Control Packets for RoE

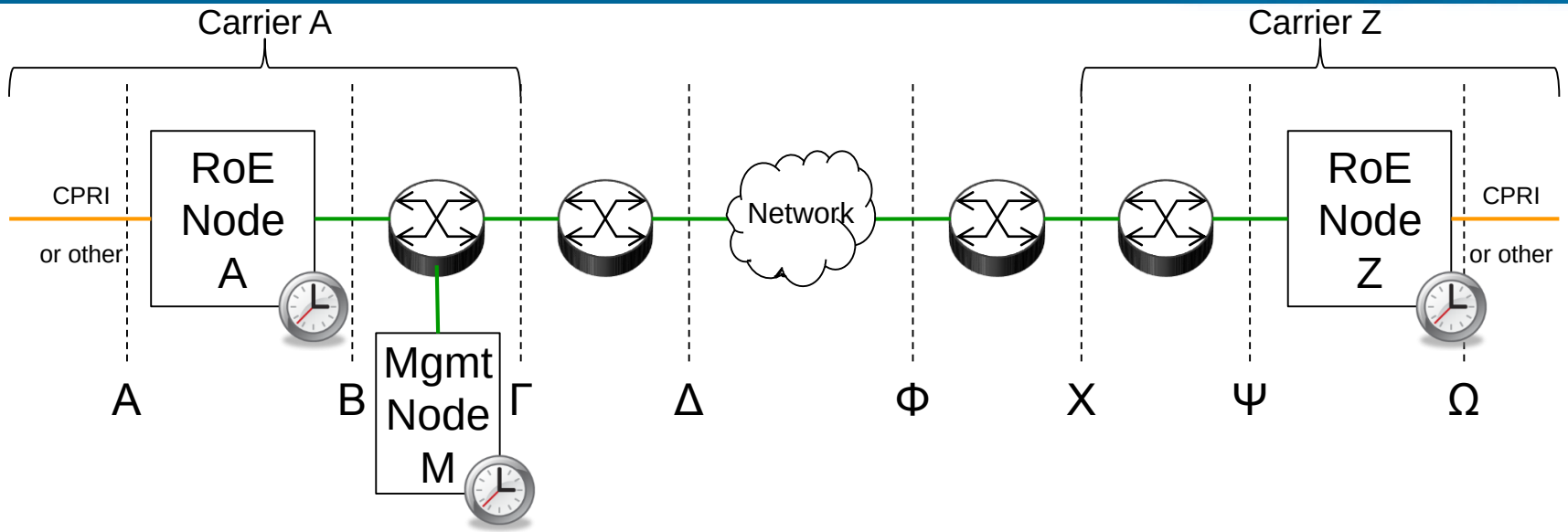
Kevin Bross

15 April 2016

Introduction

- A basic agreement on control packets is necessary to ensure multi-vendor interoperability of RoE
  - Philosophy: akin to setting IP address and subnet mask, not defining DHCP
- This presentation suggests a minimum set of control packets for RoE and management setup
  - Where does management node reside?
  - How to establish flows and flow parameters
  - Minimal data to report

Management Node M

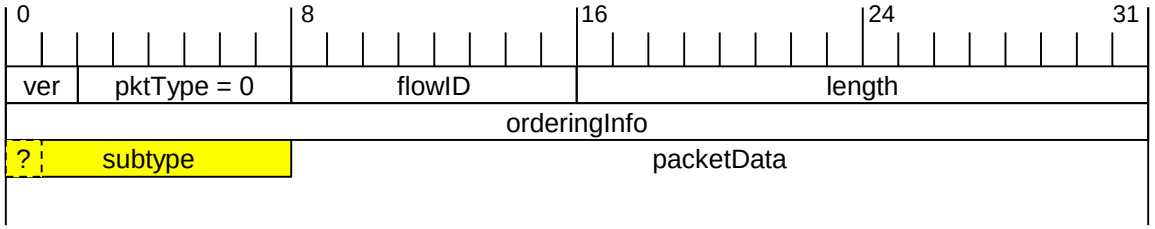


- Assume management is handled by a separate node M
  - Could be co-located with Node A or Node Z, but define architecture as separate for flexibility
  - Node M typically located in network of one carrier, but may be geographically remote

A = Alpha
B = Beta
Γ = Gamma
Δ = Delta
Φ = Phi
X = Chi
Ψ = Psi
Ω = Omega

Control Packets

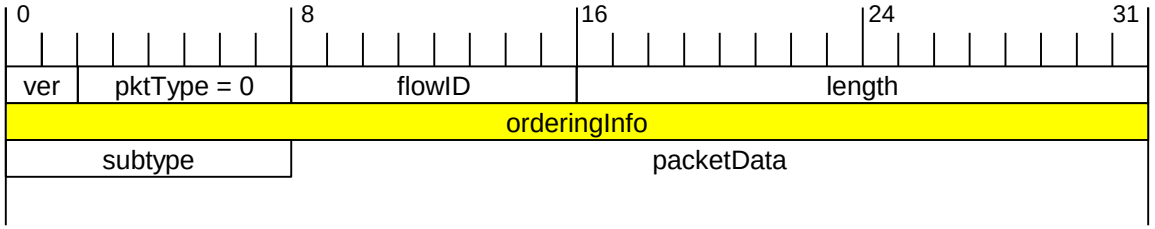
- Control packets follow the RoE header format (pktType = 0)
- All management packets have subtype



- Unlike data packets, all control packets expect a response
  - Retry packet if no response (and still valid)
    - Always respond, even if already responded to seq. #
  - Suggest MSb of subtype is set to indicate response (129 = response to subtype 1, etc.)

