Flow Cost Service

draft-gao-alto-fcs-00

Kai Gao [1] J. Jensen Zhang [2] H. May Wang [2]

Y. Richard Yang [3]

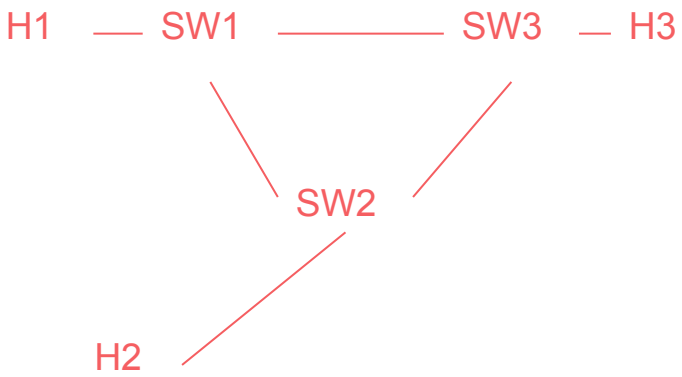
[1] Tsinghua University [2] Tongji University [3] Yale University

July 21@IETF 96

Motivation: Fine-grained Routing

Network routing trends to be fine-grained

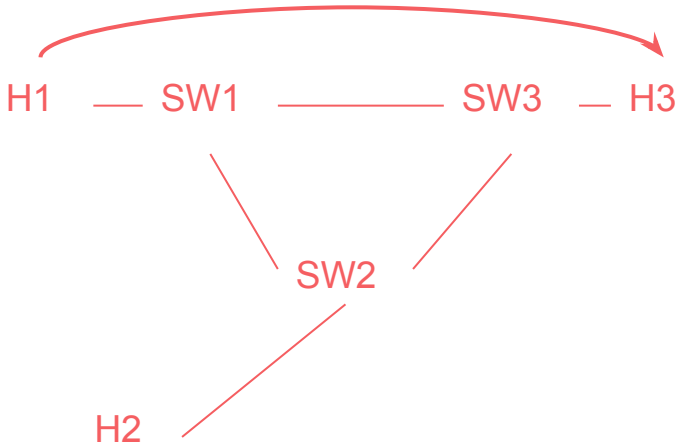
- Expressive
- Accurate



Motivation: Fine-grained Routing

Network routing trends to be fine-grained

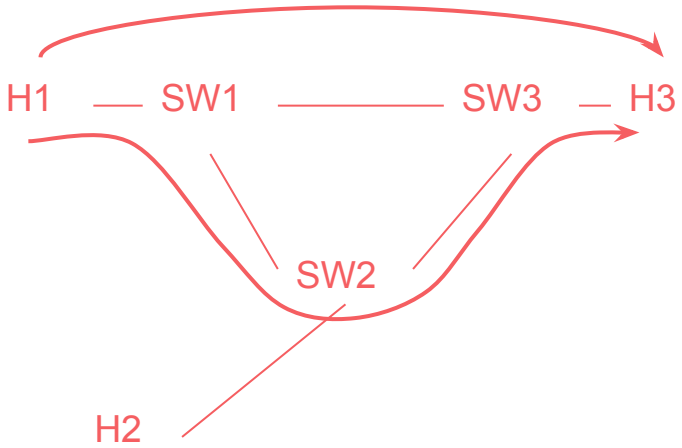
- Expressive
- Accurate



Motivation: Fine-grained Routing

Network routing trends to be fine-grained

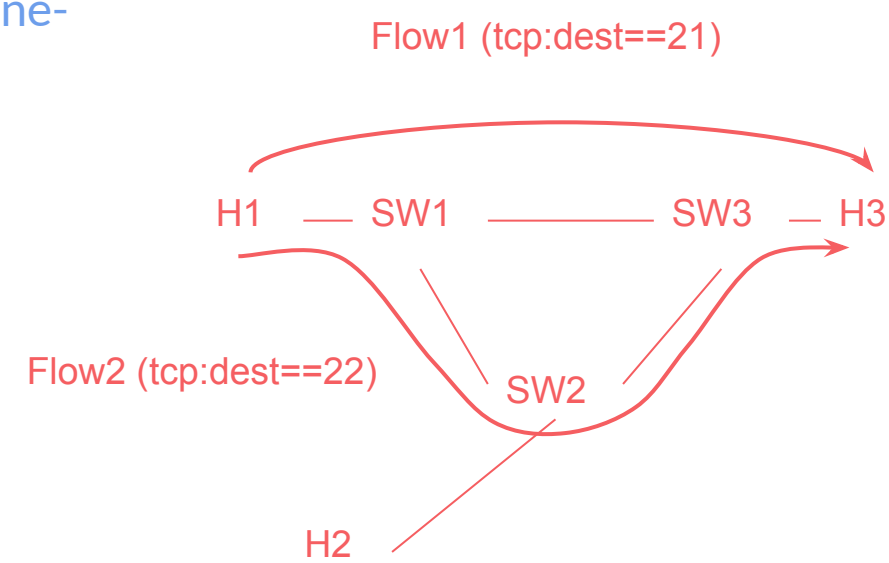
- Expressive
- Accurate



Motivation: Fine-grained Routing

Network routing trends to be fine-grained

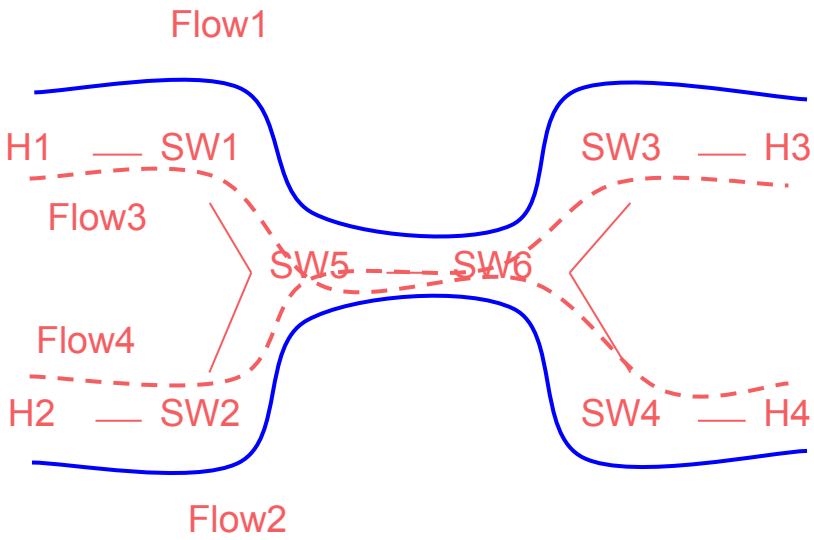
- Expressive
- Accurate



Motivation: Flow correlation

Flow correlation: the costs of different flows are related

- Side-effect
- Non-peer



Flow Expression Encoding

- Flow ID
  - Same format as a PIDName [RFC7285#Section 10.1]
