Security Level:

# Network Control Language(NCL）

## ----Make network as “What you want is what you get”

www.huawei.com

HUAWEI TECHNOLOGIES CO., LTD.

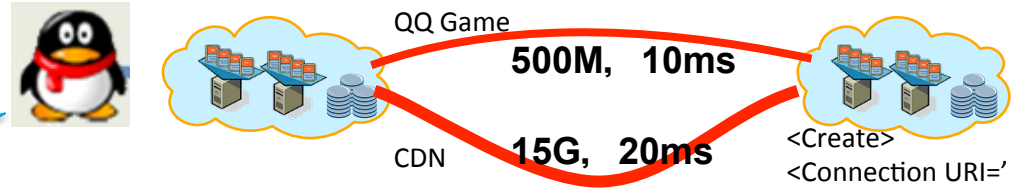
HUAWEI

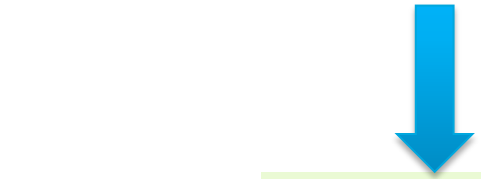
# Gap between APP and Network

- **APP wants**
  - A connection between two sites
  - A service flow with SLA
  - A customer network service chain
- **APP doesn't want**
  - Configurations of each device between two sites
  - Flow entries in each device
  - Tunnel, ACL, PBR configurations of different devices

# Device Interface is not Network Interface

Datacenters from xian to beijing:
1，two kinds of service flows with distinct SLA
2，need two connections to bear the two flows



Controller

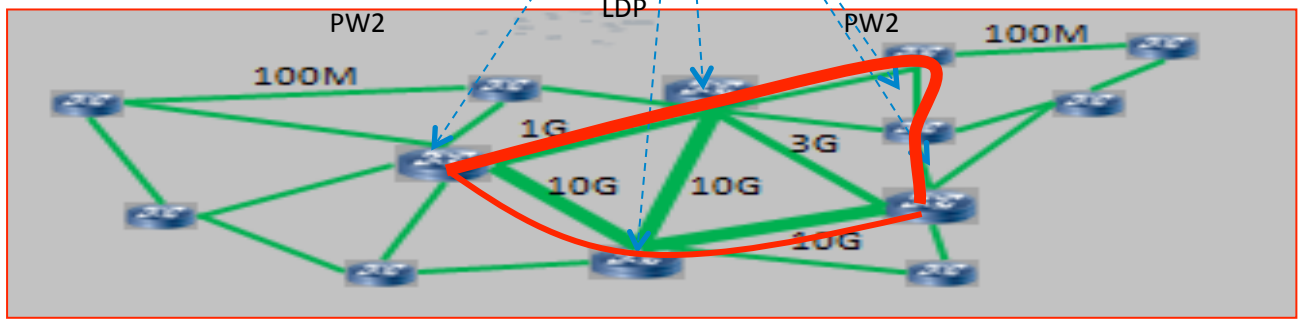
```
<Create>
<Connection URI=' 1' From='XA' To='SH' BW='500M'
Delay='10ms'></Connection >
<Connection URI=' 2' From='XA' To='SH' BW='15G'
Delay='20ms'></Connection >
</Create>
<Flow URI=' 1' Identity="DIP=xx, Pro=xxx"></Flow>
<Flow URI=' 2' Identity="DIP=yy, Pro=yyy"></Flow>
<Map Flow URI='1' Connection URI='2'></Map>
<Map Flow URI='2' Connection URI='1'></Map>
```

Face toward service logic

MPLS
LDP
Interface 0/0/0
PW1
Interface 0/0/2
PW2

IGP
MPLS
LDP

MPLS
LDP
Interface 0/0/0
PW1
Interface 0/0/2
PW2

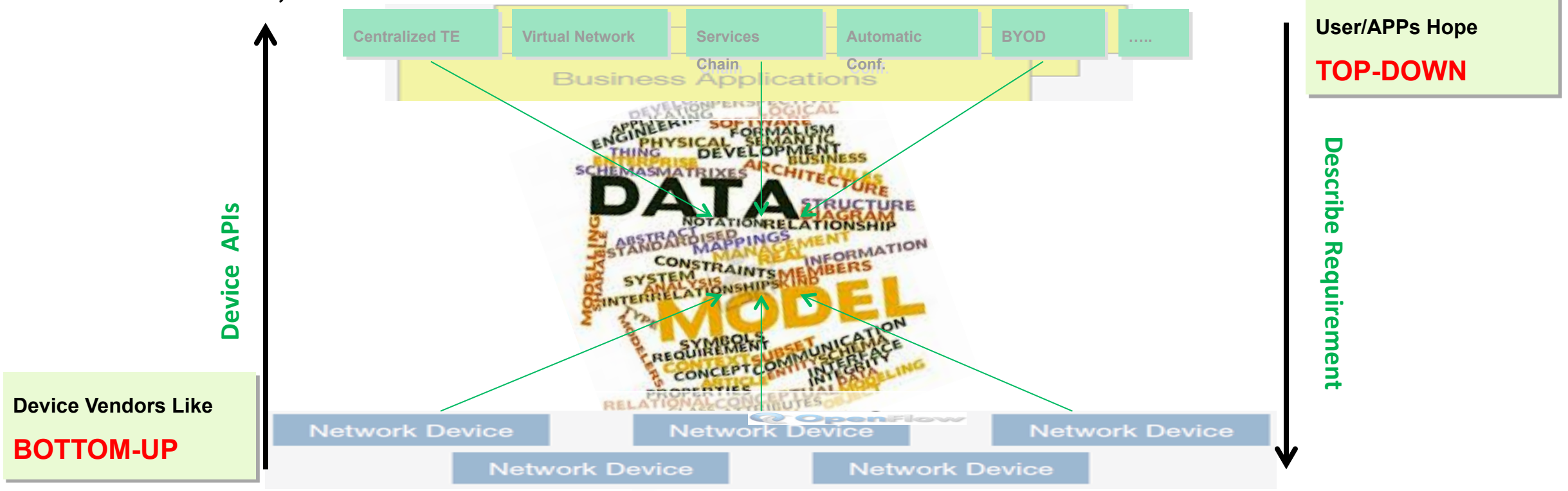



Face toward device configuration

# What should NBI be?

- NBI should be used to simplify the network operation of APP
- NBI should be describing the network requirements

- **It seems above is a simple answer of this question, but present NBI is hundreds of APIs. It is hard to be memorized, to be used and can't be standardized.**