Missing Control Packets

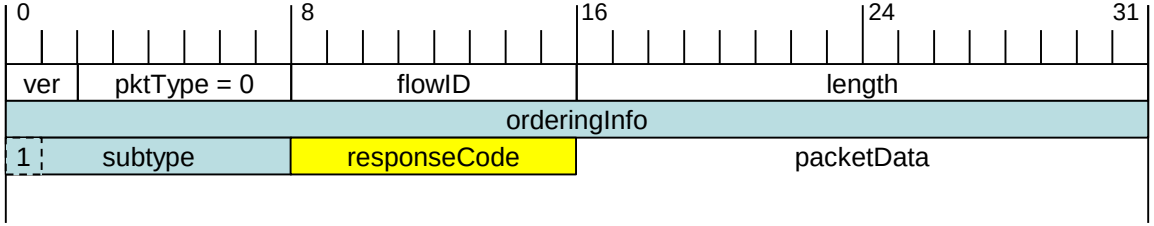
- The *orderingInfo* field for control packets shall be used to indicate a sequence #
  - A gap in sequence numbers indicates a missing control packet



  - Recommend increasing sequence number by 1 for all control packets
    - §4.2.5.2: p=32, q=0
  - Re-use the sequence number when re-trying a command? Set a high-order bit for retries?

Response Codes

- Response packets include 8-bit code indicating success or failure of command
  - The *orderingInfo* field indicates what command is being responded to

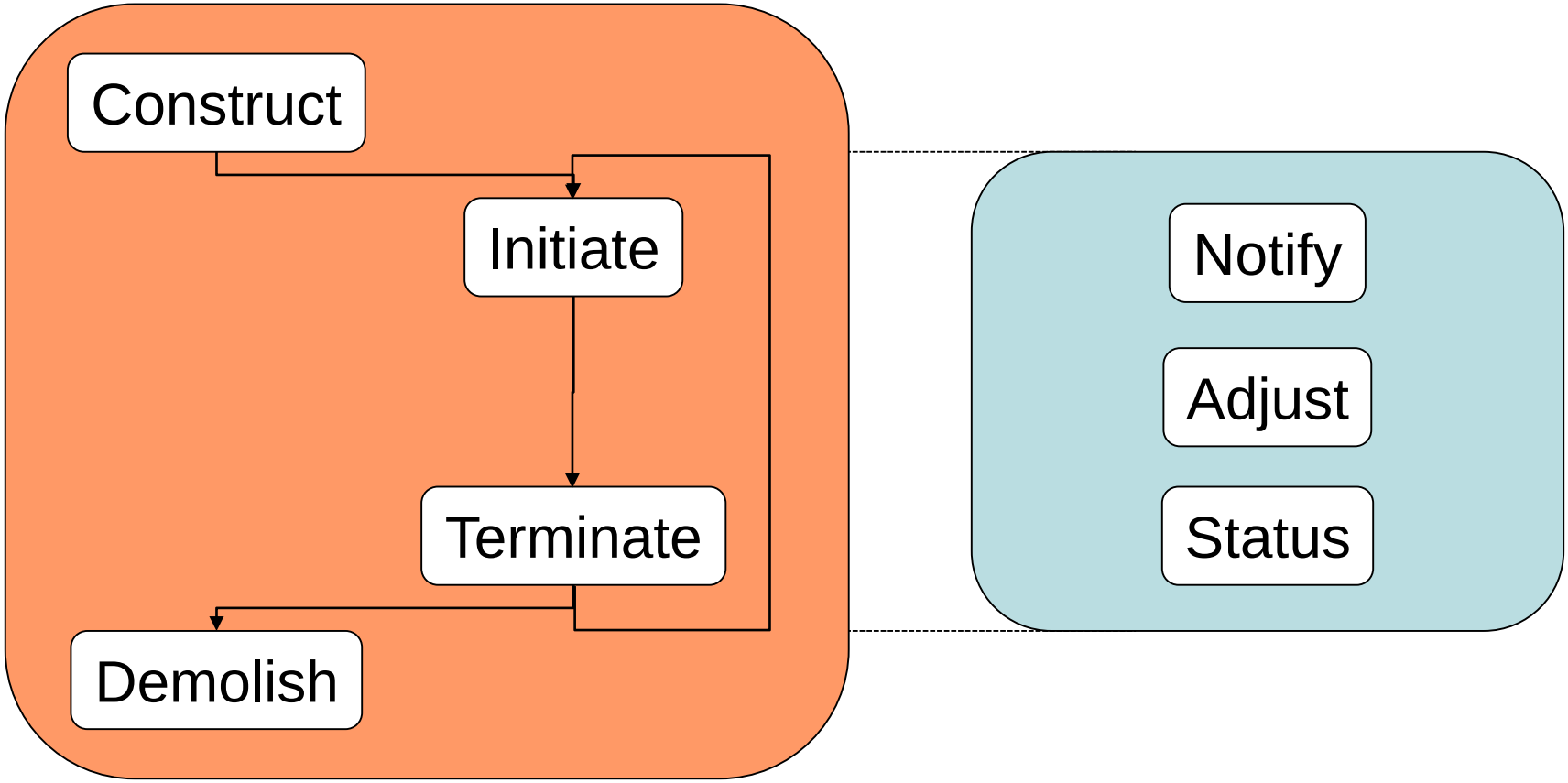


- Suggested response codes:

