ALTO Topology Extension

draft-yang-alto-path-vector-00
draft-yang-alto-topology-06

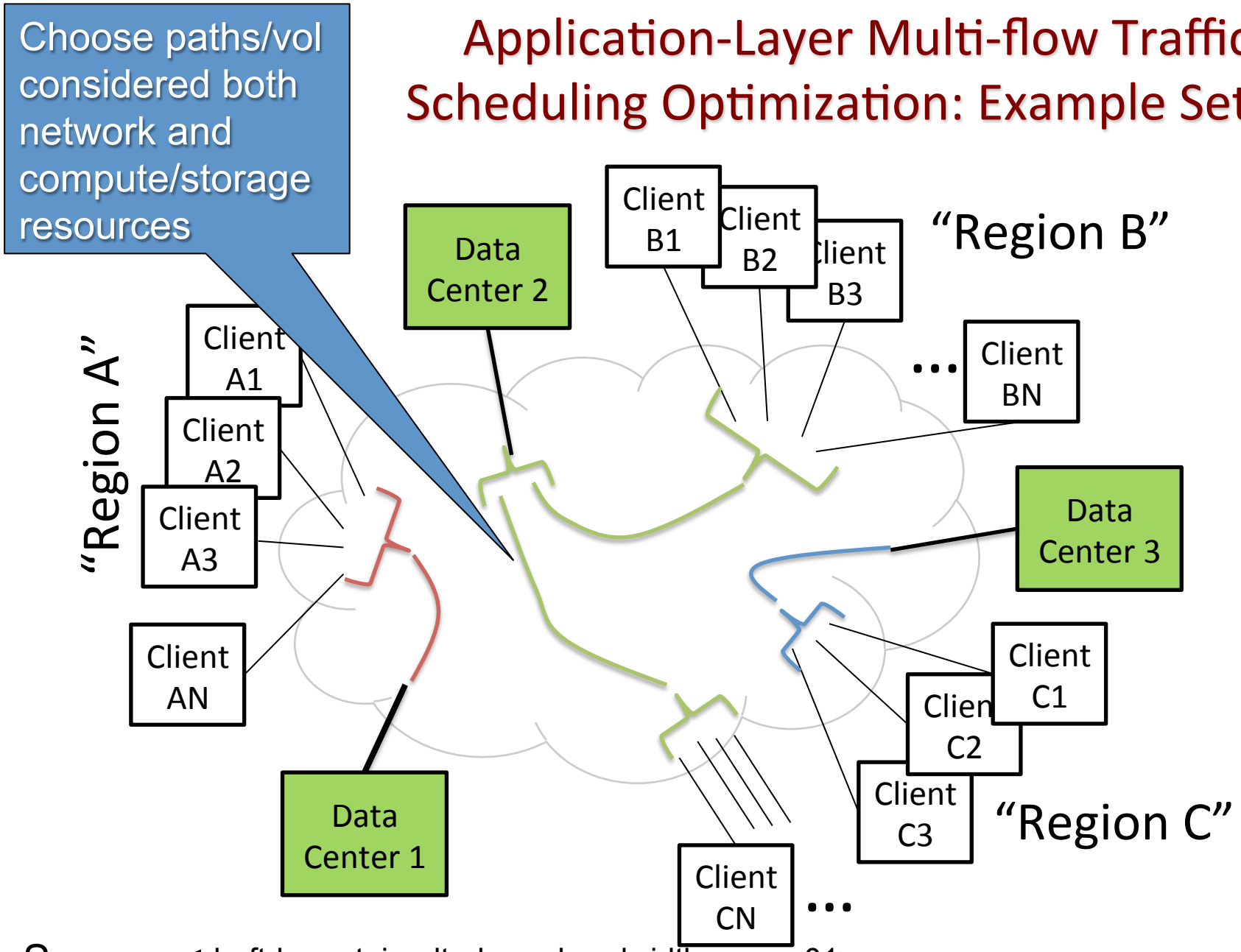
Presenter: Y. Richard Yang

March 26, 2015 @ IETF 92

Overview of Document Change

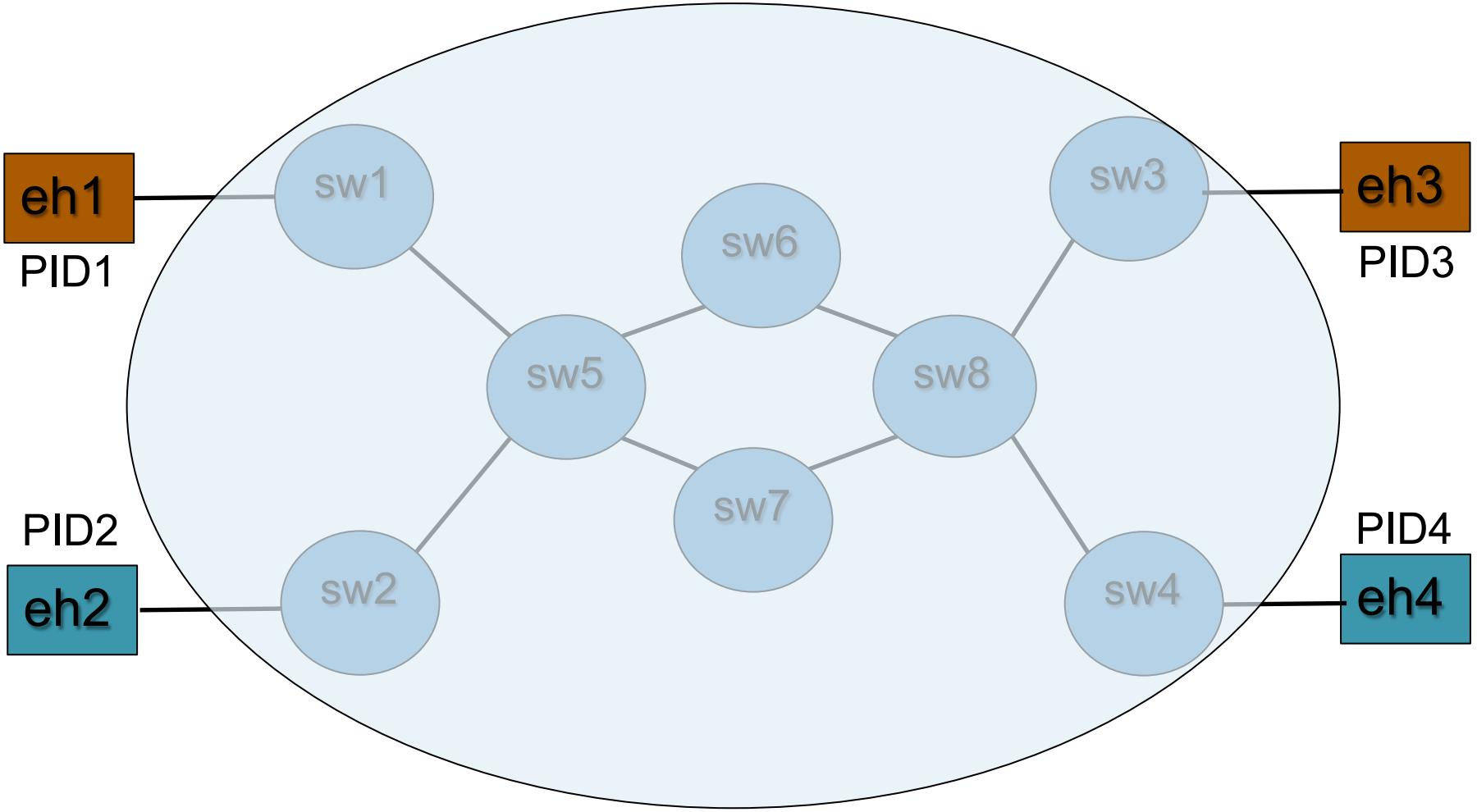
- Split `draft-yang-alto-topology` into two
  - draft-yang-alto-path-vector-00
    - focuses on path vector, as an abstraction to convey abstract network routing topology constraints
  - draft-yang-alto-topology-06
    - focuses on general ALTO node-link graphs