# YANG is design for Devices

- **IETF NETMOD WG**
- **YANG is a data modeling language used to model configuration and state data manipulated by the Network Configuration Protocol (NETCONF)----RFC6020**

```
+--rw interfaces
|  +--rw interface* [name]
|     +--rw name                            string
|     +--rw description?                    string
|     +--rw type                            identityref
|     +--rw enabled?                        boolean
|     +--rw link-up-down-trap-enable?       enumeration
+--ro interfaces-state
   +--ro interface* [name]
      +--ro name                  string
      +--ro type                  identityref
      +--ro admin-status          enumeration
      +--ro oper-status           enumeration
      +--ro last-change?          yang:date-and-time
      +--ro if-index              int32
      +--ro phys-address?         yang:phys-address
      +--ro higher-layer-if*      interface-state-ref
      +--ro lower-layer-if*       interface-state-ref
      +--ro speed?                yang:gauge64
      +--ro statistics
         +--ro discontinuity-time     yang:date-and-time
         +--ro in-octets?             yang:counter64
         +--ro in-unicast-pkts?       yang:counter64
         +--ro in-broadcast-pkts?     yang:counter64
         +--ro in-multicast-pkts?     yang:counter64
         +--ro in-discards?           yang:counter32
         +--ro in-errors?             yang:counter32
         +--ro in-unknown-protos?     yang:counter32
         +--ro out-octets?            yang:counter64
```

**YANG focus on device**, hard to describe global network information, e.g., connection, flow.

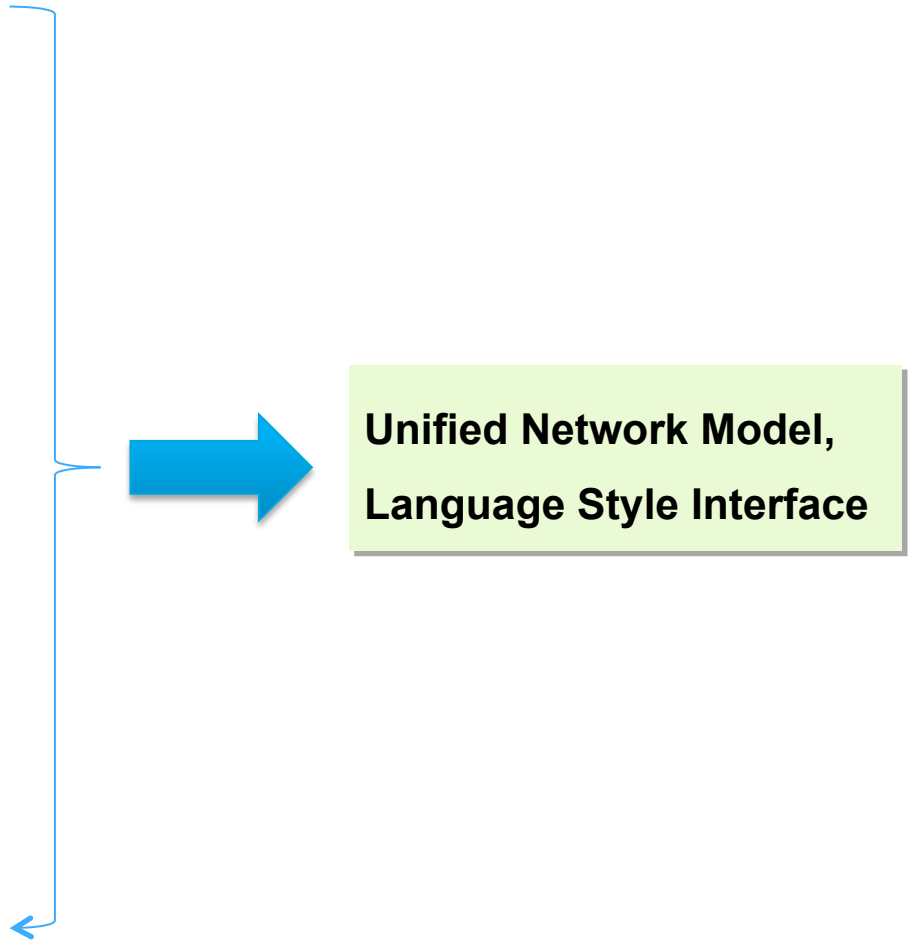
**YANG models configuration and status** data, hard to describe service logic, e.g., service chain.

**YANG is a tree structure**. For multidimensional association structure(a interface bear multiple service flow) need to put the object on multiple tree, which is hard to maintenance and synchronization.

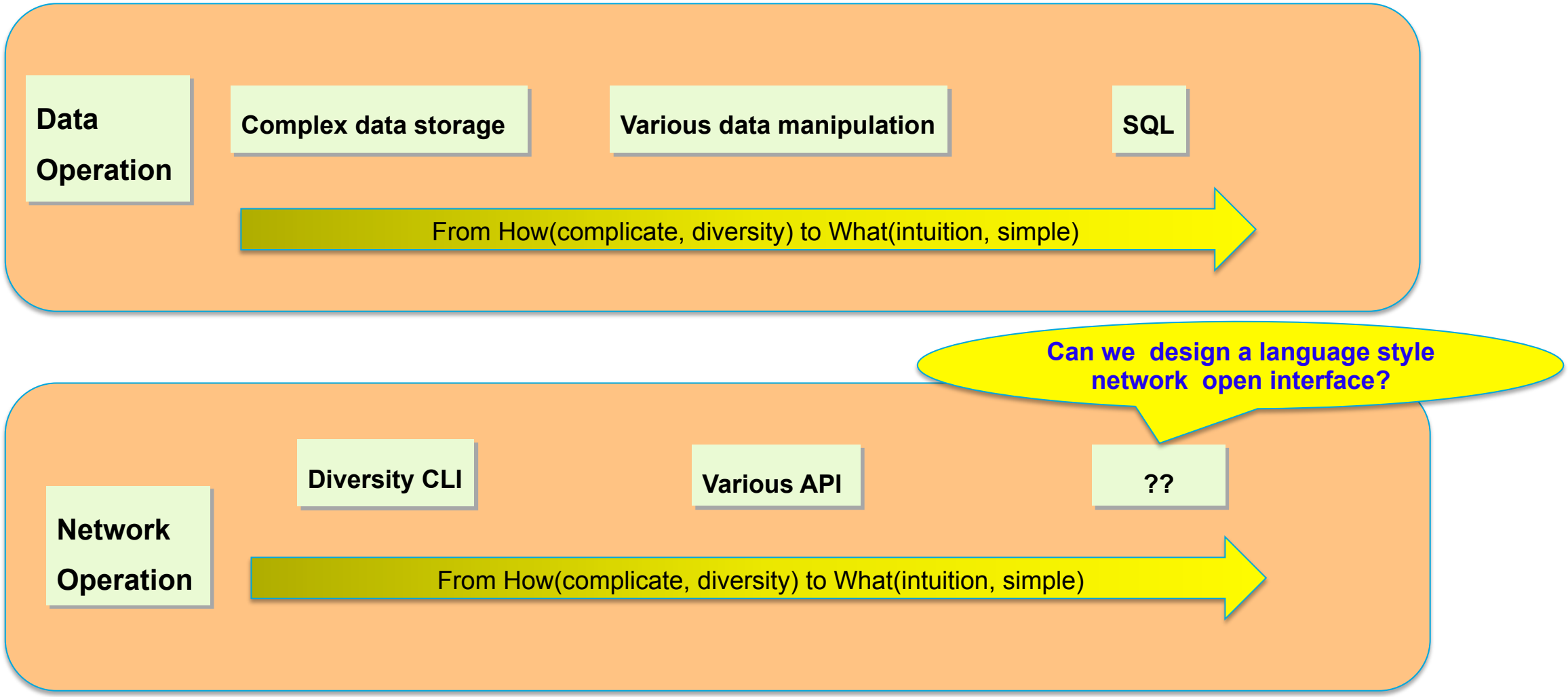
**YANG is not directly used by APP. It needs more effort for APP to learn**

