



University
POLITEHNICA
CA of
Bucharest



Faculty of
Automatic
Control and
Computers



Computer
Science and
Engineering
Department

Graphical Packet Generator

Bachelor Project – July 2014

Author

Oana Niculăescu
oana.niculaescu@cti.pub.ro

Scientific Advisor(s)

Prof. Dr. Ing. Răzvan Rughiniș
As. Drd. Ing. Mihai Carabaș
Șl. Dr. Ing. Laura Gheorghe



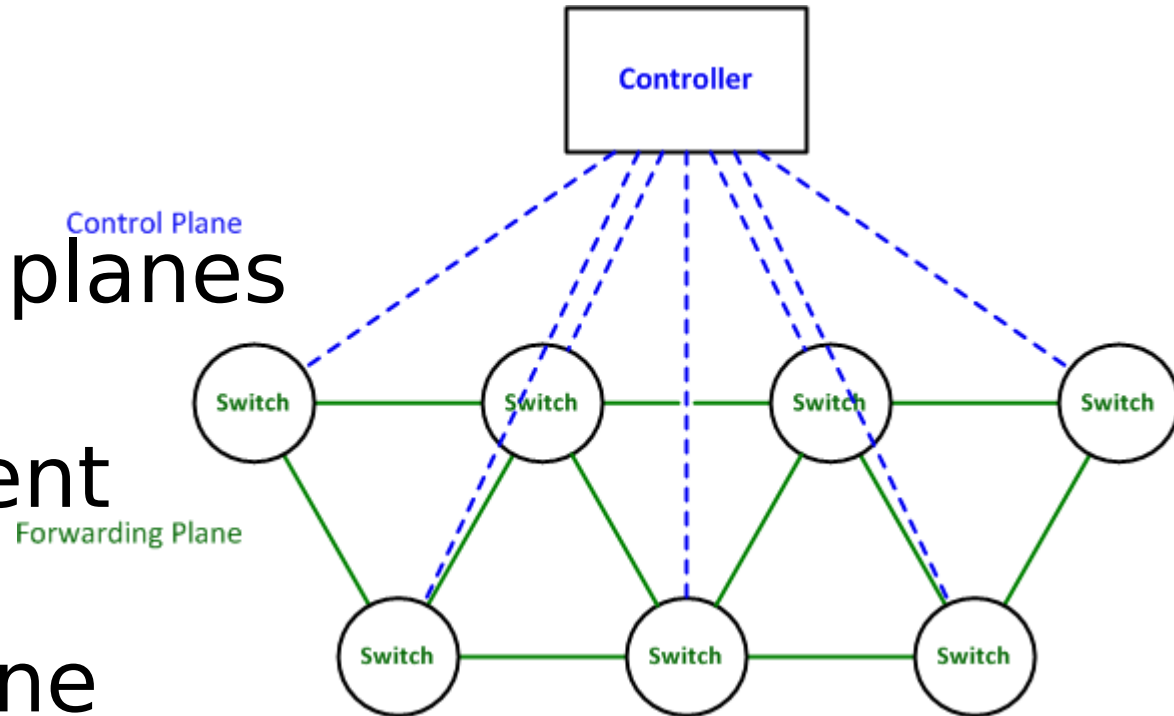
Thesis objective

- GPG application consistent with SDN principles, onePK testing
- automatically discover the topology
- packet generation inside a network from a central point



