Autonomic Control Plane update

draft-ietf-anima-autonomic-control-plane-02

Michael Behringer, Toerless Eckert,

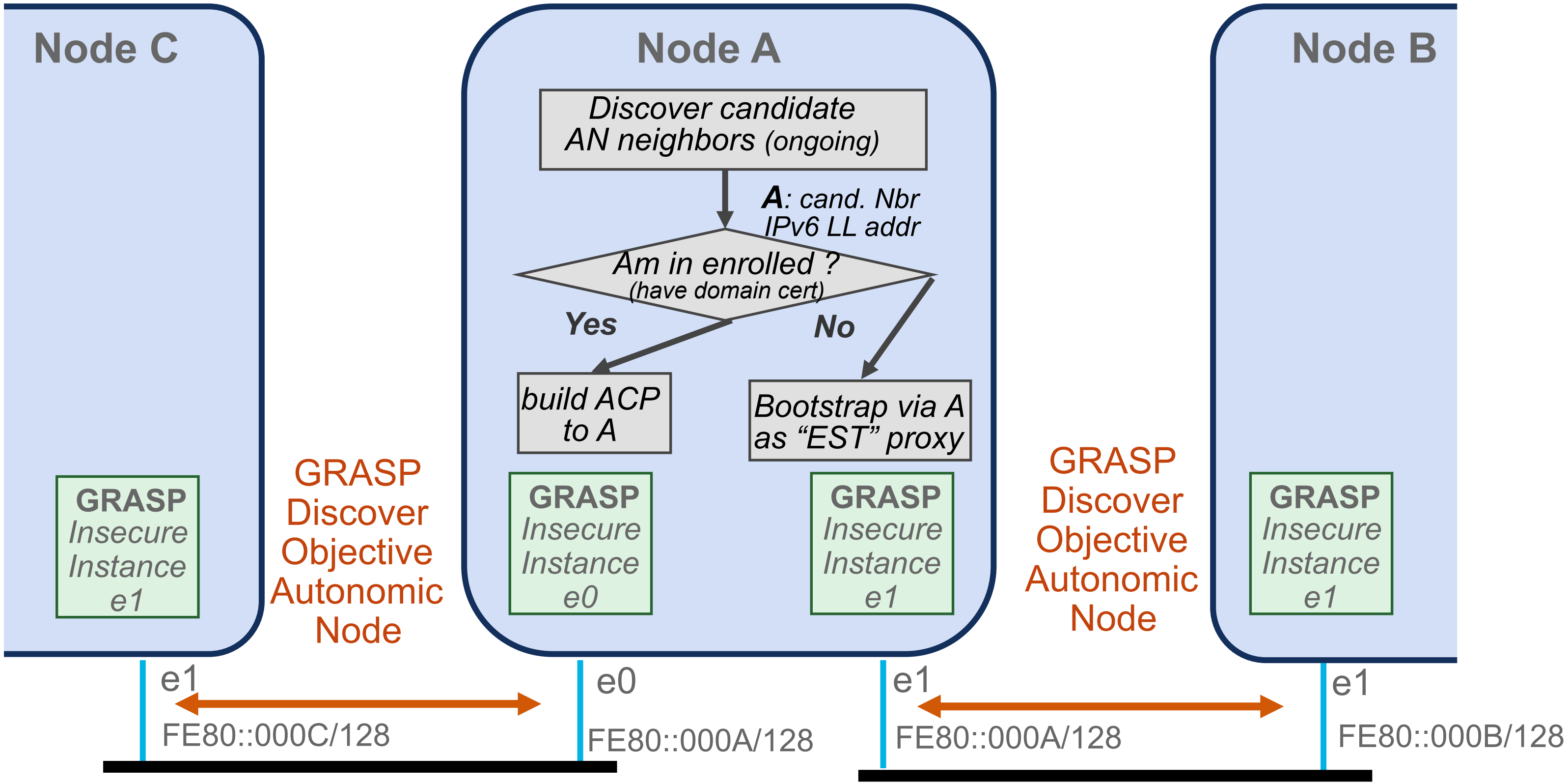
Balaji B.L., Steinthor Bjarnason

IETF 95, Apr 4. 2016, Buenos Aires

Changes since -01

- Draft includes addressing for ACP
  - Superceeds/retires draft-behringer-anima-autonomic-addressing-02
  - No changes
  - Details discussed during reference draft slot.
- Introduce specification how to bring up ACP channels
  - Leveraging GRASP
- Slides also contain explanation how Bootstrap uses GRASP
  - First use of GRASP inside ACP
  - TBD: detail into next rev of Bootstrap draft.

Insecure GRASP instances to find neighbors



Node C
Node A
Node B
Discover candidate AN neighbors (ongoing)
A: cand. Nbr IPv6 LL addr
Am in enrolled ?
(have domain cert)
Yes
No
build ACP to A
Bootstrap via A as "EST" proxy
GRASP Insecure Instance e1
GRASP Discover Objective Autonomic Node
GRASP Insecure Instance e0
GRASP Insecure Instance e1
GRASP Discover Objective Autonomic Node
GRASP Insecure Instance e1
e1
FE80::000C/128
e0
