Diameter Overload Control Solutions: Issue Discussion

draft-campbell-dime-overload-issues-01

Issues Discussion

- Issues that seem to cause confusion
- Issues that have generated list discussion
- Not Exhaustive
  - Example: No piggybacking vs dedicated application slides here
- Trying not to be specific to one particular proposal (but not always successful)

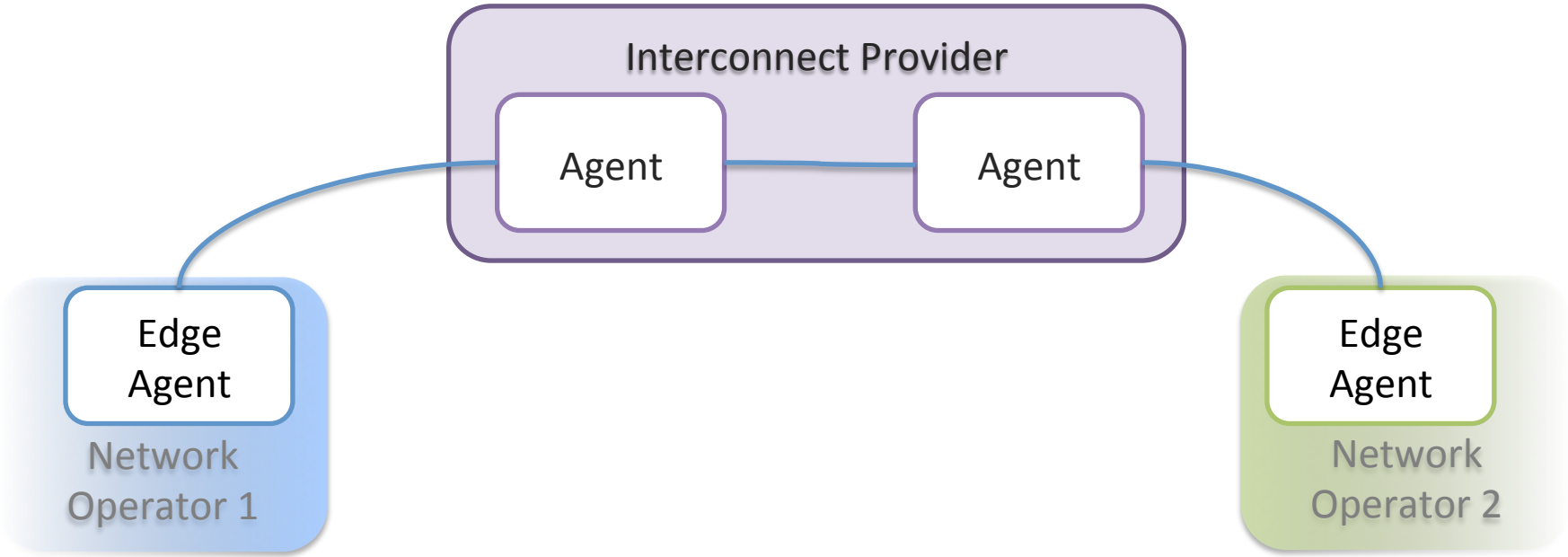
Non-Adjacent Overload Control

- Overload information exchanged between Non-Adjacent nodes
  - … that don't share a transport layer connection
  - … that are separated by a Diameter agent.

Non-Adjacent Use Cases

- Operator Interconnect – Operators exchange overload info across a third party interconnection service
  - e.g IPX
- Non-supporting agent – Nodes share overload info across a Diameter agent that does not support the OC mechanism.

