Autonomous Fast Rerouting for Software Defined Network

2012.10.29

NTT Network Service System Laboratories,
NTT Corporation

Shohei Kamamura, Akeo Masuda, Koji Sasayama

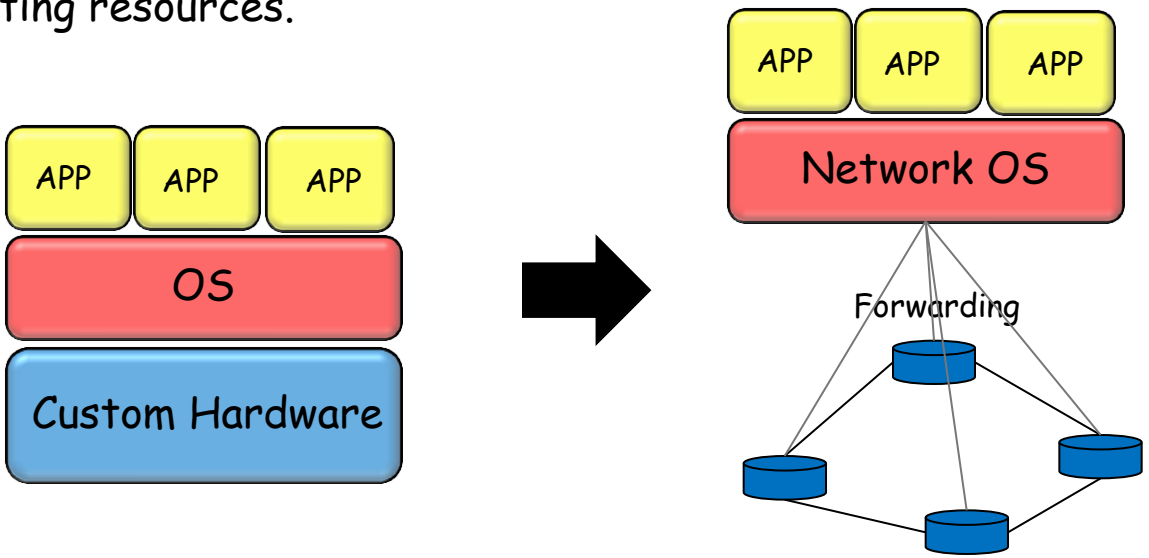
NTT

Outline

1. Background and Motivation
2. Requirements for Carrier-grade SDN
3. Problem Statement and our Approach
4. Related Works of IP Fast Rerouting
5. Autonomous Fast Rerouting for SDN (SDN-FRR)
6. Conclusion

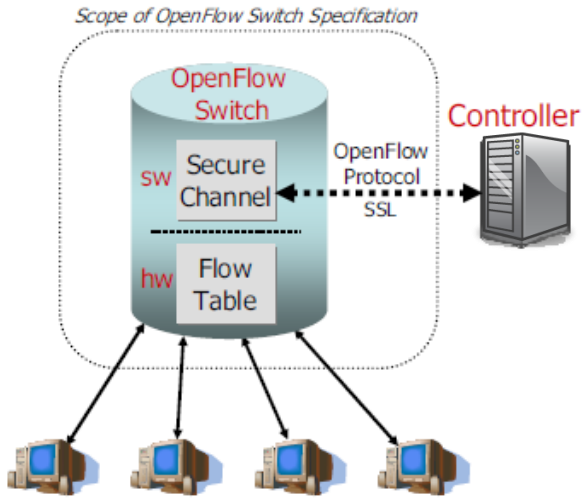
Background and Motivation

- Recently the main players of the development of networking technologies seem to be shifting to operators and users of datacenter
- Developers are eager to totally program the operation of not only their computing equipment, but also the network
- The concept of **software defined network (SDN)** is expected to release the network operation from time-consuming tasks such as manual configuration
  - similar to the way of management where the cloud operators program the usage of computing resources.



Enabler of SDN

- **OpenFlow**: Interface between NE and Software-based controller
- Separation of control functions from forwarding hardware
  - Forwarding architecture consists of OpenFlow **controller** and OpenFlow switches (**OFS**s).
  - The controller computes the routes for each OFS, and each OFS only have to forward data packets.
- Flow-based routing (switching) using **flow table**
  - Each switch has the forwarding table called flow table, and flow table consists of match fields, Instructions, and Counters