Code	Meaning
0	Success
1	Invalid Flow ID
2	Flow in Use
3	Invalid Port Number
4	Unsupported Mapper Type

Code	Meaning
5	Unsupported Mapper Subtype
6	Unsupported Packet Size
7	Variable Size Not Supported
8	Unsupported Rate
9	Unsupported Encoding Method

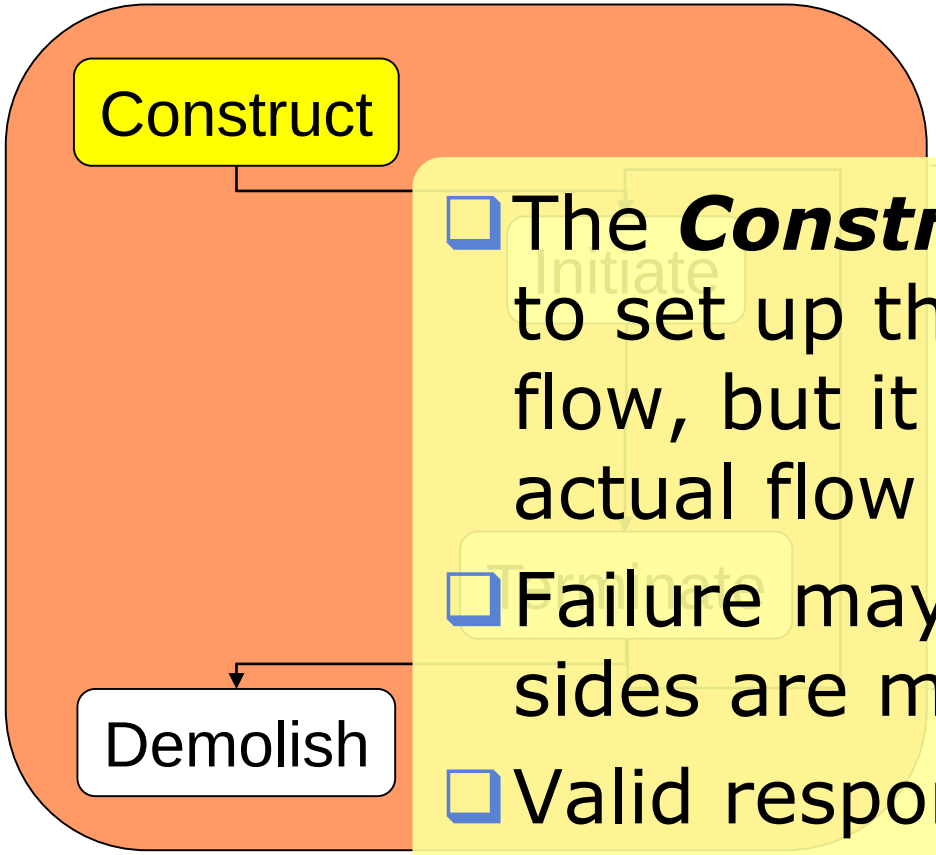
7 Core Flow Control Packets



- These are described on the following pages

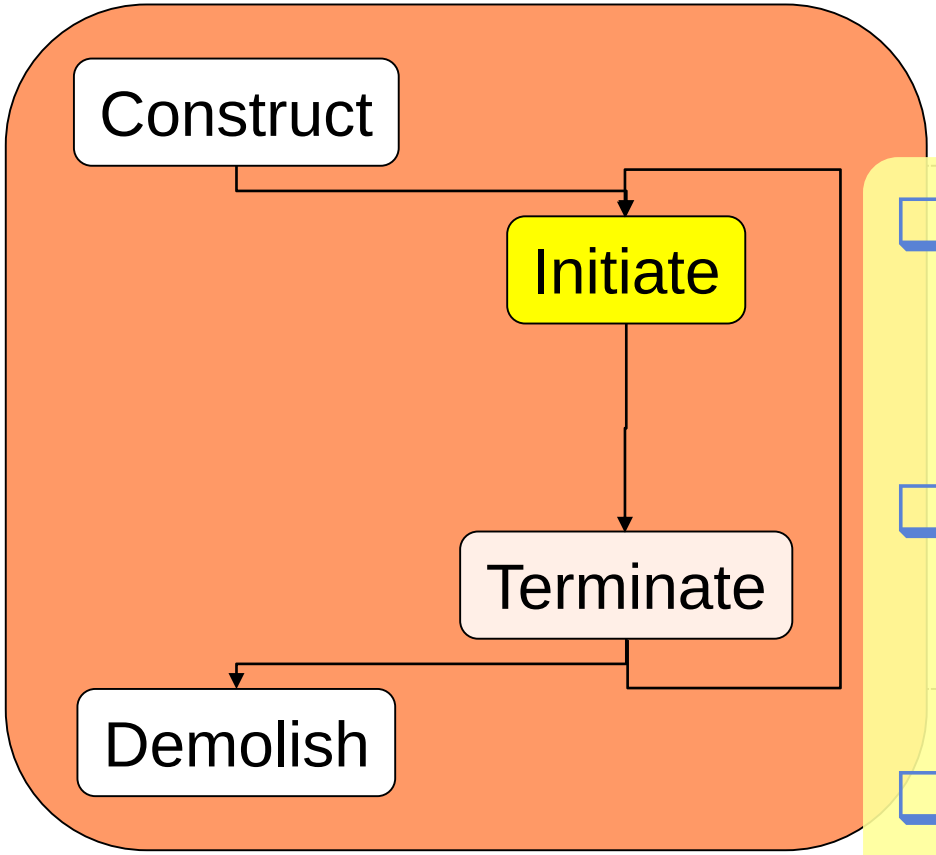
Note: names chosen to have unique initial letters

