IETF NSIS WG Overview

Julien Laganier
RESO – LIP / SUN Labs

IETF NSIS WG Overview

These slides are a digest of IETF NSIS WG proceedings.

More information can be found at http://www.ietf.org

Outline

- Problem statement and scope
- Design space for NSIS signalling protocol

NSIS Problem Statement and Scope

NSIS Initiator

- Something (NI) that starts a request for resources
- Might be
  - the end system
  - some other part of the network (ingress point)
- Actions triggered (directly or indirectly) by higher layers in the end system
- Needs to map requested resources and provide feedback information to higher layers

NSIS Forwarder

- NF assists the NI in managing resources further along the path
  - does not interact with higher layers
  - interacts with the NI and possibly others NFs
  - path segment by path segment
  - not required to be on the flow's data-path
- A path segment traverses an underlying network:
  - covering one or more IP hops
  - using some local QoS technology
  - NF maps service-specific informations to technology-related QoS parameters

NSIS Assumptions and non-ones

- NSIS signalling could run
  - end-to-end
  - edge-to-edge
  - network-to-network
- NFs are not constrained to a particular location within a NSIS domain
- NSIS does not consider pure end-to-end signalling (application layer issue, SIP WG)

NSIS Exclusions

- application and transport layer adaptation
- interaction between transport/applications and the network layer
- QoS provisionning
- interaction between the network layer and QoS provisionning
- interaction with resource management capabilities
- service definitions (e.g. QoS classes)
- QoS monitoring

The Design Space for NSIS Signaling Protocols

Overview

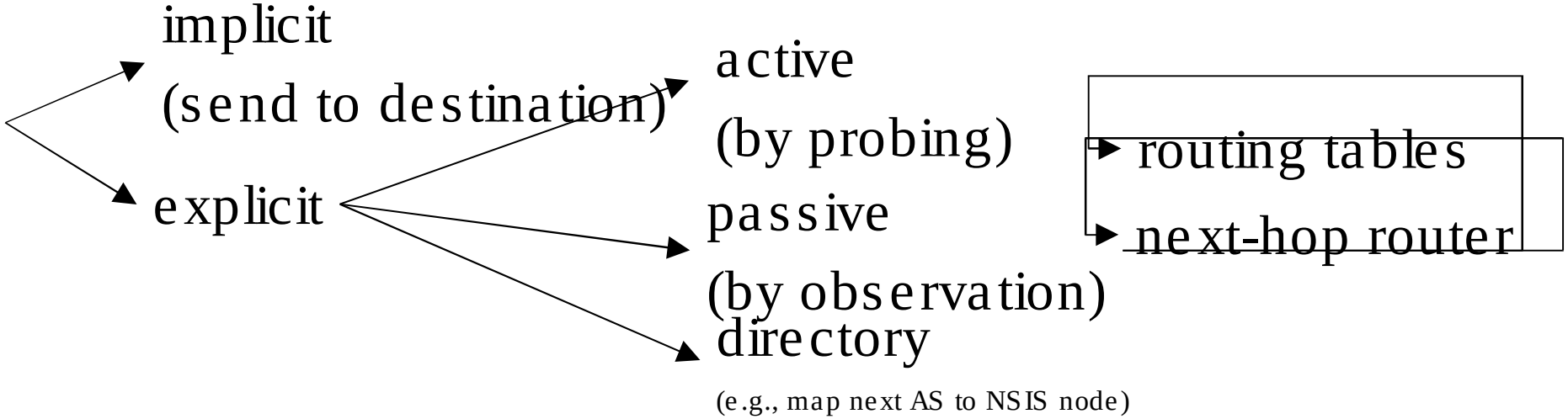
- Some NSIS assumptions
- “ Transport” layer
- Peer node discovery

NSIS model

- Want to support a variety of “signaling” applications
  - not just QoS → otherwise, why bother with 2-layer model?
  - *path-associated state management*
  - applications:
    - manage data flows along path: NAT/firewall, QoS
    - just along path:
      - active network code deposits
      - network property discovery (“traceroute-on-steroids”)
      - network property management (not just NATs/firewalls)
  - we are not designing all applications now, but should not lightly prevent future use
- bidirectional signaling support with equivalent functionality
  - NI → NR and NR → NI
  - possibly NE → NE

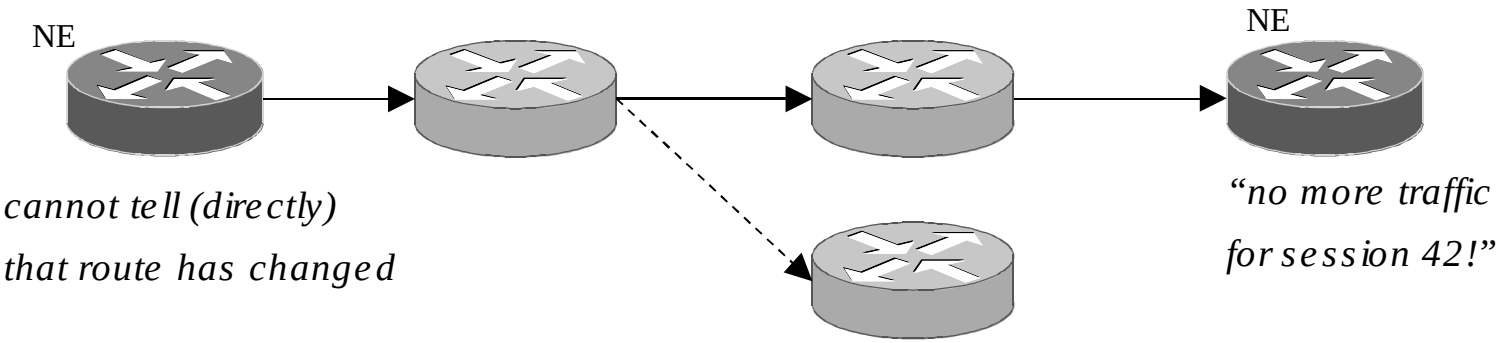
Finding NSIS peers

- The problem is not finding (all/some) NSIS elements
  - → service location problem (SLP, DNS, etc.)
- but rather finding the next NE on the data path to the NR



