I2NSF Flow protection

Case #1

David Carrel
Brian Weis
7 Nov 2018

Current Case 2

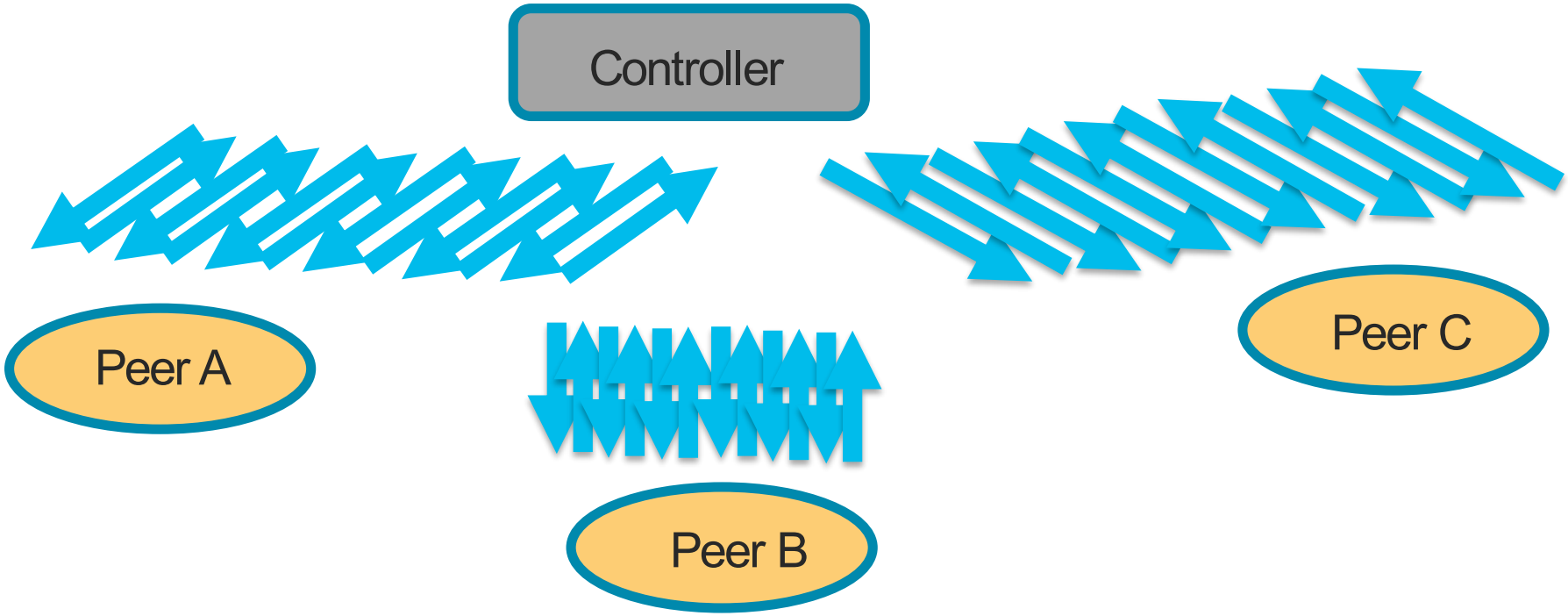
- For every combination of every pair of NSFs:
  1. Send Inbound SPIs to both NSFs
     - Wait for response from both NSFs
     - If 1 NSF is offline/busy, Controller resends or cleans up
  2. Send outbound SPIs to both NSFs.
     - Wait for responses
     - If 1 NSF is offline/busy, Controller resends or cleans up
  3. Send delete for old SPIs to both controllers
     - Wait for responses

This works, but doesn't scale.

- Creates 6 x N^2 messages for Controller and NSF per re-key
- Creates N^2 state machines for controller
- Multi-tenancy makes this even worse by N^2

- Gets even more complicated when NSFs are offline?
  - If NSF goes offline during re-key, Controller must clean up peer NSF SAs
  - If Controller goes offline, someone must clean up and resynchronize re-keying.

This works, but doesn't scale.



Other issues…

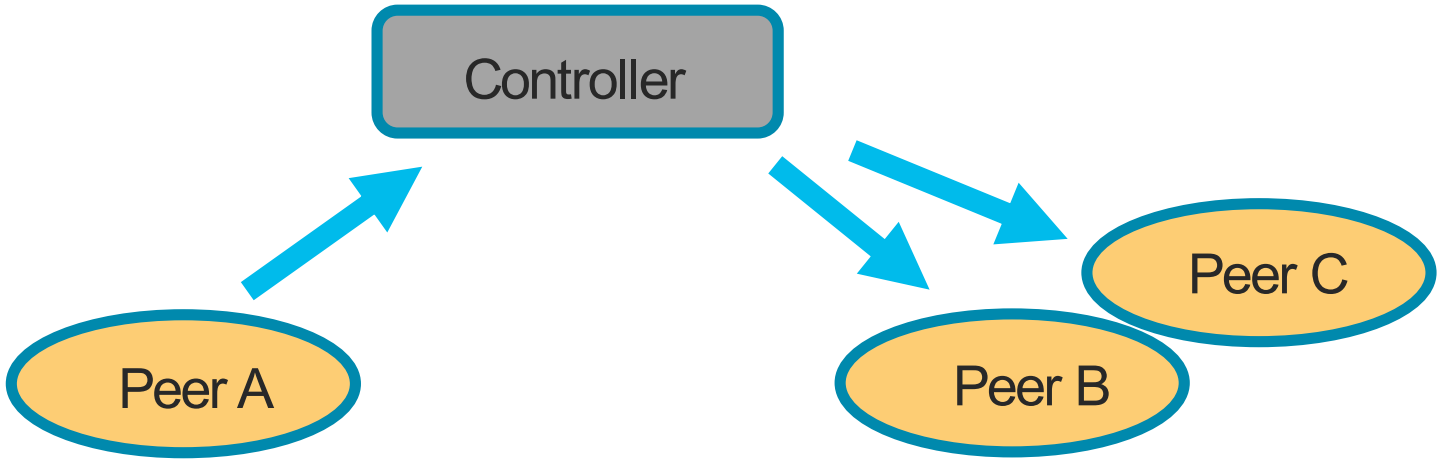
- Using multiple controllers becomes very complicated
- Controller knows ALL IPsec keys
- Case 1 and Case 2 need very different configuration models
  - **Case 1** follows typical RFC 4301
    - Controller sends SPD and PAD
    - NSF sends SAD to Controller with stats
  - **Case 2**
    - Controller sends SPD and SAD to NSF
    - NSF sends SAD to Controller with stats

Controller IKE

- Creates N messages for Controller (actually < N)
- Creates 1 messages for each NSF
- No state machines on Controller, N state machines on NSF
- Makes multiple controllers very easy – Loose synchronization
- Controller knows NO IPsec keys
- Configuration model handles SPD, PAD, and SAD similar to case 1
- Robust to NSF/Controller going offline or losing connectivity
  - No controller logic – Entire keying model is asynchronous!

Controller IKE

- Peer initiates re-key
- Loose synch provides losslessness & flexibility
- Remaining sync uses existing ESP data pkts
- Re-key of A's DH is completely async from re-key of B's DH




CISCO