

Design and Implementation of Optical Transport Network Models with Path Computation

Chang-Gyu LIM*, Moonsub SONG*

*ETRI (Electronics and Telecommunications Research Institute), Daejeon, Korea

{human, sirius}@etri.re.kr

Abstract— SDN (Software Defined Networking) controllers can manage various network services through abstraction of physical layer functionality. It can also be applied to optical transport networks. We already implemented T-SDN(Transport SDN) models and controllers especially in MPLS-TP(Multiprotocol Label Switching - Transport Profile) networks, based on OpenDaylight platform. Now, we need to expand another networks controlled by expanded T-SDN controllers. This paper proposes the expanded OTN(Optical Transport Network) models. With using proposed OTN models, we can make optical transport network services easily.

Keywords— Optical transport network, model, SDN

I. INTRODUCTION

Today's network providers hope to handle network management system by dividing the legacy network system into control and data planes. SDN(Software Defined Networking) separates these two planes and manages various network services through abstraction of physical layer functionality. It can also be applied to optical transport networks.

We already implemented T-SDN(Transport SDN) models[1] and controllers especially in MPLS-TP(Multiprotocol Label Switching - Transport Profile) networks. T-SDN controllers are based on OpenDaylight[2] platform(Lithium version). It manages MPLS-TP(Multiprotocol Label Switching - Transport Profile) network services and resources with NBI(North Bound Interface) and SBI(South Bound Interface). Now we need to expand another networks controlled by expanded T-SDN controllers. We use YANG data models for OTN(Optical Transport Network) abstraction. YANG is a data modeling language based on standard IETF document[3]. Figure 1 shows the architecture of expanded T-SDN controllers(OTN controllers).

In Section 2, we design proposed YANG data models for OTN and OTN path computation. Former T-SDN models[1] are expanded to the proposed OTN models. We also describe expanded path computation models for OTN, briefly.

Section 3 shows our implementation and the result of the OTN path computation using the proposed OTN model. We realize the OTN path computation plugin in OTN controllers.

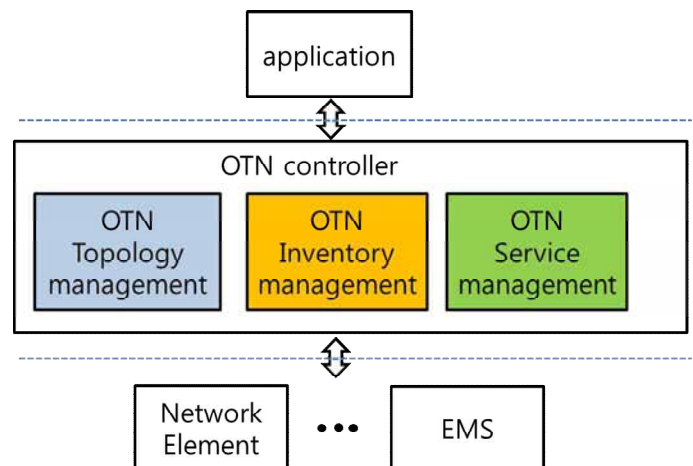


Figure 1. The architecture of OTN controllers

II. PROPOSED OTN YANG DATA MODEL WITH PATH COMPUTATION

In this section, we describe a proposed OTN YANG data model with path computation. Former T-SDN models[1] are expanded to the proposed OTN models. We also describe expanded path computation models for OTN, briefly.

Figure 2 shows the relation of the proposed OTN model and the T-SDN generic model. If you want to make new models, you can use the generic T-SDN model for designing new models.

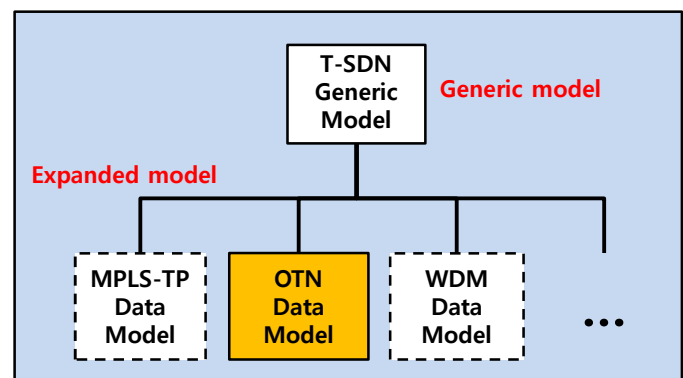


Figure 2. The relation of the proposed OTN model and the T-SDN generic model

A. OTN Inventory YANG Data Model

The OTN inventory YANG model provides the information below. Figure 3 shows the proposed OTN inventory YANG data model.