Example of flow table

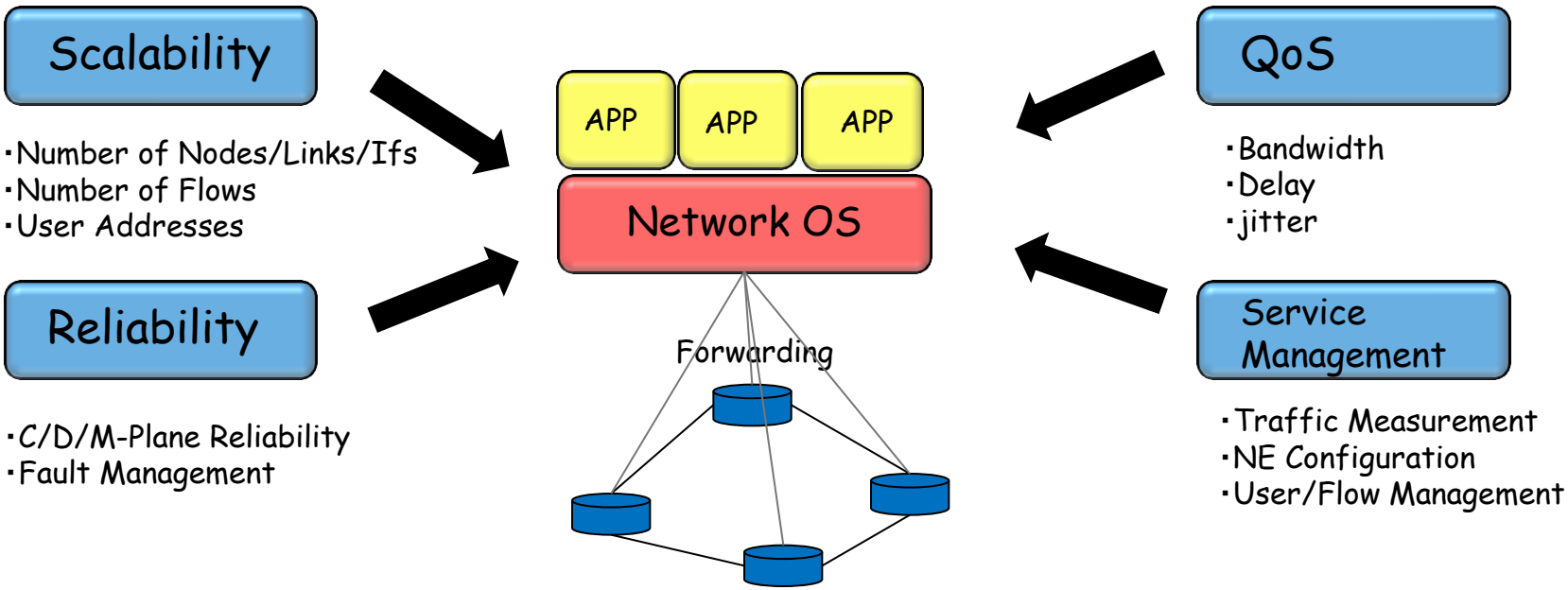
Match with (~15)-tupples

Match Fields	Instructions
DST=10.0.0.0/8 ToS==0	Forward IF #1
DST=10.0.0.0/8 ToS==1	Forward IF #2

[1]N. McKeown, et al., ``OpenFlow: enabling innovation in campus networks,'' in Proceedings of ACM SIGCOMM Computer Communication, 38(2):69-74, April 2008.

Requirements for Carrier-grade SDN

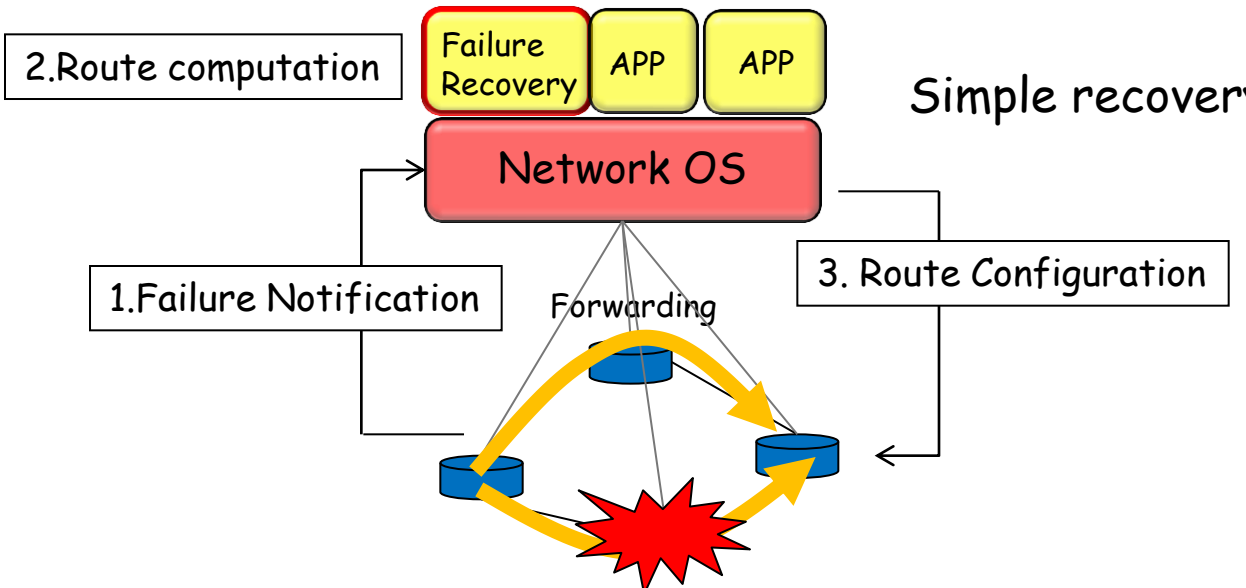
- The term **carrier grade** describes a set of functionalities and requirements that architectures should support in order to fullfill the operational part of network operators [2]
- The main requirements are (1) Scalability, (2) Reliability, (3) Quality of Service, and (4) Service Management



[2]D. Staessens, et al., "Software Defined Networking: Meeting Carrier Grade Requirements, " in Proc. of LANMAN 2011.