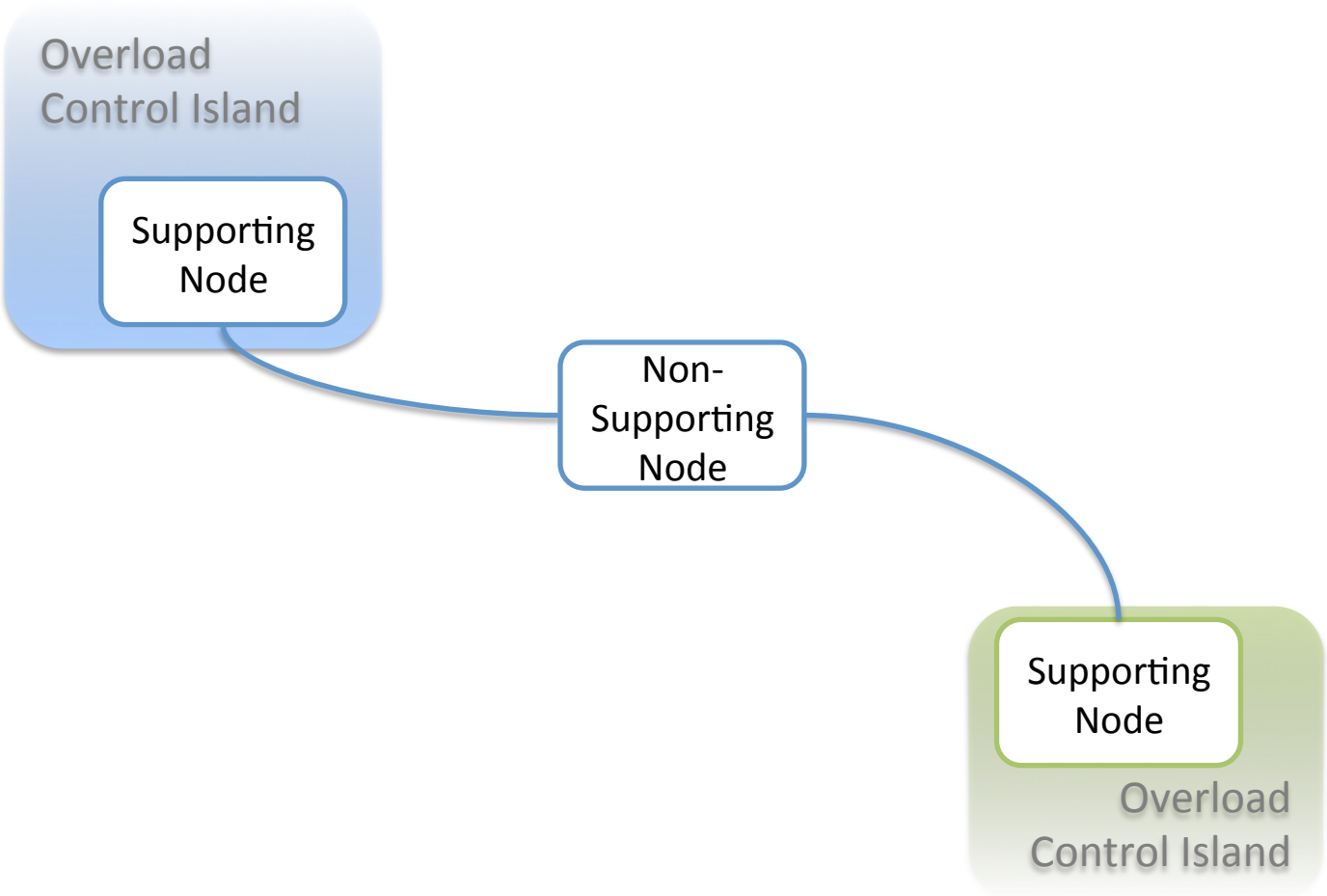
Operator Interconnect Use Case



Interconnect Characteristics

- Interconnect provider may not support OC mechanism
- Operators may not trust interconnect with OC info
- Are operators likely to exchange OC info with other operators at all?