- Add a non-normative section on how path-vector allows general application-layer traffic scheduling optimization (to be complete)

Application-Layer Multi-flow Traffic Scheduling Optimization: Example Setting



Source: <draft-bernstein-alto-large-bandwidth-cases-01>

Application-Layer Multi-flow Traffic Scheduling Optimization: Toy Example



- App computes traffic volumes for two flows:
  - PID1 (eh1) -> PID3 (eh3); PID2 (eh2) -> PID4 (eh4)

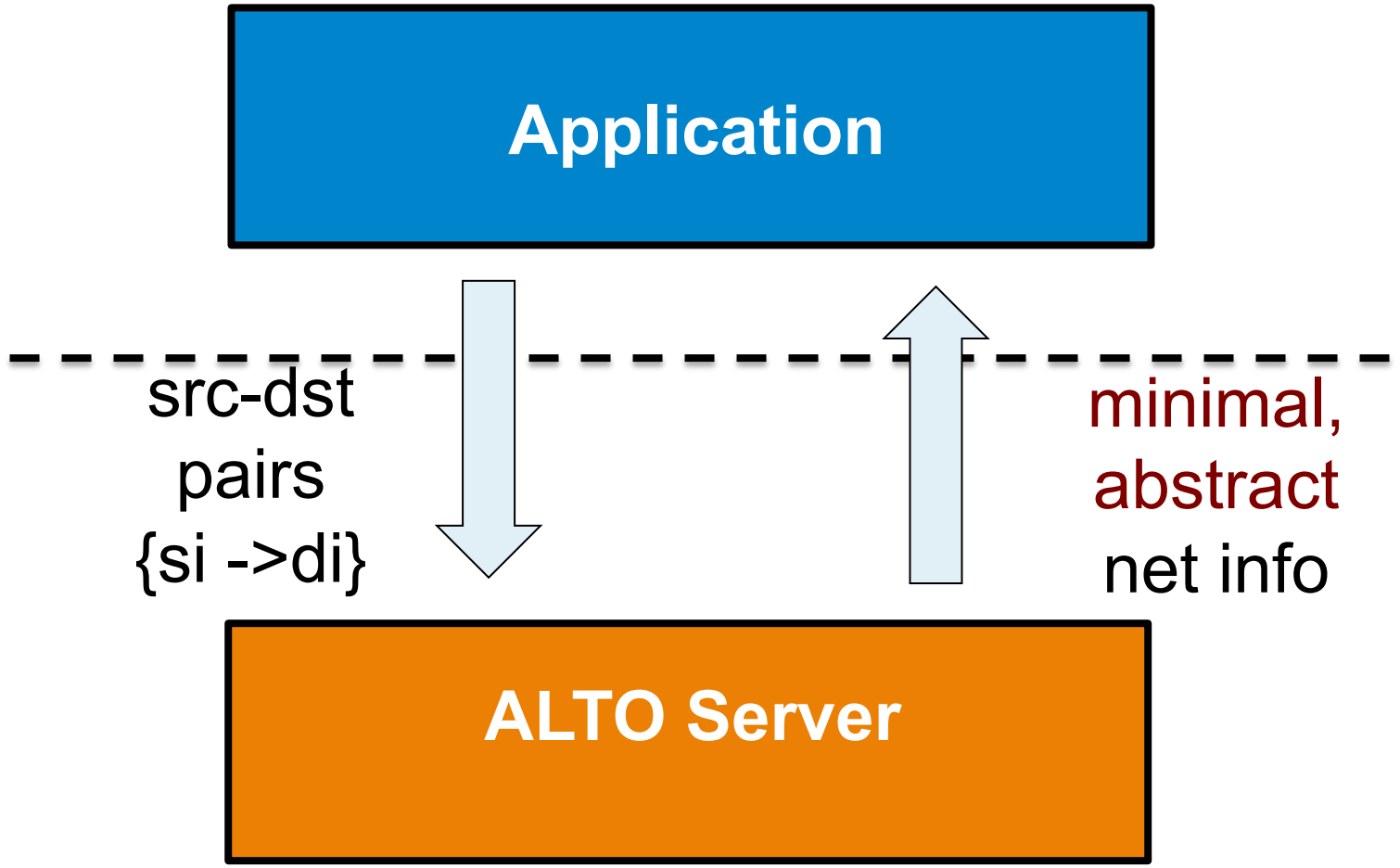
Application-Layer Multi-flow Traffic Scheduling Optimization (ALMPO): Framework

