SUPA Declarative Policy

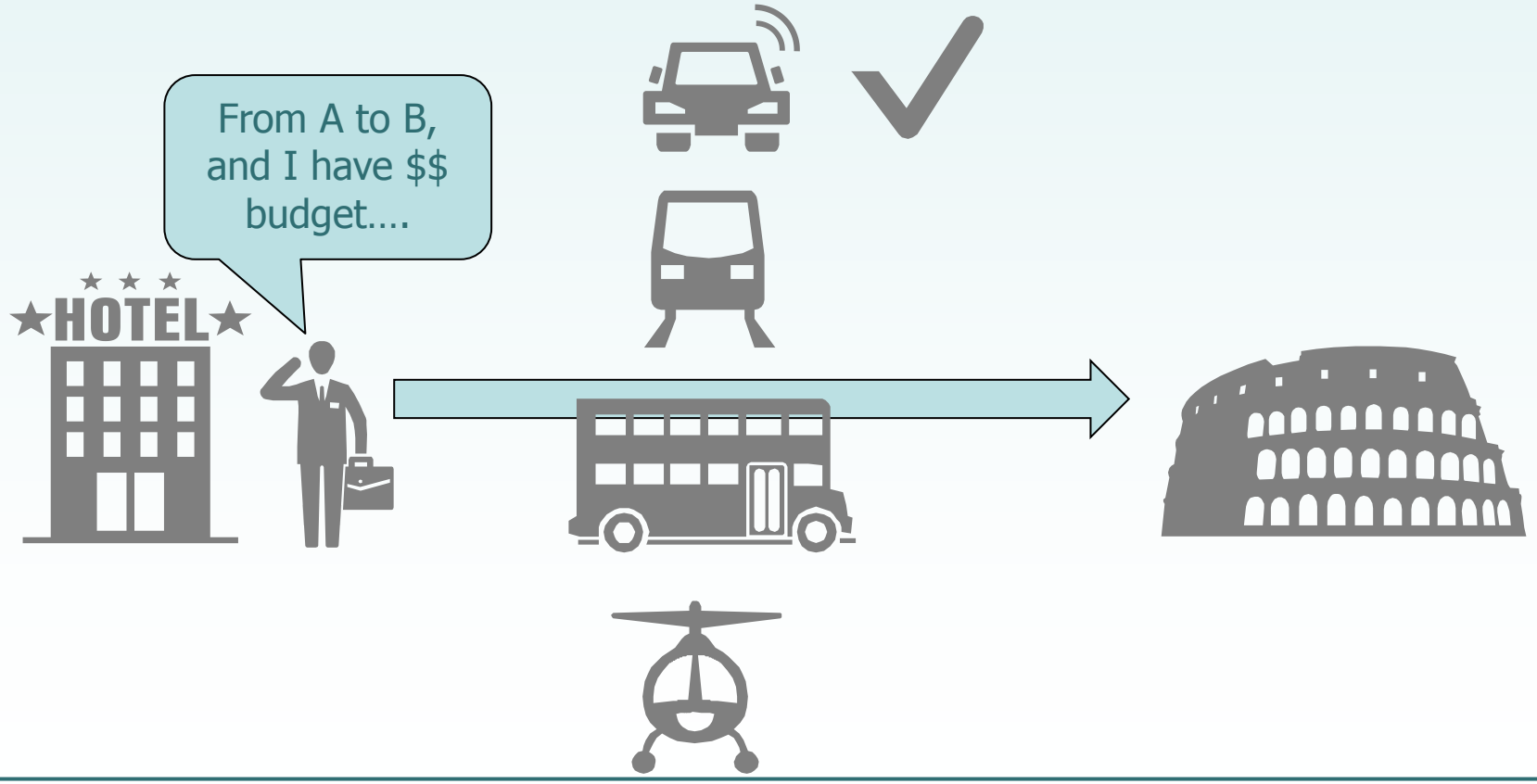
Jun Bi

Tsinghua Univ./CERNET

draft-bi-declarative-policy-00

Goal

- The goal, objective, high level request
- Express what should be done without telling how