Non-Supporting Agent

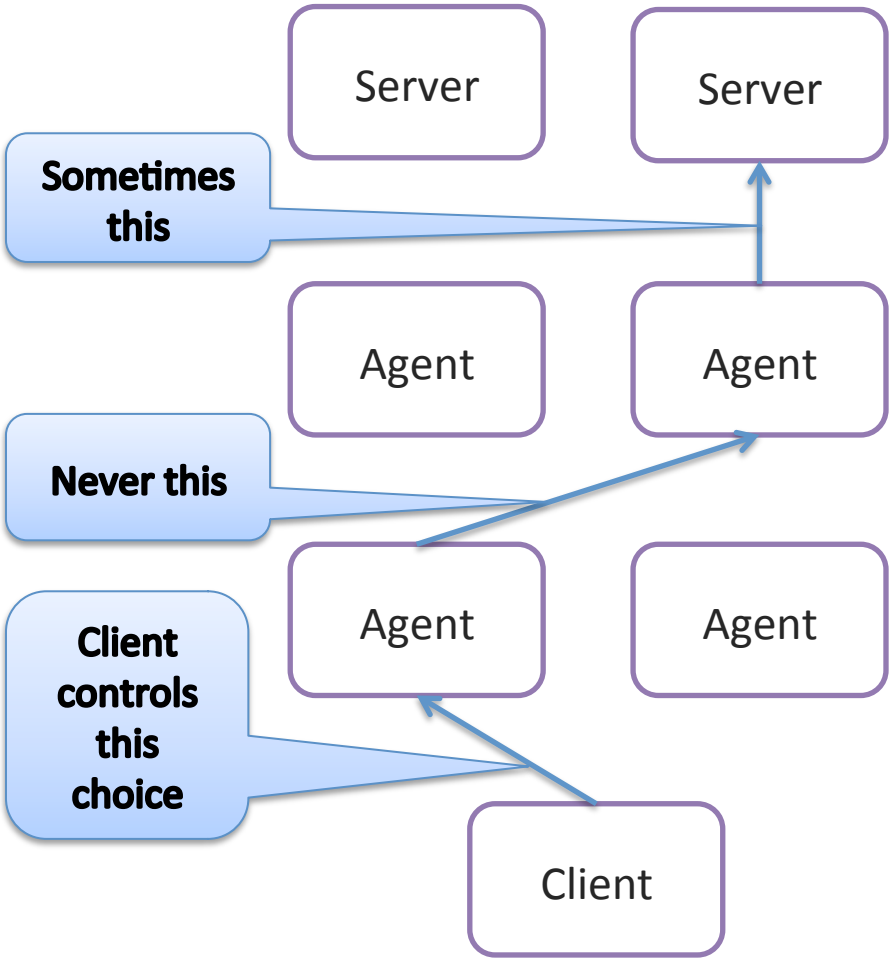


Non-Supporting Agent Characteristics

- Can create “Islands” of OC that can’t share information
- Single administrative domain?
- Will clients and servers support OC before agents do?
  - Seems likely to be the opposite

Non-Adjacent Topology Issues

- A node must know which requests may hit an overloaded node
  - A node selects its peers
  - Sometimes selects the opposite endpoint
    - i.e. Destination-Host
  - Never selects non-adjacent agents



Non-Adjacent OC Negotiation

- Nodes need to negotiate or declare OC support, and possibly negotiate parameters
- Easy for adjacent case – just use capabilities exchange
- Harder for non-adjacent case
  - Who do you negotiate with?
  - How do you route negotiation messages??
  - Multiple negotiated parameter sets on same transport connection.

Non-Adjacent OC Report Delivery

- Where do you send OC reports?
- Must know topology past the peer
  - Provisioned?
  - Discovered?
    - Is “everyone I’ve seen so far” good enough?
    - Send in Diameter responses until everyone backs off?
- Report Ordering
  - Handled at transport layer for adjacent
  - Not guaranteed for non-adjacent

Non-Adjacent OC Scopes

- Only adjacent peer can act on some scopes
  - Peer Host
  - Connection
  - Must not be sent to a non-adjacent node
  - Must not be acted upon by a non-adjacent node
- Need a way to distinguish adjacent from non-adjacent reports.