FE80::000A/128
e1
FE80::000A/128
e1
FE80::000B/128

Insecure GRASP instances, details

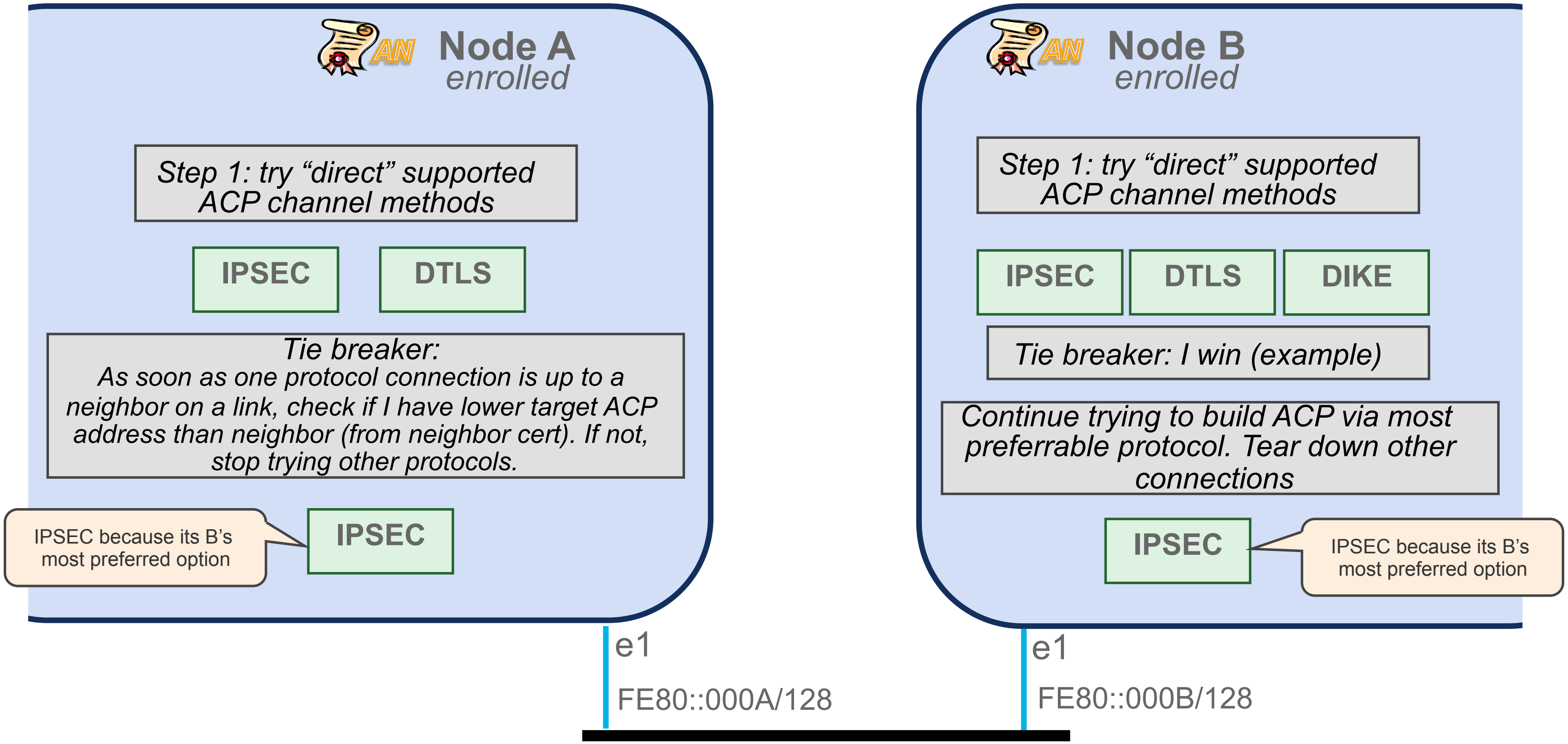
- “Instance” per physcial interface/port
  - Just GRASP via UDP, LL multicast and Ucast responses
  - No passing on of discover information across interfaces
  - No caching of learned information
  - IP TTL = 255 ?!, GARP Loop count = 1
  - Very small subset of GRASP used here.
- Do not DISCOVER/USE anything but neighbor IPv6 Link Local Address
  - Anything else is a candidate attack vector: These instances of GRASP are insecure
- But: May want to include more for diagnostics (similar to CDP/LLDP/…)
  - MUST NOT USE THIS in following ACP/Bootstrap protocols.
  - Just for (wiresharking) operator trying to diagnose possible issues.
    Even on neighboring AN device (so operator can “insecurely” diagnose problems, when enrollment/ACP build does not work).
  - The better we feel about our security to work, the less we need security by obscurity:
    - What Bootstrap options / ACP channel options we support, whether we are enrolled or not,…