Construct Packet



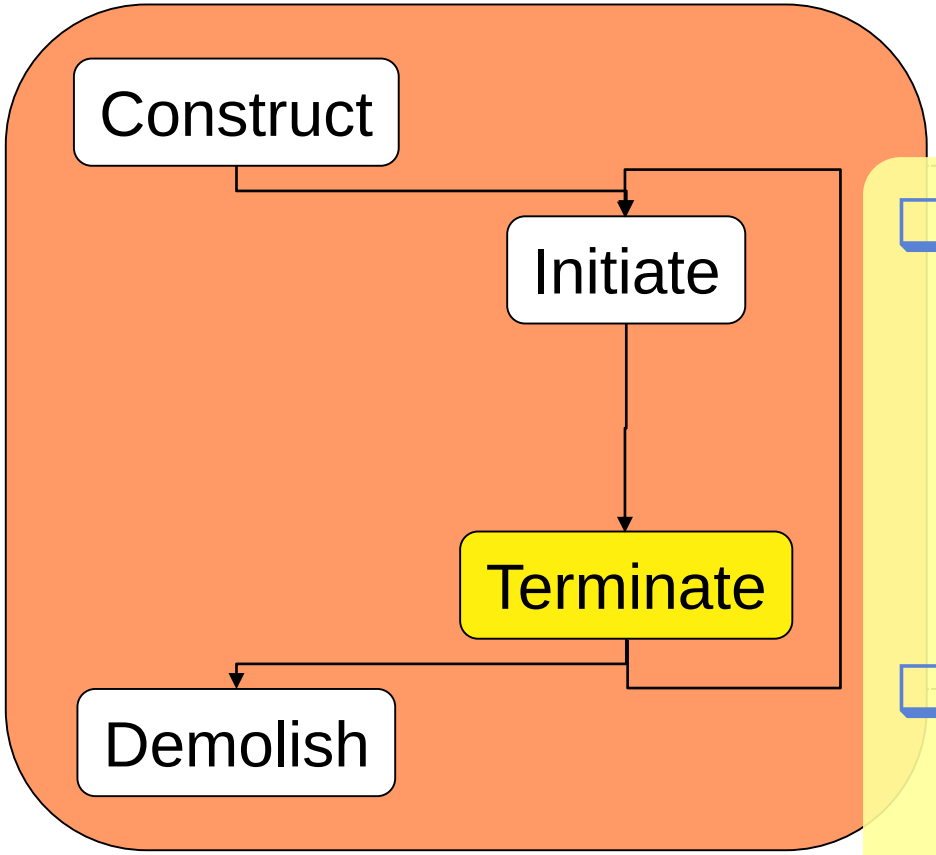
- The ***Construct*** packet is used to set up the overhead for a flow, but it does not start the actual flow
- Failure may occur, especially if sides are mis-matched
- Valid response codes: 0 - 8

Initiate Packet



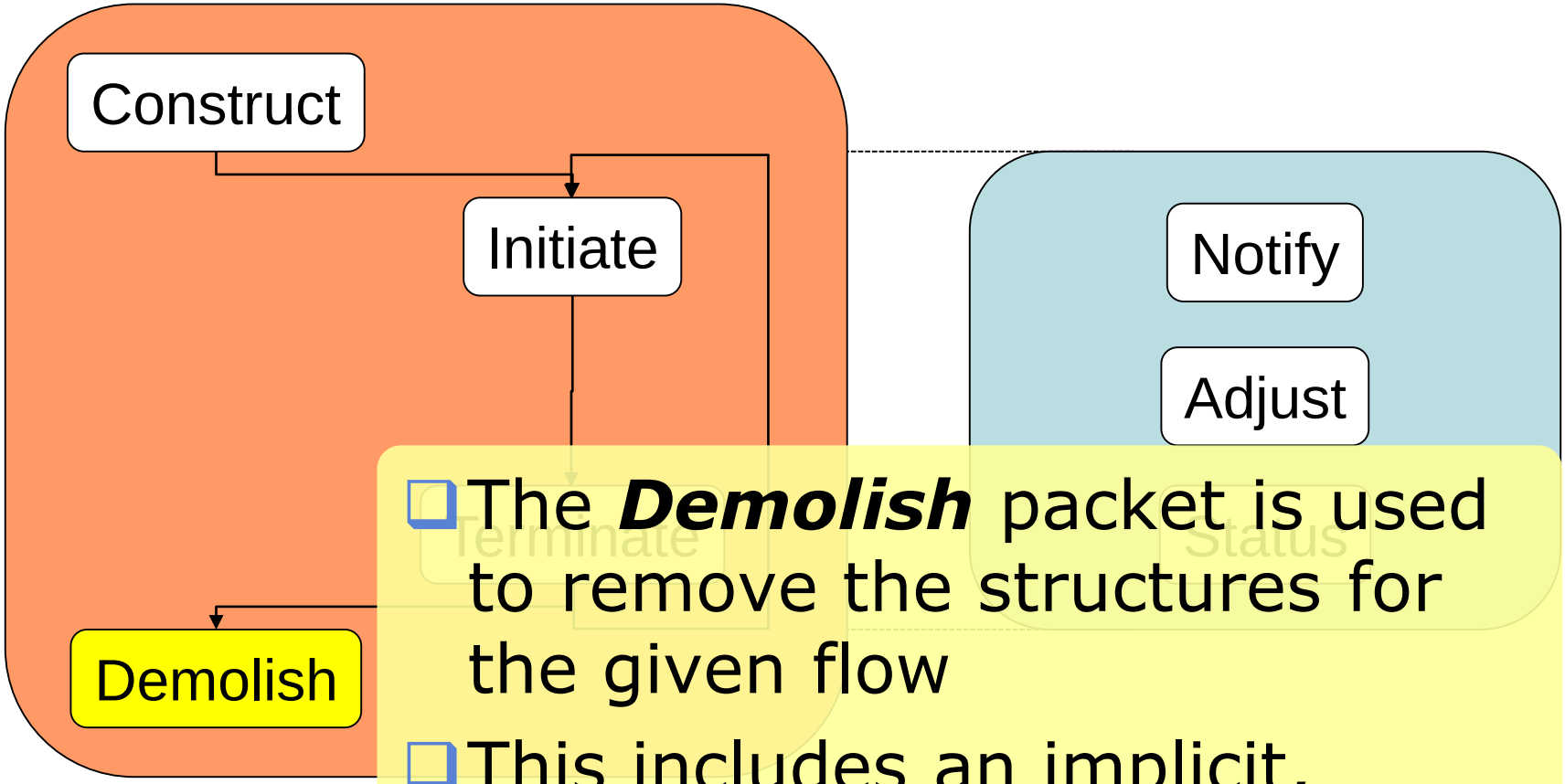
- The ***Initiate*** packet is used to start the actual data flow
- Should generally succeed if *Construct* was successful
- Valid response codes: 0, 1, 9