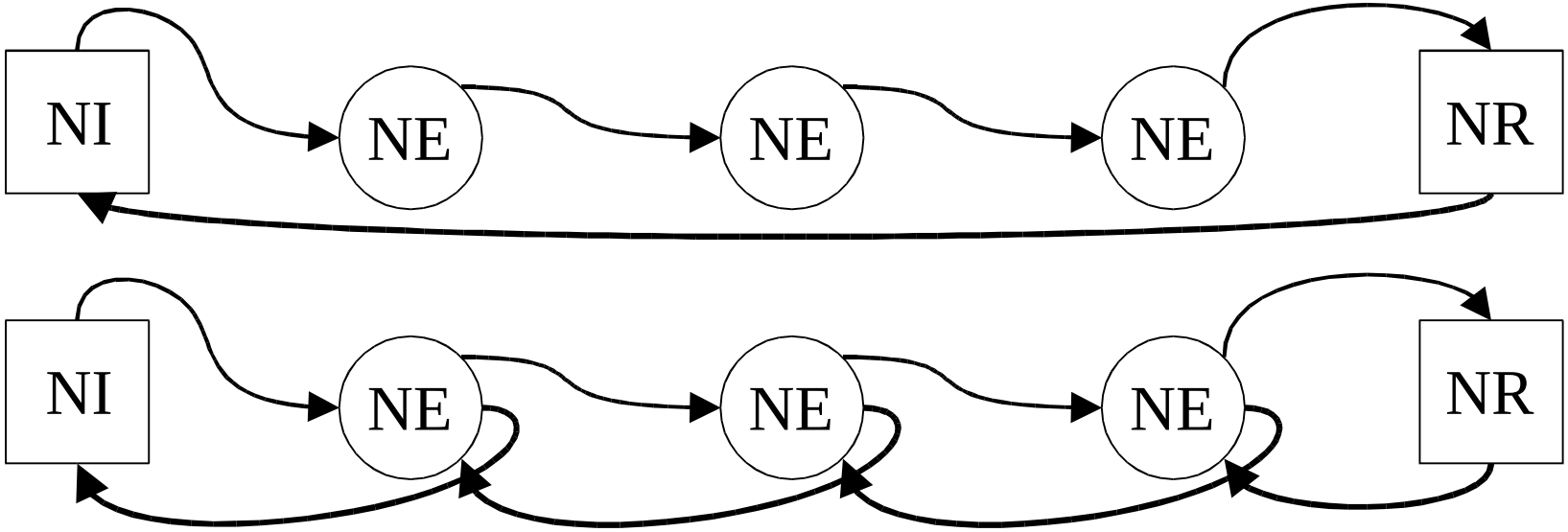
When to discover peers

- Can be triggered by NI or NE
- May not want it automatically
  - e.g., remove reservation – don't want to be first on new path!
  - good to have separation of discovery and operation
- Options:
  - for every new NI-NR session
    - including edge changes
  - for every application-layer refresh
  - requested by NI
  - when detecting a route change in the middle of the network



Next-node discovery

- Basic function, regardless of *-orientation
- generally, NI needs to establish state so that messages can flow in both directions
  - implicit assumption, could have unidirectional



Next-node discovery

- Next-node discovery probably causes operational distinction between path-coupled and path-decoupled
- path-coupled:
  - one of the routers downstream
  - always only guess at coupling
- path-decoupled:
  - some server in next AS
  - anything else make (interdomain) sense?

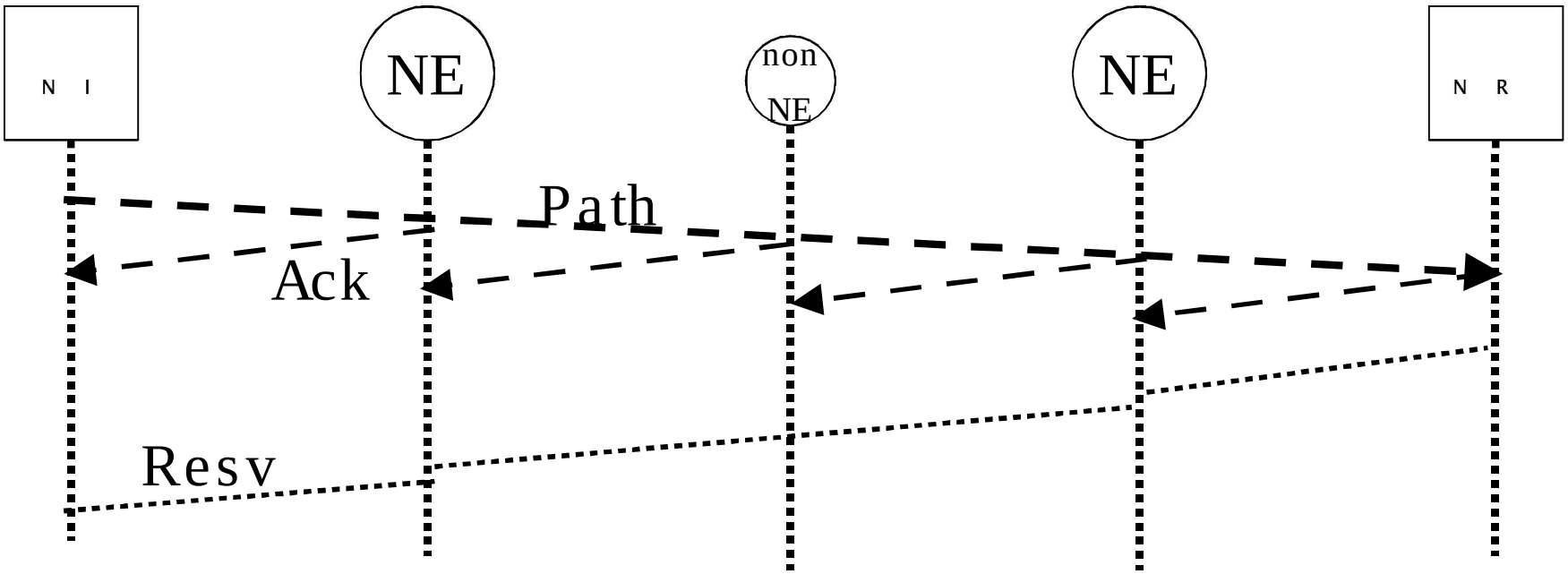
Next-node discovery: path-coupled

- All discovery is approximate
  - some node could use any feature of the discovery packet to route it differently