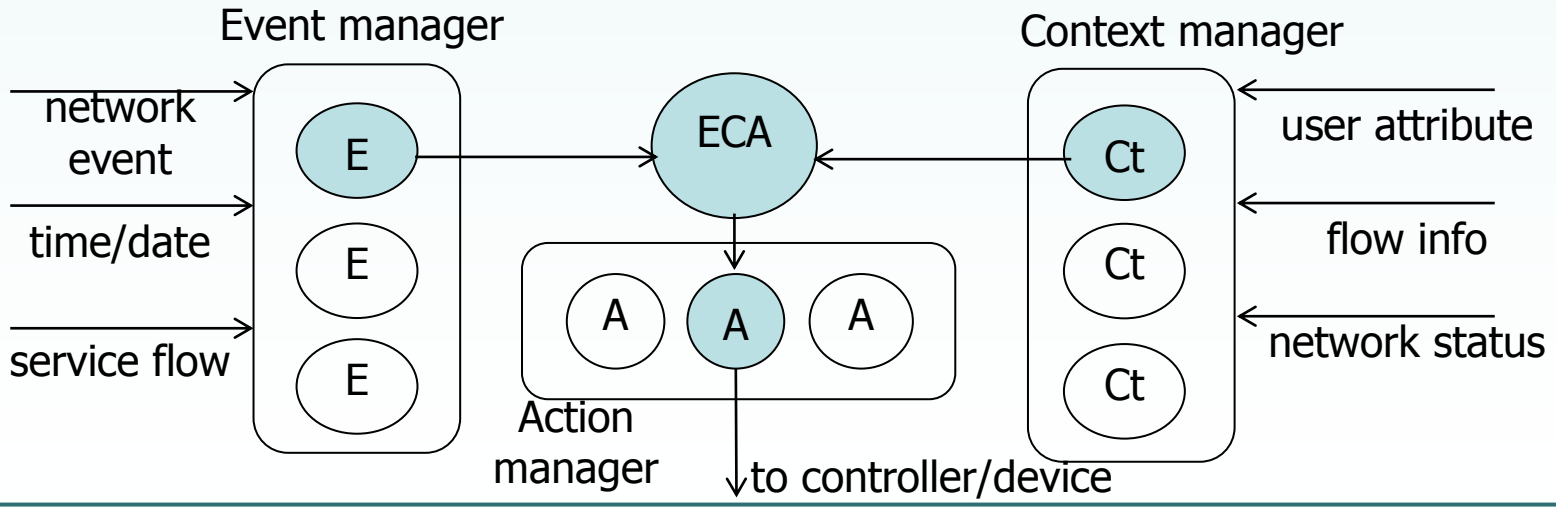


Declarative Policy

- Policy
  - Defines how policy rules are used to manage service behavior
- Policy Model
  - Defines rules for governing managed objects
  - Defines representation for rules
  - Can be used to ***govern service relationships***
- Declarative Policy
  - More abstracted
  - More service level
  - Device independent

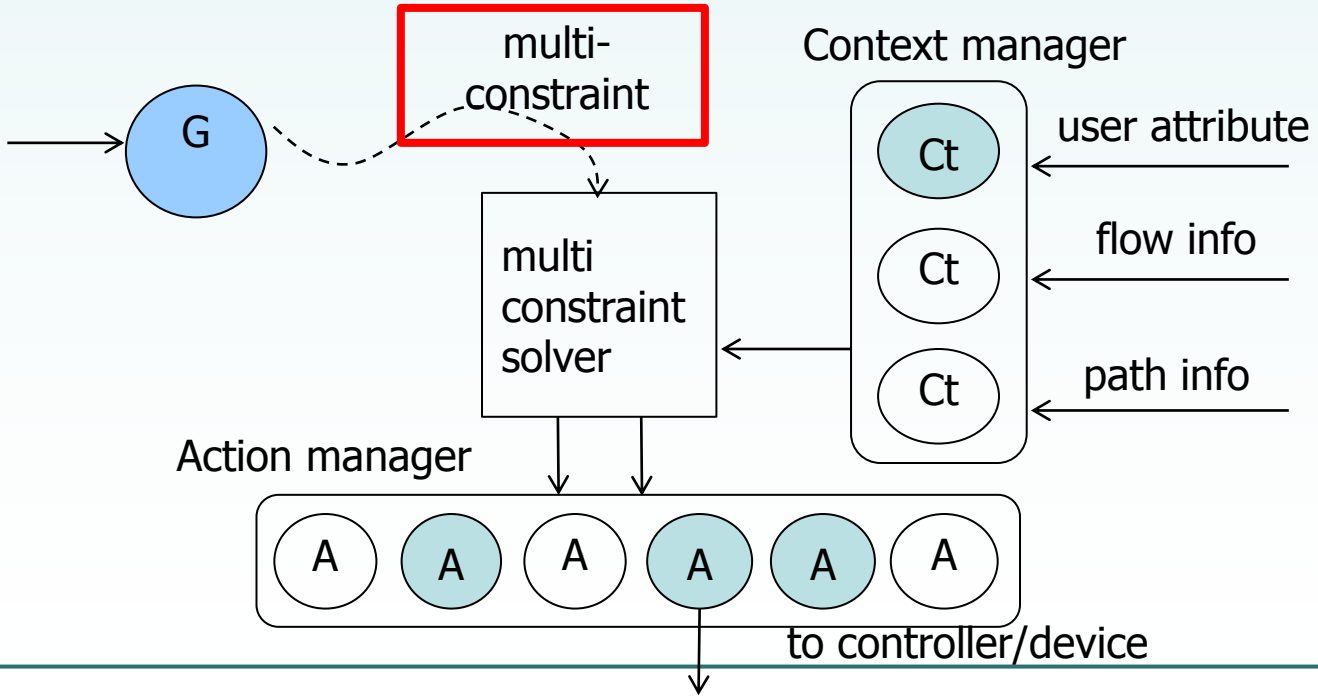
Types of Policy (1)

- **Event-Condition-Action (ECA)**
  - IF the Event is TRUE
    - IF the Condition is TRUE
      - THEN execute the Actions
  - Explicit programming of which condition and which action to be chosen (rationality is in the policy!)