Why GRASP for discovery ?

Instead of eg: mDNS, CDP/LLDP,…

- Lightweight/simple:
  - mDNS requires at least 4 type of RR for a single DNS-SD service discovery
- Isolated:
  - I do not want to see “Autonomic Device” in my user “service” browser
  - I do not want mDNS proxies to forward my AN neighbor discovery packets to other LANs.
  - I do not want to see my AN neighbor service to pop up in a poor central DNS server when mDNS<->uCAST DNS is used.
- L2 capable
  - On an AN L2 switch, I want to intercept/terminate GRASP packet (insecured on links) so I can build just ACP connections between the L2 switch and connected routers – as opposed to full mesh between all routers in the L2 LAN.
  - If I would do this with mDNS I had a lot of work with other uses of mDNS at my hand.
- Can not use CDP/LLDP because they would not go across non-AN L2 switches.

Building ACP secure channels (both sides enrolled):



AN Node A
enrolled

Step 1: try “direct” supported ACP channel methods

IPSEC

DTLS

Tie breaker:
As soon as one protocol connection is up to a neighbor on a link, check if I have lower target ACP address than neighbor (from neighbor cert). If not, stop trying other protocols.

IPSEC

IPSEC because its B’s most preferred option

AN Node B
enrolled

Step 1: try “direct” supported ACP channel methods

IPSEC

DTLS

DIKE

Tie breaker: I win (example)

Continue trying to build ACP via most preferrable protocol. Tear down other connections

IPSEC

IPSEC because its B’s most preferred option

e1

FE80::000A/128

e1