discovery = data	divergence causes	constraints
destination address	load balancing	
source & destination address	L4 load balancing?	no signaling proxies (ICMP errors misdirected to data source)
full 5-tuple	presence of router alert options?	no signaling proxies how to disentangle at end system?

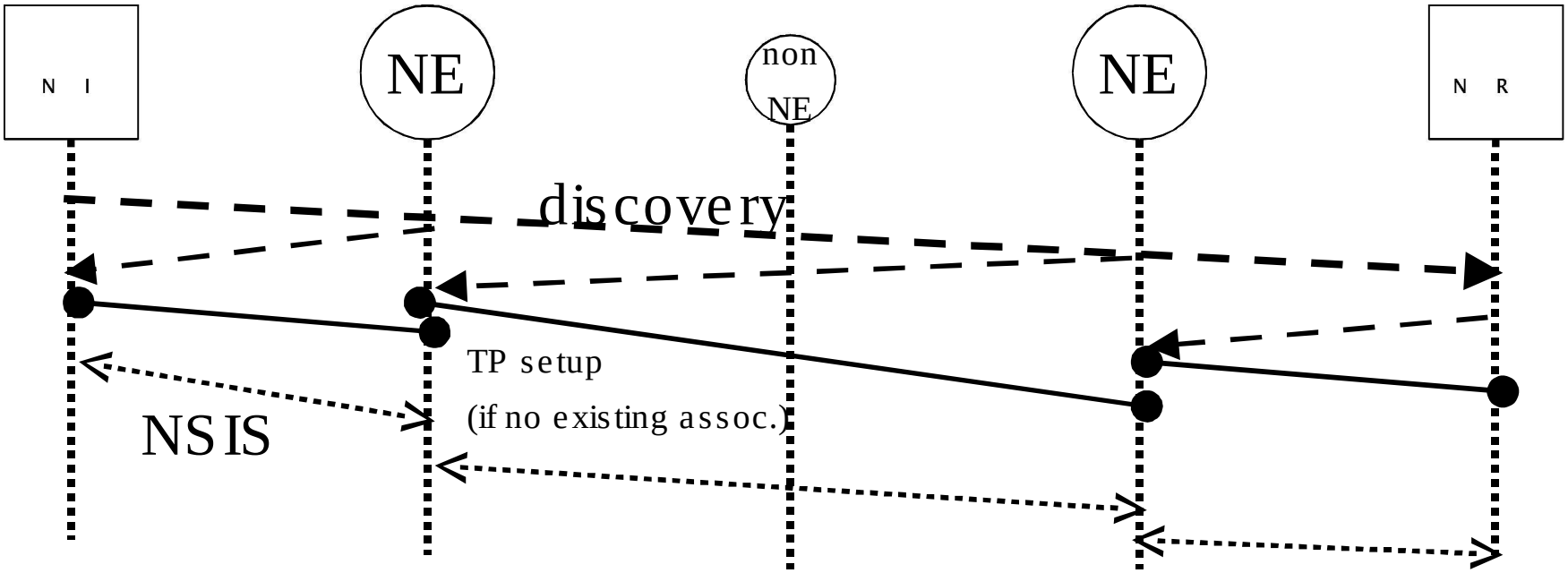
Peer discovery: RSVP style

- Forward messages (Path, PathTear) addressed to NR
- Backward messages (Resv*,PathErr) sent hop-by-hop
- Path messages: discovery + special application semantics



Peer node discovery: path-coupled

- With forward connection setup
- Only needed if next IP hop is not NSIS-aware
- Discovery messages: pure or application-enabled?



Transport requirements

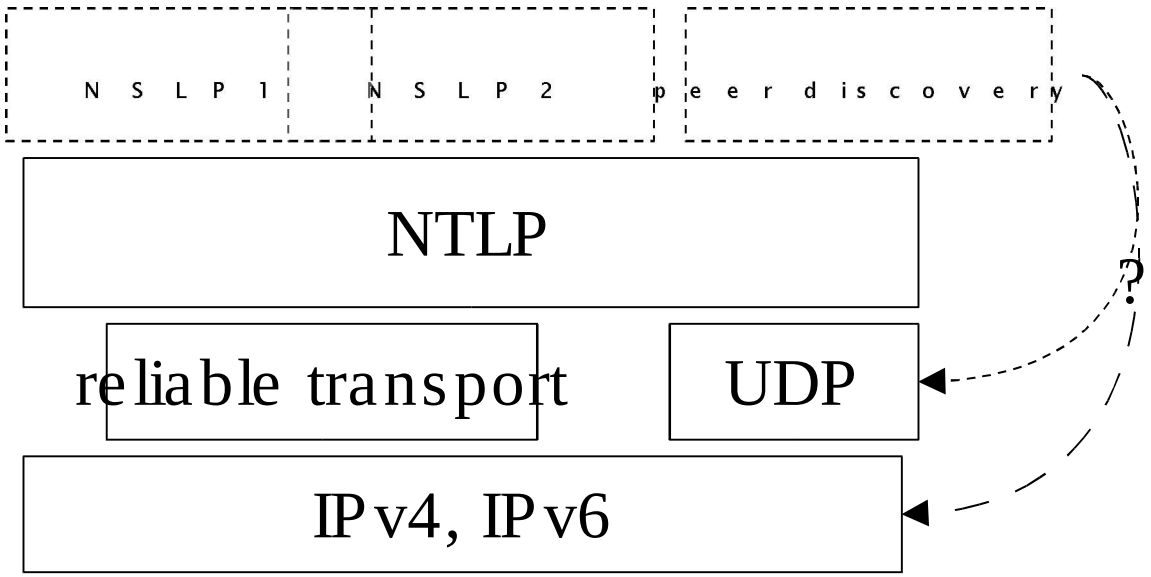
- Signaling transport users may require large data volumes:
  - active network code
  - signed objects (easily several kB long if self-contained; standard cert is ~5 kB)
  - objects with authentication tokens (OSP, …)
  - diagnostics accumulating data
- Signaling applications may have high rates:
  - DOS attacks
  - automated retry after reservation failure (“redial”)
  - odd routing (load balancing over backup link)