Classic Networking

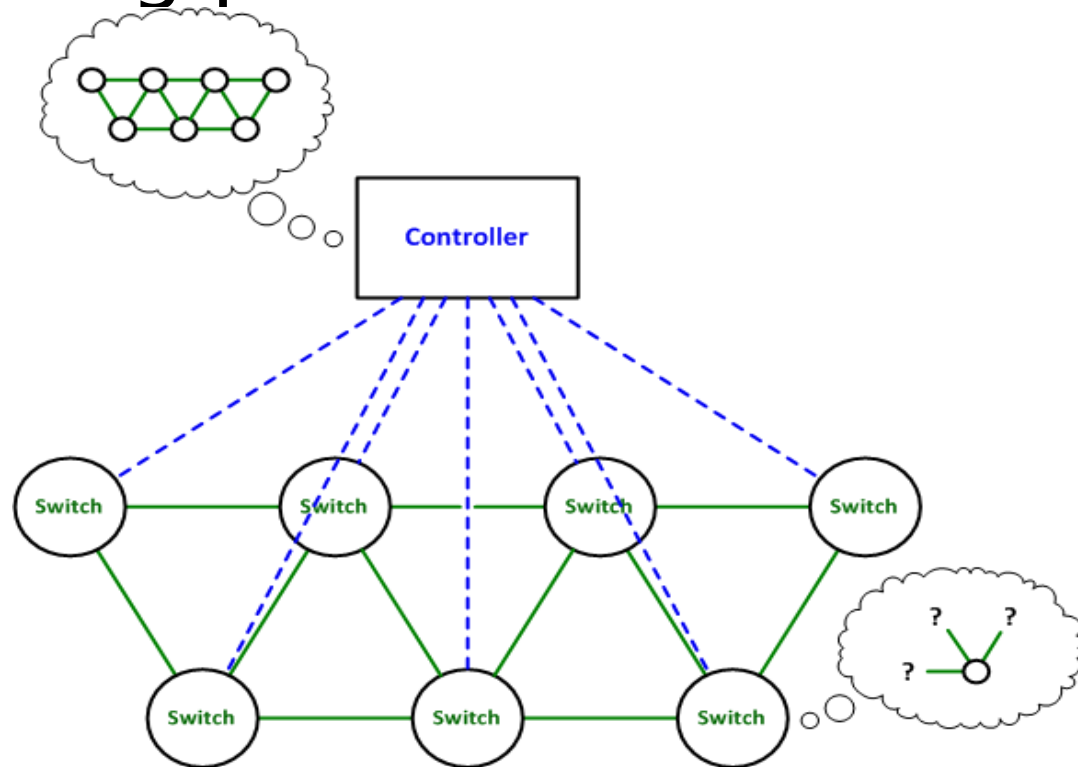
- 3 conceptual planes of operation:
 - management plane
 - control plane
 - forwarding plane