FE80::000B/128

Why: Direct building of ACP channels without “negotiation protocol”

- Not all AN devices need to be able to talk to each other
  - Acceptable, if not beneficial to only have to implement protocols required by device
  - No need for network wide “Mandatory To Implement (MTI)” protocol
- Example:
  - Enterprise Campus: Lot of Switching gear, MacSec would be ideal.
    - Low end switches may only want to support MacSec, nothing else.
    - Would be great option to have, but need to solve some MacSec specific issues first, so just a theoretical option now.
  - IoT gear on enterprise edge: only want to support dTLS (memory constraints)
    - Only “Gateway” devices would need to support both MacSec and dTLS.
- Any “negotiation” protocol might be too heavy or not easily acceptable across all possible market segments as well.
  - Negotiation protocol secure == almost same overhead as secure channel protocol (?!).
  - Therefore have the option for security negotiation protocol free negotiation (Step 1).

Expectation against "direct" secure channel protocols

- **Mutual secure authentication with LDevID (AN Domain Certificate)**
  - Peer certificate must be signed by same CA
  - Peer certificate must be valid
  - Domain Name in peer certificate must match domain name in own certificate
    - Domain Name is encoded in the "OU" field in the format of a domain name

      Example: Subject: OU=**example.com**+serialNumber=XXX-YYY-ZZZ+…

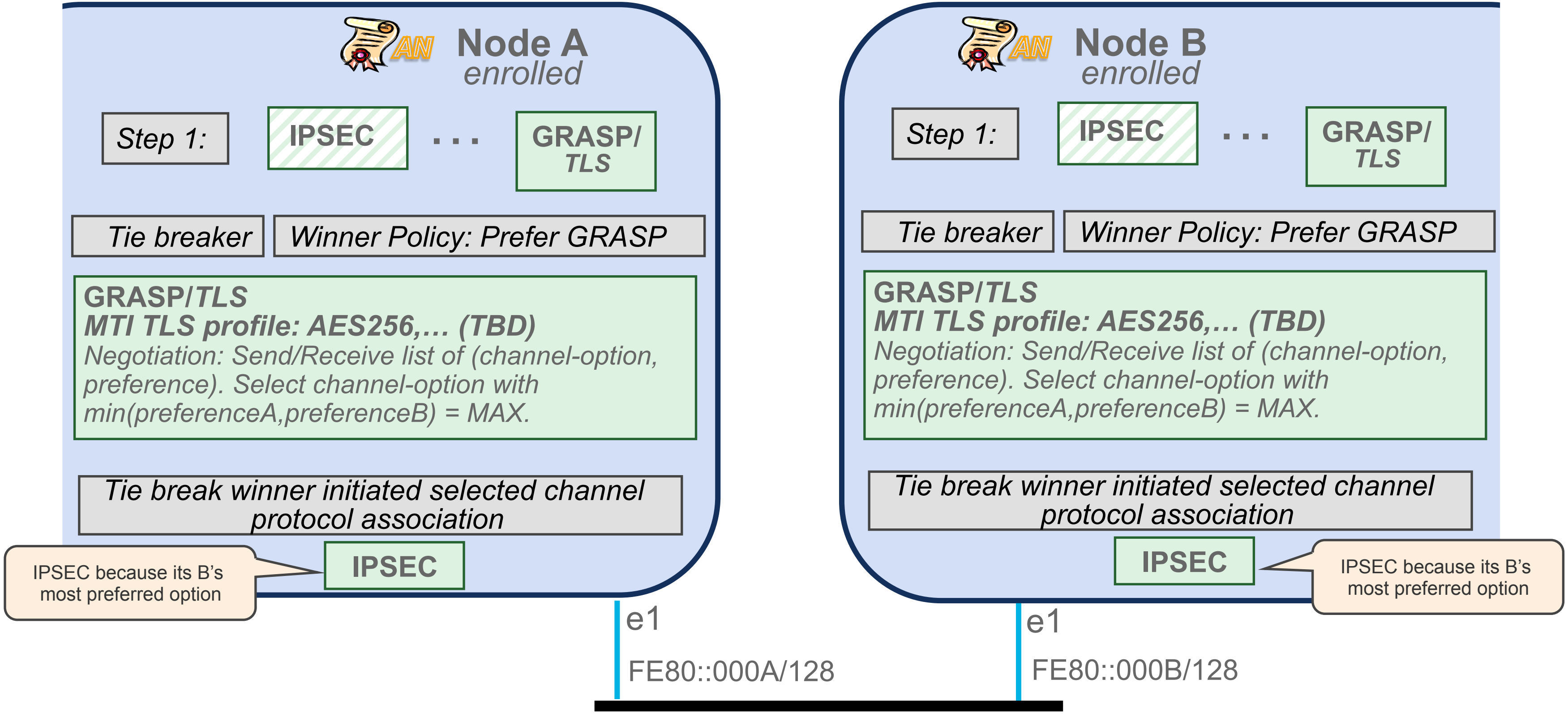
      – example.com is the AN domain name

