

ITUEvents

Machine Learning for Wireless LANs +  
Japan Challenge Introduction  
ITU-ML5G-PS-031, ITU-ML5G-PS-032  
29 July 2020

ITU  
**AI/ML in 5G**  
Challenge

*Applying machine learning in  
communication networks*

ai5gchallenge@itu.int

Register [here](#)  
Join us on [Slack](#)

Sponsors



Organizer



ITU AI/ML in 5G Challenge

Global Round in Japan

# ITU AI/ML in 5G Challenge Global Round in Japan

## Organizers



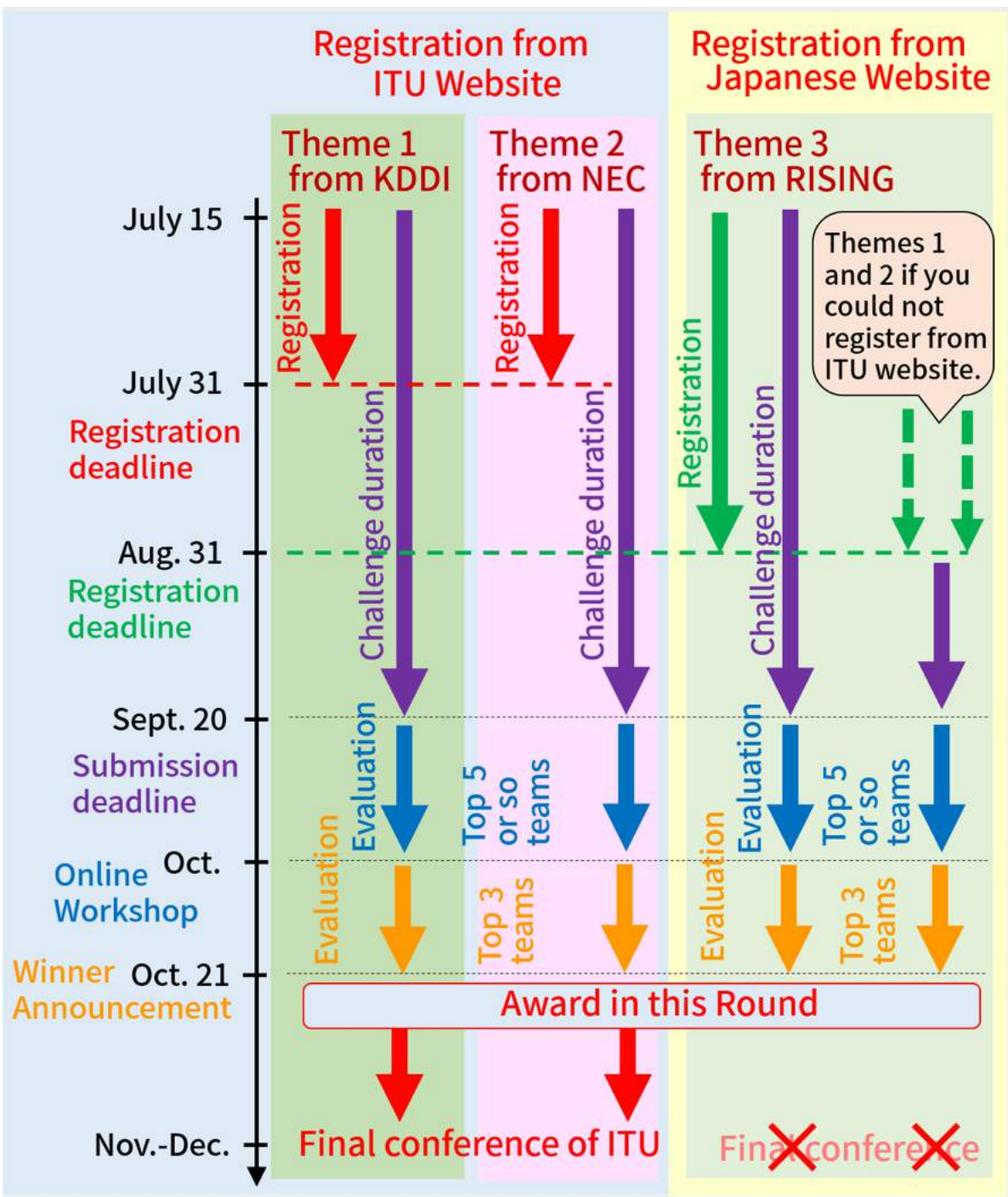
## Theme 1 from KDDI

Analysis on Route Information Failure in IP Core Networks  
by NFV-Based Test Environment

## Theme 2 from NEC

Network State Estimation  
by Analyzing Raw Video Data

Theme 3 from RISING  
(Not for Final Conference of ITU)  
WiFi Location Estimation



## Three Problem Sets:

- Theme 1 (KDDI)
- Theme 2 (NEC)
- Theme3 (RISING)

Submission Deadline: 2020/9/20



2019/11/26-27

246 participants

111 Posters

We have organized a cross-field (across 19 technical committees of IEICE) symposium to **apply AI/ML to networking**



# Today's Webinar Agenda

## Part 1: Invited Expert Talks

Advanced Traffic Classification Through In-Network Machine Learning, Prof. Akihiro Nakao (U of Tokyo),  
Machine Learning for Wireless LANs, Associate Prof. Koji Yamamoto (Kyoto University),

## Part 2: Problem Sets in the Global Challenge

ITU-ML5G-PS-031: Network State Estimation by Analyzing Raw Video Data. (Tomohiro Otani, KDDI Research, Inc)

ITU-ML5G-PS-032: Analysis on route information failure in IP core networks by NFV-based test environment.