Problem Statement

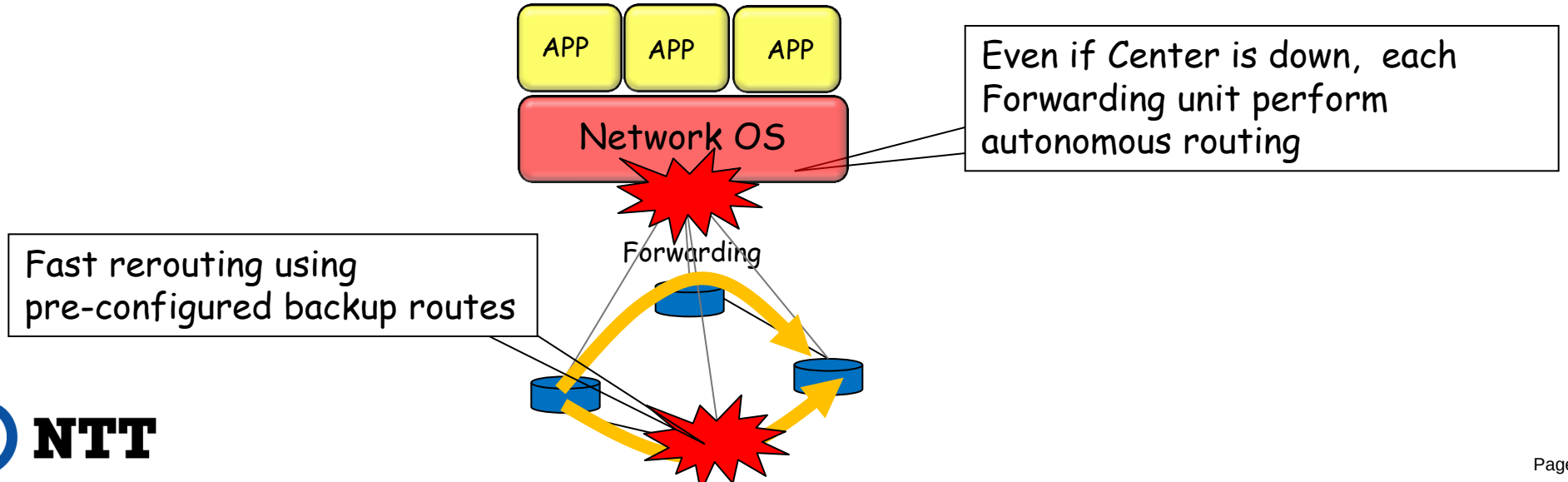
- We focus on the **Reliability for SDN**
- As an simple recovery model is implementing "Failure recovery App" on a central controller
  - ➔Transmission delay between the forwarding unit and the central controller
  - ➔Reliability depends on not only forwarding unit but also C-Plane (Central controller)
- Our problem is establishing the recovery scheme for SDN, which achieve **carrier-grade fast (sub-50ms)** recovery and does not depend on central controller



Simple recovery framework for SDN

Approach: Fast Rerouting

- The key idea is embedding **autonomy** to each forwarding unit
  - The goal is realizing fast restoration, which does not require the reactive update and global convergence, by preconfiguring backup routes
  - Even if the central controller or control-plane is down, each node behave autonomously
  - Autonomous actions are performed **in controllable range**
- Embedding autonomy is realized by pre-configured backup forwarding table, which is inspired by **IP Fast Rerouting**



NTT

Related Works of IP Fast Rerouting

- Improving IP resilience is classified as reactive and proactive approach
- IP fast rerouting was proposed for improving IP resilience using proactively configured backup routes such as MPLS protection
- There are many methods for realizing IP fast rerouting
  - We focus on **the backup topology-based IP fast rerouting**, which has both extensibility and feasibility for actual use

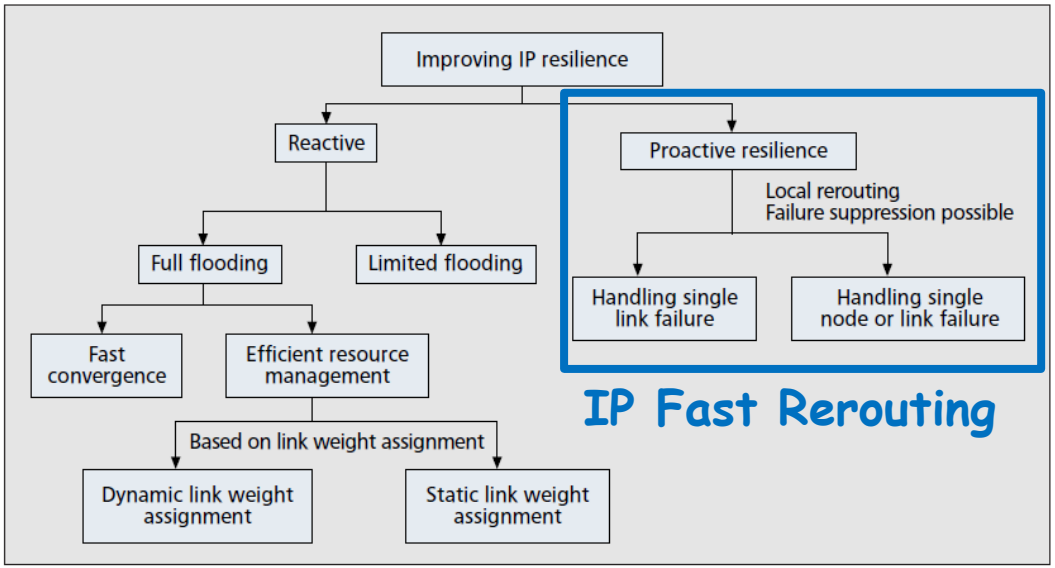
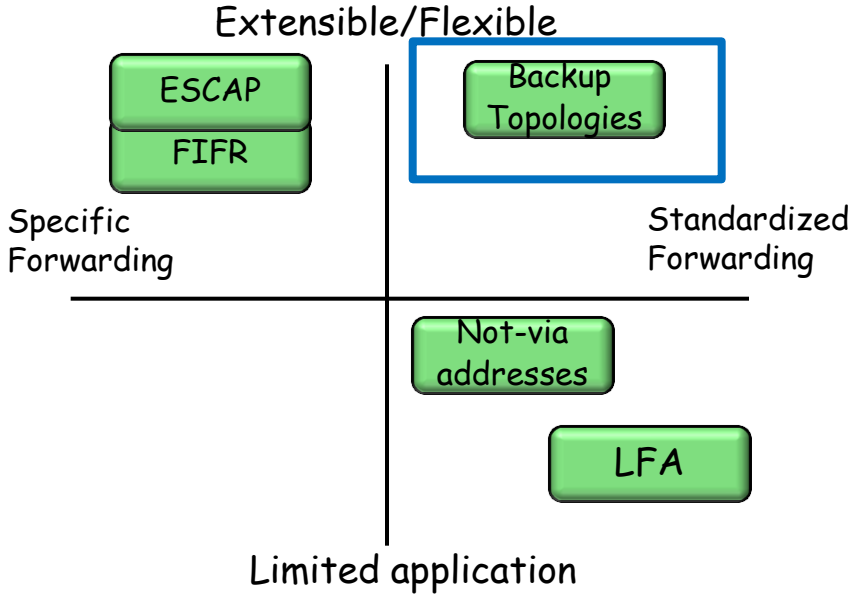