Terminate Packet



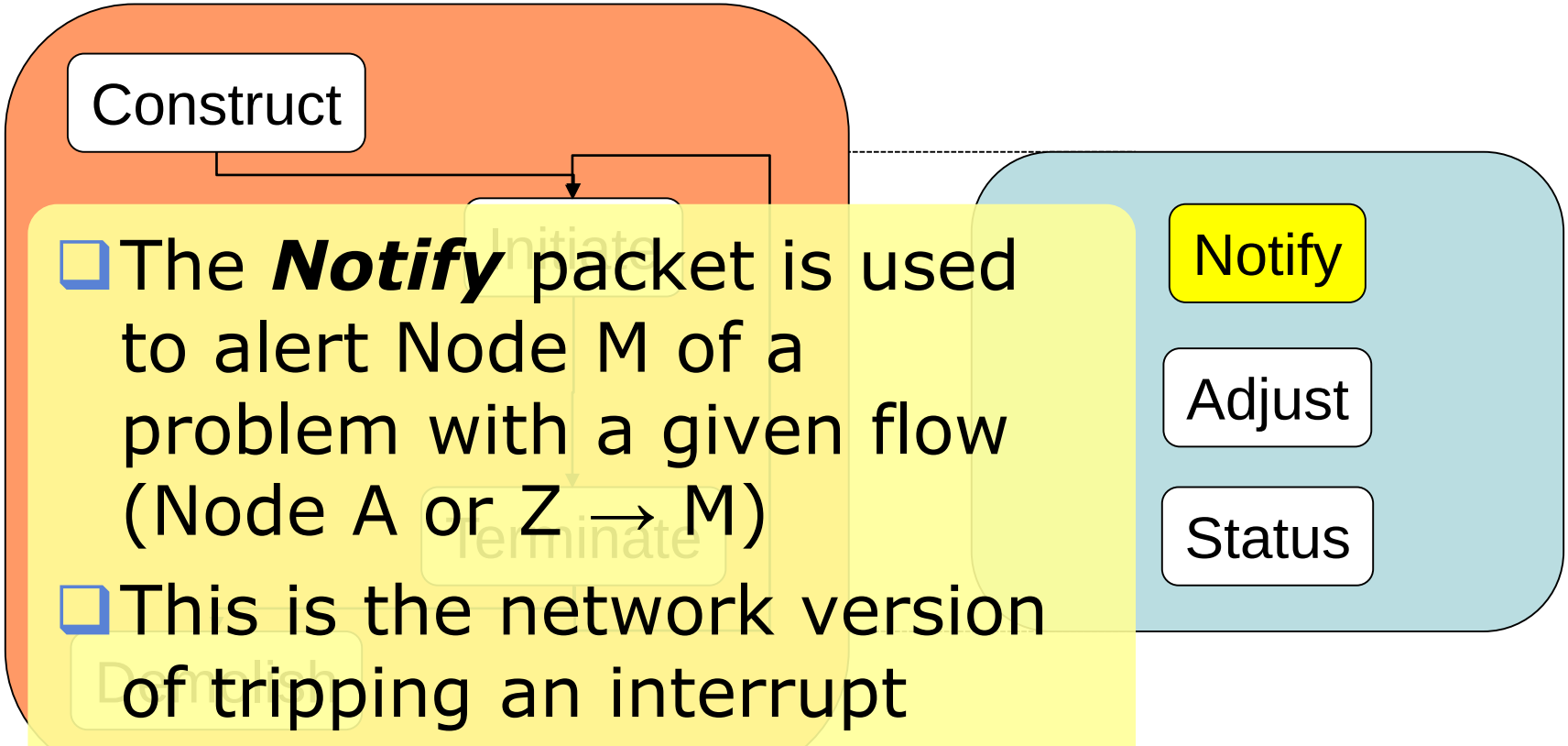
- The ***Terminate*** packet is used to stop the data flow without destroying flow setup
- Should generally succeed if *Initiate* was successful
- Valid response codes: 0, 1