- Note: these rules are not inclusive of more complex multi-domain trust options as written out in other part of the draft.
  - Eg: subdomain1.example.com may trust subdomain2.example.com.
  - If we do not define this into the basic ACP rules, it can be added later via Intent options.

- Note: Domain name also needs addressing element(s) to defining the ACP address of the device. This is covered in another part of the draft.

Building ACP secure channels (both sides enrolled)

With GRASP negotiation



AN **Node A** *enrolled*

Step 1: **IPSEC** . . . **GRASP/ *TLS***

Tie breaker *Winner Policy: Prefer GRASP*

GRASP/*TLS*
MTI TLS profile: AES256,… (TBD)
Negotiation: Send/Receive list of (channel-option, preference). Select channel-option with min(preferenceA,preferenceB) = MAX.

Tie break winner initiated selected channel protocol association

IPSEC

IPSEC because its B's most preferred option

AN **Node B** *enrolled*

Step 1: **IPSEC** . . . **GRASP/ *TLS***

Tie breaker *Winner Policy: Prefer GRASP*

GRASP/*TLS*
MTI TLS profile: AES256,… (TBD)
Negotiation: Send/Receive list of (channel-option, preference). Select channel-option with min(preferenceA,preferenceB) = MAX.

Tie break winner initiated selected channel protocol association

IPSEC

IPSEC because its B's most preferred option

Why GRASP NEGOTATION of channel protocol ?

- 0: Not necessary/beneficial when there is not enough to negotiate
  - Eg If platforms only support one option anyhow (eg: lightweight dTLS only platforms).
- But
  - 1: Tie-break-winner decides is not a generically good option:
  - 2: Can not negotiate between variants of secure channel options without separate negotiation protocol
  - 3: (Extension) can use secure channel protocols without their own mutual domain certificate based authentication.

Why GRASP NEGOTATION of channel protocol ?

- 1: Tie-break-winner decides is not a generically good option:
  - Consider negotiating for best achievable performance.
    Preference = achievable throughput.
  - TBD: Negotiation rules not well finalized (perf based only in slides, not -02 draft).
  - BrianQ: can we make it extensible ? Eg: basic negotation eg: preference based, but if both sides can support other negotiations, those will be used. ?!
- 2: Can not negotiate between variants of secure channel options without separate negotiation protocol:
  - Example: Negotiate either of:
    - IPv6 (ACP) packets (natively) inside IPsec tunnel association
    - IPv6 (ACP) packets inside GRE in IPsec transport association
    - *This is practically impossible on many products, because the implementations require the encap stack to be set up before performing authentication.*
    - *This is also theoretically hard because the encap stack would have to be guessed from the IKEv2 negotiation of traffic profile.*
    - One option may be preferred because of performance, or only one of these options may be supported.

Why GRASP NEGOTATION of channel protocol ?