Figure 2. *Classification of various proposals for IP resilience (please see the article for references).*

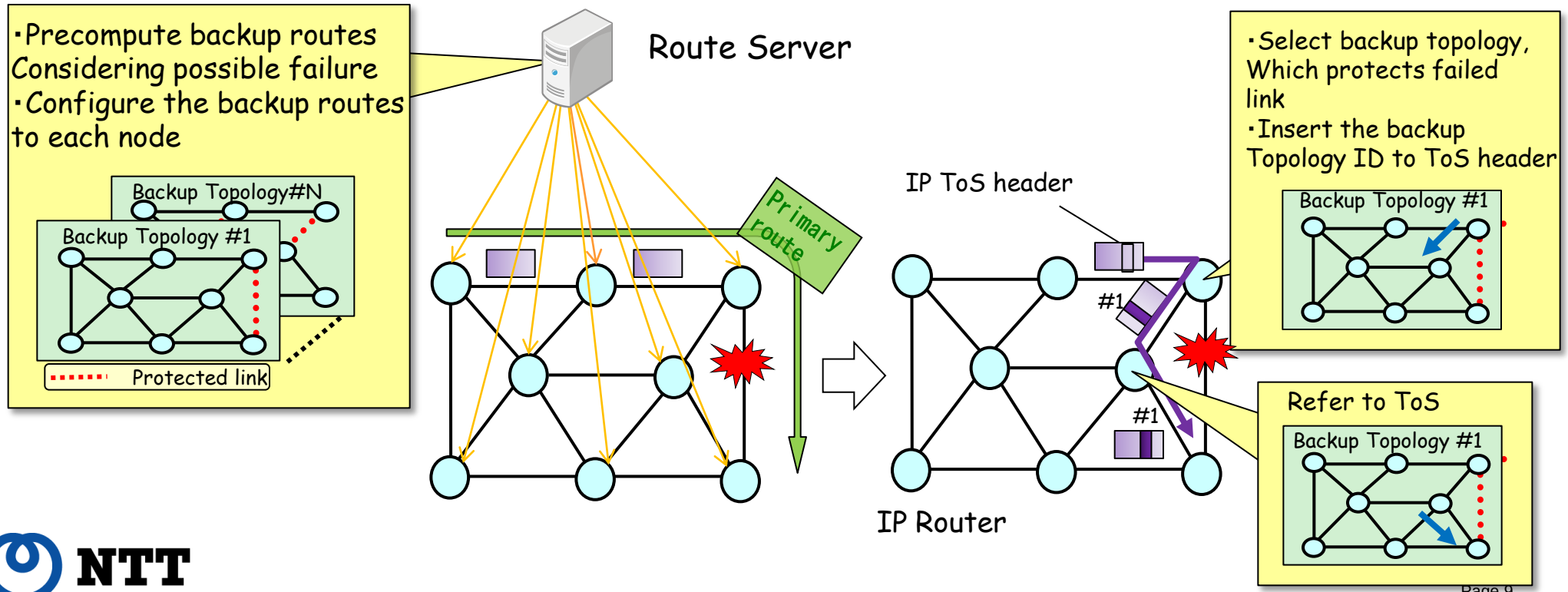


Existing works for IP fast rerouting

[3]S. Rai, et al., ``IP Resilience within an Autonomous System: Current Approaches, Challenges, and Future Directions,'' in Proceedings of IEEE Communication Magazine 2005.