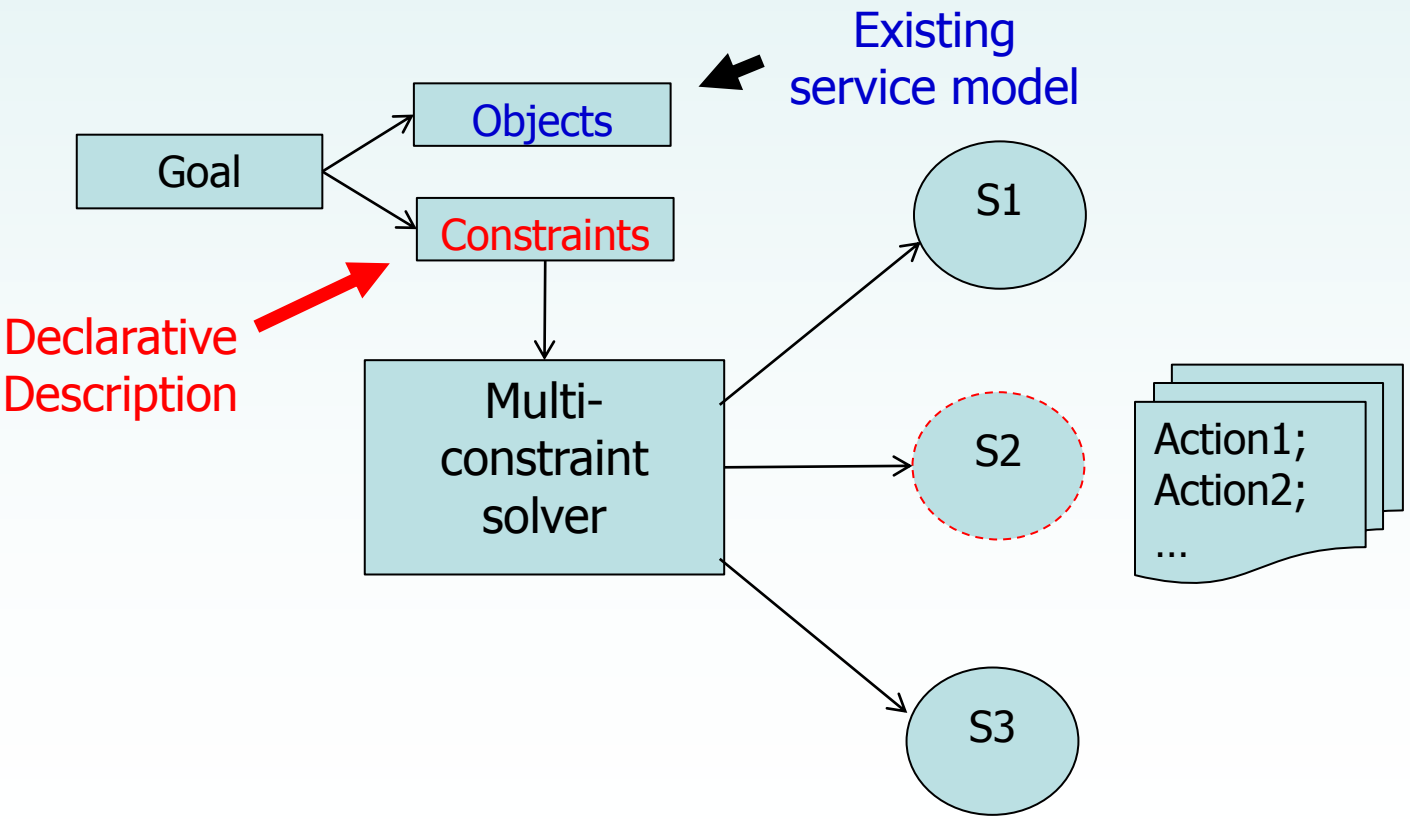
Types of Policy (2)

- **Declarative （Goal）**
  - Express ***what*** should be done, ***not how to do it***
  - Specifies criteria for choosing a set of states, any of which is acceptable
  - Rationality is generated by optimizer/planner