Takanori Iwai(NEC Corporation)

[https://itu.zoom.us/webinar/register/9815956026267/WN\\_Pdc0-r05TmujTGX0gatprw](https://itu.zoom.us/webinar/register/9815956026267/WN_Pdc0-r05TmujTGX0gatprw)

[Invited Expert Talk]  
Advanced Traffic Classification Through In-Network Machine Learning

2020/7/29

Aki Nakao  
The University of Tokyo

Professor  
Vice Dean of Interfaculty Initiative in Information Studies  
Advisor to the President of The University of Tokyo

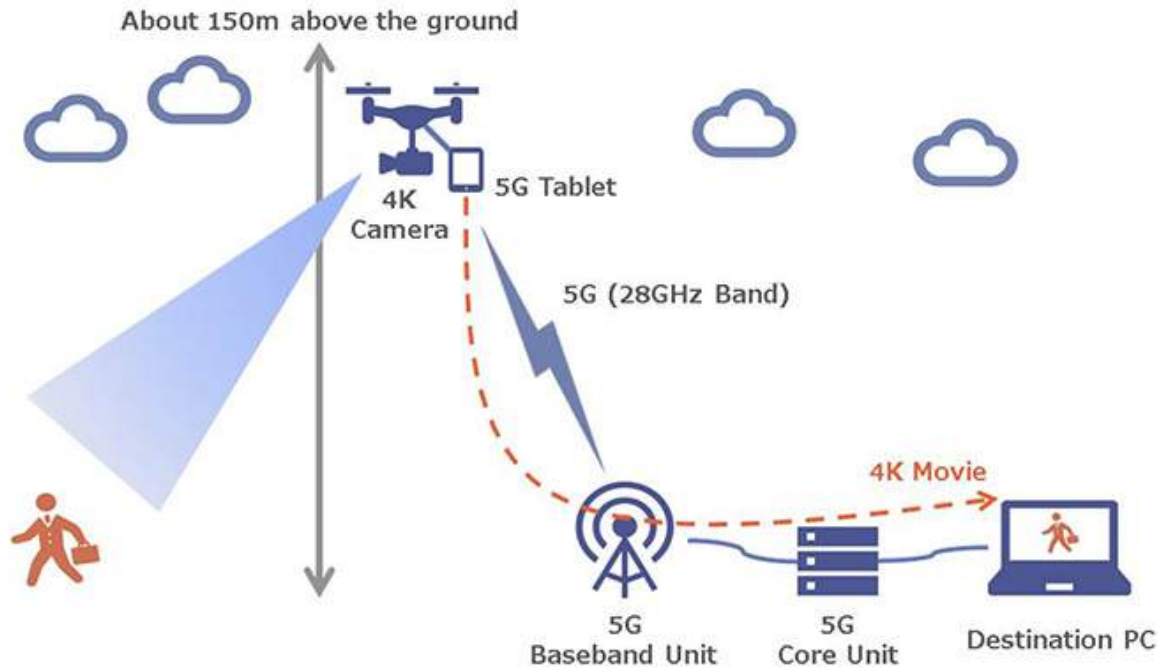
# Akihiro Nakao

- Advisor to the President of the University of Tokyo
- Vice Dean, Interfaculty Initiative in Information Studies
- Professor, the University of Tokyo
  
- Chairman of **5GMF Network Committee**
- Chairman of **Local5G Committee**, Broadband Association
- Various Roles in Ministry of Internal Affairs and Communication (MIC), **Japanese Government**
- Executive CTO of FLARE NETWORKS





# KDDI and The University of Tokyo Demonstrate Japan's First Real Time 4K Video Transmission using 5G Drone



Nakao Research Laboratory of The University of Tokyo together with KDDI has completed the field trial of a live 4K video transmission from a drone via 5G technology.

The experiment was carried out in an effort to realize consumer services that can benefit hugely from drones, such as public safety and surveillance, agriculture monitoring and disaster response.

The experiment area was set up in the university's Kashiwa campus using Samsung Electronics' 5G equipment such as a base station and a tablet. A video stream shot from the air using a 4K camera mounted on a drone was transmitted to the ground using the 28GHz frequency band designated for 5G mobile communication in the near future.

Text: Akihiro Nakao (Professor)