Lower and upper layers

- Do all nodes process all NSIS messages?
- “ omnivorous”:
  - all messages, even unknown signaling protocols
  - e.g., firewalls
  - depends on what information is common
    - common flow identification?
- “ vegetarians”:
  - only things they know and can understand

Layering

- Some terminology confusion for NTL**P** – service vs. protocol
  - we'll take protocol (and contradict framework…)
  - = functionality added to lower layer
- maybe 'messaging layer' is less overloaded



Reliability

- Most signaling applications require that end systems have reasonable assurance that state was established
  - if it wasn't important, why bother sending message to begin with ☺?
  - often, modestly time-critical:
    - human factors → call setup latency
    - economic → fast and reliable teardown
- RSVP discovered later → staged refresh timer (RFC 2961)

Reliability options (1)

- End-to-end retransmission
  - NI retransmits until confirmation by NR
    - ☺ simple – only requires NI state
    - ☺ deals with node failures
    3. usually, no good RTT estimate → flying blind
    4. doesn't work well for NR-initiated messages
    5. node processing (incl. AAA) adds delay variability → RTT very unpredictable
- Hop-by-hop below NTLP
  - share congestion state between sessions → better RTT estimate
  - re-use transport optimizations such as SACK
  - inappropriate services?
  - mandates explicit discovery (see later)

Reliability options (2)

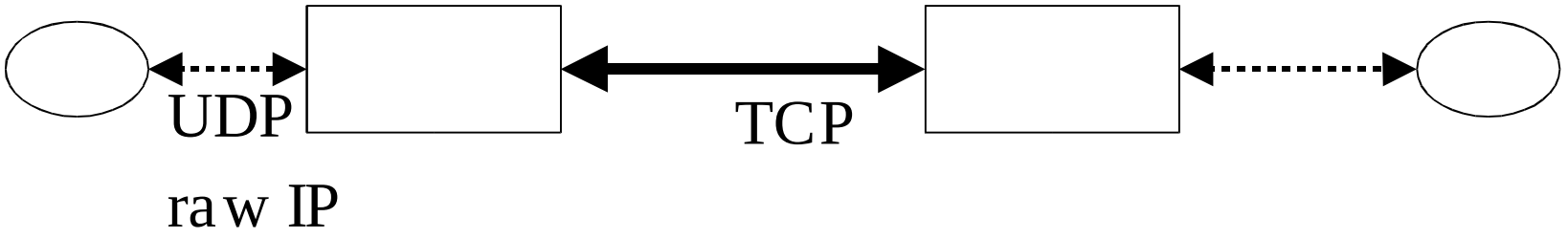
- Hop-by-hop by NTLP per session
  1. can use implicit discovery
  - RFC 2961
  - simple exponential back-off: no windowing, no SACK
    ➔ bad for long-delay pipes
  - timer estimation difficult
    - often few messages for one NSIS session
    - must only have transport semantics
- Hop-by-hop by NSLP
  - diversity of needs vs. cost
    - what does a feature cost if not used/needed?
  - what's left for NTLP in that case?

Other transport issues

- MTU discovery
  - can change during session → may force end-to-end rediscovery
  - NSIS packet size can change during transit
  - not a problem if all messages are small (< 512 bytes?)
- Congestion control → prevent network overload
  - traffic burst for state synchronization
  - retry after failures
- Flow control → prevent NE overload
  - traffic burst for state synchronization
- Security association
  - needed for any channel security
- Message bundling
  - probably interesting mostly for small (optimistic refresh?) messages
- DOS prevention
  - need validated peer → never, ever send more than one message for each request!

Transport protocol options

- None → raw IP
  - limited to IPsec for NE-NE channel security
  - can't send Path via IPsec: no idea what SA
- TCP
  - needs encapsulation (= one-word message length)
  - HOL blocking – waiting for old message
  - IPsec or TLS for channel security
- SCTP
  - easier end system diversity → relevant mostly for path de-coupled
  - avoids HOL blocking – but effect is *very* hard to actually observe (see upcoming *IEEE Network* article)
- DCCP
  - future options, but "UDP + congestion control" may not be good fit



Transport (non-)issues

- “ But xP is stateful and we want soft-state”
  - existence of transport association should not be coupled to NTLP or NSLP state lifetime
  - loss of transport does not signify anything (except maybe a reboot of the peer)
  - primarily an optimization issue: state maintenance vs. state establishment overhead
- Multicast
  - Each branch can have own transport session
  - In RSVP, only Path* are multicast
- End-to-end principle
  - not clear what the “ends” are here
  - each NE is not just forwarding, but processing and modifying messages
  - explicitly noted for performance enhancement
- Number of associations per node
  - limited by `select()`, but not `poll()`