Declarative Description

- Only describe the constraints on possible state
- No information of how to do to achieve the state



Seven Bridges of Königsberg

The problem was to devise a walk through the city that would cross each bridge once and only once

Its negative resolution by Leonhard Euler in 1736 laid the foundations of graph theory and prefigured the idea of topology

Programming : manually decompose logic an objects, permutation and combination.(complicate, rely on human knowledge and modeling)
Declarative: only describe objects and logic constraint

```
abstract sig Landmass {}
one sig N, E, S, W extends Landmass {}
```

What is land

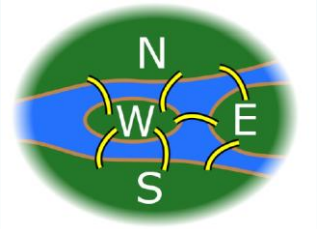
```
abstract sig Bridge { connects: set Landmass }
fact { all b:Bridge | #connects = 2 }
```

What is a bridge

```
one sig Bridge1 extends Bridge {} { connects = N + W }
one sig Bridge2 extends Bridge {} { connects = N + W }
one sig Bridge3 extends Bridge {} { connects = N + E }
one sig Bridge4 extends Bridge {} { connects = E + W }
one sig Bridge5 extends Bridge {} { connects = E + S }
one sig Bridge6 extends Bridge {} { connects = S + W }
one sig Bridge7 extends Bridge {} { connects = S + W }
```

How the 7 bridges are connected

Too easy for me.....




```
sig Path { firstStep: Step }
sig Step {
   from, to: Landmass,
   via: Bridge,
   nextStep: lone Step
} { via.connects = from + to }
fact {
   all curr:Step, next:curr.nextStep |
         next.from = curr.to
}
```

What is a path

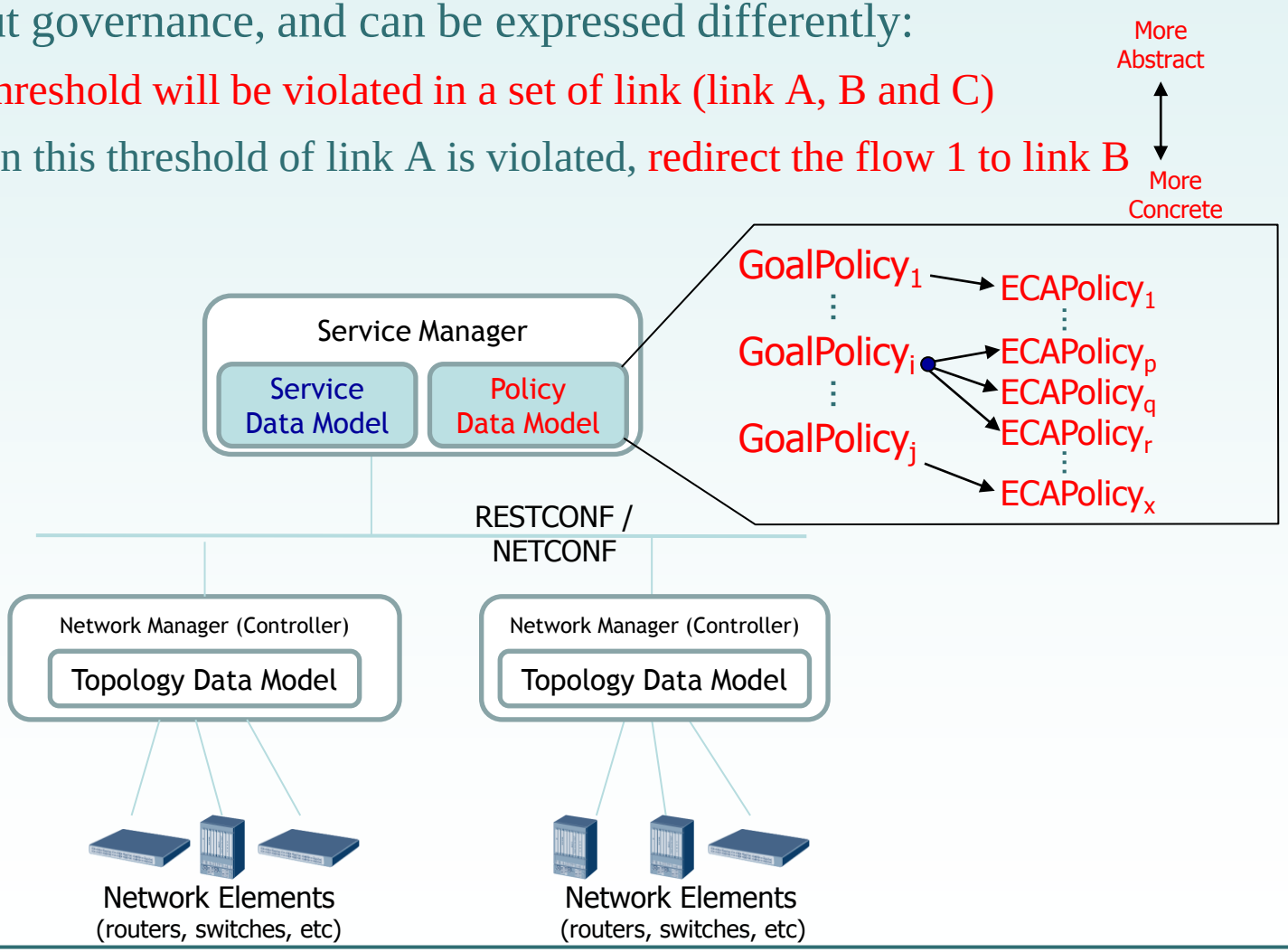
Constraint on path: no path can be used twice

```
fun steps (p:Path): set Step {
   p.firstStep.*nextStep
}
pred path() {
   some p:Path | steps[p].via = Bridge
}
run path for 7 but exactly 1 Path
```