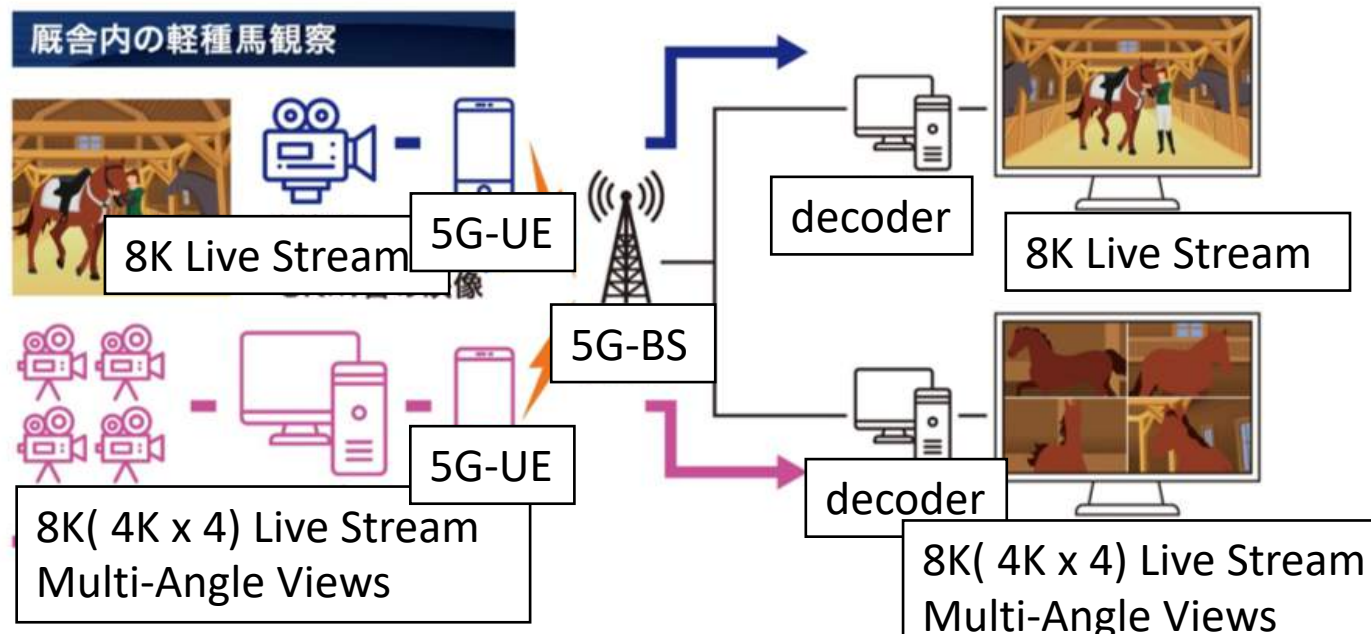
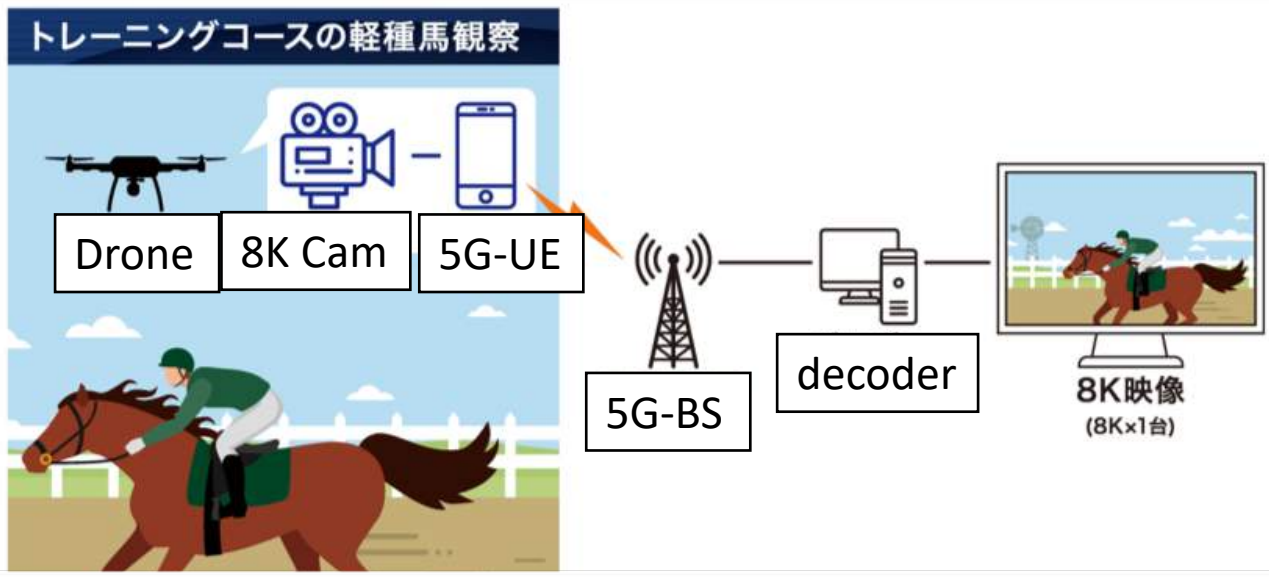
Proofreading: David Buist (Project Senior Specialist)