Software Defined Networking

- separation of control and forwarding planes
- onePK



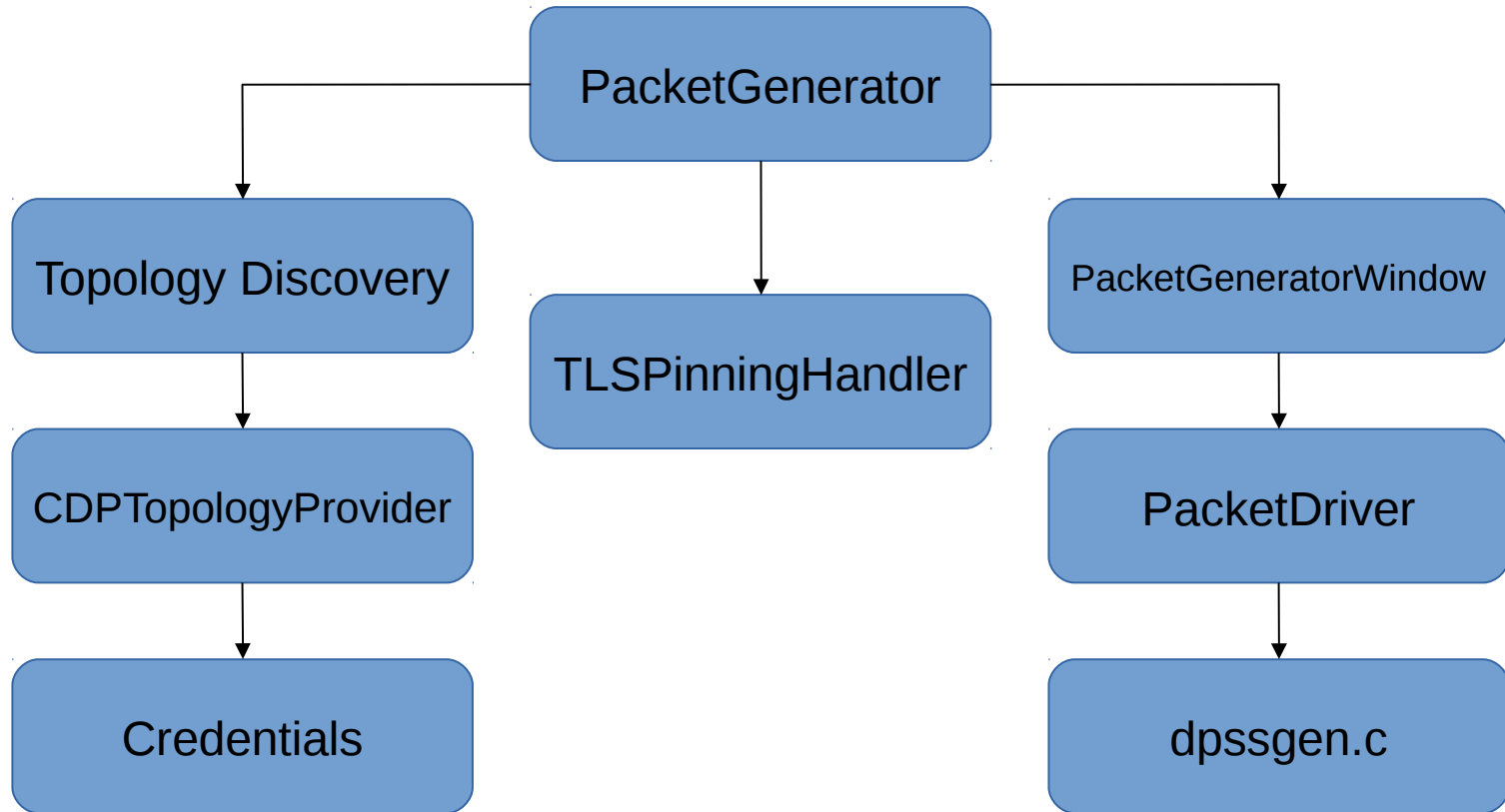


Motivation for GPG

- eliminate box-by-box configuration
- gain end-to-end traffic control
- manage the network by policy, programmatically



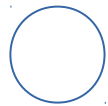
GPG Design Overview





Automatic Topology Discovery

- Discovery Service Set – Topology Discovery
 - CDP protocol discovery



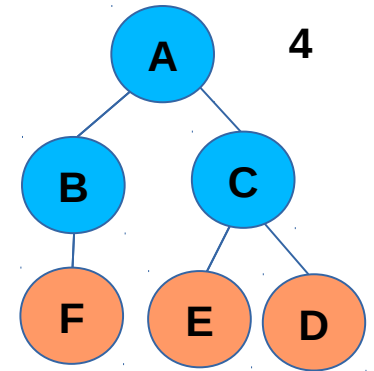
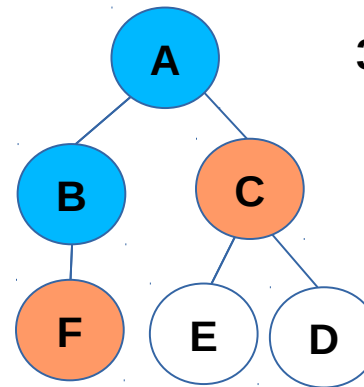
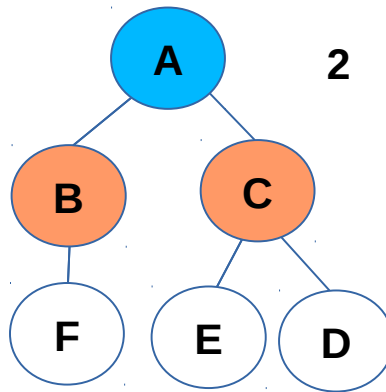
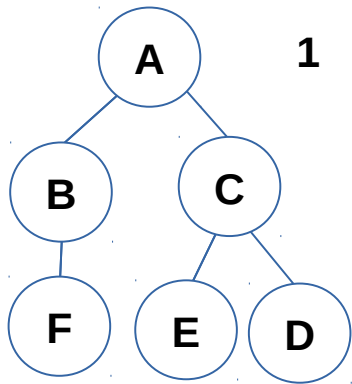
Undiscovered
node



Discovered
node



Discovered and
connected to node





Packet Generation

- Data Path Service Set
- `onep_dpss_inject_raw_packet`:
 - next hop action – explicit next hop IP
 - divert action – original packet and location set at PREROUTING
 - divert action – original packet and location set at OUTPUT