Transport (non-)issues

- State overhead
  - information about next/previous hop has to be somewhere…
- Transport header overhead
  - most messages are likely >> 40 bytes
- Transport implementation overhead
  - Conceivable end systems and routers already implement IP, UDP, TCP
    - TCP needed for DIAMETER, SNMP in routers
    - TCP on any reasonable mobile device (HTTP, SMTP, POP, IMAP, …)
  - Less clear for SCTP

Identifiers

- Need identifiers for each logical association/session
  - know whether this path has been traversed before
    - need discovery or not
  - pass to correct upper layer handler
- SIP lesson: do not overload identifiers

Identifiers should be…

- globally unique
  - otherwise, they'll have to be combined with something else
- not depend on host addresses
  - NI and NR may change during session (mobility)
  - NAT and RFC 1918-uniqueness issues
  - RSVP SENDER_TEMPLATE and SESSION object →
    - constrains applications
    - hard to match (multiple formats)
    - same session has different identifiers along a path → hard to manage
- probably not depend on globally unique host identifier (MAC) address
- constant length
  - easy to parse and compare
- cryptographically random
  - not sufficient for security, but often helps to prevent long-distance session stealing attacks
  - can often avoid a complicated hash function

Packet format issues

- Variations on type/length/value
- Type can be
  - externally described (RSVP)
    - meaning (“destination address”, “flowspec”)
    - format (IPv4 or IPv6)
  - internally described
    - implied (DIAMETER)
    - internal discriminator

A Two-Level Architecture for Internet Signalling

The Signaling Problem Domain

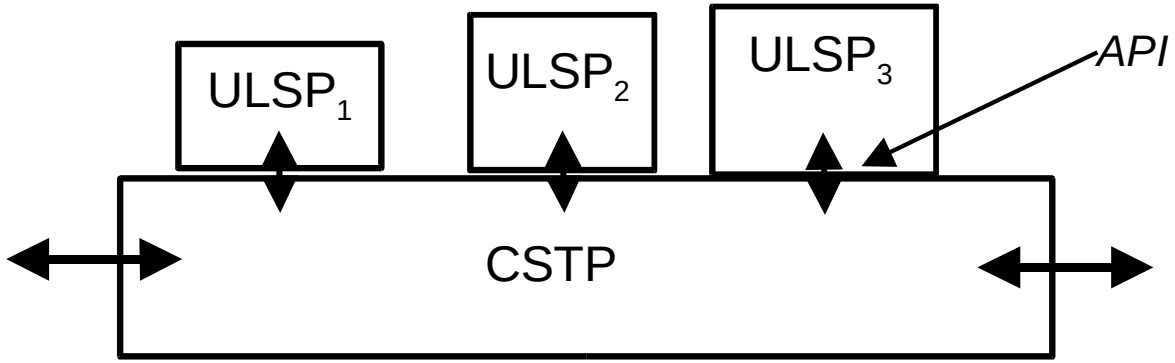
- There are many different signaling problems.
  - **QoS setup** [e.g., RSVP v1] and all its variations -- E2E vs. proxies, multi-level (inter- vs. intra-domain), mobility (?), telephony (?)
  - **Middlebox configuration** (NATs, firewalls,...) [e.g., TIST]
  - **Traffic engineering** [e.g., RSVP-TE]
  - **Link Layer & Access Net Control** [e.g.,GMPLS-TE, PacketCable, ...]
  - **Network provisioning** (VPNs, Diffserv DSCP, ...)
- Important WG decision: how to structure this space

Motherhood

- Want to take an Internet-friendly approach
  - Robust: cope with heterogeneity and with failure
  - Fundamentally simple but easily extensible
  - General: useful for future signaling applications
  - Common mechanisms
- What Can We Learn from RSVP V1?
  - It has been adapted to a wide variety of signaling problems
  - That’s also the bad news ... Advanced state of protocol chaos!

Two-Level Architecture Proposal

- Define Internet Signaling Protocol Suite (ISPS)
  - (My choice for a name for the “NSIS protocol”)
- ULSP -- (generic) Upper-Level Signaling Protocol
- CSTP -- Common Signaling Transport Protocol
  - Framework document calls these NSLP, NTLP resp.
  - CSTP may use IP, UDP, and/or TCP (see later)



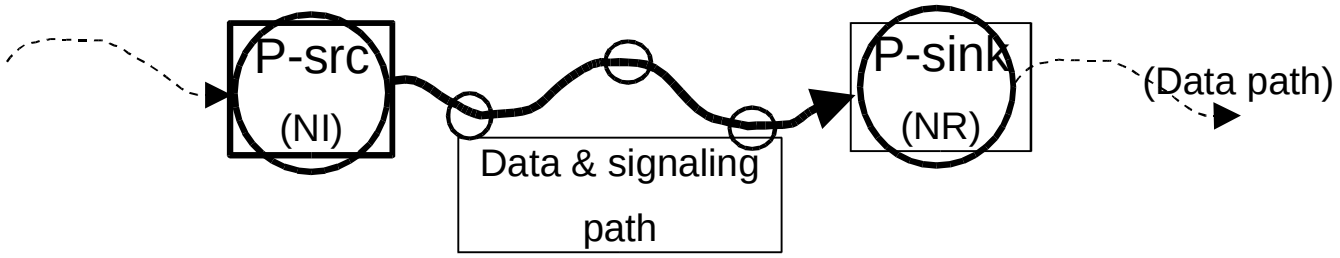
Why?

(For the same reasons there is a Transport layer in the protocol stack.)