# New explorations from the industry

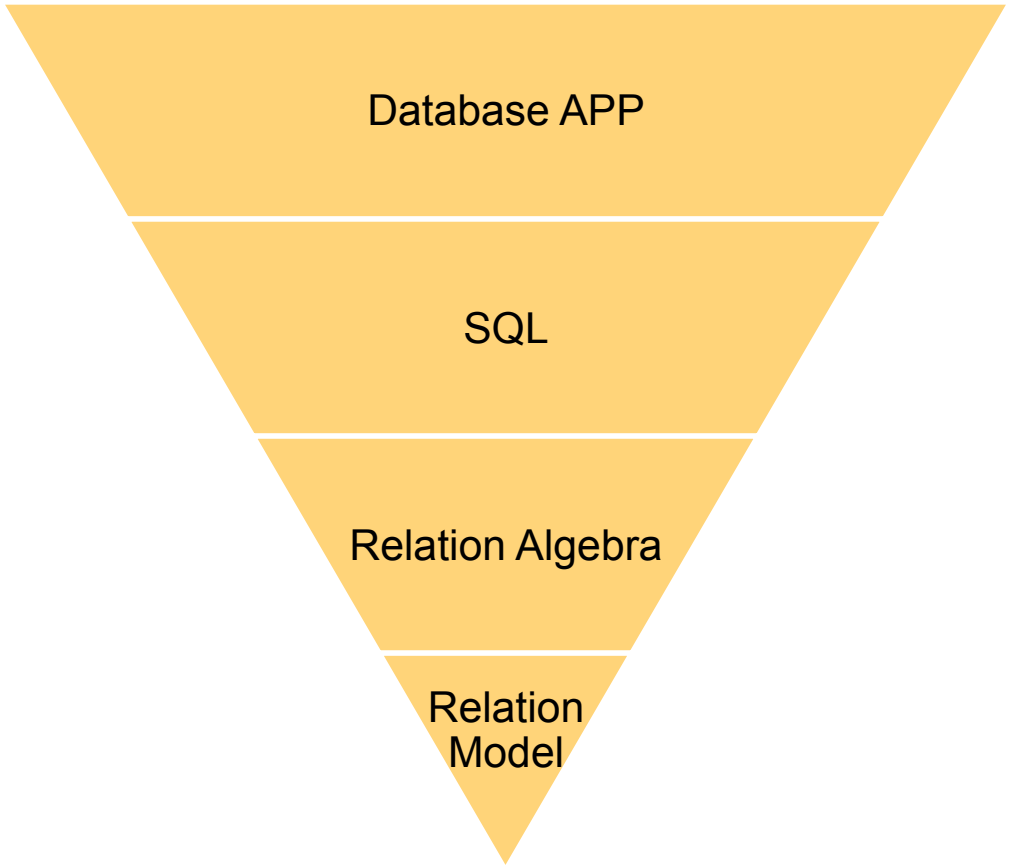
- **From Product**
  - API based Interface
  - Use case specific interface
- **From Organization**
  - NBI WG in ONF
  - I2RS in IETF
- **From Academic**
  - Flow Programming Language---Frenetic
  - Chip programming Language---P4