Demolish Packet



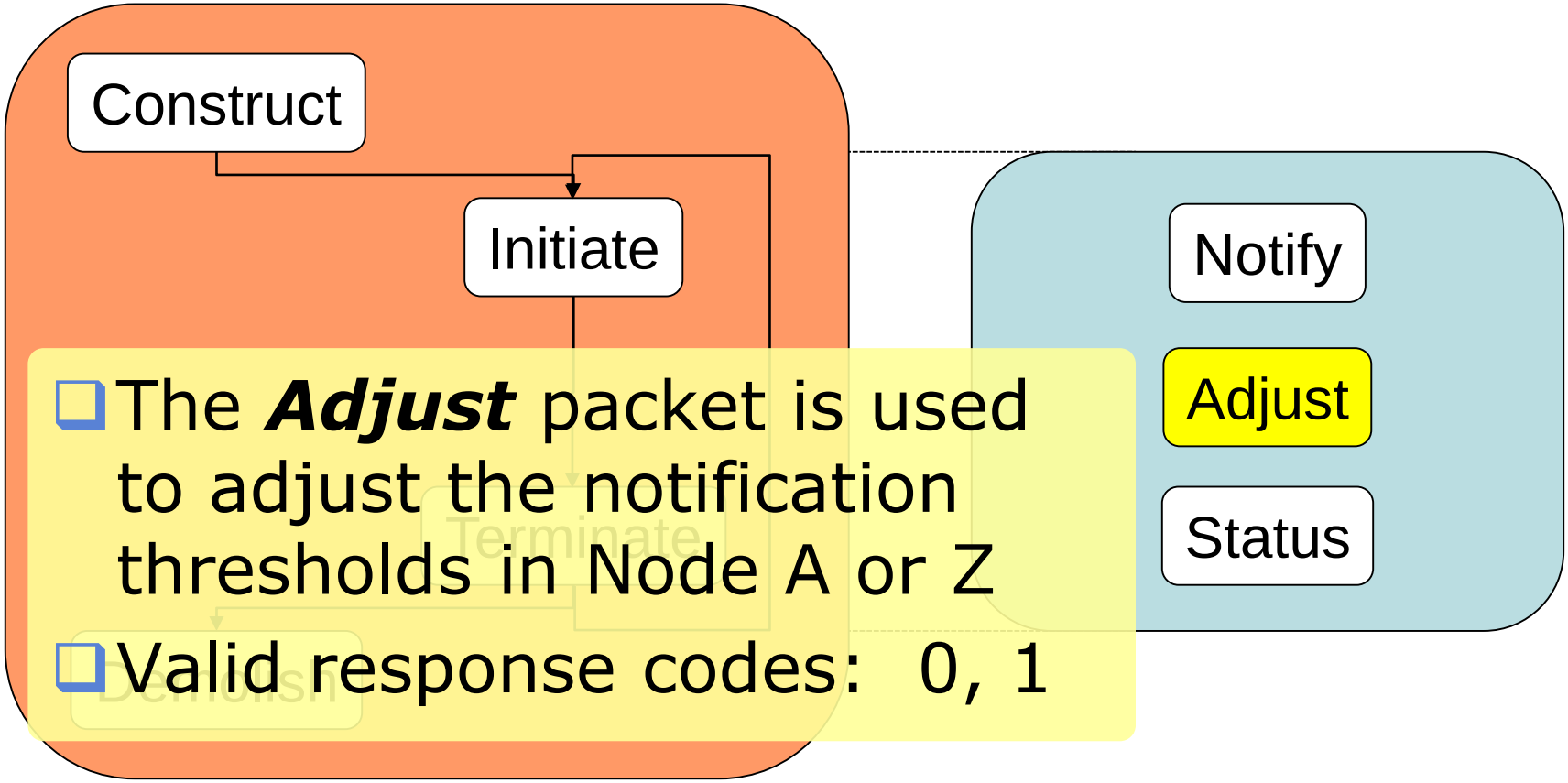
- The ***Demolish*** packet is used to remove the structures for the given flow
- This includes an implicit, immediate *Terminate*
- Valid response codes: 0, 1