- Augmentation of the generic T-SDN inventory node: OTN nodes need a tunnel list, a tunnel cross-connect list and an access interface list which is composed of its id(key value), containers and attributes.
- Augmentation of the generic T-SDN inventory RPCs: RPC's input or output parameters are expanded. There are OTN tunnels', tunnel cross-connects' and access interfaces' RPCs for CRU(Create, Read, Update) operation respectively.
- Augmentation of the generic T-SDN inventory notifications: There are notifications of updated OTN nodes, node-connectors, tunnels, tunnel cross-connects and access interfaces respectively.

```
augment "/tin:nodes/tinv:node" {
  ext:augment-identifier "otn-inventory-node";
  list "tunnel" {
    key "tunnel-id";
    .....
    leaf node-ref {type tnode:node-ref;}
    leaf tunnel-id {type ttunnel:tunnel-id;}
    leaf rate-type {type otn-tunnel-rate-type;}
    leaf multiple-rate {type string;}
    leaf active-path-status {type ttypes:active-path-status-type;}
    leaf src-node-ref {type tnode:node-ref;}
    leaf dest-node-ref {type tnode:node-ref;}
    leaf access-if-ref {type taccessif:access-if-ref;}
    container working-path {
      leaf node-connector-ref {.....}
      leaf path-status {.....}
      leaf higher-tunnel-ref {type ttunnel:tunnel-ref;}
      leaf-list tributary-slot {type uint16;}
    }
    container protection-path {.....}
    leaf protection-type {type ttypes:tunnel-protection-type;}
    leaf service-mapping {type ttypes:enabler-type;}
  }
  list "tunnel-xc" {
    key "tunnel-xc-id";
    .....
  }
  list "access-if" {
    key "access-if-id";
    .....
  }
}
```

Figure 3. OTN inventory YANG data model: augmentation of the generic T-SDN inventory node

B. OTN Topology YANG Data Model

The OTN topology YANG model provides the information below. Figure 4 shows the proposed OTN topology YANG data model.

- Augmentation of the generic T-SDN topology tunnel: OTN topology tunnels need a rate type and a multiple rate, etc.

- Augmentation of the generic T-SDN topology link: OTN links need a high-order tunnel reference and an available number of odu0s, etc.
- Augmentation of the generic T-SDN topology node-connector: OTN node-connector need an available odu0s tributary slots, etc.

```
augment "/topo:network-topology/topo:topology/topo:node/topo:node-connector" {
  ext:augment-identifier "otn-topology-node-connector";
  leaf-list supported-odu0s-tributary-slot {
    type uint16;
  }
  leaf-list available-odu0s-tributary-slot {
    type uint16;
  }
}
augment "/topo:network-topology/topo:topology/topo:link" {
  ext:augment-identifier "otn-topology-link";
  leaf high-order-tunnel-ref {
    type ttunnel:tunnel-ref;
  }
  leaf maximum-odu0s {
    type uint32;
  }
  leaf available-odu0s {
    type uint32;
  }
}
```

Figure 4. OTN topology YANG data model: augmentation of the generic T-SDN topology node-connector and link

If we use OTN networks, it should be considered that ODU(Optical Data Unit)0 tributary slot allocation is necessary. A specific ODU0 tributary slot value is permitted only once per OTN node-connector. Therefore, we expand a generic topology node-connector to a ODU0 tributary slot allocation enabled node-connector with adding supported/available-odu0s-tributary-slot leaf-lists. Figure 4 is shown. Otn-topology-node-connector is created by the augment method. Expanded node-connector has supported/available ODU0s tributary slot leaf-lists. We also expand a generic topology link to a ODU0s bandwidth aware link with adding maximum/available ODU0s bandwidth attributes. This is also shown at Figure 4. OTN networks must have a higher ODU tunnel. OTN links also have a high-order tunnel reference attribute.

C. OTN Service YANG Data Model

The OTN service YANG model provides the information below. Figure 5 shows the proposed OTN service YANG data model.

- Augmentation of the generic T-SDN service: OTN services need a service constraint list, a source/destination node container and OTN service attributes.
- Augmentation of the generic T-SDN service RPCs: RPCs' input or output parameters are expanded. There are OTN expanded service RPCs for CRU operation respectively.