- Typed header field
  - `<protocol-name>:<field-name>` (Subset of OpenFlow match fields)

```
Flow expression:

“ssh-flow”: {
  “ipv4:source”: “192.168.1.2”,
  “ipv4:destination”: “192.168.1.3”,
  “tcp:destination”: 22,
  “ethernet:vlan-id”: 20
}
```

Flow-based vs. Endpoint-based

```
Object {
  FlowFilterMap     flows;
} FlowCostRequest : MultiCostRequestBase;

Object {
  [CostType    cost-type;]
  [CostType    multi-cost-types<1..*>;]
  [CostType    testable-cost-types<1..*>;]
  [JSONString  constraints<0..*>;]
  [JSONString  or-constraints<0..*><0..*>;]
} MultiCostRequestBase;

Object-map {
  FlowId -> FlowFilter;
} FlowFilterMap;

Object-map {
  TypedHeaderField -> JSONValue;
} FlowFilter;
```

```
Object {
  CostType        cost-type;
  [JSONString     constraints<0..*>;]
  EndpointFilter  endpoints;
} ReqEndpointCostMap;

Object {
  [EndpointDescriptor srcs<0..*>;]
  [EndpointDescriptor dsts<0..*>;]
} EndpointFilter;

EndpointDescriptor :=
  protocol:address:port |
  protocol:address
```

