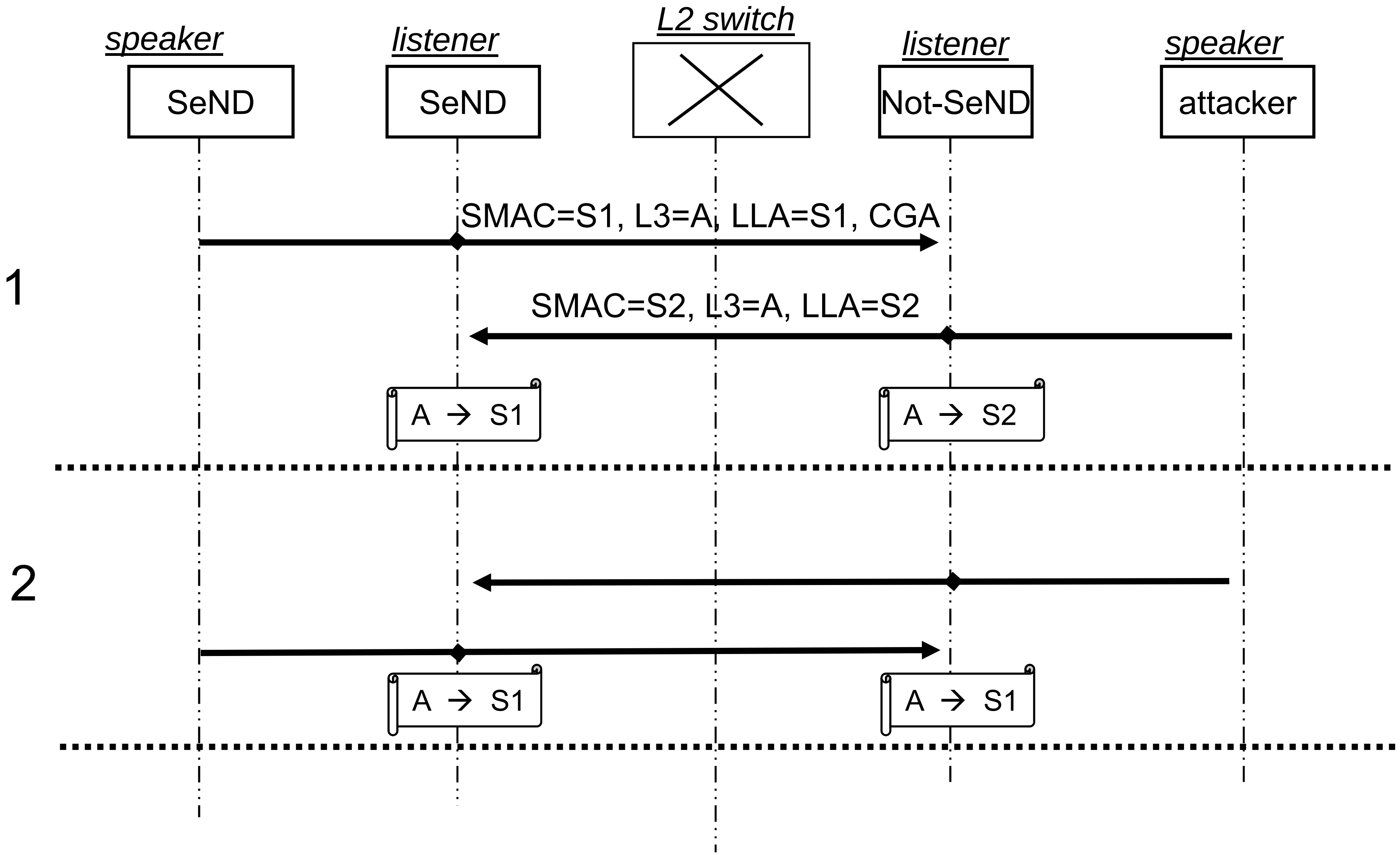
IETF-75

draft-levy-abegnoli-savi-plbt-00

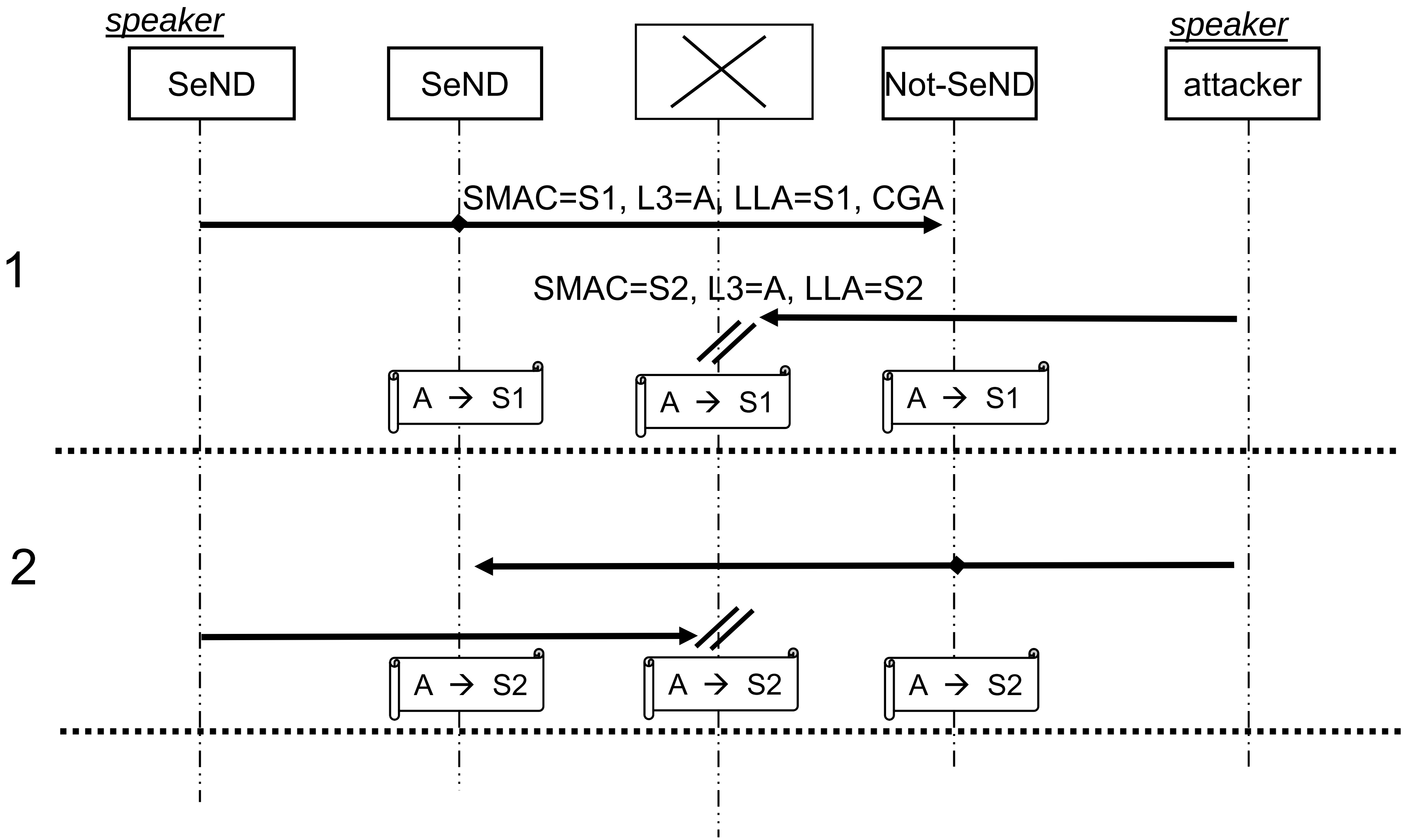
Goals

- Build and maintain a L3/L2 binding table on the L2 switch *as accurate as possible (definition of "accurate"?)*
- Focus on SLACC address assignment
- Separate control and data plane
  - Bindings established with control plane protocols
  - DAD and data packets can be used as "hints"
  - Simple rustic data plane decisions: bridge, drop, count, punt to cpu
- Analyses and possibly handle "hard" cases:
  - SLACC
  - Multiple switches, with mix of savi & non-savi
  - Mix of trusted and untrusted switch ports
  - Mix of cryptographic, static, dhcp, slacc addresses
  - NDP-LLA and SMAC can be different
  - Heterogeneous movement requirements