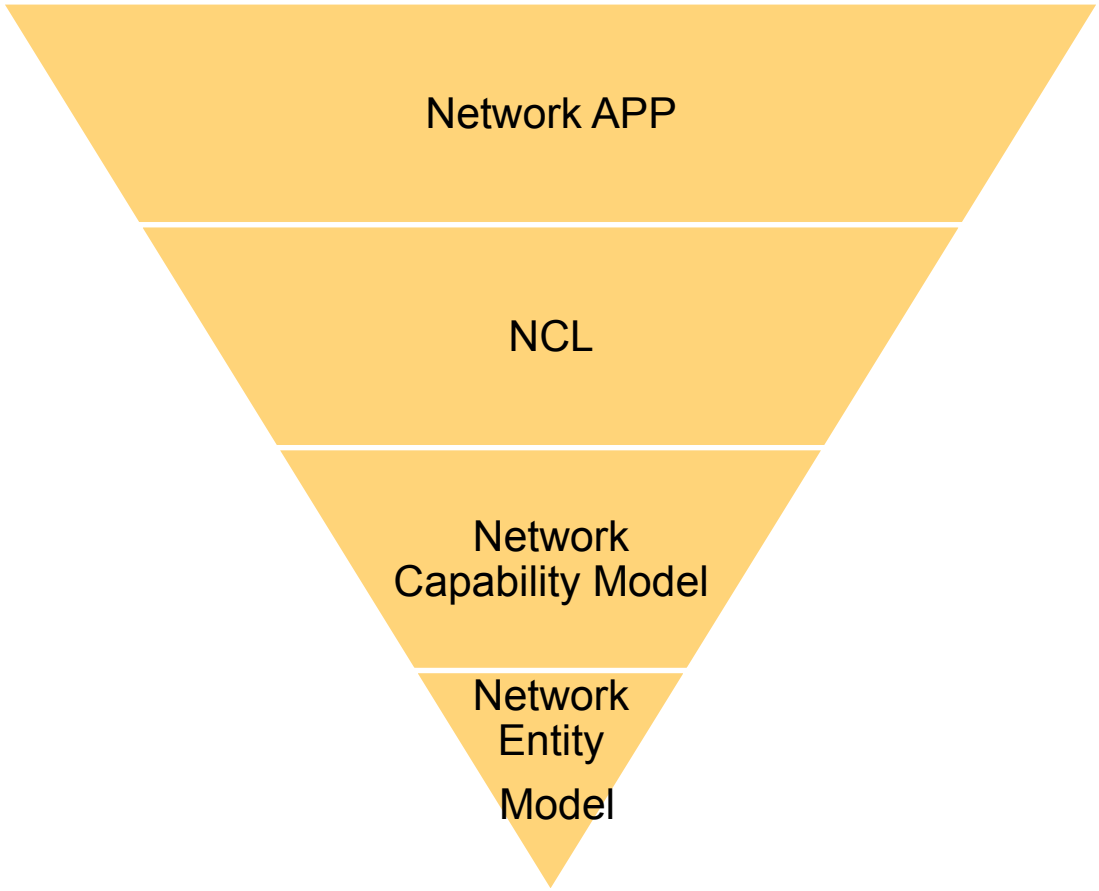


# What can we Learn from another successful story – SQL?

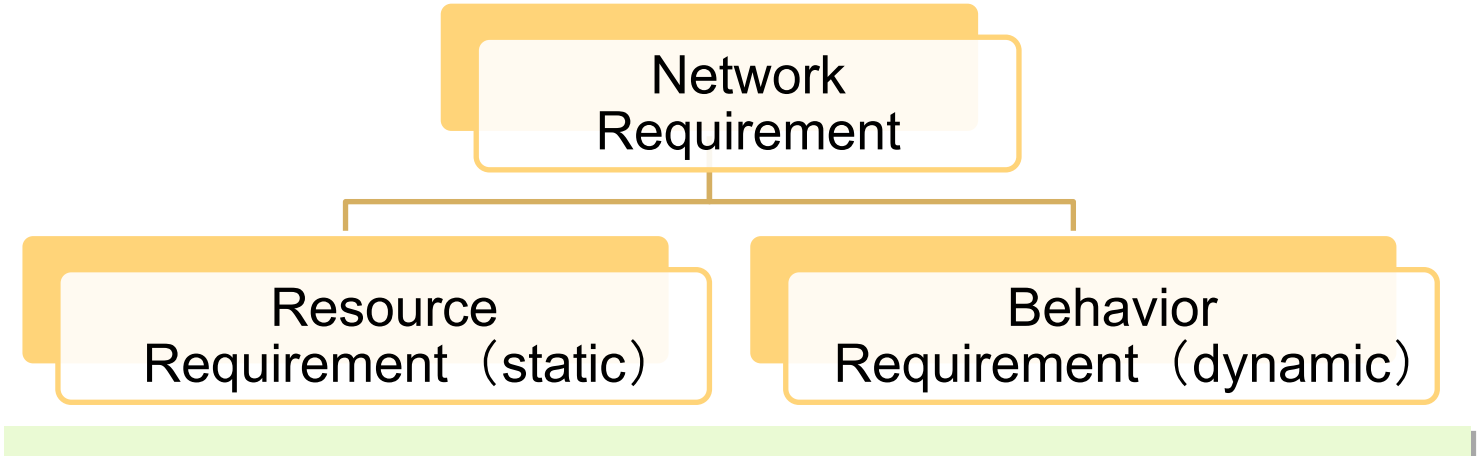


# How to design a language like SQL



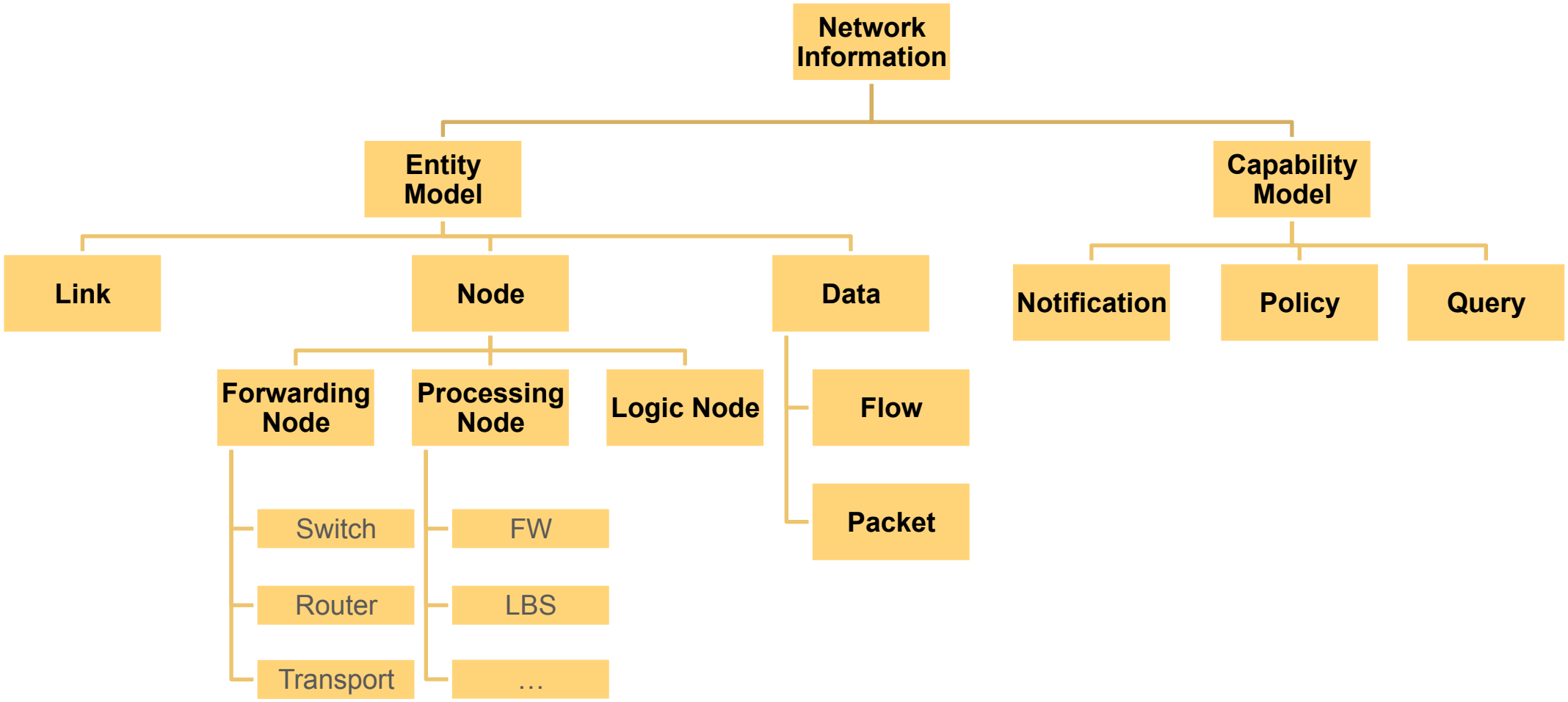
# NCL is a Language style NBI




- **NCL is a domain specific language (DSL), which is used to describe network resource and behavior requirements by APP/User. It defines network open entity model and network open capability model.**

- **A compiler resolves NCL to understand APP/User requirements and execute them in network.**

# Top-Down design Network Abstraction Model



# Resource Access

| | | |
|---|---|---|
| Entity Model | node | **Node/UnNode** entity_id **Type** {FN\|PN\|LN} **Owner** node_id **Properties** key1 ,value1 |
| | link | **Link/UnLink** entity_id **Endnodes** (node1_id,node2_id) **SLA** key,value **Properties** key1 ,value1 |
| | flow | **Flow/UnFlow** entity_id **Match/UnMatch** key1, value1\|**Range(**value, value**)** \| **Mask(**value, value**) Properties** key1 ,value1 |