- App has a set of K flows {si -> di}
- Path pi for si->di is computed by network
- App computes traffic volume xi on path pi (si -> di)
  - $x = [x_1, x_2, \ldots, x_K]$
  - App traffic volume optimization framework:

min/satisfy obj(x)
s.t.
x satisfies app constraints
x satisfies network constraints

Convey minimal network info to inform app the parameters

App/Net Interaction



Two Types of Network Constraints/Parameters

- Individual path, e.g.,
  - obj = x_1 * routingcost(s_1 -> d_1) + x_2 * routingcost(s_2 -> d_2)
  - 100 ≤ x1 ≤ bw(s_1 -> d_1)

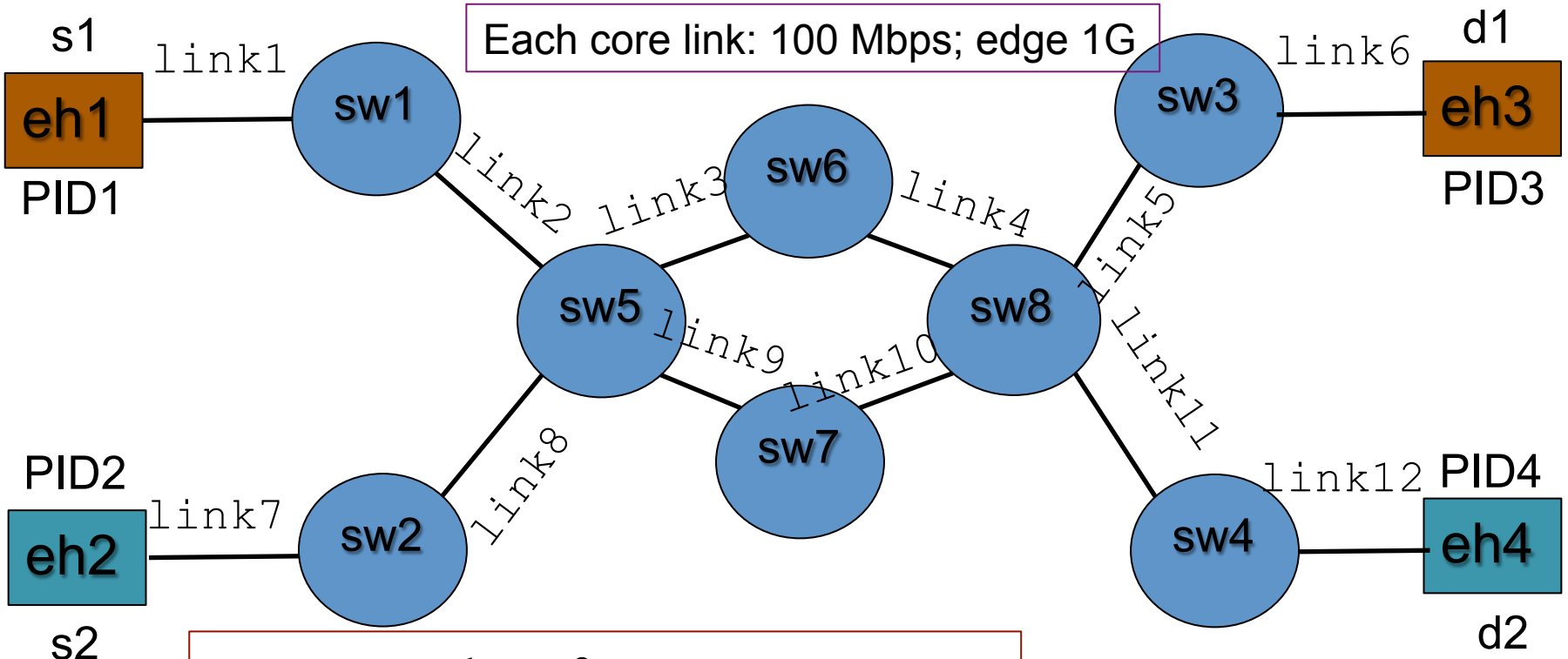
Existing E2E cost maps work well already.