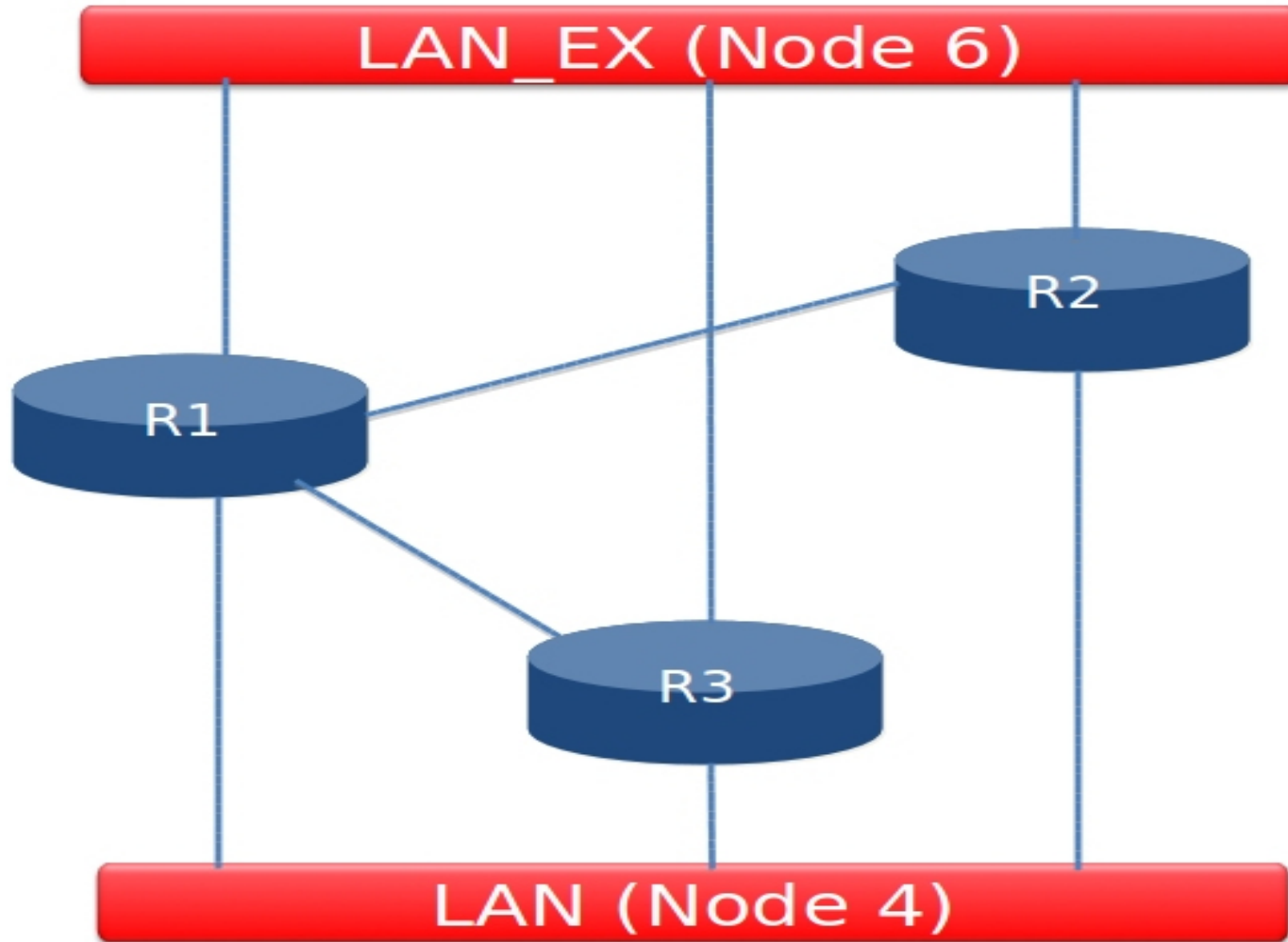


Implementation Results

- a common interface
- automatic topology discovery,
consistent view of the network
- packet generation from a central
point



Testing Topology





The Interface & Packet Generation

DPSS Packet Generator Settings for Router1

Destination MAC:
(e.g., 00:11:22:33:44:55)

Source MAC:
(e.g., 00:11:22:33:44:55) ☒ Use Interface MAC

Ether Type:

Interface:

Number of Packets: Inter-Packet Gap: ms

DSCP Value:

TTL:

Source IP:

Destination IP:

Protocol: ☐ Raw ☐ UDP ☒ ICMP

Protocol Number:

ICMP Payload:
(space-separated hex)

```
d6 af b6 95 32 41 73 5b b4 89 a8 31 d0 64 43 d3
cf 3d eb 23 92 41 e8 8e 65 ef ef b0 e9 7e c0 c8
84 f7 de 80 66 9a 13 5c df d8 c9 63 85 98 5d e6
cb 25 ee 05 ed ef 45 75 8f 52 35 4e 7d bb 0b 45
```

Source Port:

Destination Port:

Sent 3 packets



Testing Results & API Limits

- only level 2 protocols supported for network discovery
- no neat way to get the MAC address
- only level 2 packets can be created inside the API
- feedback for the generated flows



Future Work

- a more efficient way to discover the topology
- use of an external tool to create packets inside the API (libnet)



Conclusion

- Graphical Packet Generator
- automatic network discovery
- packet generation from a central point
- testing the application and the restrictions of the API