# 8K Live Streaming with 5G for Remotely Monitoring Race-Horses

2019/11 Released

## ~World-First 5G Drone 8K Live Streaming~



# 5G Live Videos Streaming and Realtime Control of Under-Water Drone

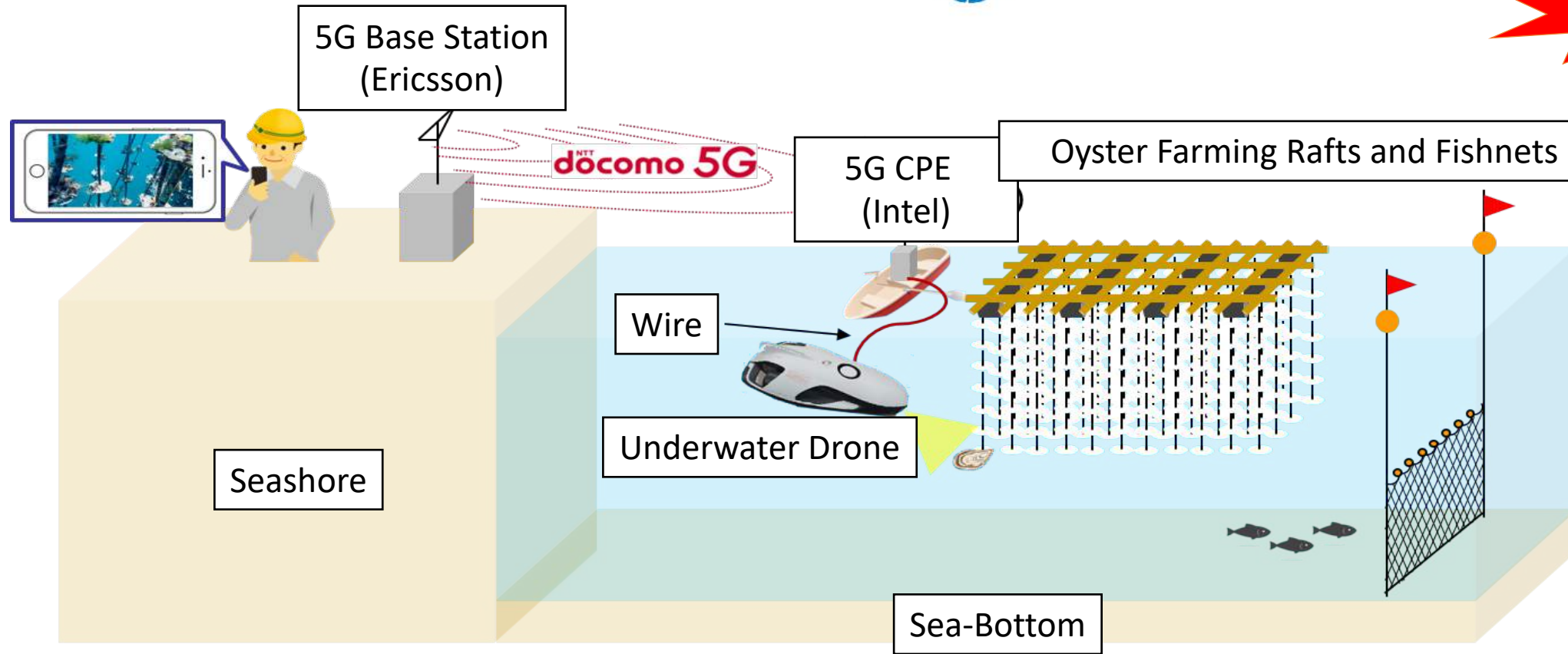