- Coupled paths, e.g.,
  - any link e: sum(x_i: (s_i -> d_i) uses link e) ≤ bw_e

Convey such info will involve network topology.

Q: Do we need to disclose whole paths?

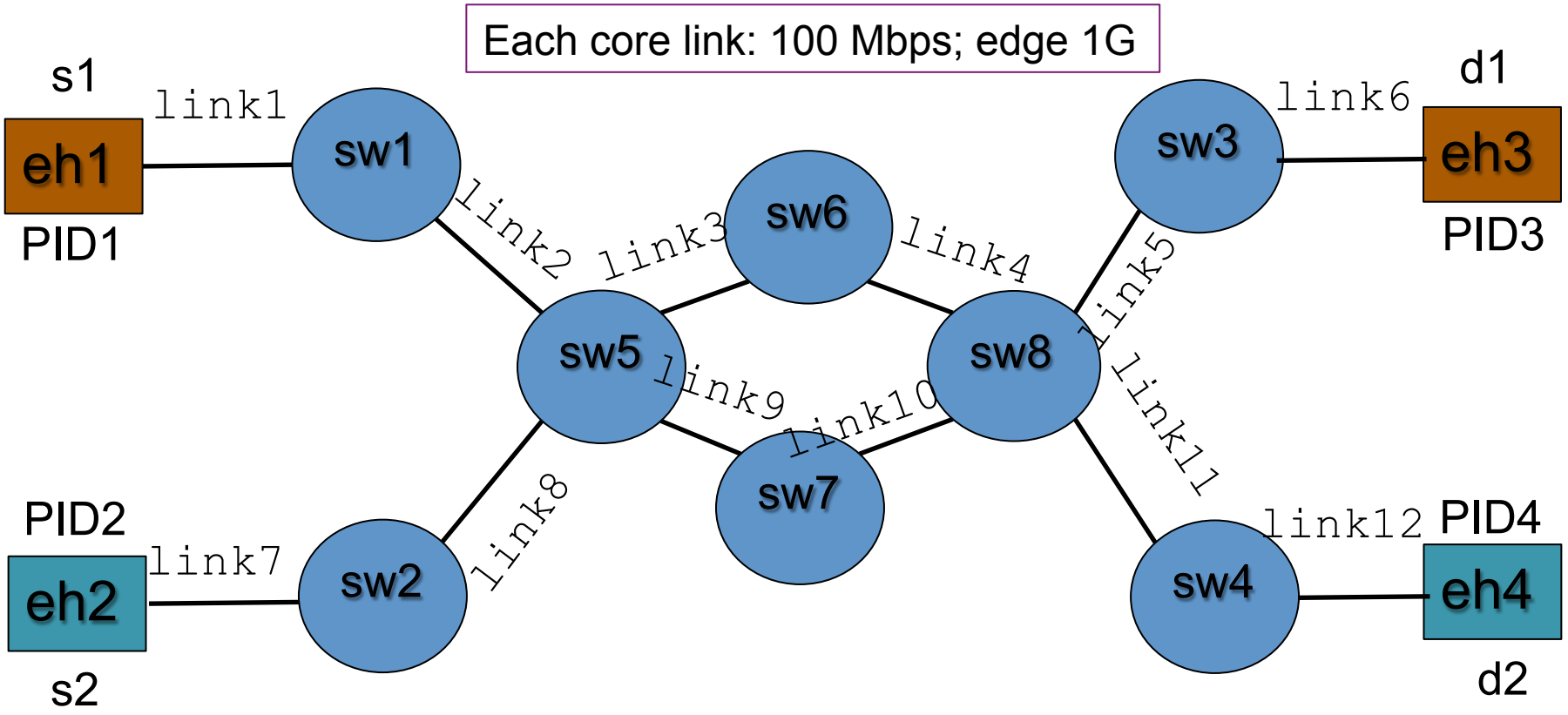
Insight: Example



	x1	x2	var <= cap	
link1	1	0	x1	1G
link2	1	0	x2	100M
link3	1	1		100M
Link4	1	1		100M
link5	1	0		100M
link6	1	0		1G
…				
linkM				

Fast (redundancy elimination) algorithm to compute minimal network elements topology to expose.