# Policy and Notification

| | | |
|---|---|---|
| Capability Model | Query | **Query** key **Value** {value} **From** entity_id |
| | Policy | **Policy/UnPolicy** policy_id **Appliesto** entity_id **Condition** {expression}<br>**Action** { “forwardto” \| “drop” \| “gothrough” \| “bypass” \| “guaranteeSLA” \| “Set” \|“Packetout“\|Node\|UnNode\|Link\|Unlink}<br>**Commit**<br>**Withdraw** |
| | Notification | **Notification** entity_id **On** key **Every** period **RegisterListener** callbackfunc |

# Connection Management

| Connection Management | **ConnectController** conn_id **Address** ip_address **Port** port_num<br><br>**Disconnect** conn_id |
| --- | --- |

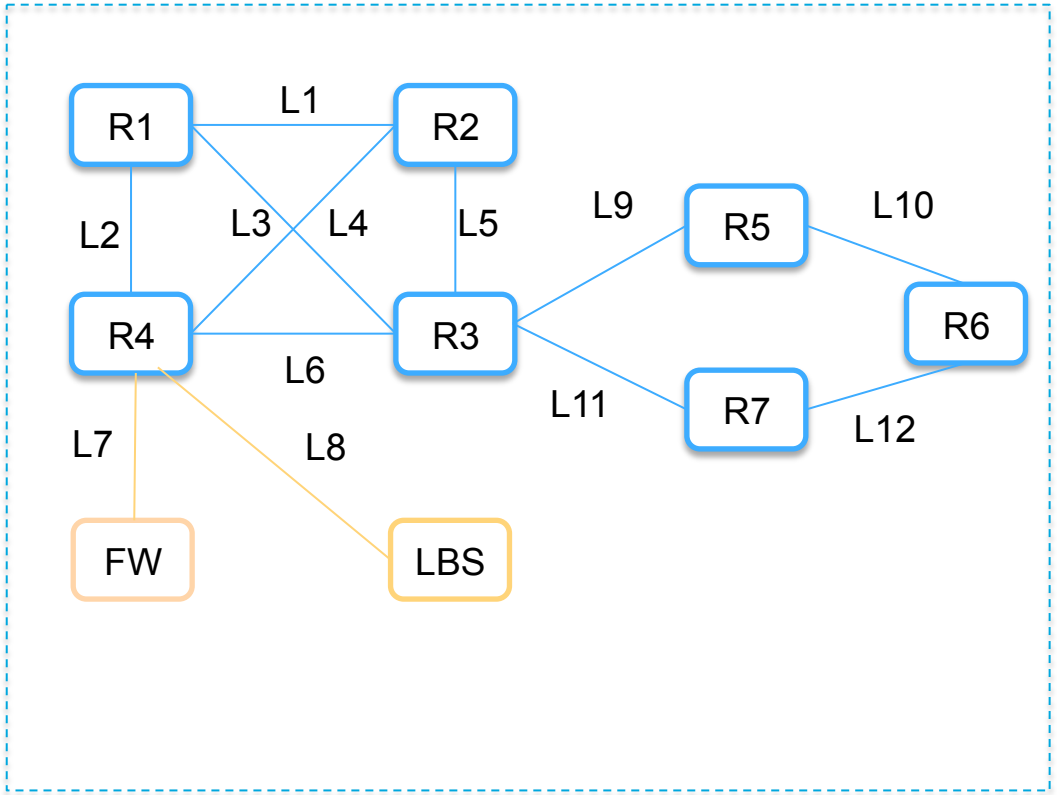
# Eclipse plug-in support NCL

- **Embed NCL into Java**
  - **Application development code is Java**
  - **Network operation use NCL, which is embed into java**
  - **Exchange parameters between java and NCL**
  - **Describe Simplify and intuitivly what APP want, don't care device implementation**

•Example

```
...
String srcip_1 = "10.0.0.1";
String dstip_1 = "10.0.0.2";
int in_port = 1;
URI flow_id = null;

#NML {flow_id = Flow "fname" Match "srcip" ,srcip_1, "dstip" ,dstip_1, "inport" ,in_port}

Calendar c = Calendar.getInstance();
int currentHour = c.get(Calendar.HOUR_OF_DAY);
if( currentHour >= 6 && currentHour <= 18){

#NML {Policy flow_id Do "DENY" }
        System.out.println("deny policy is applied");

}else{

#NML {Policy flow_id Do "MODIFY" , "dstip" , "10.0.0.4" AndDo "ALLOW"}
        System.out.println("allow policy is applied");
}
...
```