Bindings accuracy



Bindings accuracy



Approach

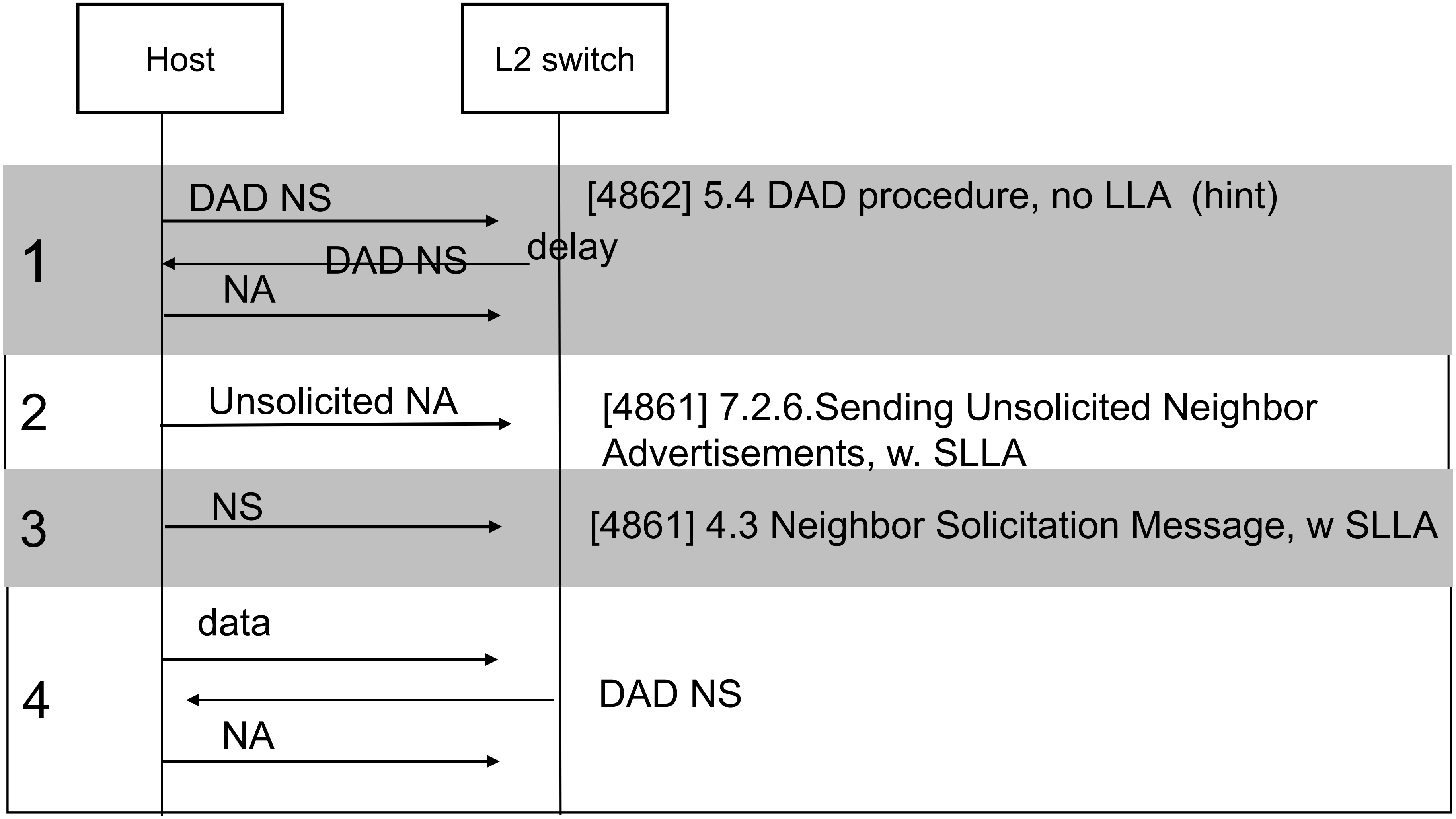
- Bindings are created “by L3 and above” information, not L2
- For SLACC, inspect all [NDP] messages
- Relies heavily on configuration
- Collision algorithm: “Prefer” some bindings over others (not LCFS, not FCFS)
- Maintain state accuracy for better movement handling
- CGA friendly
- Movement scenarios are a function of all the above