Notify Packet



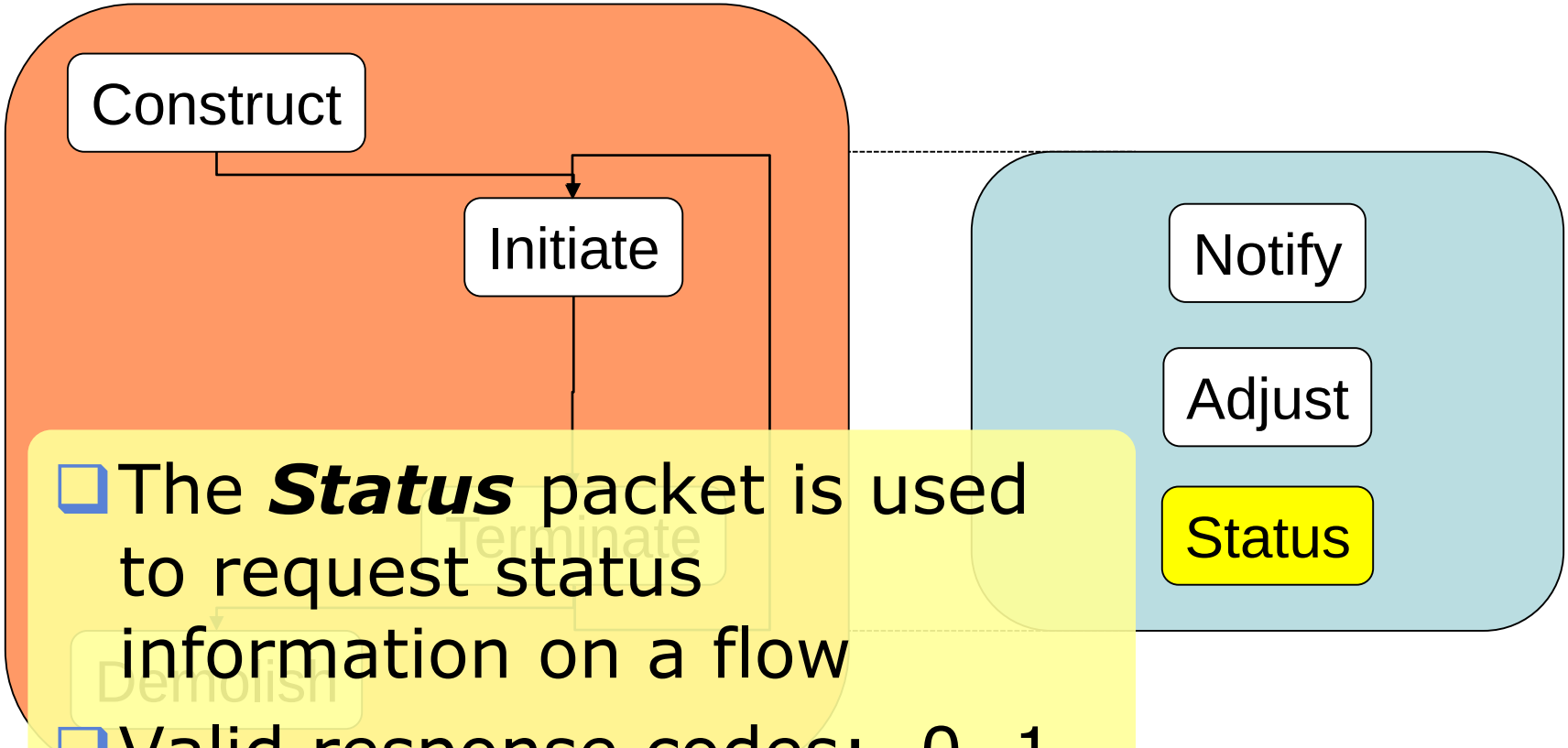
- The ***Notify*** packet is used to alert Node M of a problem with a given flow (Node A or Z → M)
- This is the network version of tripping an interrupt signal in a chip
- Valid response codes: 0, 1

Adjust Packet



- The ***Adjust*** packet is used to adjust the notification thresholds in Node A or Z
- Valid response codes: 0, 1

Status Packet



- The ***Status*** packet is used to request status information on a flow
- Valid response codes: 0, 1
  - The response packet includes the status information in its packetData field

Two Sets of Flow Stats

- Two sets of flow statistics maintained by Node A and Node Z
  - Full data has statistics since *Initiate* packet
    - Full data is reset with each successful *Initiate* packet
  - Incremental data has statistics since last reset by *Adjust* packet
- Odometer analogy
  - Car odometer has lifetime distance
  - Trip odometer has distance since last reset
- How management nodes use this data is out of scope for 1904.3

Control Packet Data

- The following slides present suggested data to go with each control packet
- All control packets have standard header
  - Includes flowID
- Fields are generally in multiples of bytes
  - Since control packets are not sent as frequently as data packets, no need to bit-pack control packet fields like standard header
- Format notes:
  - Field lengths are in bits
  - All packets shown are command packets, except for Status Response packet
    - All other responses only have return code

Construct Packet Data

Field	Bits	Meaning
subType	8	Construct Control Packet (0x01)
radioPort	8	Radio port to encapsulate (i.e., if multiple CPRI ports)
mapperType	8	1 = tunnel, 2 = agnostic, 3 = aware, 4 = native
mapperSubType	8	Encoding type (8b/10b or 64b/66b), or other sub-definition
flowSource	48	MAC address of flow source (i.e., Node A)
flowDest	48	MAC address of flow destination (i.e., Node Z)
flowMgmt	48	MAC address of management node (Node M)
nomPacketSize	16	Nominal packet buffer size (excluding K-chars)
maxPacketSize	16	Maximum packet size in flow (excluding K-chars)
dataRateKbps	32	Radio data rate in 1000 bps units (headroom > 1 Tbps)

- This packet sets up infrastructure for flow
  - Data is sent to Node A and Node Z