2019/11 Released



東京大学  
THE UNIVERSITY OF TOKYO

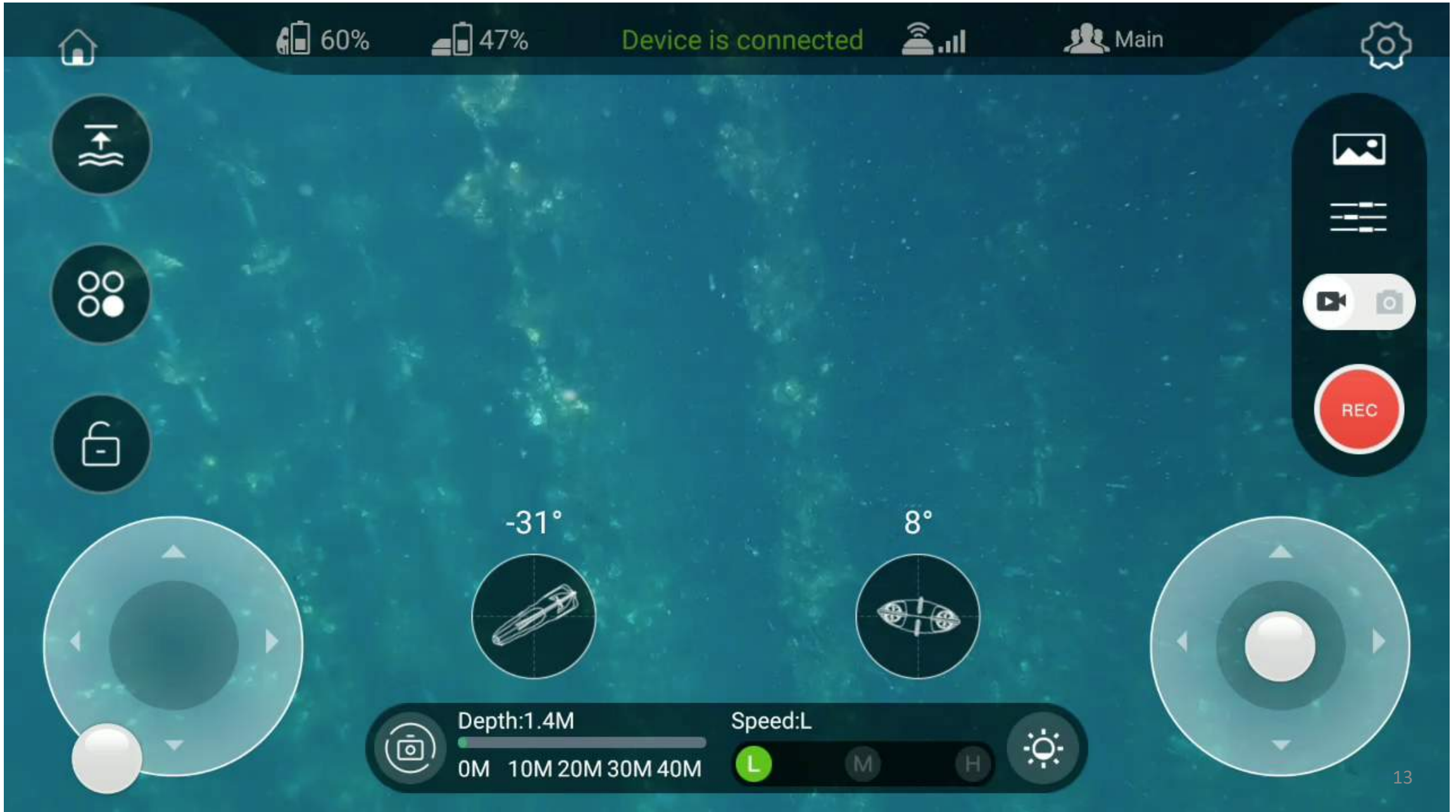
NTT docomo

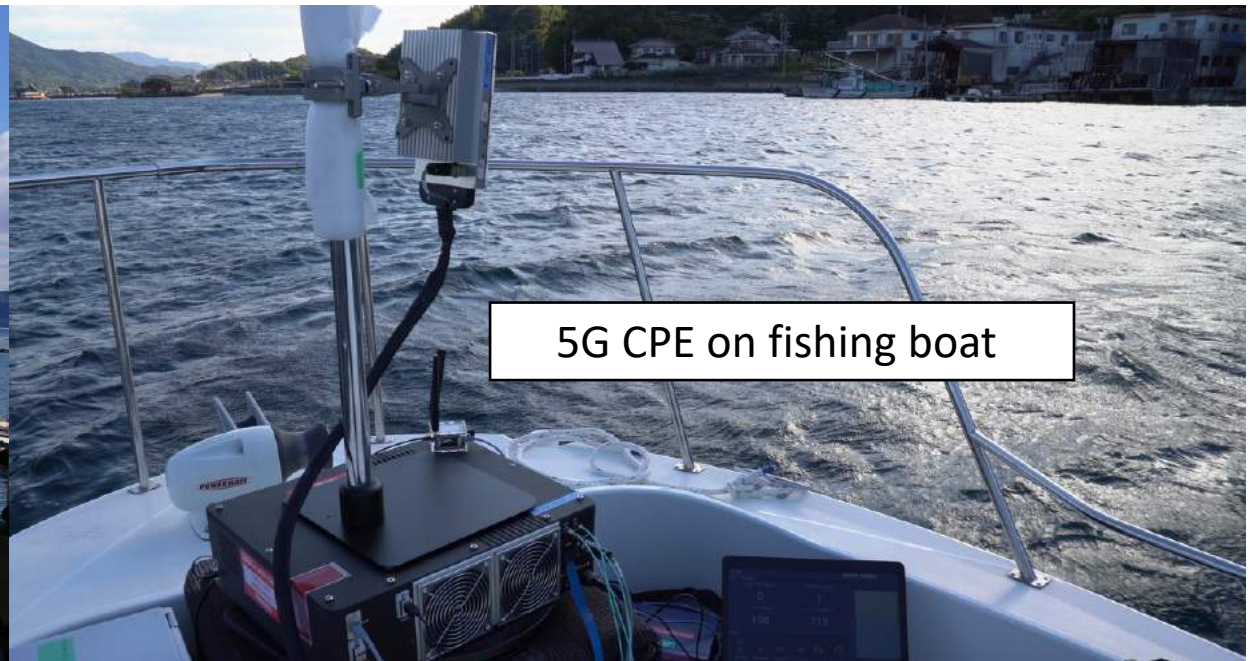
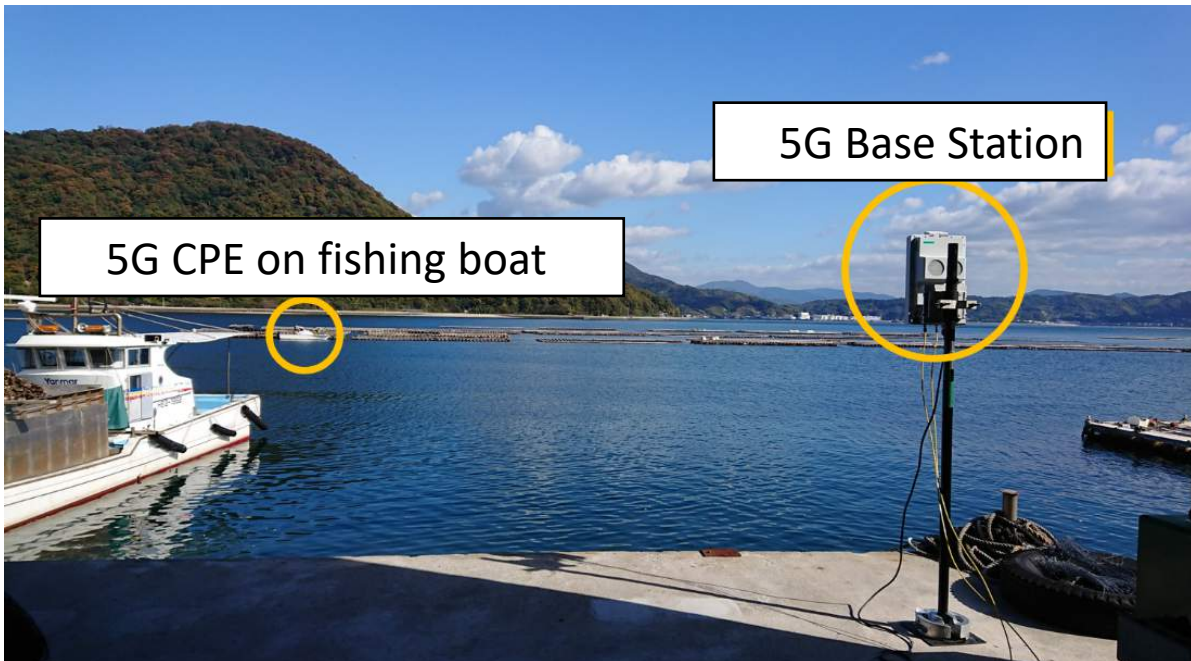
Press  
Release!