Learning process

- Binding learnt by snooping all NDP traffic:
  - NS, NA, RS, RA [REDIR, CPS, CPA]
- L3 bound to NDP-LLA. SMAC used as a hint.
- Preference algorithm for collisions (First-come is "one" element of it)
- Entry state tracked by the switch
- End-devices optionally NUD'ed for maintaining binding table entries in most current state

Inspect all NDP messages …

Three (+1) chances to learn the binding



Host

L2 switch

1

DAD NS

[4862] 5.4 DAD procedure, no LLA (hint)

DAD NS

delay

NA

2

Unsolicited NA

[4861] 7.2.6.Sending Unsolicited Neighbor Advertisements, w. SLLA

3

NS

[4861] 4.3 Neighbor Solicitation Message, w SLLA

4

data

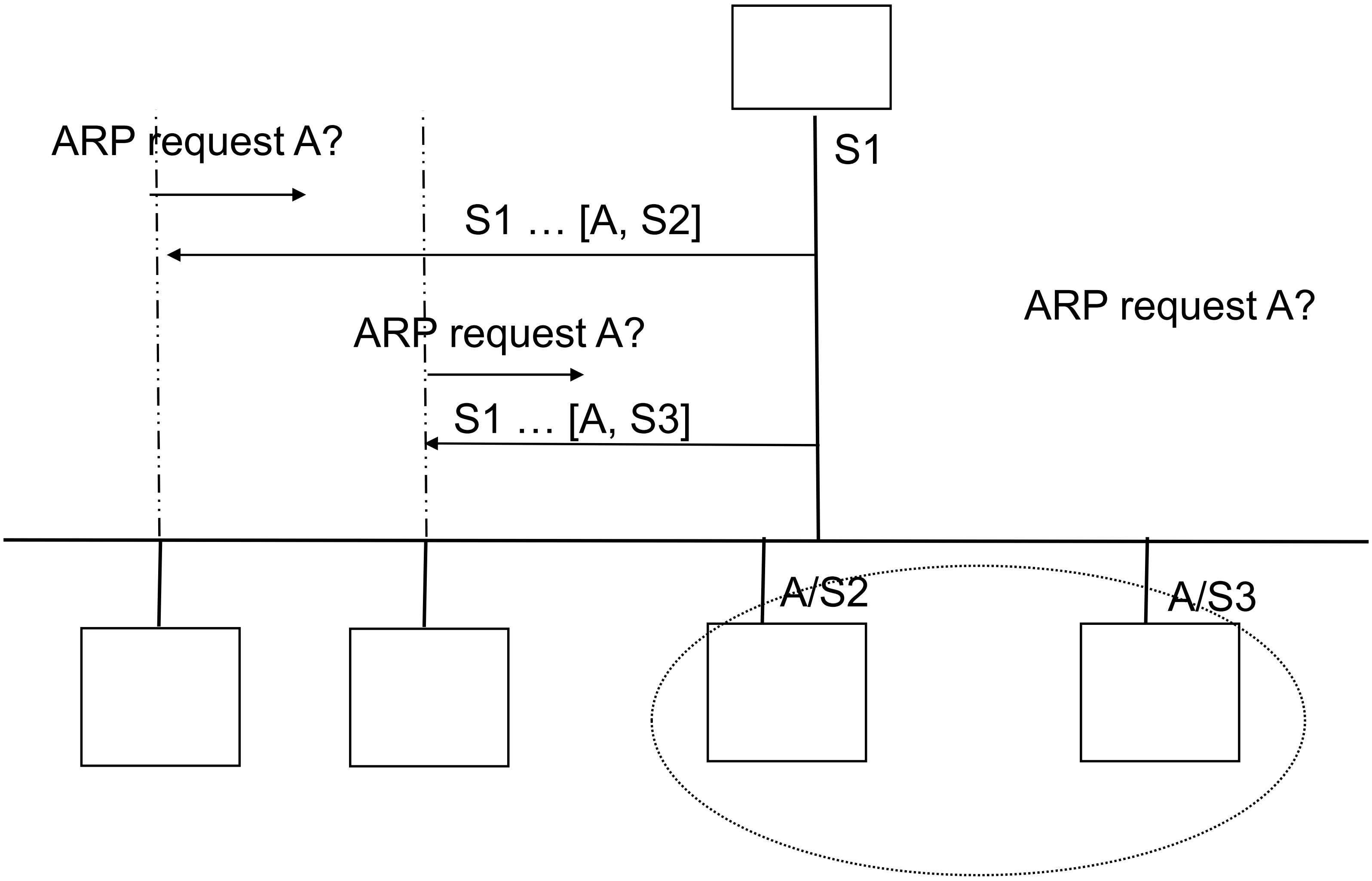
DAD NS

NA

Inspect all NDP messages …
L3 bound to NDP-LLA

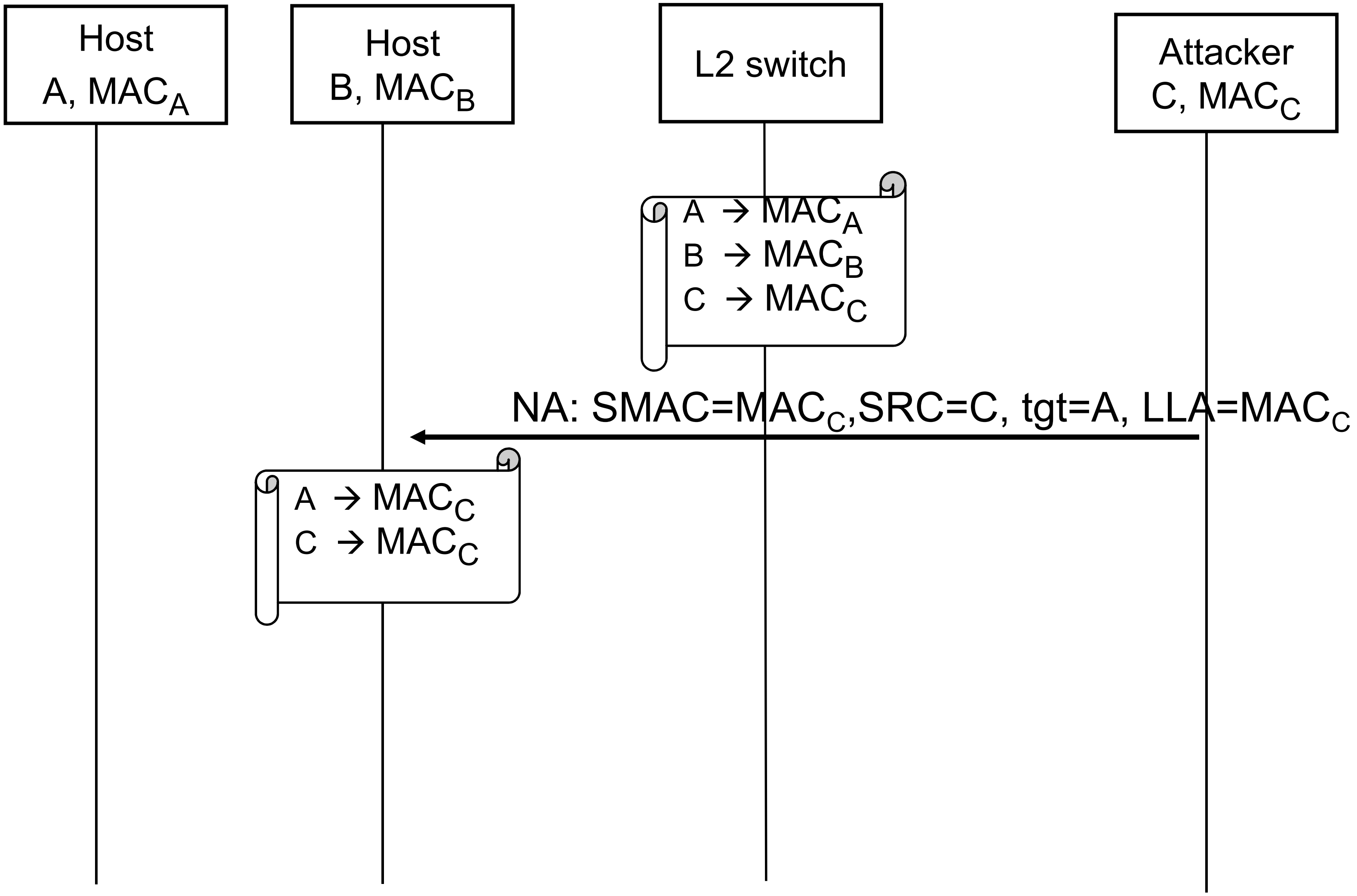
- Currently, there is nothing that forces the SMAC and the LLA to be the same
- Some “border-line” scenarios can exploit this:
  - LLA server (NetPet)
  - Multiple SMAC, multiple LLA on same interface
  - ND-proxy
- DAD NS is used as a hint, to “lock” a new entry in the binding table. But LLA is the value bound to the IP@.
- Data packet could be used as a well as a hint
- Checking SMAC==LLA is a policy matter
- In the absence of LLA within T0, switch sends DAD NS