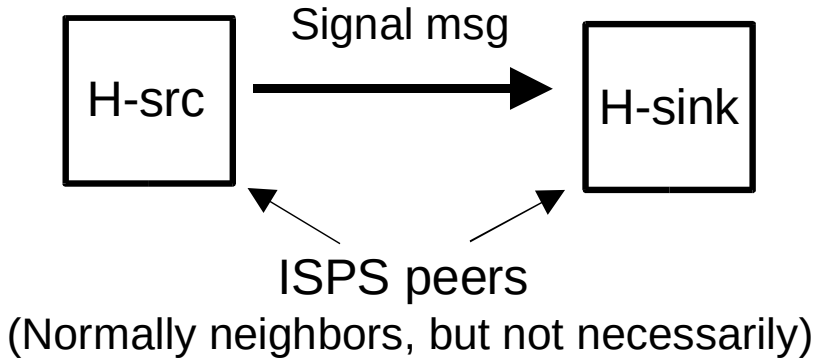
- Modularity is a Good Thing
  - Simplify design of new signaling applications
  - Independent evolution of signaling applications and the underlying transport mechanism
  - Key element: CSTP service model and API
- Why only two levels?
  - Can we further modularize the ULSP level?
  - Good idea, but moving beyond engineering into research here.

Functional Partition

- ULSP implements E-to-E (NI-NR) semantics



- CSTP handles peer-peer communication
  - Payload: *Signaling App. Proto. Unit* (SAPU)



CSTP Functions

- Design goal: general building block for wide range of signaling applications
- Proposed CSTP-level functions
  - Reliable ordered delivery* of (trigger) messages
  - Soft-state refresh*
  - Fragment/reasm SAPUs*
  - Routing interface
  - Hop/Hop security*
  - Congestion control
  - Neighbor discovery and up/down determination

(* = optionnal)

CSTP Service Model

- Send: deliver SAPU and maintain/timeout as soft state
    - Option: deliver SAPUs reliably and in order [1].
- Tear: (H-src -> H-dest) explicitly remove SAPU state.
- Rev-Tear: (H-dst to H-src) delete soft state. [not in I-D]
- SendInfo: deliver SAPU (not soft state)

Note:

[1]: Can reliable delivery option be chosen internally by CSTP on hop/hop basis, rather than chosen by the ULSP?

CSTP Service Model (2)

- CSTP service model is subtly different from Request, Release, Notify, Accept, model ... of traditional telephony signaling.
  - Send roughly analogous to ‘Request’
  - Tear, Rev-Tear roughly analogous to ‘Release’
  - SendInfo rough analogous to ’Notify’
  - But ‘Accept’, ‘Reject’ are higher-level concept -- at ULSP level.

CSTP Protocol

- The CSTP protocol(s) proposed in the Draft are incomplete; intended to establish plausibility. (Move to an Appendix)
- CSTP was originally designed to provide RSVP-style transport directly over IP
  - Call this CSTP/IP.
  - Could alternatively use (S)TCP connections underneath CSTP, to obtain reliable ordered delivery – CSTP/(S)TCP.
  - Still need soft state mechanism for deleting unused state.

More Technical Details

- Assumed that an SAPU is opaque to the CSTP level.
  - Strong assumption: e.g., CSTP layer then does not know about flow identification, which may be arbitrarily encoded into SAPU.
    (cf. RSVP v1 Session, SenderTemplate objects)
  - Then SAPU level cannot know about previous/next hop per-flow; the ULSP has to maintain this information for a path-coupled signaling operation.
    - Alternative approach: back off on generality, make standard-encoding of flow identification visible to CSTP. Needs more thought.

ULSP Level

- Every ULSP can use its own encoding for its SAPUs.
  - This flexibility is good, but it would also be good to:
- Define a standard encoding format for SAPUs.
  - The RSVP packet format -- small fixed header + sequence of typed data objects -- has worked well.
- Could define a concrete API to CSTP
  - To enable 3rd-party ULSP software

NSIS Framework issues

Key Issue I

- What is the actual NTLP/NSLP divide?
  - [Almost] all other issues depend on it
  - Seem to be many options
- Consider a ‘core’ NSLP at every NE?
  - Share application logic without putting it in NTLP
- Capture the split explicitly with an API?
  - When to start and where to put it?
- See later slides for discussion

Key Issues II and III

- Mobility interactions pop up again and again in many places
  - Very little open discussion and analysis so far
  - Nature of input from seamoby unclear
- A main change from RSVP→NSIS is multi 'application' support
  - Discussion is still very QoS focussed; don't know how multi application support should be thought about
  - Cf. other experiences of overloading: AAA, DNS, HTTP
  - What steps are needed to prevent chaos

1: Sender/Receiver

- Relates to current NI/NF/NR definitions
- Assumption:
  - Layer split will make signalling transaction initiation/response a signalling application property
  - Means NI/NF/NR become NSLP concepts
  - Consequence: significant editorial updates…

2: NI/NR/NF Rights, Proxies

- Mainly an issue about authorisation structure
- Assumption:
  - Layer split will assign this as NSLP issue
  - Update current discussion as QoS ‘example’
- Proxies: just back-to-back NR+NI
  - Whether you like them depends on application
  - Framework will impose no restrictions
    - “ Signalling application can do arbitrary violence”

3: Local or Long-Distance

- Should entities relate only to peers or over longer distances
  - Affects notification handling (need for reverse routing), “reservation identifier” meaning
- Assumption:
  - Layer split will make NTLP ‘single hop’
  - Applications opting out of reverse path forwarding will have to work out how to secure e2e notifications
  - Framework will discuss for QoS, not finalise

4: Protocol Design Issues

- Addressing (peer-peer vs. e2e)
  - Assumption: say both are needed, depends on detailed design of NTLP
- Need for supporting protocol extensions
  - Not identified in framework so far, leave open
- Layer split & use of supporting transport
  - See later detail & leave as design question

5: Flow Identification

- What parameters to use/are needed to identify a flow and its path and where are they visible
  - Assumption: layer split will put something minimal in NTLP as defining the path
  - Assumption: HA approaches for MIP support are ruled out
  - Assumption: framework will explain problems with policy forwarding, up to NTLP to handle them

6: More Mobility Issues

- Picture of NSIS signalling in overall context of macro-mobility and seamoby-like protocols is missing
  - Impacts what optimisations are ***really*** useful
  - Existing text needs update & review anyway
- Framework for “reservation identifier” is consequently unclear
  - As is what layer it lives in