Flow-based vs. Endpoint-based (Cont.)

```
{
  “cost-type”: {
    “cost-mode”: “numerical”,
    “cost-metric”: “routingcost”
  },
  “flows”: {
    “l3-flow”: {
      “ipv4:source”: “192.168.1.1”,
      “ipv4:DESTination”: “192.168.1.2”
    },
    “optional-l3-flow”: {
      “ipv4:sourcE”: “192.168.1.1”,
      “Ipv4:destination”: “192.168.1.2”,
      “ethernet:sOuRce”: “12:34:56:78:00:01”,
      “ethernet:destination”: “12:34:56:78:00:02”
    }
  }
}
```

```
{
  “cost-type”: {
    “cost-mode”: “ordinal”,
    “cost-metric”: “routingcost”
  },
  “endpoints”: {
    “srcs”: [“ipv4:192.168.1.1”],
    “dsts”: [
      “ssh:192.168.1.2”,
      “http:192.168.1.2”,
      “tcp:192.168.1.3:6655”
    ]
  }
}
```

Flow-based vs. Endpoint-based (Cont.)

- Filter encoding: `EndpointFilter -> FlowFilterMap`
- Response encoding: `EndpointCostMap -> FlowCostMap`
- Capability: No special capabilities `-> FlowCostCapabilities`

Cost Confidence for Ambiguous Paths

- The problem of ambiguous paths exists for both FCS/ECS
- Cost confidence: indicate the ambiguity of a query
- Examples:
  - Combine the results of all paths and use standard deviation:
    `1 - | deviation / mean |`
  - Select only one path and use the probability:
    `P(|selected path|)/P(all possible path)`

```
{"meta": {"cost-type": {
  "cost-mode": "numerical",
  "cost-metric": "routingcost"
  },},
  "flow-cost-map": {
    "l3-flow": 10,
    "l3-flow-aggr": 50,
    "optional-l3-flow": 5,
  },
  "flow-cost-confidences": {
    "l3-flow": 70,
    "l3-flow-aggr": 40,
    "optional-l3-flow": 90
  }
}
```

Error and Warning

- Three kinds of errors: Conflict/Missing/Unsupported
- Allow accurate location of errors
- Can be extended to allow partial failures and partial recoveries (useful when combined with incremental updates)

```
object-map {
  FlowId -> FlowCostError;
} FlowCostErrorMap;

object {
  [TypedHeaderField conflicts<2..*>;]
  [TypedHeadreField missing<2..*>;]
  [TypedHeaderField unsupported<1..*>;]
} FlowFilterError;
```

Compatibility

- Support all cost types and possible extensions
  - Multi-cost
  - Calendar
  - Path vector
- Support incremental updates
- Have no side-effect on legacy clients/servers

Summary

- Expand the ID space for endpoints (support fine-grained routing)
  - Original (ECS): IP addresses/prefixes
  - `draft-wang-alto-ecs-flow`: Tuples encoded as URI
  - FCS: Tuples similar to OpenFlow match
- Introduce the flow-based filter
  - Use case: flow scheduling
  - ECS may not be efficient
- Response and errors
  - Flow-based cost map
  - Cost confidence: evaluating the effects of ambiguous paths
  - Flow-based error map

Future work

Design related:

- How can clients give accurate queries?
- How about if the client cannot decide the flow configuration?
  - For example, a client must query a flow with tcp:source port for fine-grained result. But the client cannot decide which tcp:source port will be used when the application executed.

Implementation related:

- How to explore ambiguous paths efficiently to compute cost confidence

Thank you!

Backup Slides

Motivation: Fine-grained Routing

Network routing trends to be fine-grained

- Expressive
- Accurate