- Augmentation of the generic T-SDN service notifications: There is a notification of updated OTN services.

```

augment "/ser:set-service/ser:input" {
  ext:augment-identifier "otn-set-service-input";
  list service-constraint{
    key "index";
    leaf index {type ttypes:constraint-id;}
    uses ttypes:common-constraint;
    leaf service-constraint-type {type ttypes:service-constraint-type;}
    .....
  }
  container source {
    leaf node-ref {type tnode:node-ref;}
    leaf node-connector-ref {type tport:node-connector-ref;}
    leaf access-if-type {type taccessif:access-if-type;}
    list port-constraint{
      key "index";
      leaf index {type ttypes:constraint-id;}
      uses ttypes:common-constraint;
      leaf port-constraint-type {type ttypes:port-constraint-type;}
    }
  }
  container destination {
    .....
  }
}
augment "/ser:service-updated" {
  ext:augment-identifier "otn-service-updated";
  uses otn-service;
}

```

Figure 5. OTN service YANG data model: augmentation of the generic T-SDN service RPC input and notification

D. OTN Path Computation Plugin

We already proposed and implemented the path computation plugin for MPLS-TP networks[4]. We can use these path computation plugin and path computation request remote procedure call model with adding to assignment of ODU0 tributary slot attributes in each node-connector. We also reuse former input/output parameter models.

III. IMPLEMENTATION AND RESULT

We implement the OTN path computation plugin using the proposed OTN model in OTN controller. Figure 6 shows the block diagram of the OTN controller with the OTN path computation plugin and the OTN topology simulator plugin which is a OTN topology, OTN node and OTN link data loader. The OTN topology simulator plugin transmits abstraction topology, node, link data to the topology data store in MD-SAL(Model Driven Service Abstraction Layer) which is supported by OpenDaylight controller. When operators call the path request RPC, there are two types of OTN tunnel paths. The one is an outer tunnel for ODU service tunnels. This tunnel cannot have services with various types of client signals directly. The other is an ODU service tunnel with client signals. An ODU service tunnel must need an outer tunnel. The OTN path computation plugin can calculate both types of the request tunnel paths.

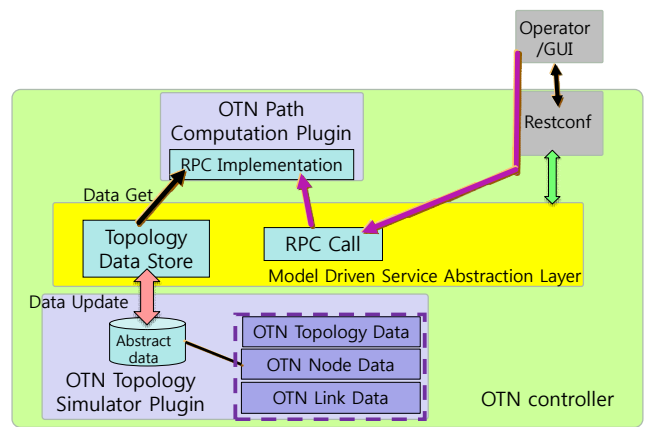


Figure 6. A block diagram of OTN controller

Figure 7 shows the specific node and link information presented by web browser.