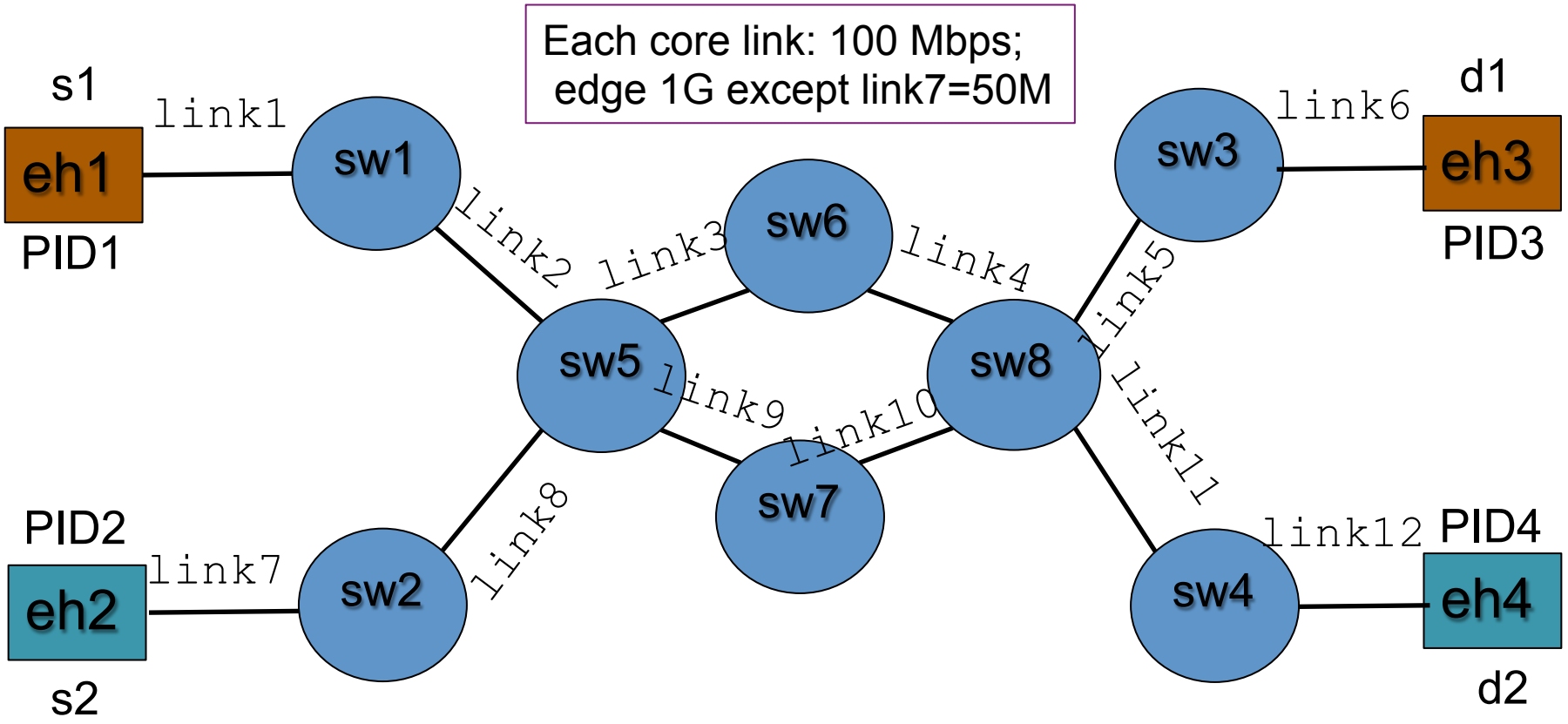
Insight: Example



s_1 -> d_1: { ane_1(≤100M) }

s_2 -> d_2: { ane_1(≤100M) }

Insight: Example



s_1 -> d_1: { ane_1(≤100M) }

s_2 -> d_2: { ane_1(≤100M), ane_2(≤50M) }

Summary

- Path-vectors of abstract network elements provides simplified, abstract routing topology information to applications to convey coupled constraints, such as shared bottlenecks, shared reliability risk.
  - The path vectors may not even form a graph, but conveys the coupling info
- The amount of simplification depends on network computation level

Path Vector: Example

HTTP/1.1 200 OK
Content-Length: TDB
Content-Type:
application/alto-costmap+json

```
{  "meta" : {
     "dependent-vtags" : [
       { "resource-id": "my-default-network-map”,
          "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e” },
       {"resource-id": "my-topology-map",
          "tag": "4xee2cb7e8d63d9fab71b9b34cbf76443631554de"
       }
     ],
     "cost-type" : {“cost-metric”: “bw”, "cost-mode"  : "path-vector”  },

     "cost-map" : {
        "PID1": { "PID1":[], "PID2":["ne56", "ne67"], "PID3":[], "PID4":["ne57"]
         },
         "PID2": { "PID1":["ne75"], "PID2":[], "PID3":["ne75"], "PID4":[]
         }, …
     }
}
```

Next Step

- Adopt the path vector approach as one key component of ALTO topology extension

- Missing piece: how to convey ANE properties?

Node-Link Graph Status Update: Key Issue

- Two approaches
  - Unified PID and network node
  - Separate network node from PID
- Mailing list discussion favors a separation design
- Implication of a separation design
  - There are two types of links: between two network nodes vs between a network node and a PID
  - Indicate by link type

```
"links" : {
  "e1" : {"type": "pid-attach",
           "src" : "PID1", "dst": "sw1",
           "costs": ..
  },
  "e2" : {"type": "transit",
          "src" : "sw1", "dst": "sw2",
          "costs": ..
  }, …}
```

Next Step

- How to proceed with the node-link graph effort?