- 3: (Extension) can use secure channel protocols without their own mutual domain certificate based authentication.
  - Letting secure channel protocol repeat domain certificate authentication when this was already done as part of the TLS connection for GRASP is duplication of effort.
    - Other security scheme also are based on leveraging the symmetric master key derived from TLS for further security associations.
  - This would be an extension, because as the result of it, the selection would not only be a (eg: numeric) secure channel protocol to start, but also additional parameters (eg: symmetric master key to use for the security association).
- Q: What is the minimum amount of GRASP negotiation we would feel happy with for the ACP RFC
  - … that would keep the door open for these options without creating the work to resolve all those details for the first ACP RFC ?
  - If we can not come to fast conclusions, then the option to use GRASP/TLS for negotiation of ACP secure channel could be moved out into a separate document ?!

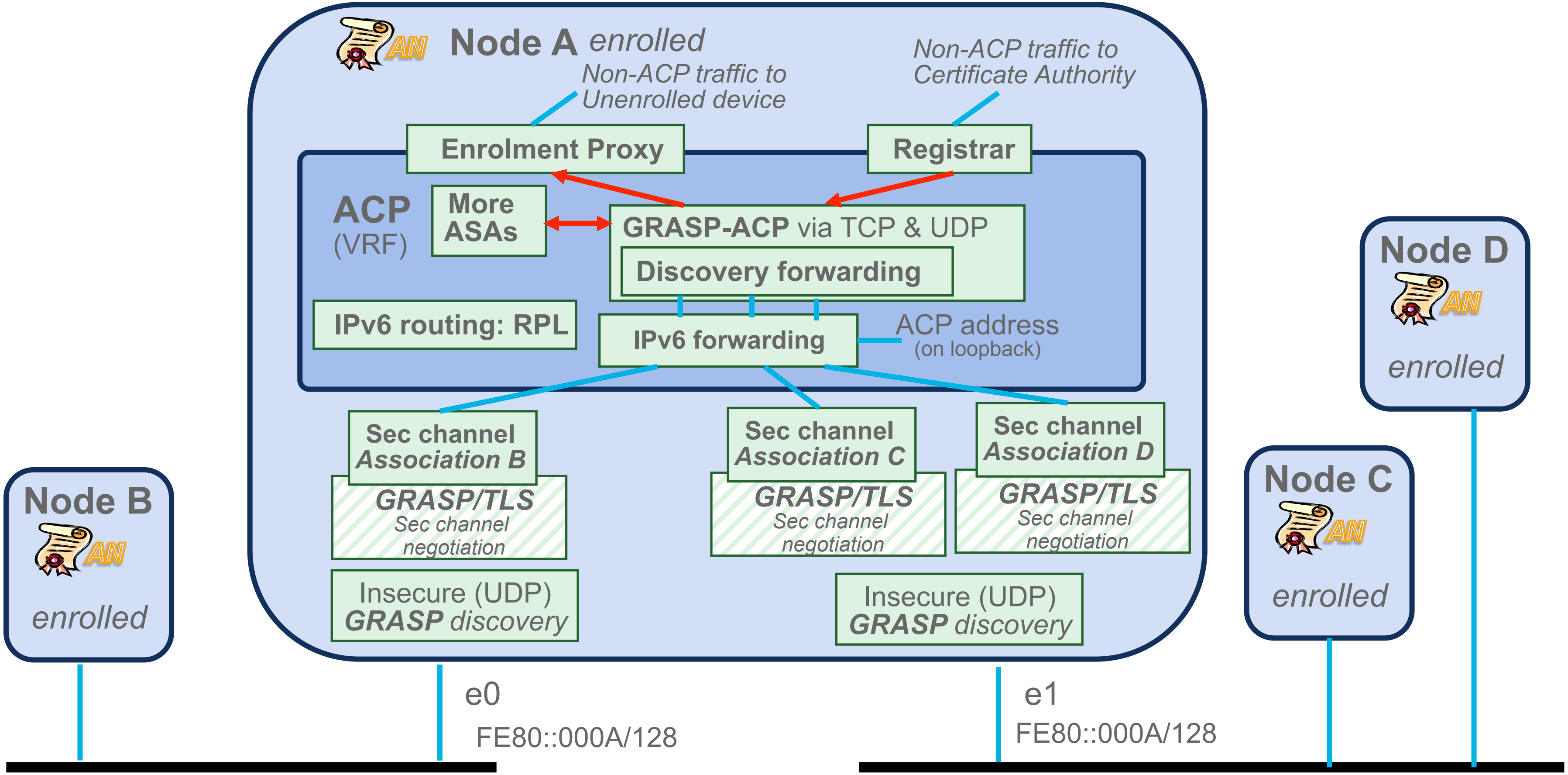
Why GRASP NEGOTIATION of channel protocol ?