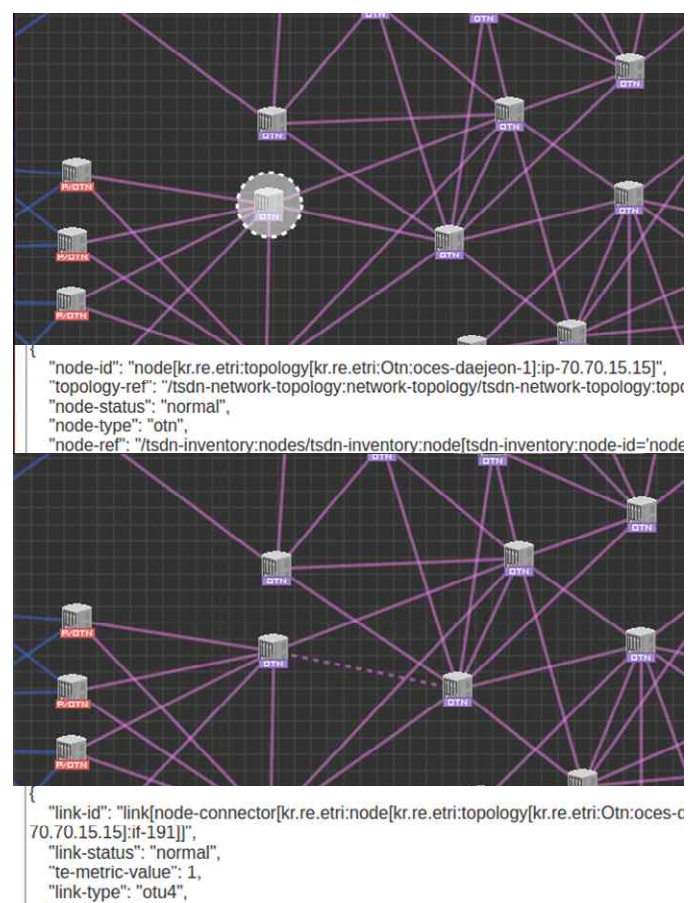


Figure 7. Node and link information presented by web browser

Figure 8 shows the JSON(JavaScript Object Notation) input example of the OTN path(ODU service tunnel) computation request. There are some input parameters like source and destination nodes and the number of the requested path candidates.

```

{
  "tsdn-pce:input":{
    "service-mapping": "enable",
    "network-type": "otn",
    "source": {
      "node-ref": "/tsdn-inventory:nodes/tsdn-inventory:n
daejeon-1[ip-70.70.40.40]"
    },
    "destination": {
      "node-ref": "/tsdn-inventory:nodes/tsdn-inventory:n
daejeon-1[ip-70.70.22.22]"
    },
    "rate-type": "odu0",
    "multiple-rate": "1",
    "path-computation-constraint": [
      {
        "index": "1",
        "value": "4",
        "constraint-type": "tunnel-number-of-candidate"
      }
    ]
  }
}

```

Figure 8. JSON input example of the OTN path computation request

Figure 9 shows the result of the OTN path request. The result has 4 candidate paths. Two of the candidate paths is drawn in Figure 9.

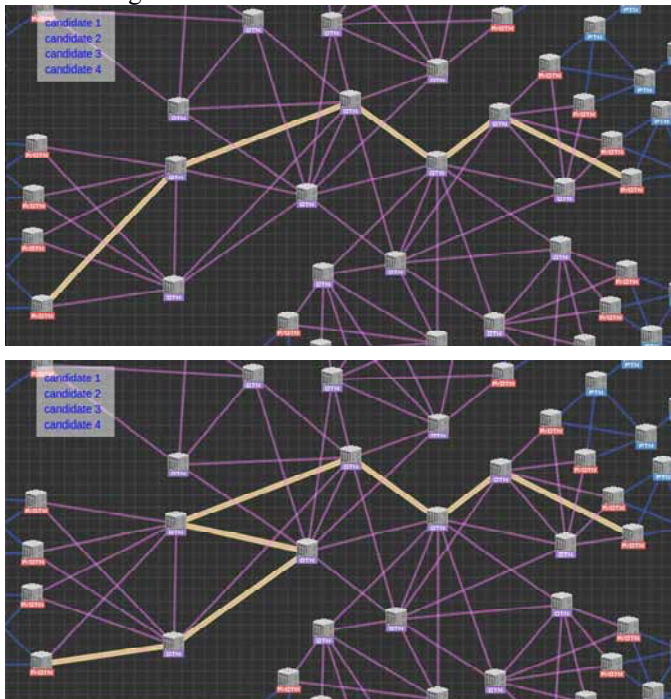


Figure 9. Result of the path request

IV. CONCLUSIONS

This paper proposes the expanded OTN models with the path computation. We also implemented the OTN path computation plugin using the proposed OTN models. With using proposed OTN models, we can make optical transport network services easily.

ACKNOWLEDGMENT

This work was supported by the NRF(National Research Foundation of Korea) grant funded by the MSIT(Ministry of Science and ICT) (NRF-2016M1B3A1A01937737).

REFERENCES

- [1] C. Lim, S. Pahk and Y. Kim, "Model of Transport SDN and MPLS-TP for T-SDN Controller", in ICACT2016, 2016, pp. 526-530.
- [2] <http://www.opendaylight.org>
- [3] M. Bjorklund, RFC6020 YANG-A Data Modeling Language for the Network Configuration Protocol, IETF, Oct. 2010.
- [4] C. Lim and S. Pahk, "Model of Transport SDN for Path Computation", in ICACT2017, 2017, pp. 309-312



Chang-Gyu LIM is a senior engineer of SW & Contents Research Laboratory, ETRI, Korea. He received his Master degree at KAIST in 2002. His key research interests are: Future Internet, Software Defined Networking and Transport Network.



Moonsub SONG is a principal engineer of SW & Contents Research Laboratory, ETRI, Korea. He received his Master degree at Chonbuk National Univ. in 2001. His key research interests are: Unmanned Vehicle, Software Defined Networking and Future Internet.