throughput!

Uses hop-by-hop IPv4 or IPv6 secure tunneling (v4 vs v6 depending on what devices support)

- Does not require to add IPv6 if not already available on device
- Does not require routing on devices (aka: easy to support on L2-only switches)

Management via QUIC/HTTP/HTTPS (e.g.: RESTCONF) just need to configure local cACP app as proxy

- Likely standard feature available in management systems / apps

TCP/UDP management traffic need to go through proxies – need to figure out best available proxy mechanism.

Shim socket library for applications to automatically select cACP (slow) or data-plane (fast) connection to destination device (data-plane of course not necessarily possible).

Questions ?

- Please let me know if you are interested in this
- Take it to the ANIMA mailing list to discuss!