Backup Slides

Bigger Context: Link Type as Property Graph

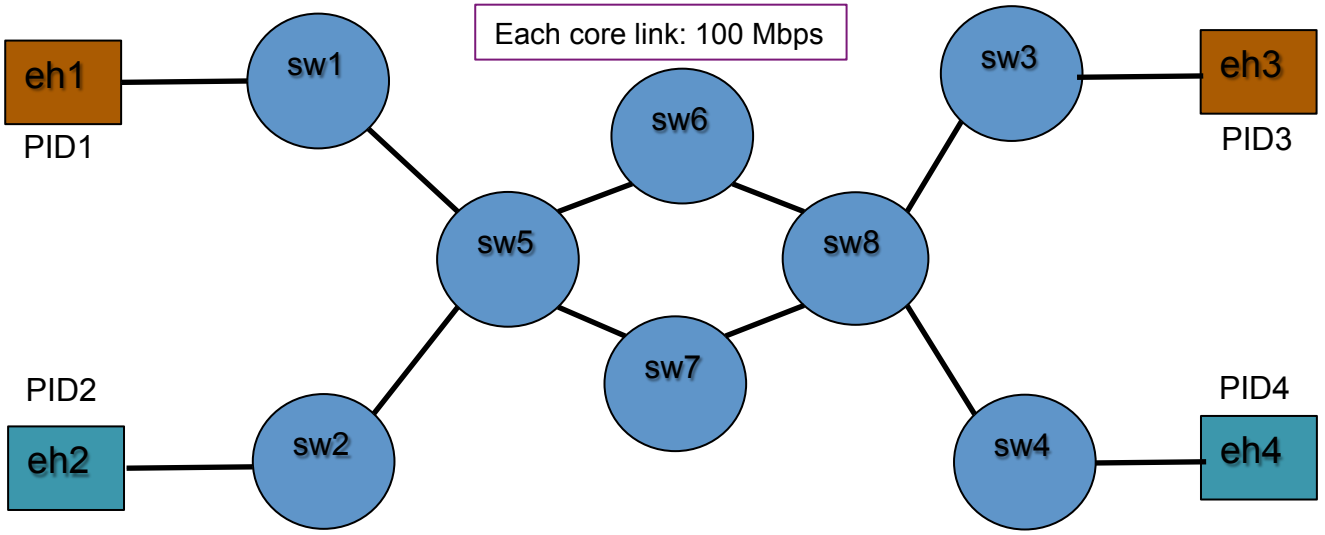
- The "separation design" is a type of property graph
  - A graph consists of a set of element objects, where each object can have any number of key/value pairs associated with it (i.e., properties)
    - Vertex: an object with a unique identifier.
      - each vertex has a set of outgoing edges.
      - each vertex has a set of incoming edges.
      - [each vertex has a collection of properties defined by a map from key to value]
    - Edge: also an object with a unique identifier
      - each edge has an outgoing tail vertex.
      - each edge has an incoming head vertex.
      - each edge has a label that denotes the type of relationship between its two vertices.
      - [each edge has a collection of properties defined by a map from key to value.]

Design Choice: General Flow

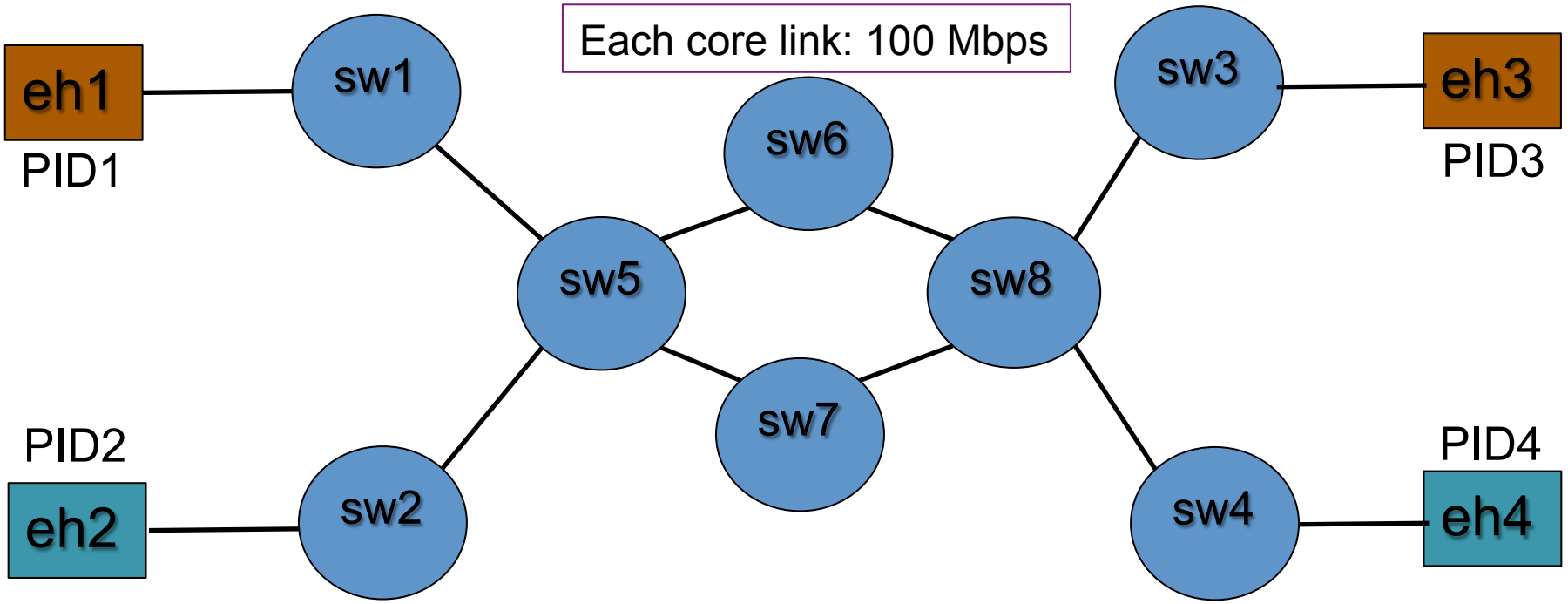
- Handle the case of multi-path routing such as ECMP

```
object {
   JSONNumber w;      // flow weight
   JSONString    ne;     // network element
} FlowElement;

object {
      cost-map.DstCosts.JSONValue -> FlowElement<0,*>;
      meta.cost-mode = ”flow”;
} InfoResourcePVCostMap : InfoResourceCostMap;
```