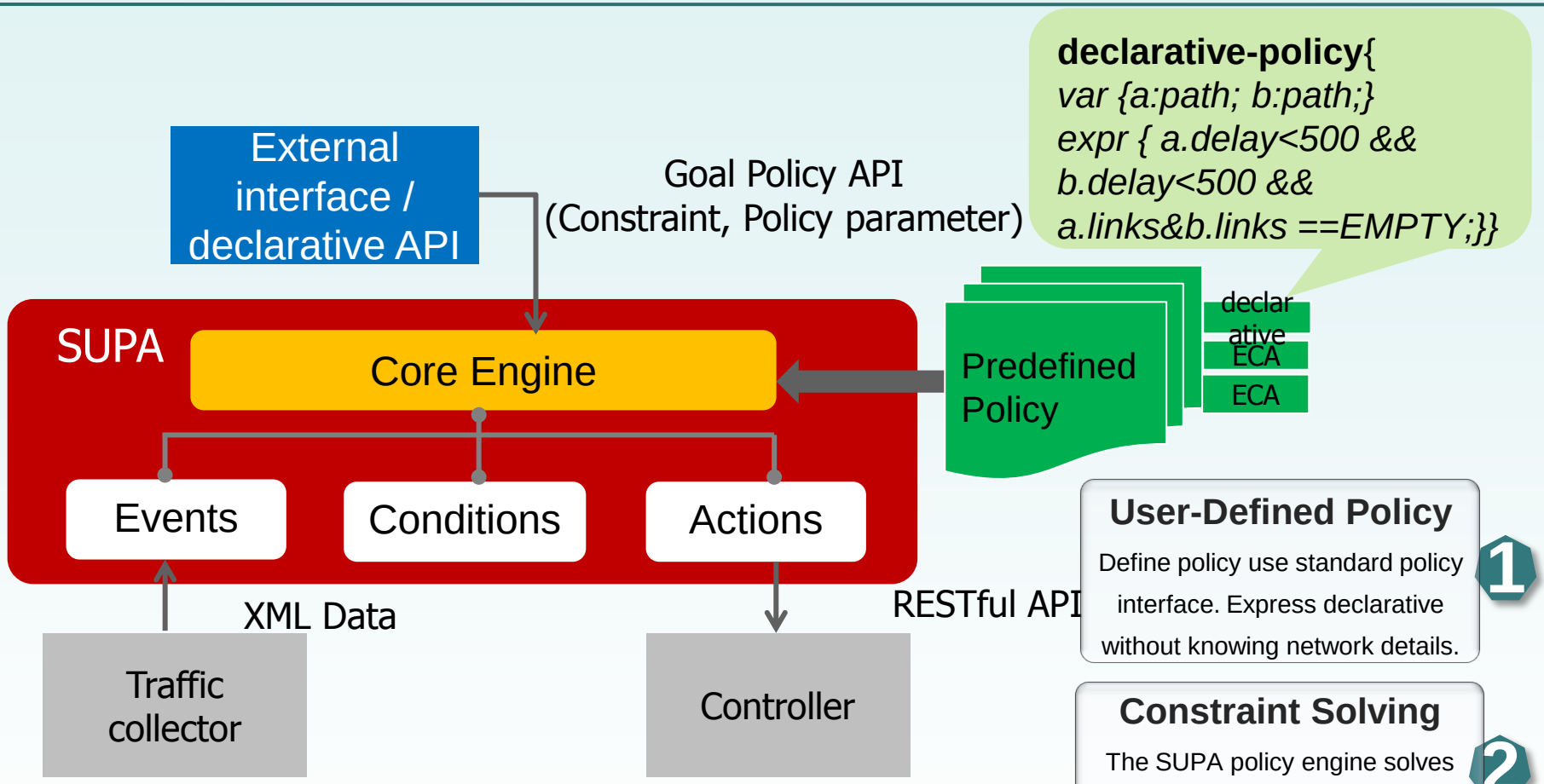
Output, no solution

Policy in Network Management

- The **usage of policy rules to manage the behavior** of managed entities
- Policy is about governance, and can be expressed differently:
  - Goal: No threshold will be violated in a set of link (link A, B and C)
  - ECA: When this threshold of link A is violated, redirect the flow 1 to link B