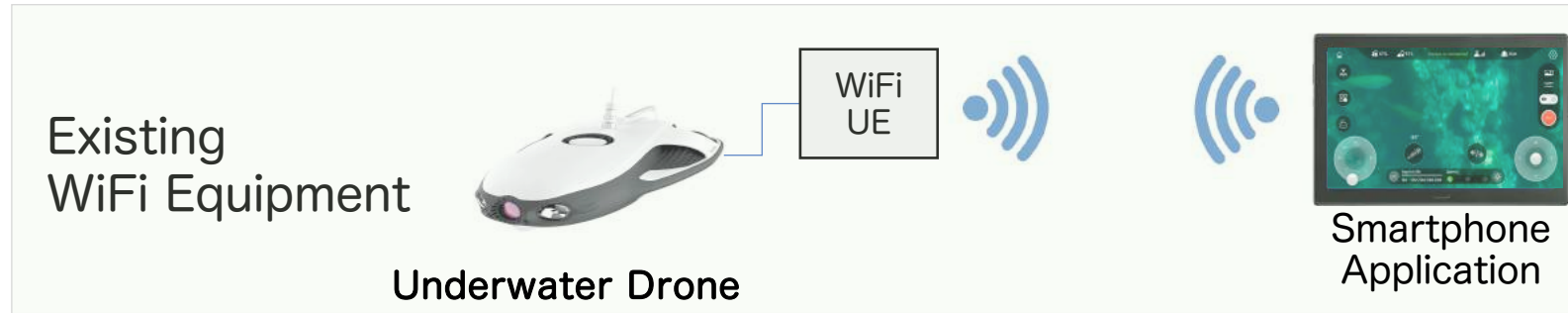
- Remote realtime monitoring Oyster Farming Rafts and Fishnets through water-drone
- **5G base station at the seashore and 5G CPE on the fishing boat**
- **URLLC for controlling under water drone**
- **eMBB for live video streaming**
- 28GHz millimeter wave band over 100-150m distance between the boat and the seashore

# 5G Live Videos Streaming and Realtime Control of Under-Water Drone

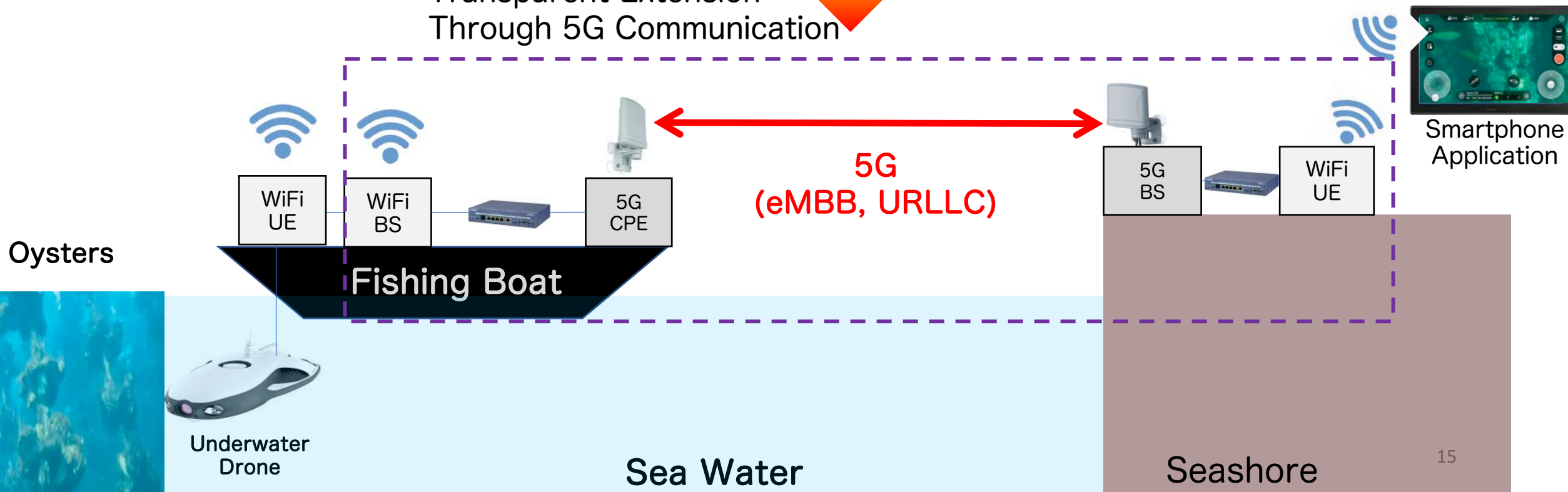




# 5G Transparent Extension of Control Range of WiFi equipment (underwater drone)



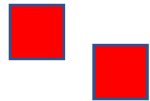
Transparent Extension Through 5G Communication