- Is there even a good contender for this use of GRASP ?
- Aka: this looks like a perfect option to explore GRASP.

Details

- GRASP/TLS connection should be kept alive even when secure channel protocol is running.
- Profiles:
  - Which channel options MAY/SHOULD/MUST be implemented
  - Propose at least one “GENERIC” autonomic device profile

GRASP inside (and outside) ACP



GRASP inside ACP

- GRASP inside ACP:
  - ASAs to find each other and then to negotiate with each other.
  - ASA can be multiple hops away. GRASP needs to forward discovery messages.
- For Autonomic Infrastructure
  - Only need to consider registrar / enrolment proxy “ASA” – aka: Bootstrap.
    - Specifiy in Bootstrap draft.
  - All other requirements against GRASP inside ACP are for future “more” ASAs.
    - Can refine upon working on ASA after recharter
- Target for ACP draft:
  - GRASP inside ACP should use UDP only for discovery (link-local)
  - TCP for all other GRASP-ACP messages
    - No TLS required because GRASP-ACP runs only via secure channels
    - TCP preferred over UDP so we can avoid having to bother about fragmentation and flow control.

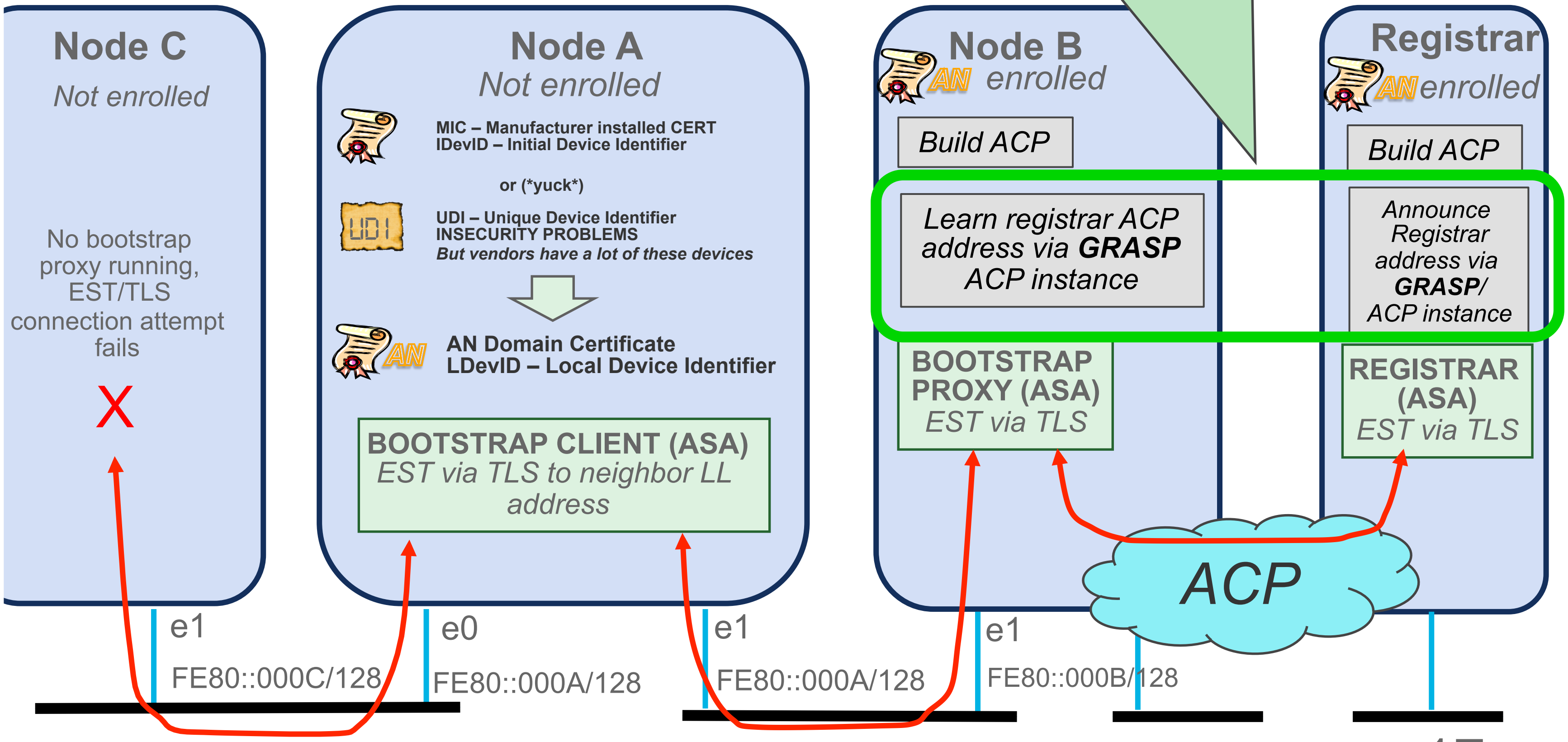
Bootstrap and GRASP

Depends on ACP being built between Proxy and Registrar

TBD: Specify GRASP “registry” For “AN Registrar” objective.

NOT PART OF ACP DRAFT

INFORMATIONAL FOR WG MEETING



Node C

Not enrolled

No bootstrap proxy running, EST/TLS connection attempt fails

X

Node A

Not enrolled