Issue#1 with binding L3 to SMAC



Server farm
ARP disabled

Issue#2 with binding L3 to SMAC



Host
A, MAC_A

Host
B, MAC_B

L2 switch

Attacker
C, MAC_C

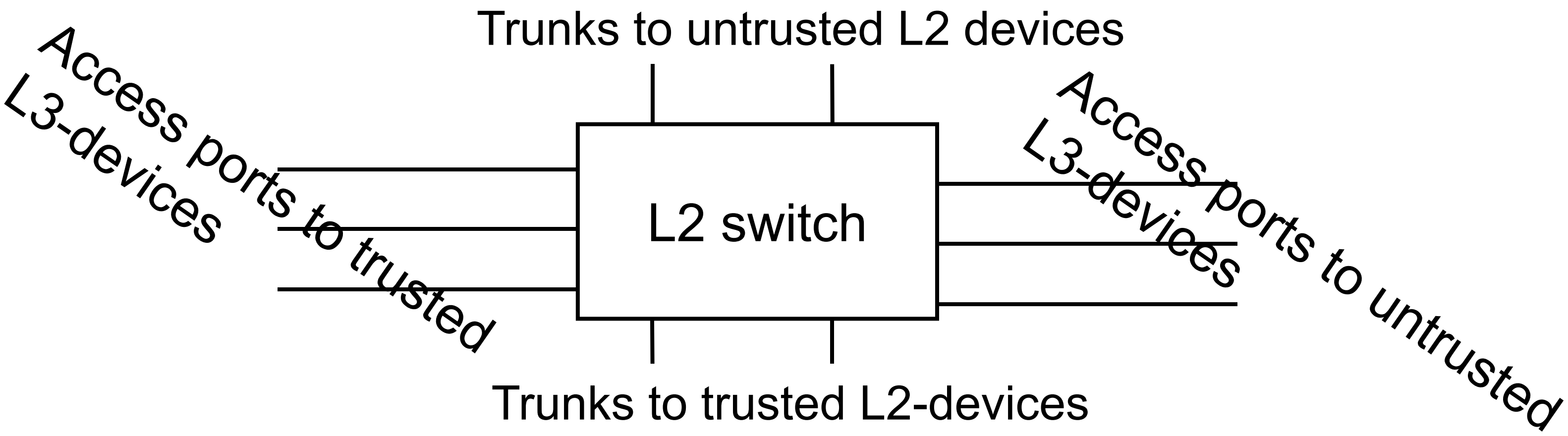
A → MAC_A
B → MAC_B
C → MAC_C

NA: SMAC=MAC_C,SRC=C, tgt=A, LLA=MAC_C

A → MAC_C
C → MAC_C

Switch port taxonomy

- L2 switch ports classified into:



Trunks to untrusted L2 devices

Access ports to trusted L3-devices

L2 switch

Access ports to untrusted L3-devices

Trunks to trusted L2-devices

- This classification is driven mostly by configuration
- Bindings evaluated based on:
  - where they are learn from
  - the credentials that they carry

- A lot of policies can be driven by this classification:
  - Binding entry replacement or update
  - movements
  - SAVI policy (drop, log, punt, rate limits, etc.)