# Without Network Slicing on Per-App Basis

UE

eMBB  
(enhance Mobile Broadband)

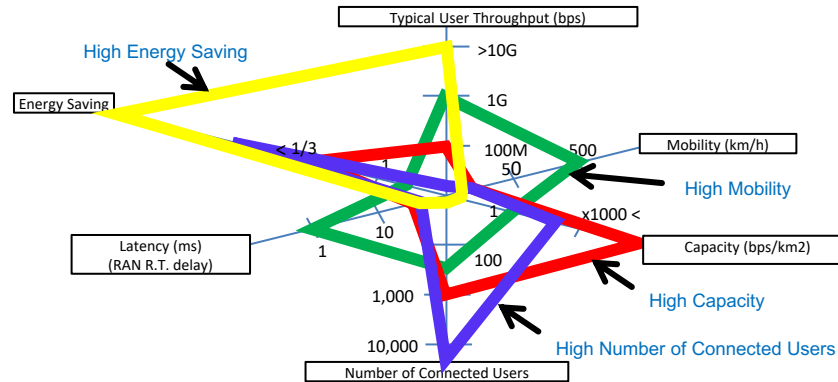
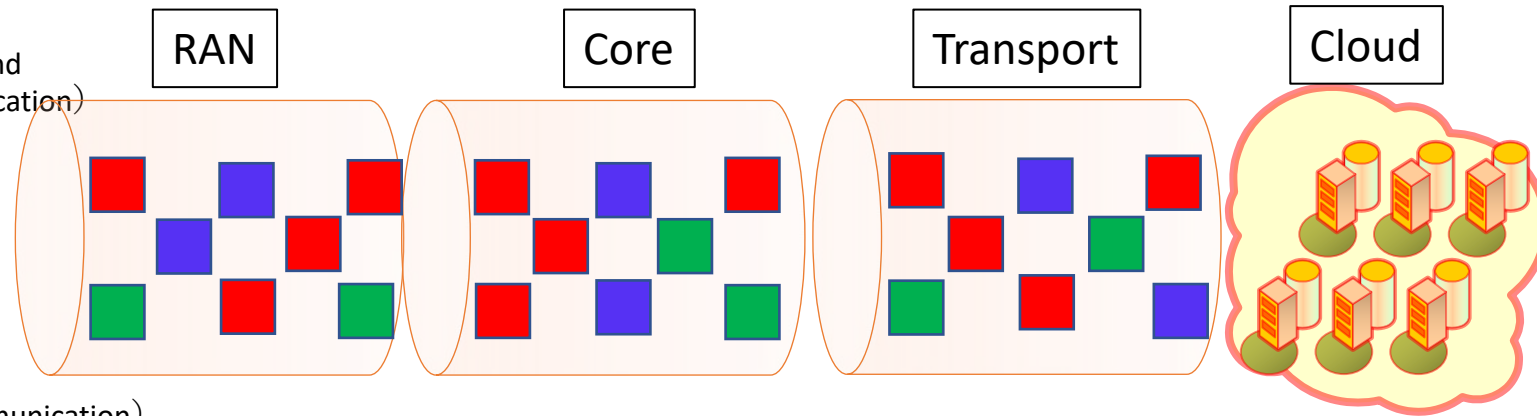


Applications with different QoS requirements share the same end-to-end communication channel

URLLC  
(Ultra Reliable and Low Latency Communication)