Initiate Packet Data

Field	Bits	Meaning
subType	8	Initiate Control Packet (0x02)
flowMode	8	0 = real data, 1 = test data (more later)
startTime	32	Presentation time for when flow starts
transitOffset	32	Timestamp offset to handle transit/buffer time
timeoutLimit	16	Milliseconds without data to cause *Alert* packet
seqnumLimit	16	Consecutive missed sequence numbers to cause *Alert* packet

- Data doesn't start until *Initiate*
- Need to start sending data at startTime – transitOffset (adjust rollover)
- Last 2 fields indicate when to raise alarms

Terminate Packet Data

Field	Bits	Meaning
subType	8	Terminate Control Packet (0x03)
stopTime	32	Presentation time for when flow starts

- Used to cleanly stop data flow
  - stopTime allows both sides to expect when data will stop (and not raise alarms)
- After this, can either demolish flow or can re-start flow

Adjust Packet Data

Field	Bits	Meaning
subType	8	Adjust Control Packet (0x04)
transitOffset	32	Timestamp offset to handle transit/buffer time
timeoutLimit	16	Milliseconds without data to cause *Alert* packet
seqnumLimit	16	Consecutive missed sequence numbers to cause *Alert* packet
resetFlag	8	0 = don't reset incremental statistics, 1 = do reset inc. stats

- Primarily used to adjust notification thresholds
- Can also walk in changes to transitOffset
  - For example, due to network reconfiguration
- Can optionally reset incremental statistics

Notify Packet Data

Field	Bits	Meaning
subType	8	Notify Control Packet (0x05)

- Method for Node A or Node Z to notify Node M when something is wrong
  - Based on thresholds from *Initiate* or *Adjust*
- Node M can issue a *Status* command to get statistics from either/both nodes

Status Packet Data

Field	Bits	Meaning
subType	8	Status Control Packet (0x06)
statusType	8	0 = full statistics, 1 = incremental statistics

- Method for Node M to request statistics from Node A or Node Z
  - Request full or incremental statistics
- Statistics reported in response packet
  - See next pages

Status Response Data (1)

Field	Bits	Meaning
subType	8	Status Control Packet Response (0x86)
statusType	8	0 = full statistics, 1 = incremental statistics
flowStatus	8	0 = new, 1 = active, 2 = terminated
radioPort	8	Radio port encapsulated (i.e., if multiple CPRI ports)
extStartTime	96	Start time for flow (32b seconds, 32b ns, 32b sub-ns)
extReportTime	96	Report time for flow (32b seconds, 32b ns, 32b sub-ns)
transitOffset	32	Timestamp offset to handle transit/buffer time
bytesProcessed	64	Total data bytes processed through this flow
packetsProcessed	64	Total good data packets processed through this flow

- This is the first part of the *Status* response
  - Note: If flowStatus = 2, extReportTime reflects the time the flow was terminated

Status Response Data (2)

Field	Bits	Meaning
currentSeqnum	32	Current data sequence number (if seqnums in this flow)
seqnumOverflows	64	Number of times the sequence number has wrapped
minBufferTime	32	Minimum time any good packet was in egress buffer
maxBufferTime	32	Maximum time any good packet was in egress buffer
avgBufferTime	32	Mean time any good packet was in egress buffer
minPacketSize	32	Minimum size of data packet (past header & K-chars)
maxPacketSize	32	Maximum size of data packet (past header & K-chars)
avgPacketSize	32	Mean size of data packet (past header & K-chars)
dataRateKbps	32	Radio data rate in 1000 bps units (headroom > 1 Tbps)

- This is *Status* response part 2

Status Response Data (3)

Field	Bits	Meaning
totalMissedPkts	64	The total number of missed packets (including late packets)
rowMissedPkts	32	The maximum number of missed packets in a row
totalLatePkts	64	The total number of packets received after presentation time
rowLatePkts	32	The maximum number of late packets in a row
transitOffset	32	Timestamp offset to handle transit/buffer time
timeoutLimit	16	Milliseconds without data to cause *Alert* packet
seqnumLimit	16	Consecutive missed sequence numbers to cause *Alert* packet
timeoutAlerts	32	Number of timeout *Notify* events
seqnumAlerts	32	Number of sequence number *Notify* events

- This is *Status* response part 3

Test Mode

- Spec should not auto-set transit time through Ethernet network like CPRI
  - Need to understand headroom requirements to account for jitter and network anomalies
  - May want to correct for asymmetry in links
- Test Mode provides data that Node M can use to set transit time expectations
  - Send sample data through network
    - At target data rate
    - From Node A to Node Z
    - Collect data on transit time (next page)
    - Then set transit time expectations with real data
  - Test mode data does not send data to radio

Test Mode Transit Times

- First 32 bits of each Test Mode data packet is used for transit time measurement:
  - Similar to 1588 2-step time synch method
  - First 32 bits in packet N = timestamp when packet N-1 was sent
  - Receiver timestamps when each packet is received
  - Receiver calculates each packet's transit time
- Stats for test mode data altered
  - min/max/avgBufferTime is actually calculated from the transit times at the receiver

Summary

- Node M manages RoE flow between Node A and Node Z
- 7 control packets proposed
  - *Construct* a flow (match features, set up flow)
  - *Initiate* flow (send data)
  - *Adjust* flow (change threshold, transit data)
  - *Terminate* flow (stop sending data)
  - *Demolish* flow (clear data structures)
  - *Notify* Node M if threshold exceeded
  - *Status* (request/deliver statistics on flow)
- Test mode can help gauge transit time
  - Node M can use data to set *Initiate* parameters