Preference value

A. Defines preference “flags”

- A flag is either a property of the port from which the entry was learnt or a property of the binding itself
- Flag_n = 2 ^{n}
- Preflevel = ∑flags
- From less to most preferred, proposed “n” values are:

1. TRUNK_PORT: the entry was learnt from a trunk port (connected to another switch)
2. ACCESS_PORT: the entry was learnt from an access port (connected to a host)
3. LLA_MAC_MATCH: LLA (found in NDP option) and MAC (found at layer2) are identical
4. TRUSTED_PORT: The entry was learnt from a trusted port
5. TRUSTED_TRUNK: The entry was learnt from a trusted trunk
6. DHCP_ASSIGNED: the entry is assigned by DHCP
7. CGA_AUTHENTICATED: The entry is CGA authenticated
8. CERT_AUTHENTICATED: the entry is authenticated with a certificate
9. STATIC: this is a statically configured entry

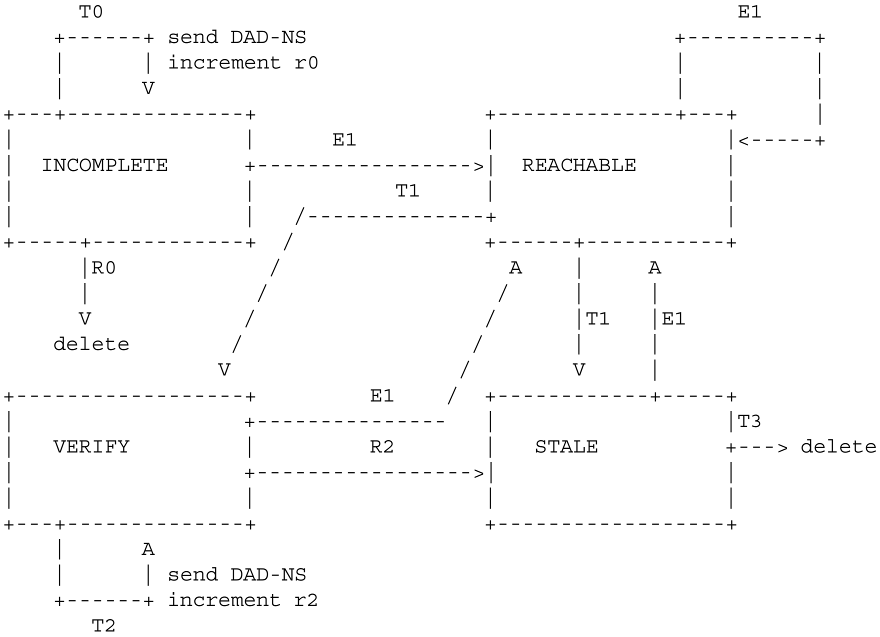
Preference algorithm

B. Define the rules (applied in this order). Updating an entry attribute is:

1. Allowed when the preflevel carried by the change is bigger than the preflevel stored in the entry.
2. Denied if the preflevel carried by the change is smaller than the preflevel stored in the entry
3. Allowed if preflevel >= TRUSTED_PORT
4. Denied for entry in state INCOMPLETE if the change is not associated with the port where this entry was first learnt from.
5. Allowed if the change is associated with a trusted port (and preflevel are equal)
6. Denied is the entry is in state REACHABLE or VERIFY (and preflevel are equal)
7. Allowed otherwise

State machine

- The state of an entry is part of the collision algorithm
- For instance, entries in REACHABLE are “locked” to their attributes (movement disallowed unless preflevel >= TRUSTED_PORT)



```
      T0                                                  E1
    +------+ send DAD-NS                             +----------+
    |      | increment r0                            |          |
    |      V                                         |          |
+---+--------------+                  +--------------+---+      |
|                  |      E1          |                  |<-----+
|  INCOMPLETE      +----------------->|  REACHABLE       |
|                  |           T1     |                  |
|                  |   /--------------+                  |
+-----+------------+  /               +------+-----------+
      |R0            /                  A    |     A
      |             /                  /     |     |
      V            /                  /      |T1   |E1
    delete        /                  /       |     |
                 V                  /        V     |
+------------------+          E1   /  +------------+-----+
|                  +---------------   |                  |T3
|   VERIFY         |          R2      |   STALE          +---> delete
|                  +----------------->|                  |
|                  |                  |                  |
+---+--------------+                  +------------------+
    |      A
    |      | send DAD-NS
    +------+ increment r2
       T2
```

Key: address/zoneid

Attributes: port, preflevel, owner, lla

Others: state