7: More Complex Scenarios

- Periodically, points raised about the even-cleverer-things that could be done
  - More complex routing interactions (automatic traffic engineering)
  - More complex charging/authorisation models
- Assumption:
  - Text can be included if people want to send it, but it will be marked as out of scope for current work

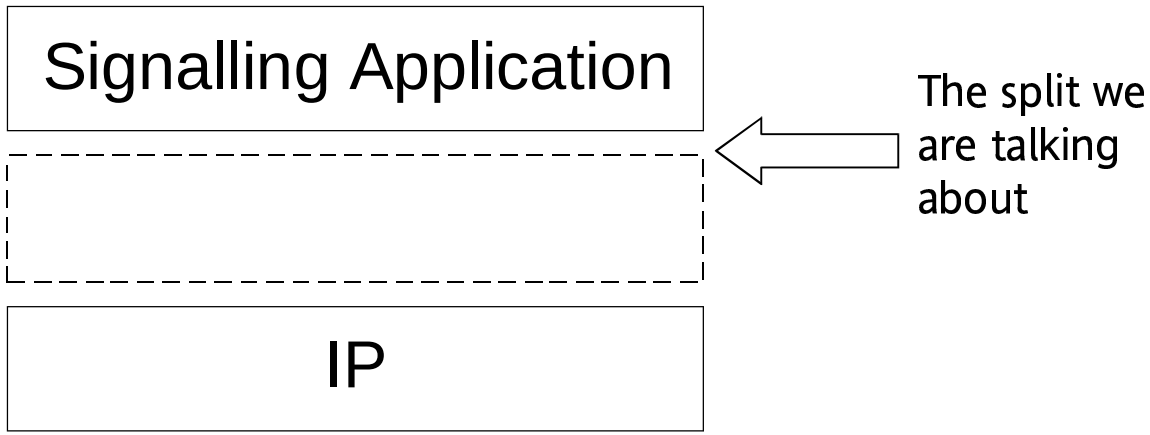
8: Pathdecoupledness

- Existing text is scattered about in framework; could leave as is, but…
- Could say how it gets incorporated economically, also w.r.t. layer split
  - Assumption[?]: NTLP is responsible for delivering messages to ‘right’ nodes, NSLP for describing desired network behaviour
  - Therefore[?]: Path-decoupled case is handled by off-path NTLP variant under common NSLP

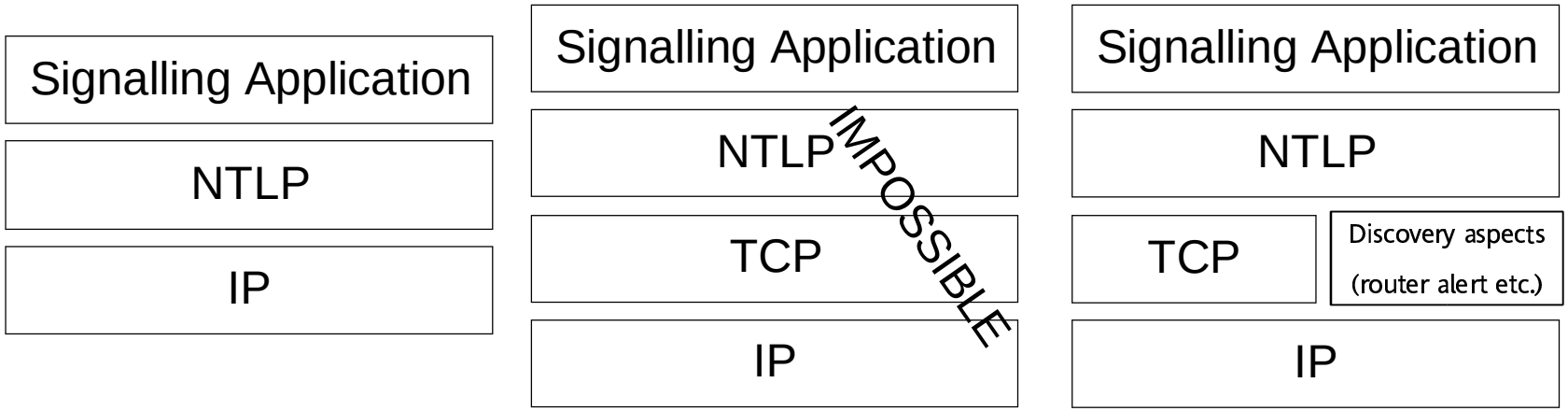
NTLP/NSLP Split Issues

What is the NTLP? (part I)

- The framework says 'the NSIS transport layer' is everything below the signalling application layer and above the IP layer



What is the NTLP? (part II)



- Prefer not to distinguish these cases at the moment
  - Do we need to?

What is the NTLP? (part III)

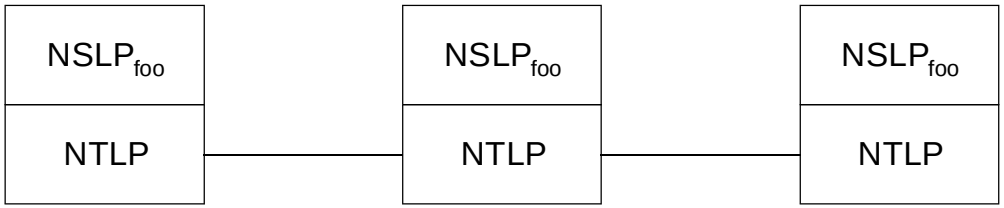
- What are the guiding principles?
  - Should all functionality generic to many NSLPs go into it?
    - Or, expect a building block approach for them?
  - Some functions just ‘go together’
  - Presumably, try to include functions which intimately interact with lower layers
    - Routing? Mobility??
  - Minimise ‘size’ of inter-layer API?

Agreed?