Multi-flow Scheduling



- ECMP for eh1 -> eh3, single path through sw6 for eh2 -> eh4
  - PID1 -> PID3: [{“ne”: “sw5-6”, “w”: 0.5},
    {“ne”: “sw6-8”, “w”: 0.5},
    {“ne”: “sw5-7”, “w”: 0.5},
    {“ne”: “sw7-8”, “w”: 0.5}]
  - PID2 -> PID4: : [{“ne”: “sw5-6”, “w”: 1},
    {“ne”: “sw6-8”, “w”: 1}]

Discussion: Design Choices

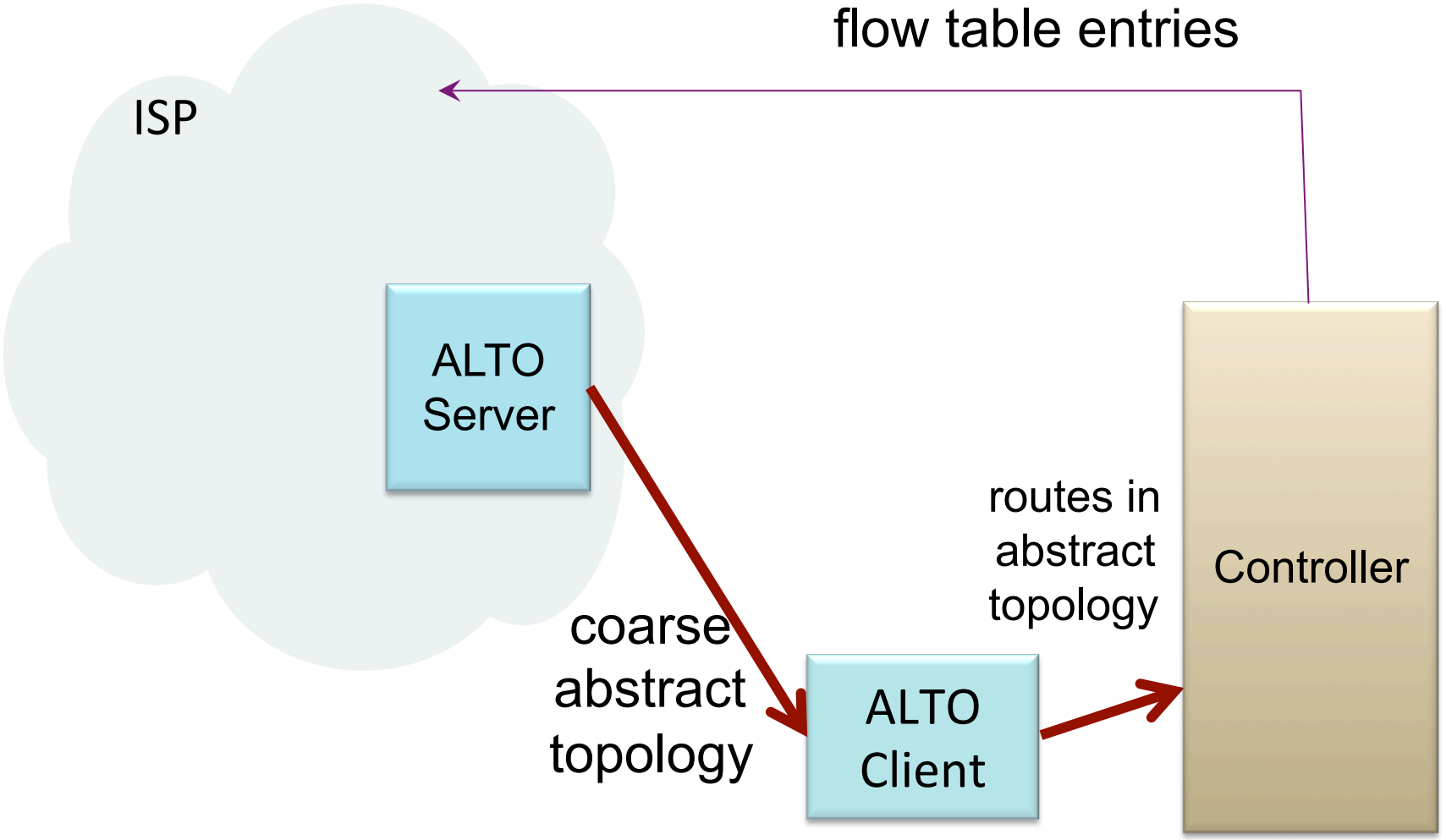
1. Both path vector and flow
2. Extend the path vector mode (no flow mode):
   PIDi -> PIDj: an object of two arrays

```
"cost-map" : {
        "PID1": { "PID1": { "pv": [], "w": [] },
                  "PID2": { "pv": ["ne56", "ne67"], "w": [1, 1] },
        },
…
```