SUPA Policy Engine Demo



User-Defined Policy 1

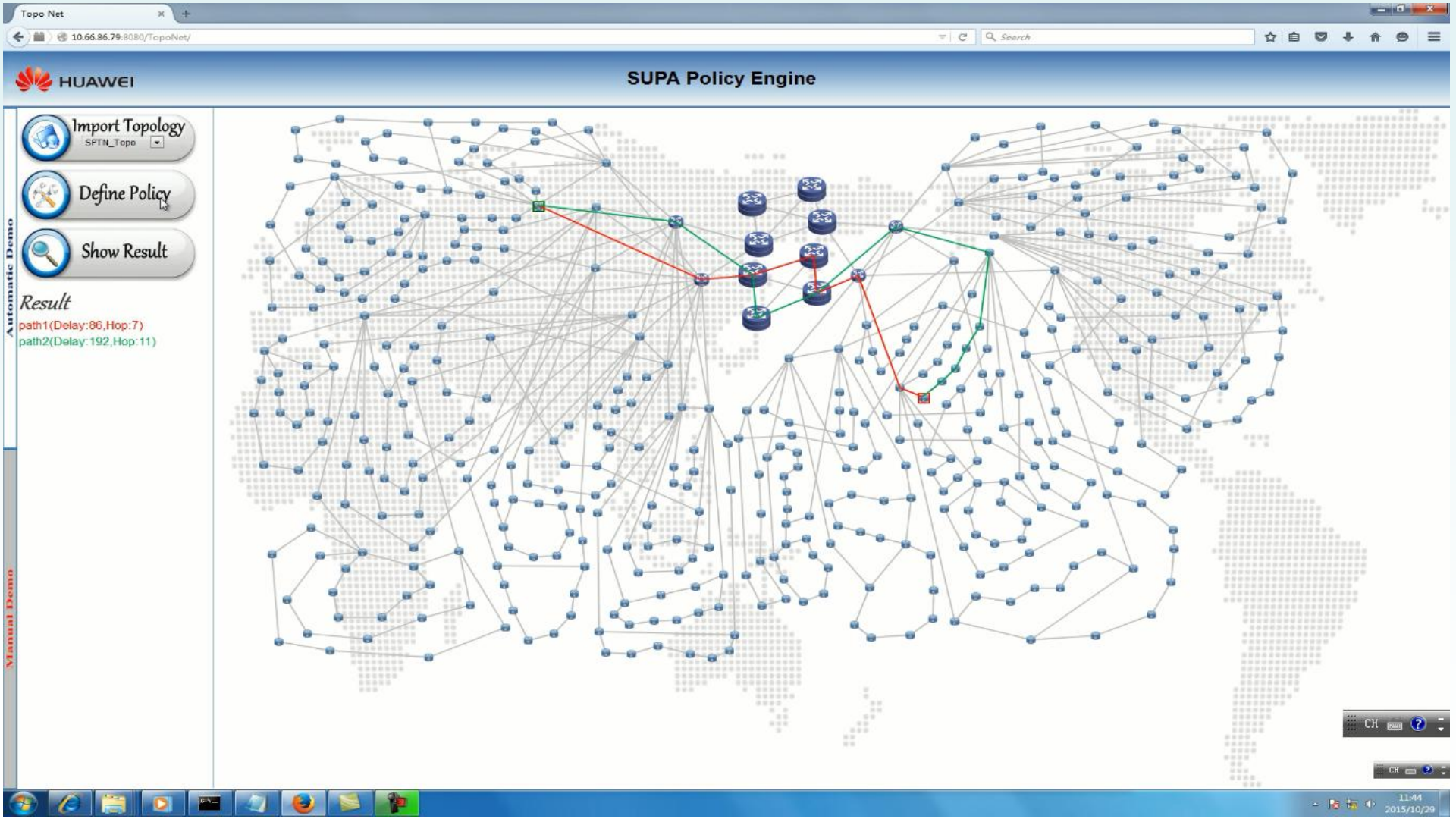
Define policy use standard policy interface. Express declarative without knowing network details.

