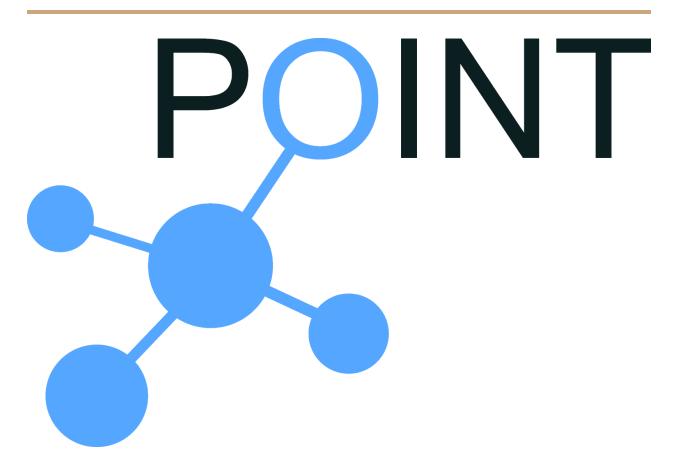
H2020 iP Over IcN- the betTer IP (POINT)

## **Design-ICN-SDN-Application**POINT ICN-SDN Application Design



List of Authors: George Petropoulos, Konstantinos Katsaros

1.Introduction 2.Implementation

## 1.Introduction

The ICN-SDN application implements the server side of the TM-SDN interface and supporting functionalities. It implements relevant methods and functionalities, including:

- The definition of the TM-SDN protocol format using Google Protocol Buffers, the generated header and source files, as well as the supporting methods to read and write Google Protobuf objects to streams.
- The server side of the ICN-SDN interface, which handles all the cases of received messages and coordinates next functions.
- The resource management functions, which provide unique node, link and internal link identifiers, and update the state of the topology.
- Supporting configuration methods, dealing with reading of network interfaces and initialization and update of Click files (used for bootstrapping purposes).

The ICN-SDN is assumed to be running on the same machine as Topology Manager does. This will facilitate the topology update in case of new or removed links, the resilience handling in case of link failures and the traffic engineering functions for path calculation. In the future, the ICN-SDN will be integrated into the Topology Manager, in order to make the resource assignment process part of the whole topology management functions.

## 2.Implementation

In terms of implementation, the ICN-SDN application is implemented as a boost:asio server application, and consists of the following source files. The key methods are also presented briefly, while more detailed documentation is exported at doc/ folder of the source repository, generated by Doxygen.

- server.cpp: The server class, implementing the server side of the TM-SDN interface, and also listening for bootstrapping protocol messages.
  - void start\_accept(): The method which listens for incoming TM-SDN messages and acts accordingly. In case of ResourceRequest messages it calls the resource\_manager module and returns the assigned identifiers included in ResourceOffer message. In case of TrafficMonitoring and LinkStateMonitoring messages, it prints the received information.
  - o int main(int argc, char\* argv[]): The main method initializing the required threads: the server process and bootstrapping message listener.
- resource\_manager.cpp: The resource manager class. It will generate unique node, link and internal link identifiers, and also keep track of assigned ones.
  - string generate\_node\_id(string node\_information): The method which generates a new node identifier, given node's information. It is used to

generate unique node identifier for requests performed by the SDN controller. New nodes are identified by their Openflow identifier.

- void generate\_link\_id( TmSdnMessage::ResourceRequestMessage::RecourceRequest request, TmSdnMessage::ResourceOfferMessage::RecourceOffer \*offer): The method which creates a resource offer for a received resource request. It is used to generate unique link identifier for resource requests via the TM-SDN interface. It is the main method which orchestrates the rest of functions for the generation of unique resources for SDN nodes. It first checks whether the node has not unique identifiers and then creates, otherwise returns the already provided ones. All are then included in the resource offer to be sent back to the SDN controller.
- o string generate\_link\_id(string node\_information, string attached\_node\_information): The method which generates a new link identifier, given node's information. It is used to generate unique link identifier for requests performed by the SDN controller. New links are identified by the SDN controller-internal identifiers of the 2 nodes being connected.
- string generate\_internal\_link\_id(string src\_node\_id, bool flag): The method which generates a new link internal identifier. It is used to generate unique internal link identifier for a new node.
- void addNode(string src, bool tmrv): The method which adds a new node to the topology graph.
- o void addNodeConnector(string src\_node, string dst\_node, string lid\_bits, string ilid\_bits): The method which adds a new node connector to the topology graph.
- void updateGraph(): The method which updates the topology graph.
- configuration.cpp: The class for creating and updating click configuration.
  - void initialize click(): The method which initializes click.
  - void update\_click(string new\_node\_id, string new\_rv\_fid, string new\_tm\_fid, string new\_internal\_lid, string new\_link\_id): The method which updates the click configuration with the new parameters.
  - void read\_interfaces(): The method which reads the available network interfaces.