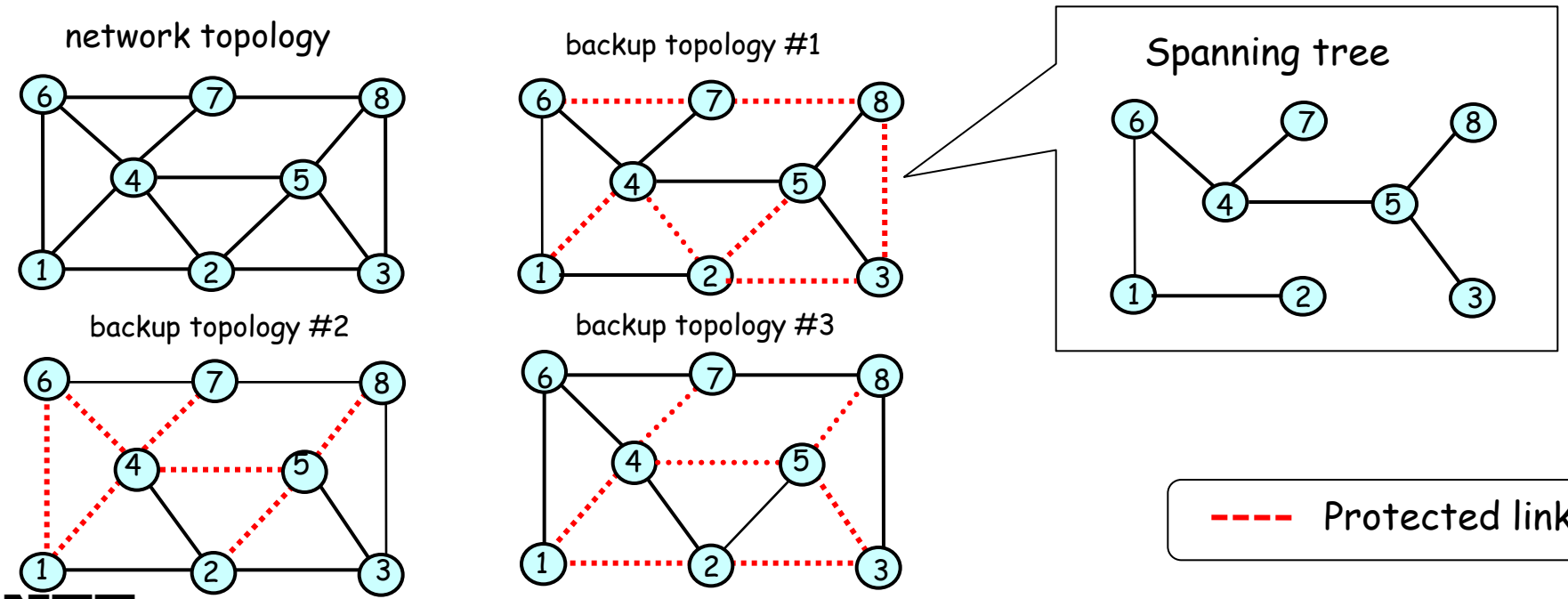
IP Fast Rerouting using Backup Topologies

- **Backup Topologies** are distributed by route server
  - Backup routes are precomputed based on backup topologies, which provides shortest path without **protected link**.
- Each router forwards the packet according to backup route based on backup topology by referring to the packet header.



IP Fast Rerouting using Backup Topologies

- An arbitrary single failure is protected at least one backup topology
- For reducing the number of backup topologies, one backup topology protects multiple links
  - the number of backup topologies is proportional to the size of the forwarding table
- To minimize the number of backup topologies, each backup topology is made such that the topology, excluding the protected links, becomes a spanning tree [4]



[4]S. Kamamura, et al., "Scalable Backup Configuration Creation for IP Fast Reroute," IEICE Trans. Commun. Vol. E94-B, No-1, Jan. 2011.

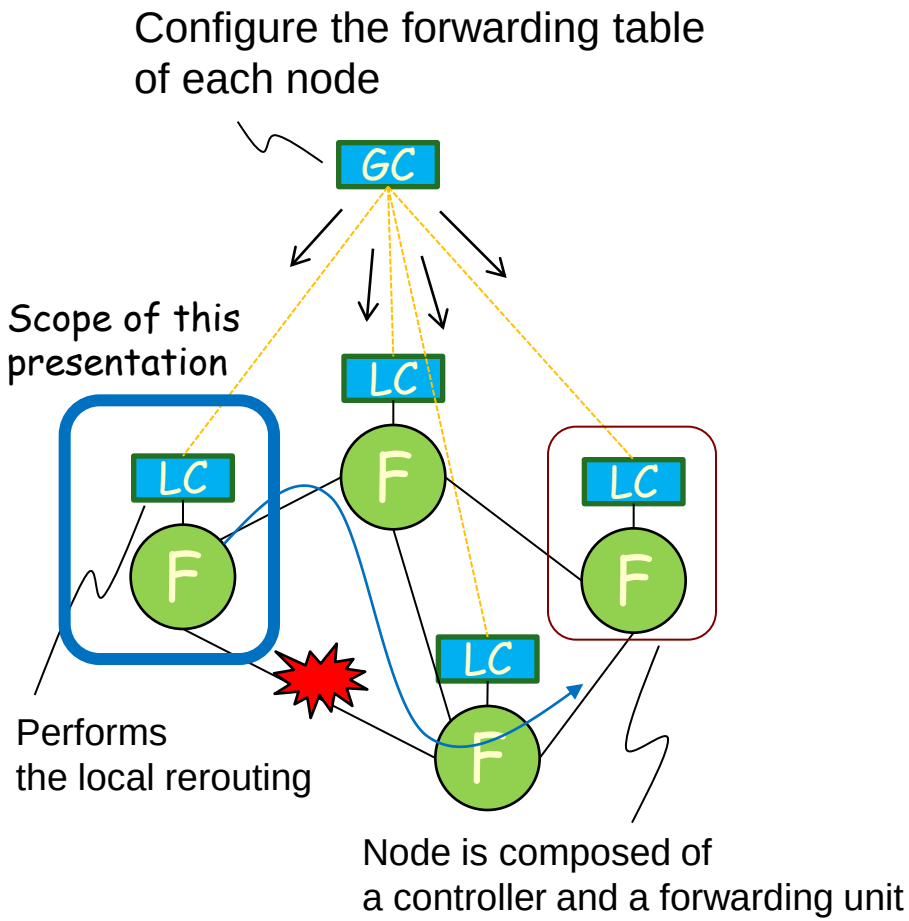
Autonomous Fast Rerouting for SDN

- We embed the autonomy to SDN framework
- We utilize **transparency of forwarding control** functions of **OpenFlow**
  - OpenFlow physically separates its control functions from the node, and they are provided as programmable area to operators
  - As a result, our proposal realizes the **user-driven and common implementation for IP fast rerouting** without any extension of OFS.