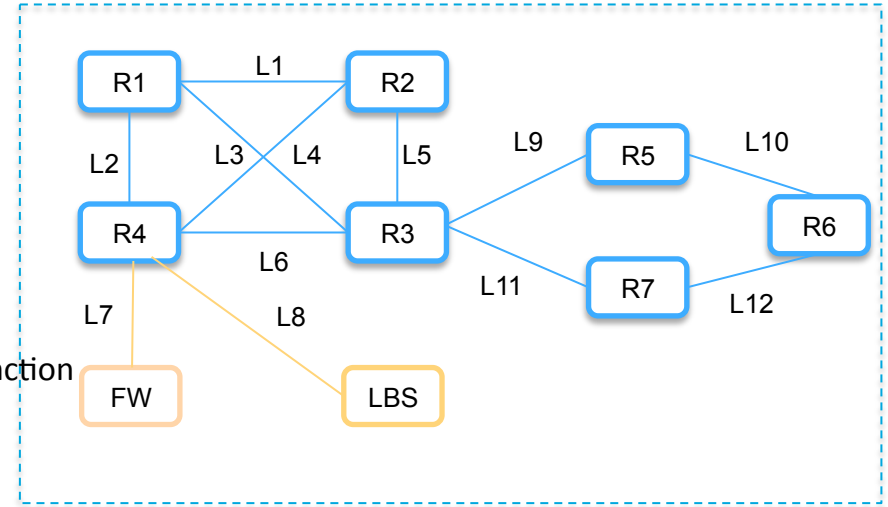
# Examples: Create Virtual Network



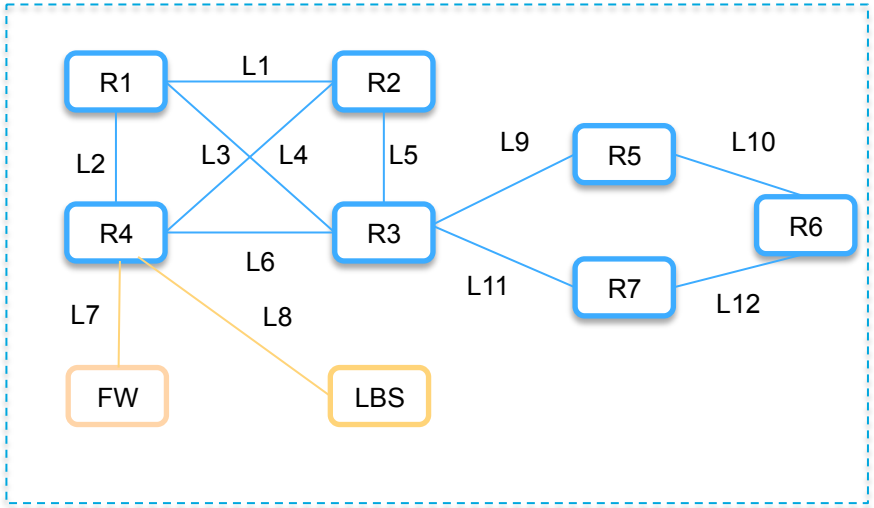

```
class  VNDemo {
    public URI flow_id = null;
    …
    public URI node_lbs_id = null;
    public URI[] Link_ids= new URI[12];
    public URI[] policy_ids = new URI[12];

    void  contructNetWorkNode() {
     //generate node instaces
      #NML {
      Node node_r1_id  Type "Forwarding"  Owner null Items "NAME","R1"
       …
      Node node_fw_id Type "Processing"  Owner null Items  “ServiceType”, ”FW”  NAME","Firewall"
      Node node_lbs_id Type "Processing"  Owner null Items  “ServiceType”,
    ”LBS” ,"NAME","Loadbance“
      }
    }
    void  BuildTopo(){
    //generate links to form the network topology
      #NML{
      Link Link_ids[0] Endnodes (node_r1_id,node_r1_id) Layer "L3"  Owner null Items
    "NAME","node1_link_node2", "Bandwith",5.0
      ….
      }
    }
```

# Examples: Policy and Notification

```
   ……
        if (ret_code == 0) {
#NML        {
              Query “Qos”  Value {pe.ActualQos} From pe.flowspec.flow_id
              Notification media.flow.flow_id On “Qos" , "AveBand" Every  60  RegisterListener callbackFunction
            }
        } else {
            pe.error = getErrorMsg(ret_code);
        }
        return;

    }else{
        return;
    }
  }
}

public void callbackFunction(URI media_id, GrantedQoS realQos, String reason){
            Media media = (Media) currentSession.medialist.get(j);
                if( media.media_id == media_id){
#NML                    Policy  policy_ids[2]  Appliesto node_r1_id  Condition {Utiliz_L1 >= Utiliz_L2} Action "forwardto“,  link_ids[1]  With  flow_id
#NML                    Commit policy_ids[2],  policy_ids[3]
                    return;
                }
            }
        }
 }
```