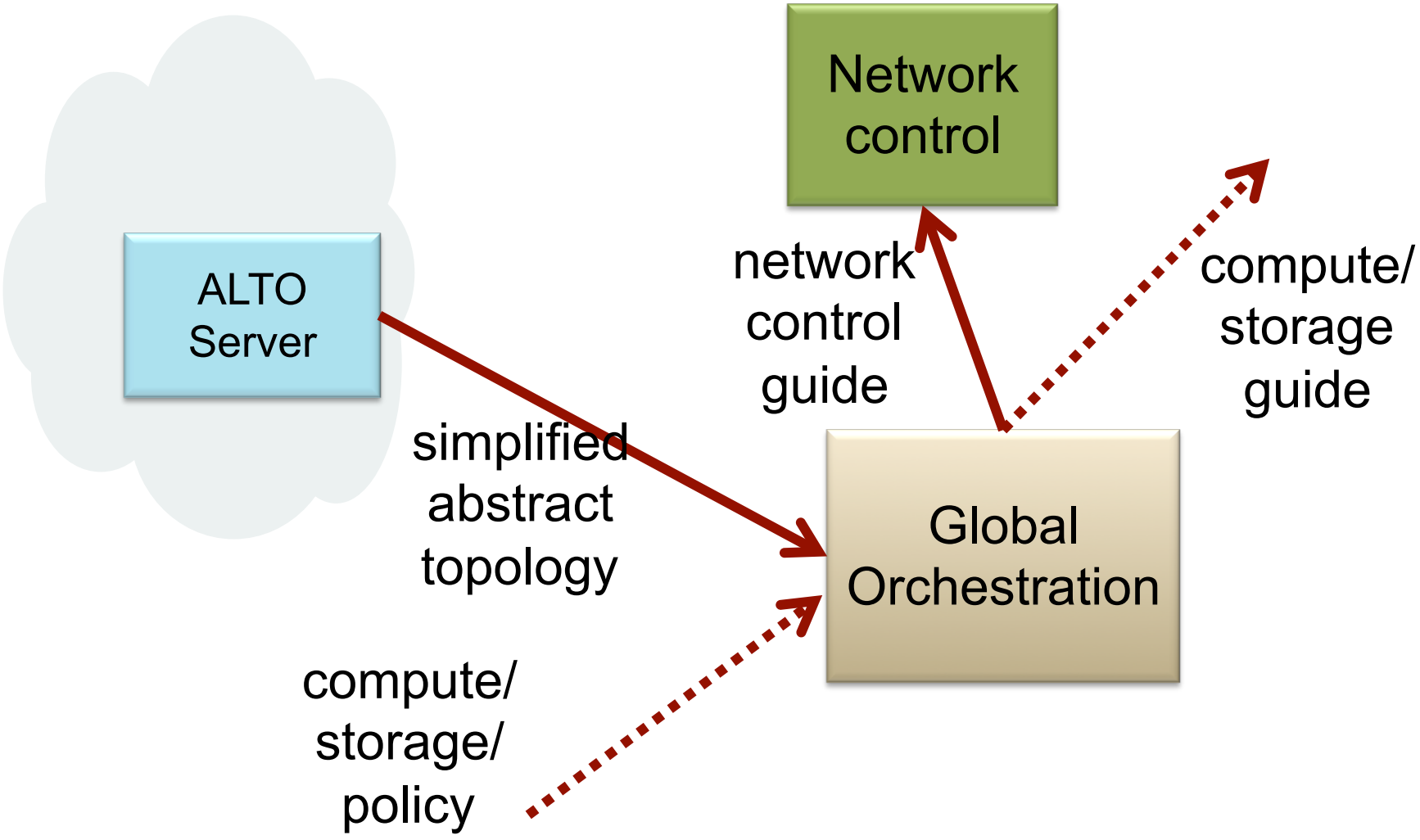
3. Extend the flow mode (no path vector mode):
   PIDi -> PIDj: an array of objects each with two elements

```
"cost-map" : {
        "PID1": { "PID1": [ {"pv": [], "w": [] },
                  "PID2": { "pv": [{"ne": "ne56", "w": 1},
                                   {"ne": "ne67", "w": 1 }
        },
…
```

App-Path-Selection Work Flow



App-Path-Selection Work Flow



YANG Model

```
module: alto-service-topology
  +--ro resources
     +--ro topology-maps*
        +--ro topology-map [resource-id]
           +--ro resource-id    alto:resource-id
           +--ro tag            alto:tag-string
           +--ro nodes* [node-id]
           |  +--ro node-id              node-id
           |  +--ro node-properties* [property-name]
           |     +--ro property-name     string
           |     +--ro property-value?   string
           +--ro links* [link-id]
              +--ro link-id    link-id
              +--ro src        union
              +--ro dst        union
              +--ro type       string
              +--ro cost* [cost-metric]
                 +--ro cost-metric    alto:cost-metric
                 +--ro value
```