NTT

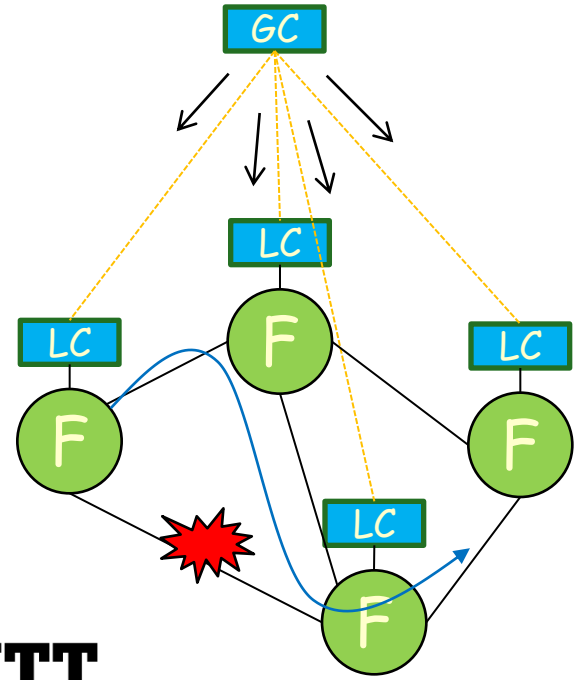
Network model

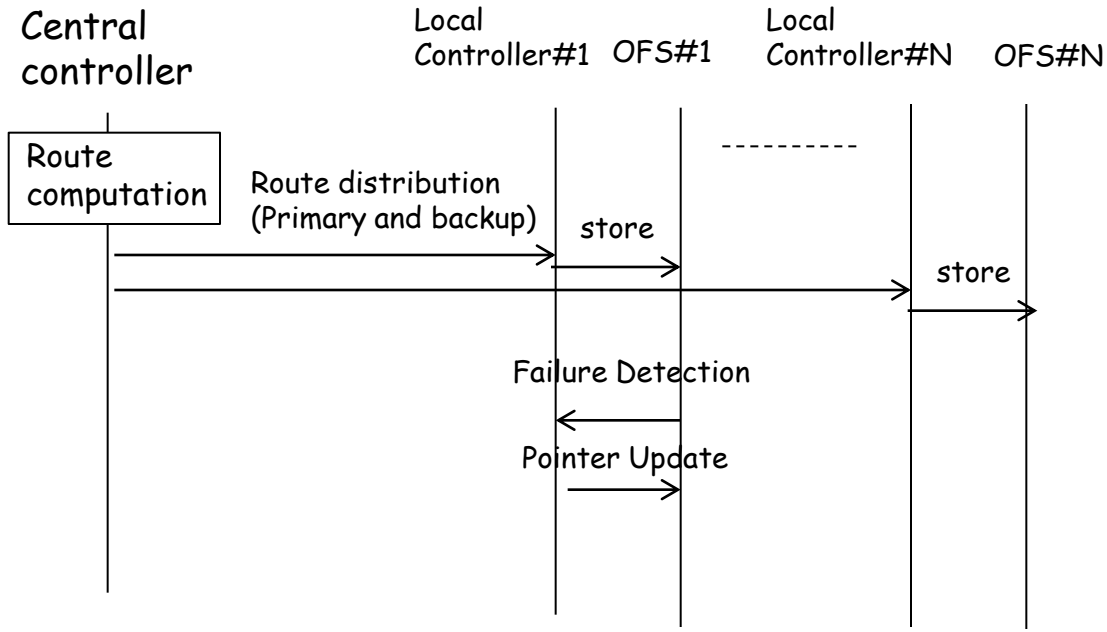
- There are **two types of controllers**: a global controller, which controls the whole network, and local controllers, which performs local restoration
- Each local controller is assigned to each open flow switch, and we regard the set of OFS and local controller as one node.
- Key features
  - ignore the propagation delay between the global controller and local one, and online recomputation for restoration
  - Simple local restoration does not depend on the state of global controller



Overview of network control

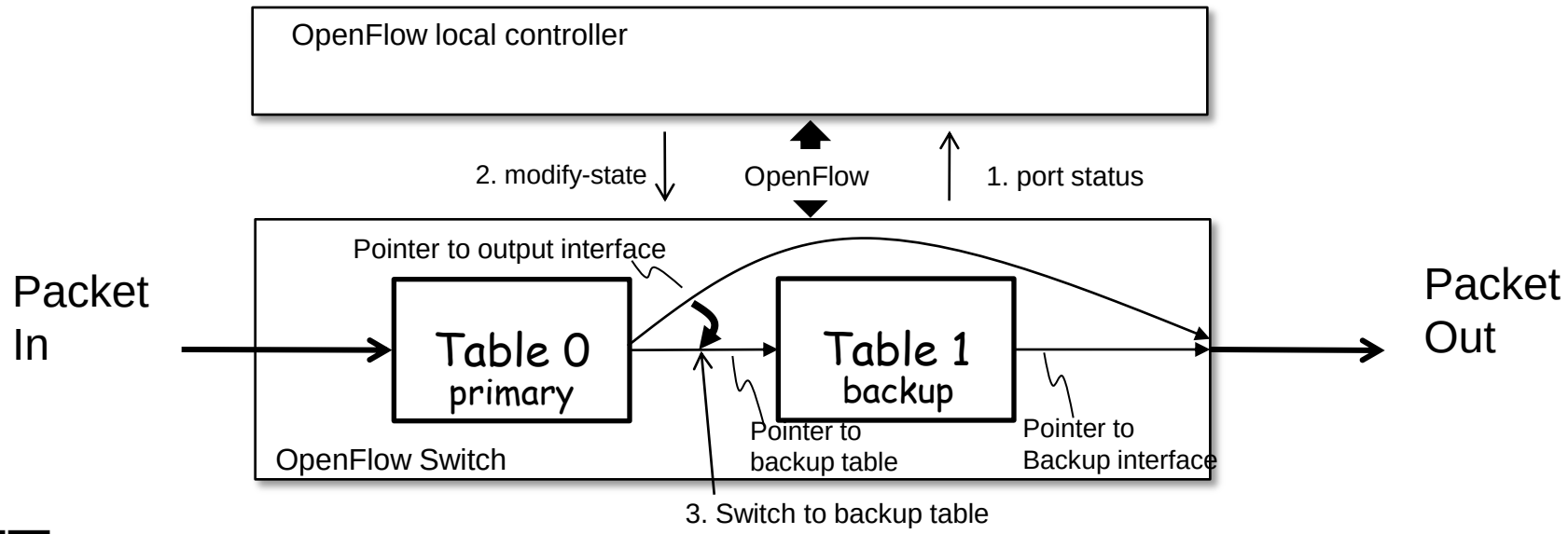
- Central (Global) controller distribute primary routes and backup routes to each OFS
  - Backup routes computed from backup topologies are aggregated, and stored as one backup table
- When OFS detects an failure, it performs local repairing without the central controller