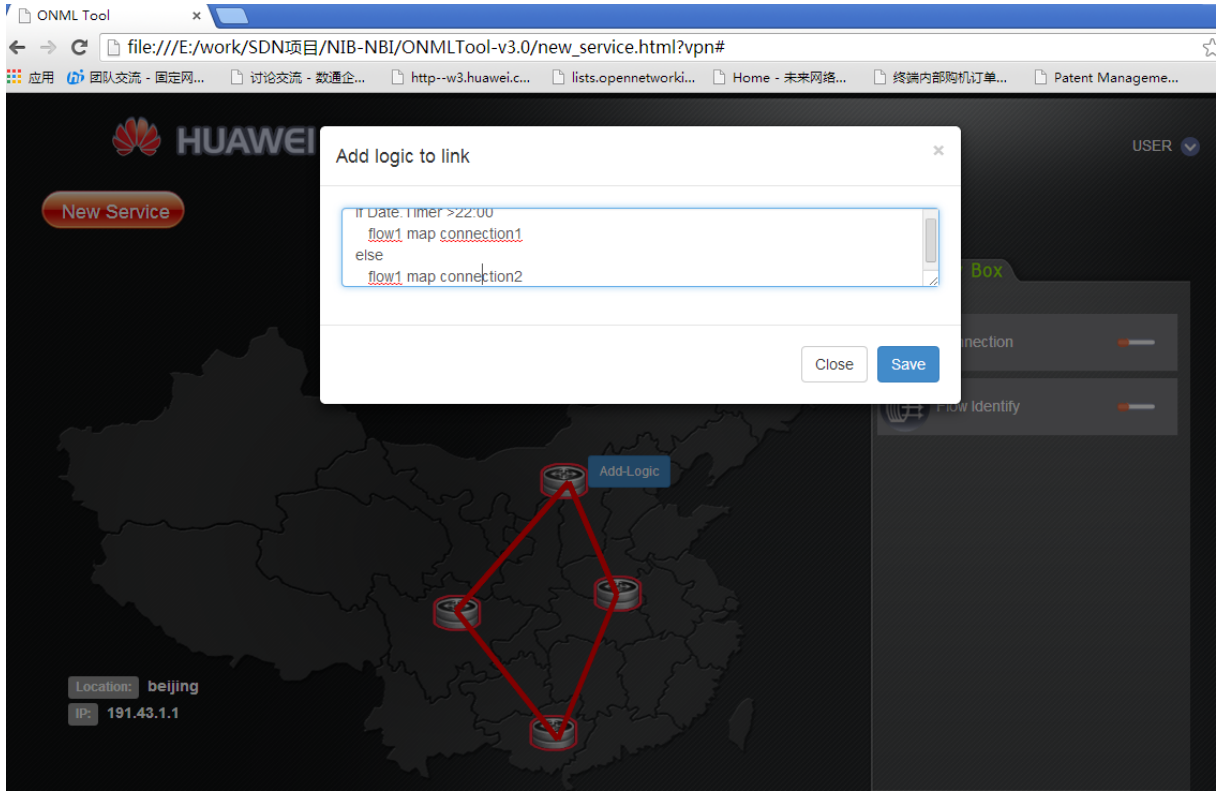


# Examples: Query Topology



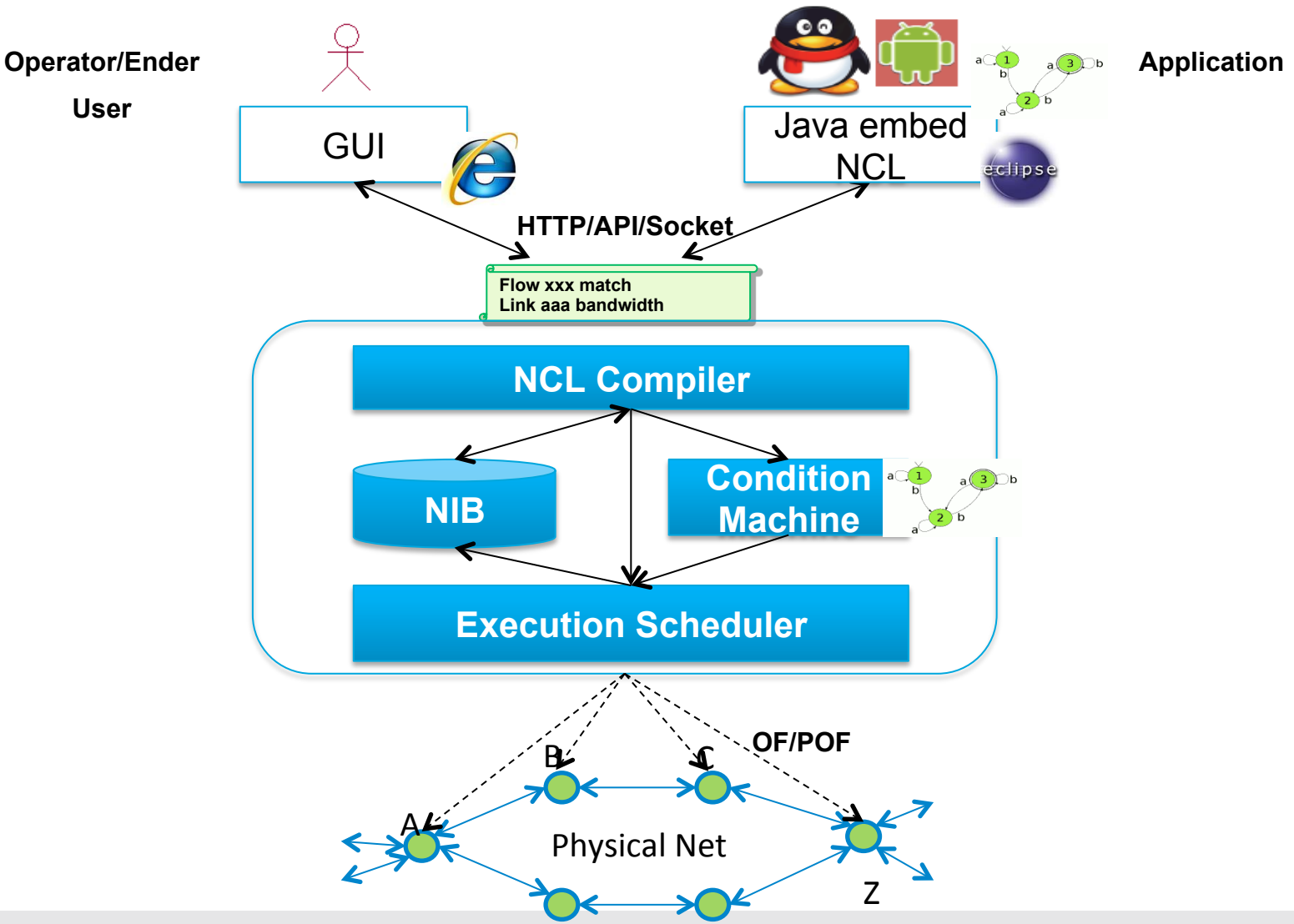

```
void   QueryNetworkTopo (URI vn_id){
                List<URI> nodeidlist = null;

#NML { Query  “NodeList” Value {nodeidlist}   From vn_id}

        for(int i = 0;  i  <  nodeidlist.size(); i++){
              String   nodename = “”;
              List<URI> linkidlist = null;

#NML      { Query “Name” Value {nodename} From nodeidlist.get(i) }

              System.out.println(“node %d : id is %s, name is %s”, i, nodeidlist.get(i).toString(),  nodename);

#NML      { Query “LinkidList” Value {linkidlist } From nodeidlist.get(i) }

              for(int j = 0; j < linkidlist.size(); j++){
                    String linkname = “”;
#NML              { Query “Name” Value {linkname} From linkidlist.get(j) }
                    System.out.println(“link %d : id is %s, name is %s ”, i, linkidlist.get(j).toString(),  linkname);

              }
        }
}
```

# An Integrate Tool for NCL




```
<NML>
<User ID=001,Name=”Tencent”, UserGroup=”OTT”,Passwd=MD5(“XXXX”)>
</User>
/* 创建虚拟网络*/
<Create>
"links":[
   {"id":"L1", "endnodes": ["R1", "R2"], "layer":3, "bandwidth":"10G"},
   {"id":"L2", "endnodes": ["R1", "R4"], "layer":3, "bandwidth":"10G"},
   {"id":"L3", "endnodes": ["R1", "R3"], "layer":3, "bandwidth":"10G"},
   {"id":"L4", "endnodes": ["R2", "R4"], "layer":3, "bandwidth":"10G"},
   {"id":"L5", "endnodes": ["R2", "R3"], "layer":3, "bandwidth":"10G"},
   {"id":"L6", "endnodes": ["R3", "R4"], "layer":3, "bandwidth":"10G"},
]
</Create>
/* 标示流*/
"policies":[
   {"id":"p1", "owner": {"type":"flow", "id":"Austin2Chicago"},
    "conditions":[{"type":"time", "from":"7:00", "to":"18:00"}], "actions":
[{"type":"forward", "path": ["R1","R4","R3"]}]
   },
   {"id":"p2", "owner": {"type":"flow", "id":"Chicago2Austin"},
    "conditions":[{"type":"time", "from":"7:00", "to":"18:00"}], "actions":
[{"type":"forward", "path": ["R3","R4","R1"]}]
   },
]
/*特定流策略*/
<script>
        If   Date.timer<22:00 and Date.timer >4:00
             flow1 map connection1
        else
             flow1 map connection2
</script>
</NML>
```

# NCL work

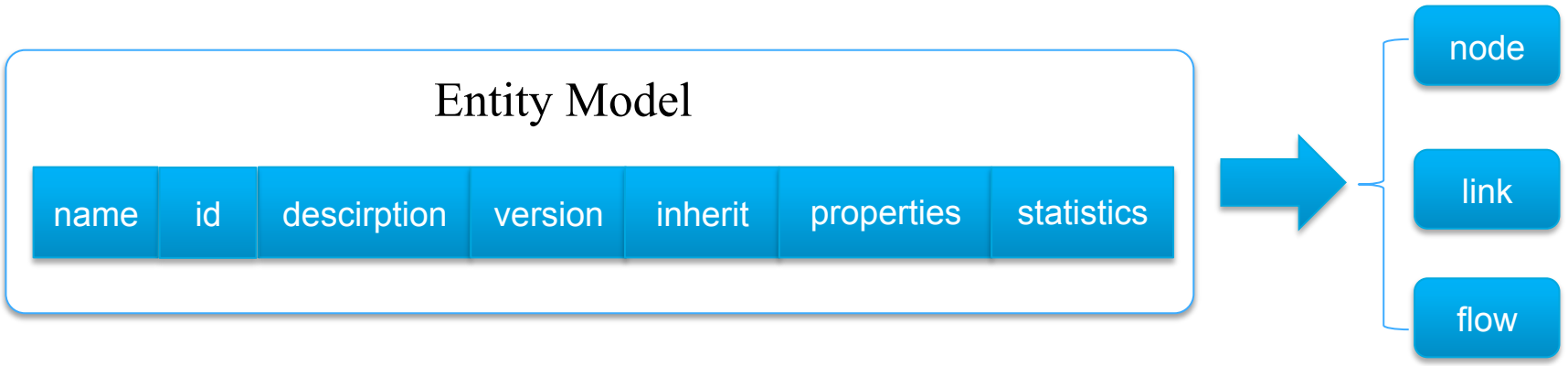

Operator/Ender
User
GUI
Java embed
NCL
Application
HTTP/API/Socket
Flow xxx match
Link aaa bandwidth
NCL Compiler
NIB
Condition
Machine
Execution Scheduler
OF/POF
A
B
C
Physical Net
Z