Overload Scopes

- What are these “scope” things, really?
  - “classifiers” that determine the set of messages that need to be reduced
  - Overload “contexts”

Overload Scopes

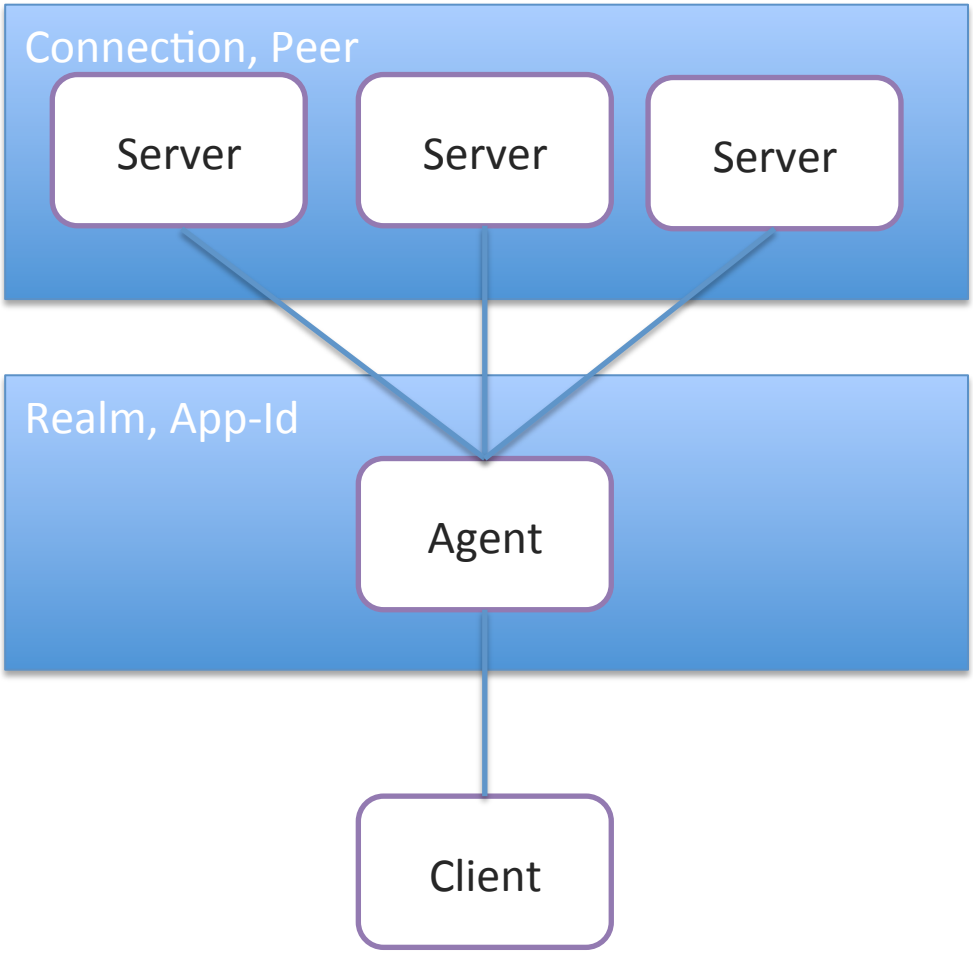
- Req 31
  - Node may be overloaded for some purposes but not others
  - MUST allow granular overload reports, to avoid over reporting
  - MUST allow (at least):
    - Node
    - Realm
    - Application
  - MUST be extensible

Scope Authority Concept

- A node generating an overload report needs full knowledge of the reported scope
  - i.e. has “scope authority” for the scope
- This is easier for some scopes that indicate a specific server
  - Peer Node
  - Connection
  - Destination-Host
- Harder for broader scopes
  - Realm
  - Application-ID