Realization of Fast Rerouting using OpenFlow (1/2)

- For realizing FRR, we utilize the **pipeline processing** with multiple flow tables, which is specified by OpenFlow v1.1
- Before a failure occurrence, entry on a primary flow table indicates the output interface
- If a failure is detected by the local controller, it modifies the entry of primary flow table so that it indicates the backup flow table



Realization of Fast Rerouting using OpenFlow (2/2)

・Before a failure occurrence,
Packet "10.0.0.1" is sent through IF#1

…g IP network (IP-FRR)

Table 0 (before a failure detect…

Match fields		Instr…ons
ToS	DA	
#0	10.0.0.0/8	Forward to IF #1
. .	. .	. .
#1~#63	*	Lookup Table1

modify-state

Table 0 (after a failure)

modify-state

Match fields		Instructions
ToS	DA	
#0	10.0.0.0/8	Set ToS = #1 Lookup Table1
		. .
		Lookup Table1

・After a failure occurrence,
Controller changes instructions fields;
・change the ToS value
・lookup table1 instead of send to IF#1

DA: Destination Address
*: Wild card

Pointer to backup table for relay nodes

・For Relaying nodes on the backup route, they should also forward the packets using backup table
・This is realized by referring to Pointer to backup table.
・Because ToS of packet changes, they do not match primary entries but match pointer to backup table, and it also refer to Table 0

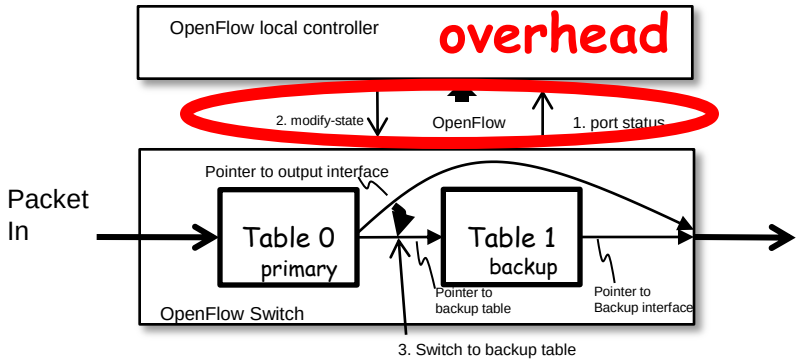
・On the table 1, output interface is determined by looking up it with a key composed of ToS value and DA

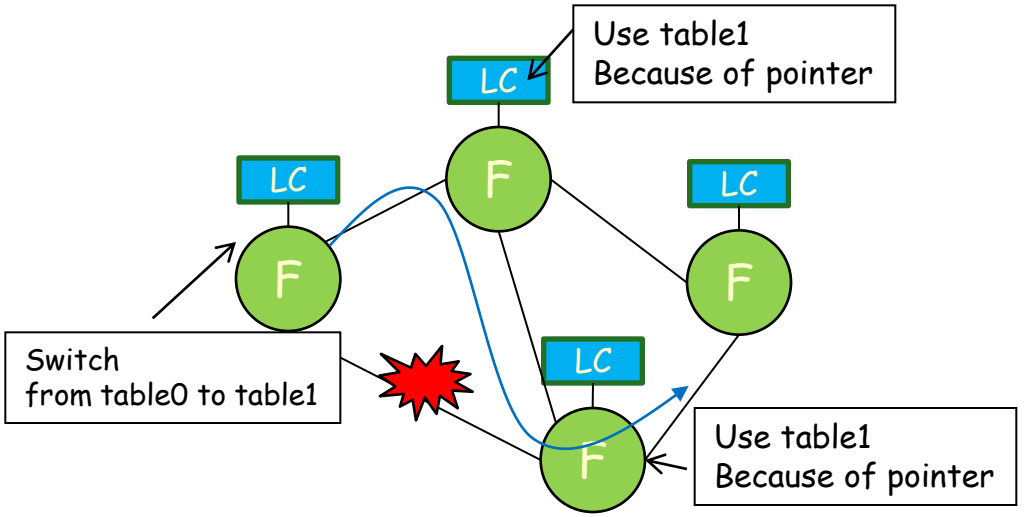
Table 1 (Backup Table)