# Thank You

www.huawei.com

# Entity Model

- **The entity model provides a fundamental abstraction for both basic network objects (such as basic network element, link, and flow) and extended objects (such as firewall, load-balancer, and DPI).**



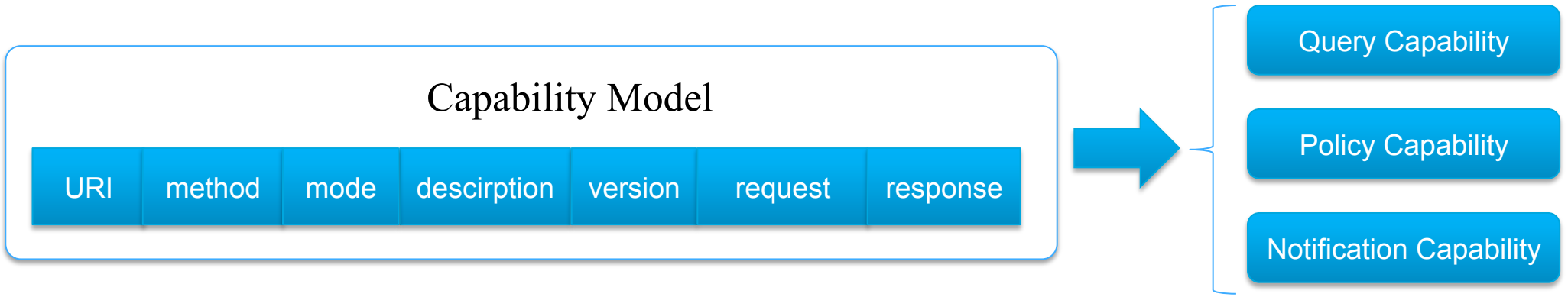

Entities derived from the entity model

# Entities

- **Node**
  - Forwarding node
    - deal with L2/3 forwarding.
    - forward packets according to the forwarding table and modifies packet heads.
  - Processing node
    - provide L4-7 network services, and will modify the body of packets
  - Logical node
    - conceal internal topology of a network and exposes properties as one entity
- **Link**
  - describe the connectivity between nodes.
- **Flow**
  - describe a sequence of packets with certain common characters

# Capability Model

- **Capability model describes a set of network functions and operations that is opened to the user**
- **Two operation modes are defined in the capability model:**
  - Synchronous mode: e.g. a creation of virtual network.
  - Asynchronous mode: e.g. port failure notification.




Capabilities derived from the capabilities model

# Reference Capabilities

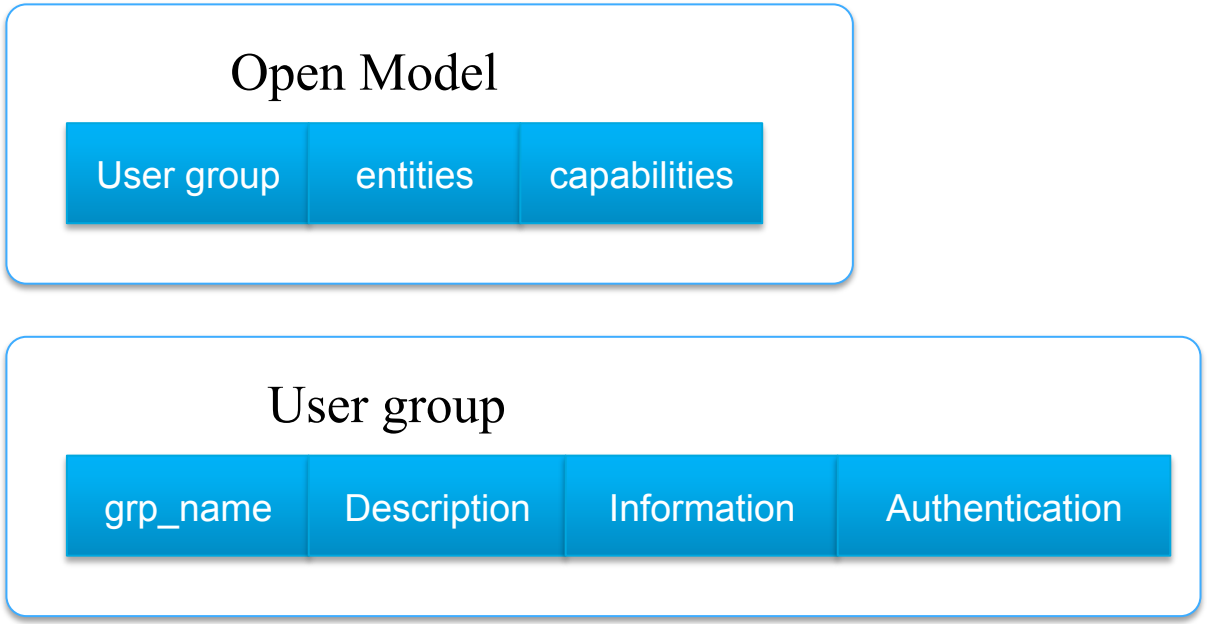
- **Query capability**
  - Query entities and their properties and statistics
- **Notification capability**
  - capability of notifying App about the information change of entities
  - in asynchronous mode
- **Policy capability**
  - capability of programing the behavior of specific entities
  - Policies can be applied to any entity
  - All the policies follow the same pattern "with <condition>, to execute <action>"

**Condition: time, utilization, load, other keywords in "statistics"(?)**

**action: drop, forward(line, in-path, off-path, chain); mirror(same as forward), set(bandwidth, QoS, capacity, other keywords in "entity para"(?)) ,modify(?),**

# Open Model

- **describe information about user group including authentication and account information**
- **define visibility of entity and capability which opened for different user groups.**



# Setup Information Model Using Meta Models

- **Identify entities to be used in the use-case**
- **Specialize entity models with parameters and statistics**
  - Customize parameters and statistics base on inheritance.
  - For example, a processing node model is derived from the basic entity model, a router model then is derived from the forwarding node model, and a customized router model is derived from the router model.
- **Specialize capabilities**
  - Define capabilities to be provided to users, and how to provide
    - Including： uri, request format, response format, …