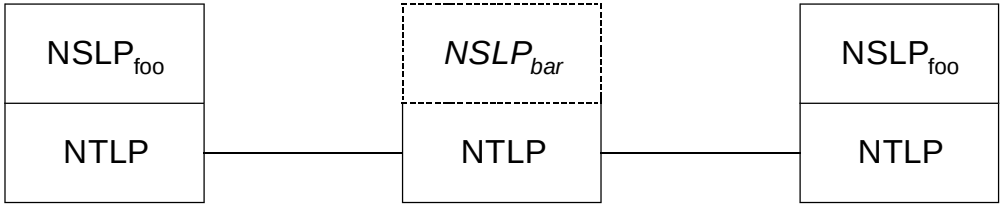
- NTLP only has simple messages (‘send data’)
- NTLP uses internal data to forward messages (i.e. doesn’t rely on NSLP to control routing)
  - This does not rule out route recording
  - NTLP actions at each node cooperate to build e2e path
- NTLP should not address sender/receiver orientation issues
  - It only discovers downstream peers, but must be capable of forwarding messages upstream
  - Is reverse routing required by all NSLPs?
    - Can we save on stored routing state if not?

Single-hop or Multi-hop

- Are NSLP peers joined by a single NTLP ‘hop’?



- Or, might the link between ‘like’ NSLPs run over more than one ‘hop’ concatenated together?



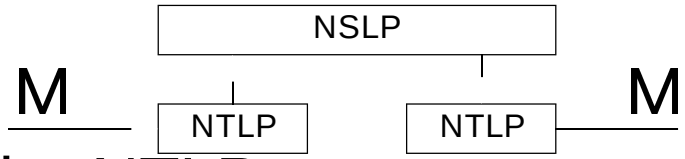
- Does this non-end-to-end-ness of the NTLP restrict the appropriate functionality?
  - Or, should we design it so case (2) doesn’t arise – i.e. enable it to ‘skip’ interior nodes?

Should be in NTLP (if at all)

- Traditional transport-layer like functionality
  - Congestion control
  - Various reliability aspects
    - Recovery from congestive loss?
    - Guaranteed delivery?
  - Bundling & Segmentation
  - In order delivery
  - Duplicate detection/removal
  - Framing
  - Flow control
- Do different NSLPs have different requirements?
  - And should any of these be made optional?

Flow Identification

- Should there be flow information (e.g. 5-tuple) in NTLP for NATing and policy routing reasons?
  - NB: not full packet classifier – in signalling application
  - Should wildcarding be allowed?
- Can NSLPs update objects/fields in NTLP?
  - E.g. addresses, other identifiers
  - Yes? When that message is locally delivered:



NB: flow of control, not message

  - Does the NTLP put any constraints on this (e.g. timing)?

State Management

- Should the NTLP provide a state management service to signalling applications?
  - Should it provide a service to NSLPs to create opaque state blobs and manage it for them?
  - Should the NTLP be used just to get opaque data to between NSLP peers?

State Management (more)

- Different NSLPs have different qualitative state management requirements/semantics.
  - E.g. soft state (explicit or implicit refresh), timeout = deletion or just ‘maybe not kept’, even hard state…
- Different quantitative aspects
  - E.g. Lifetimes, lifetime precision, refresh criticality
- What would integration with the NTLP buy you?
- Putting state management functions in NTLP makes the NTLP API much more complex
- If NSLP peers aren’t always NTLP peers then even harder to define and analyse

Scoping

- Do we need message/object scoping?
  - E.g. the ability to restrict certain messages to certain ‘regions’ of the network
  - Relates also to ‘last node’ problem
- Should message/object scoping be performed in the NTLP or NSLP?
  - Does the NTLP define scopes, but not enforce them?

Rerouting / Mobility Events

- Where and how are mobility events detected? (i.e. who cares about them – NTLP or NSLP?)
- How are they handled?
  - Does NTLP or NSLP perform merging/deletion?
  - Is it signalling application specific? (may depend on authentication/authorisation aspects)

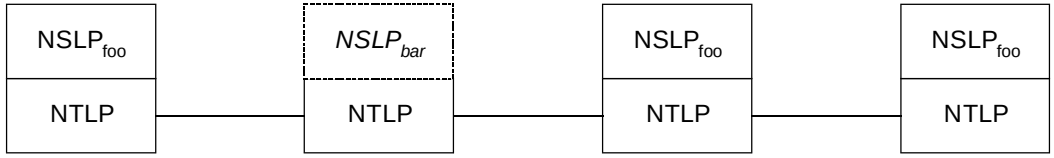
  - Note: this issue relates to where any “reservation identifier” should go
  - Note: these events may involve previously non-adjacent xxxP peers talking to each other

State/session teardown

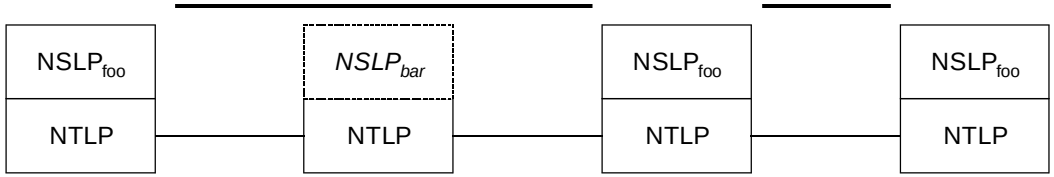
- How are NTLP and NSLP state related?
  - Does tearing down NSLP state automatically teardown supporting NTLP state?
  - Should NSLPs be allowed to do this?
- Dead peer or transport layer connectivity loss
  - Does the NTLP have a concept of a connection between peers?
  - What should NTLP do if the connection fails?
    - Tell NSLP (and let it make the decision)
    - Deletes NSLP state
    - Or, abandon the concept after all…
  - When might you want NSLP state to persist in such circumstances?

Security Aspects

- Where do we provide message protection (confidentiality, integrity) protection?
  - Peer-to-peer at NTLP?



  - Peer-to-peer at NSLP only?



- Which types of DoS attack should be prevented at the transport level?

That's it !

http://www.ietf.org