Match fields		Instructions
ToS	DA	
#1	10.0.0.0/8	Forward to IF #2
#2	10.0.0.0/8	Forward to IF #3
. .	. .	. .
#63	10.0.0.0/8	Forward to IF #3

Overhead of our SDN-FRR

- Though there are no overheads for the restoration time on the relaying OFS, **failure detecting OFS requires the partial update process** for the restoration
- The time for updating one entry is about 146μs [5], the overhead of restoration time is "**146μs x number of updated entries**"





[5]P. Francois et al., ``Achieving sub-second IGP convergence in large Ipnetworks,'' ACM SIGCOMM Computer Communication Review Volume 35 Issue 3, July 2005

Example of Numerical Result

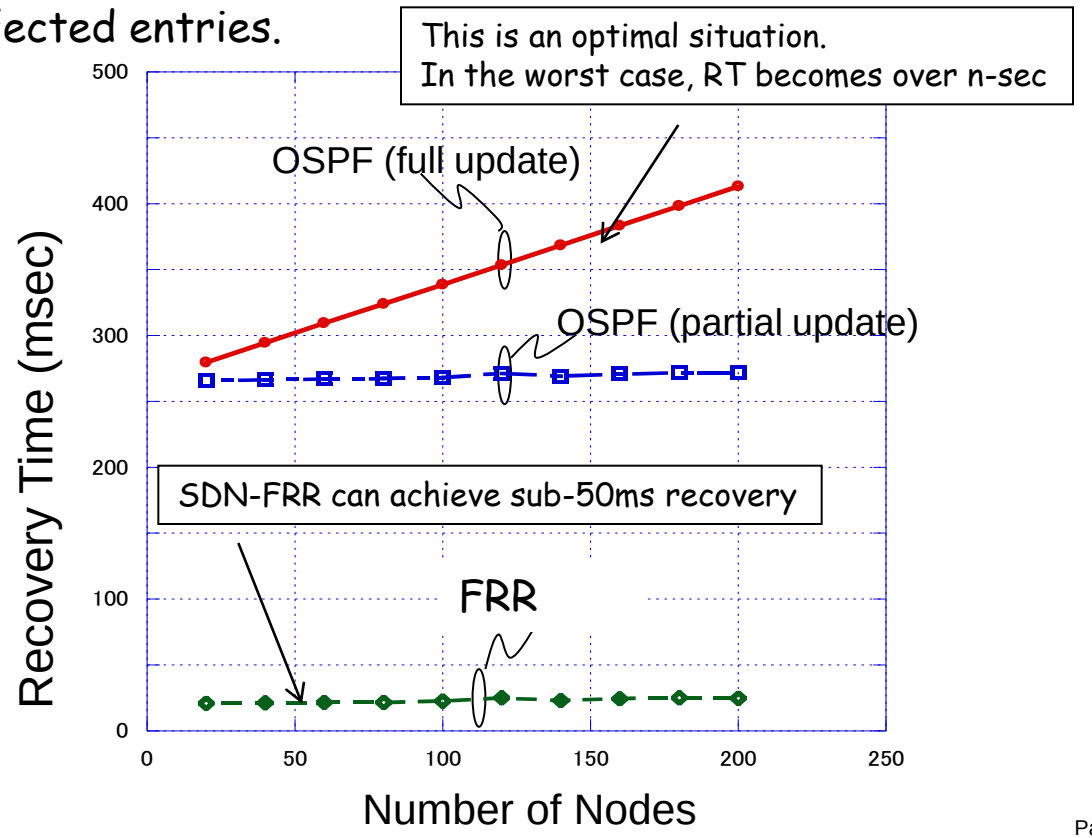
- Assuming to apply IP network, we evaluated the restoration time compared to the current distributed protocol (OSPF)
- Though our FRR architecture requires the partial update process for the restoration, it is negligible, and it achieves **sub-50ms recovery**
  - There are a small number of affected entries.

Restoration time formulation with SDN-FRR

$$T_{frr} = T_{detect} + T_{update}$$

Restoration time formulation with OSPF

$$T_{ospf} = T_{detect} + T_{lsao} + T_{flooding} + T_{spf-delay} + T_{spf-calc} + T_{update}$$

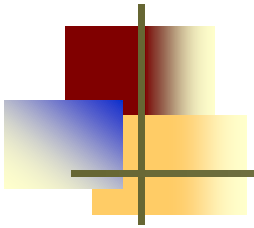


Conclusion

- Conclusion
  - We focus on the problem to realize the **carrier-grade reliability for SDN**
    - Fast Recovery (sub-50ms) and independency of the C-Plane state
  - We **embed the autonomy** inspired by IP fast rerouting to SDN
  - We provides implementation framework of SDN-FRR using OpenFlow, and can achieve **sub-50ms restoration**

- Future works
  - will implement our IP fast rerouting method for proof of concept
  - will develop the recovery optimization framework that uses both local and global repairing as the situation demands.

NTT



Backup Slides

Simulation Conditions

For the variables $T_{spf-calc}$ and T_{update}, we measure them by computer simulation. $T_{spf-calc}$ is equal to the Dijkstra shortest-path calculation time, whose computation increases as the square of the number of nodes. For the T_{update}, we count the number of entries in the forwarding table for each router, and it is multiplied by $146\mu s$. For the partial update, we compute the worst case where the link, which has the most number of flows, fails. Above simulations are performed in the system with CentOS 5.5, quad-core Xeon 1.86GHz, and 32GByte memories. For the fixed value T_{detect}, T_{lsao}, $T_{flooding}$, and $T_{spf-delay}$ are set to fixed values 20ms, 12ms, 33ms and 200ms, respectively.

NTT