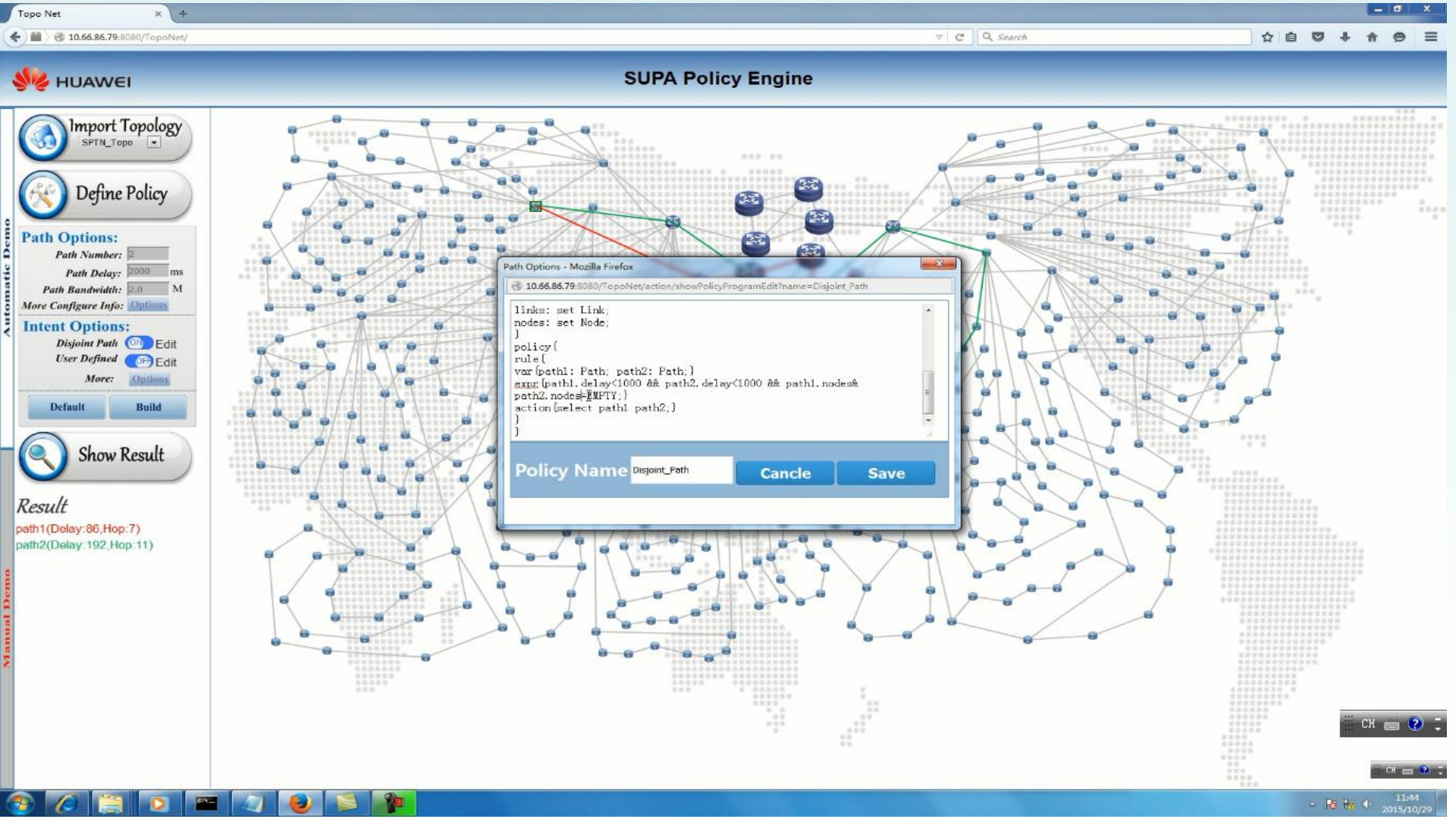
Constraint Solving 2

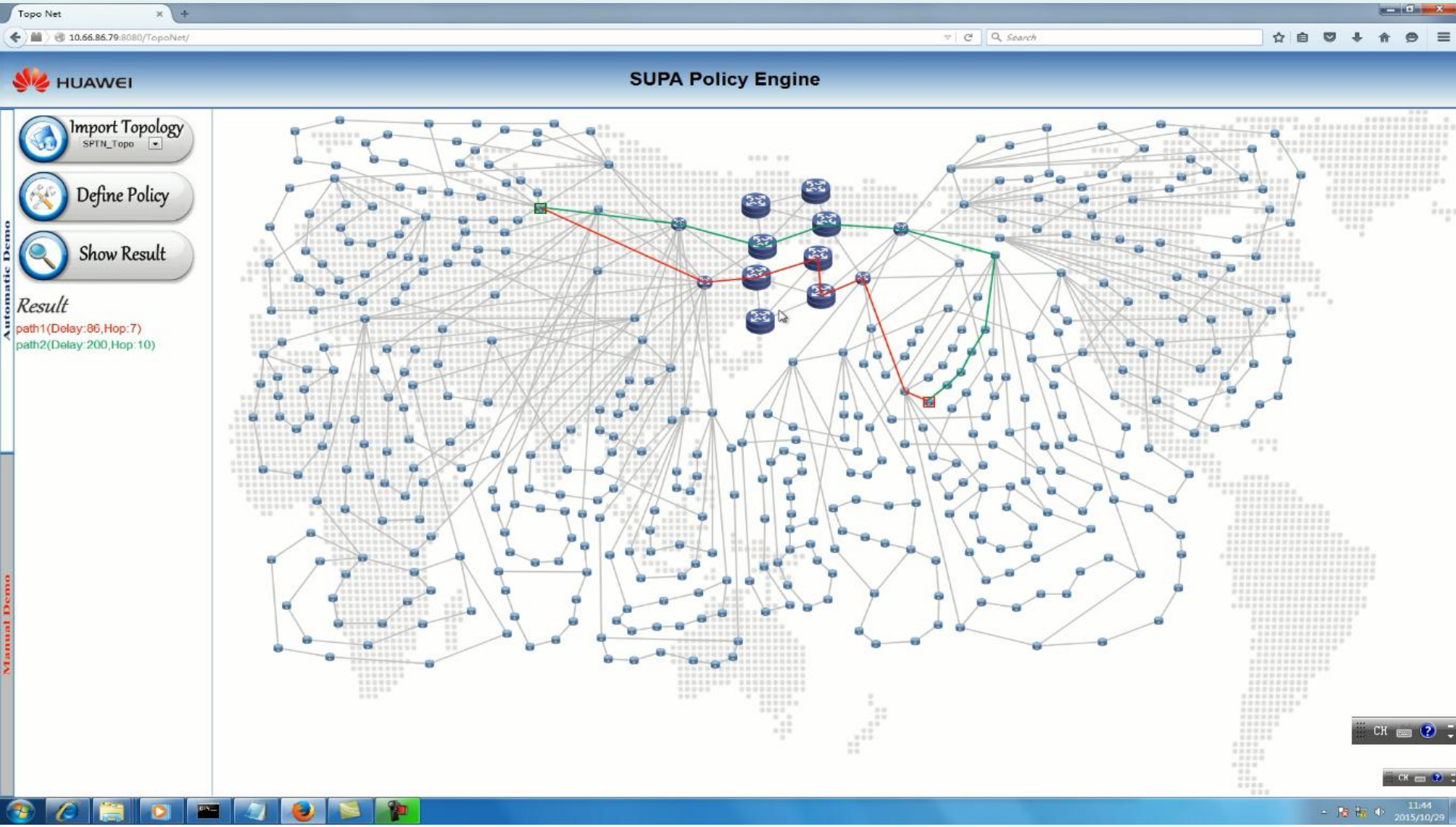
The SUPA policy engine solves the multiple constraints and service requests.

Policy Enforcement 3

Output a solution and enforce the policy to network infrastructures with actions.


Topo Net
10.66.86.79:8080/TopoNet/
Search
HUAWEI
SUPA Policy Engine
Import Topology
SPTN_Topo
Define Policy
Show Result
Result
path1(Delay:86,Hop:7)
path2(Delay:192,Hop:11)
Automatic Demo
Manual Demo
CH
11:44
2015/10/29


Topo Net
10.66.86.79:8080/TopoNet/
HUAWEI
SUPA Policy Engine
Import Topology
SPTN_Topo
Define Policy
Path Options:
Path Number: 2
Path Delay: 2000 ms
Path Bandwidth: 2.0 M
More Configure Info: Options
Intent Options:
Disjoint Path ON Edit
User Defined OFF Edit
More: Options
Default
Build
Show Result
Result
path1(Delay:86,Hop:7)
path2(Delay:192,Hop:11)
Automatic Demo
Manual Demo
Path Options - Mozilla Firefox
10.66.86.79:8080/TopoNet/action/showPolicyProgramEdit?name=Disjoint_Path
links: set Link;
nodes: set Node;
}
policy{
rule{
var{path1: Path; path2: Path;}
expr{path1.delay<1000 && path2.delay<1000 && path1.nodes&
path2.nodes=EMPTY;}
action{select path1 path2;}
}
}
Policy Name
Disjoint_Path
Cancle
Save
CH
11:44
2015/10/29


Topo Net
10.66.86.79:8080/TopoNet/
Search
HUAWEI
SUPA Policy Engine
Import Topology
SPTN_Topo
Define Policy
Show Result
Result
path1(Delay:86,Hop:7)
path2(Delay:200,Hop:10)
Automatic Demo
Manual Demo
CH
11:44
2015/10/29

There will be a SUPA Policy Engine Demo at BnB on Thursday evening. Welcome to join us.

Questions?