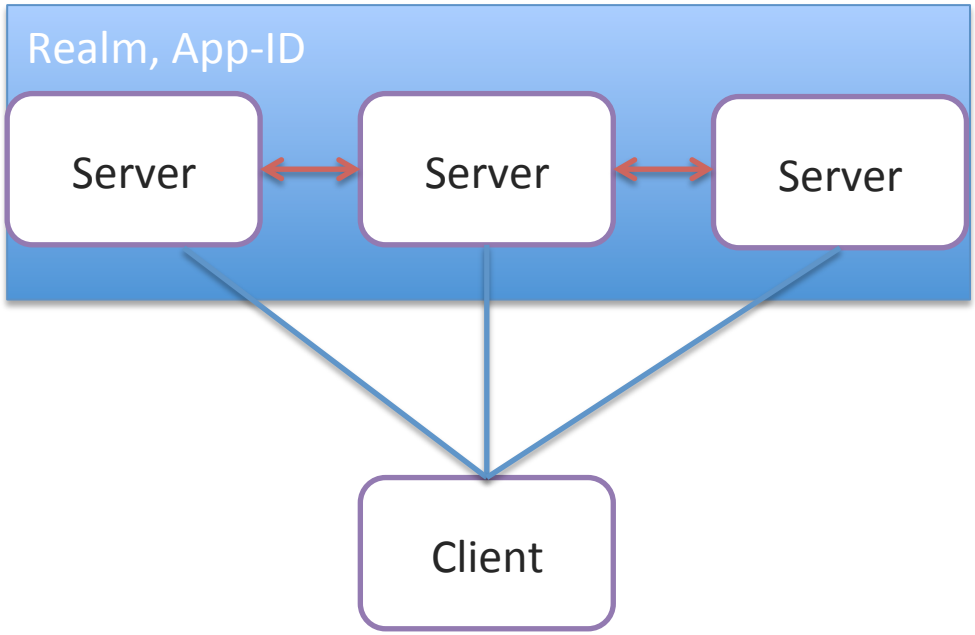
Scope Authority

- Connect, Peer-Host
  - Only needs to know about self
- Realm, App-ID
  - Requires knowledge of all nodes that can handle it



Scope Authority

- Server support of larger scopes is hard
  - Must sync load between all servers in group



Scope Categories

- Explicit – Values explicitly specified in the scope AVPs
- Implicit – Values inferred from the Diameter message that carries the report
  - Only make sense for piggybacked solutions
  - Example: Connection means “this connection”
- Baseline – A scope that always applies
  - Can refine scope by adding more scope AVPs
  - Cannot grow scope beyond baseline
    - » Example: JJacques proposed baseline
    - » “this Realm” + “this App-Id” + “this Connection”
  - draft-tschofenig-dime-overload-arch has a single implicit, baseline scope
- Default – A scope that applies if no scope is specified?
  - Do we need this?

Scope Combinations

- roach-dime-overload-ctrl allows multiple Scope AVPs on same overload report
  - Different types refine the scope
    - Defines a set intersection
    - Example: Realm + Application-Id – request must match both Realm _and_ Application-ID
  - Same types expand the scope
    - Defines a set union
    - Example: Application-Id1 + Application-Id2 – the request must match one or the other Application-Ids.

Mandatory Scopes

- Nothing mandatory to _send_.
- Do we need any mandatory to understand?
  - Options:
    - At least one mandatory to accept for interop purposes
      - Example: Connection, maybe Application-Id
    - No mandatory to accept
      - But either require at least one scope to be specified or define a default
      - Or define a baseline scope

Deprecate scopes?

- Should we deprecate the following?
  - Host (aka Peer-Host)
    - Mostly redundant with Connection
      - But might save having to send the same report to the same node multiple times, depending on node design.
    - May be hard to implement with certain cluster or blade architectures
  - Session
    - Too granular
    - Likely to have thousands or millions of sessions

Thanks for Listening!