MIC – Manufacturer installed CERT
IDevID – Initial Device Identifier

or (*yuck*)

UDI – Unique Device Identifier
INSECURITY PROBLEMS
But vendors have a lot of these devices

AN Domain Certificate
LDevID – Local Device Identifier

BOOTSTRAP CLIENT (ASA)
EST via TLS to neighbor LL address

Node B

AN enrolled

Build ACP

*Learn registrar ACP address via **GRASP** ACP instance*

BOOTSTRAP PROXY (ASA)
EST via TLS

Registrar

AN enrolled

Build ACP

*Announce Registrar address via **GRASP**/ ACP instance*

REGISTRAR (ASA)
EST via TLS

ACP

e1 FE80::000C/128

e0 FE80::000A/128

e1 FE80::000A/128

e1 FE80::000B/128

Open Issues – resolve until Berlin.

- Finalize Specification:
  - Detailling GRASP message payload formats for the three uses of GRASP
  - Details of RPL parameters
    - Proposal details from Pascal Thubert exist
  - Complete set of requirements against Cert content
    - Currently: only defining “domain-name” in ACP draft
    - But need element(s) for ACP address
    - Overall responsibility -> Bootstrap document ???
  - Adressing:
    - Target ACP RFC will only describe use of Zone 0
    - Additional zone use (eg: ACP with MPLS/VPN zoning) for later documents.

Open Issues – resolve until Berlin.

- Architecture items
  - Draft contains useful extensions (eg: authentication across domain).
  - Want to keep in document – but not clear how we would refine to implement
  - Will mark these sections as “architecture only”, full functional details for future documents.