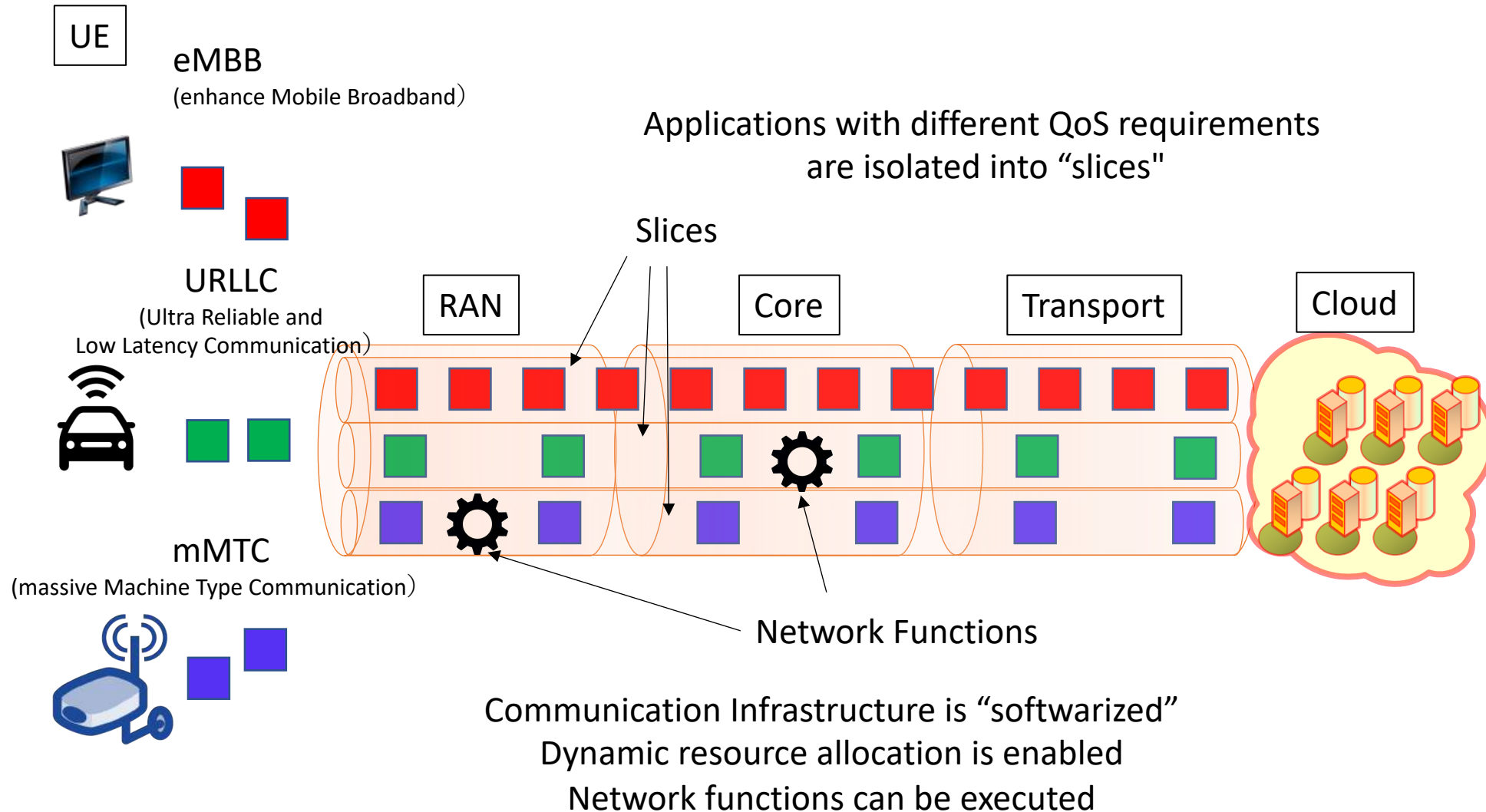


mMTC  
(massive Machine Type Communication)

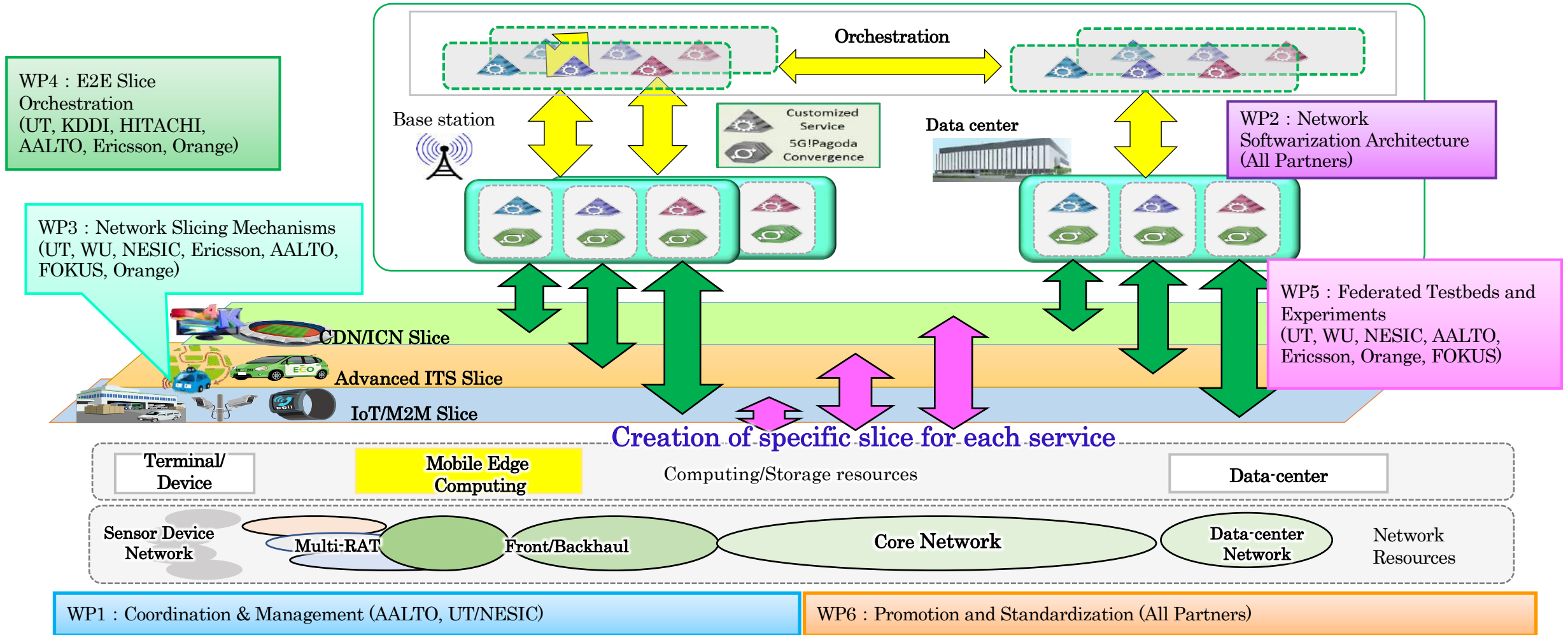




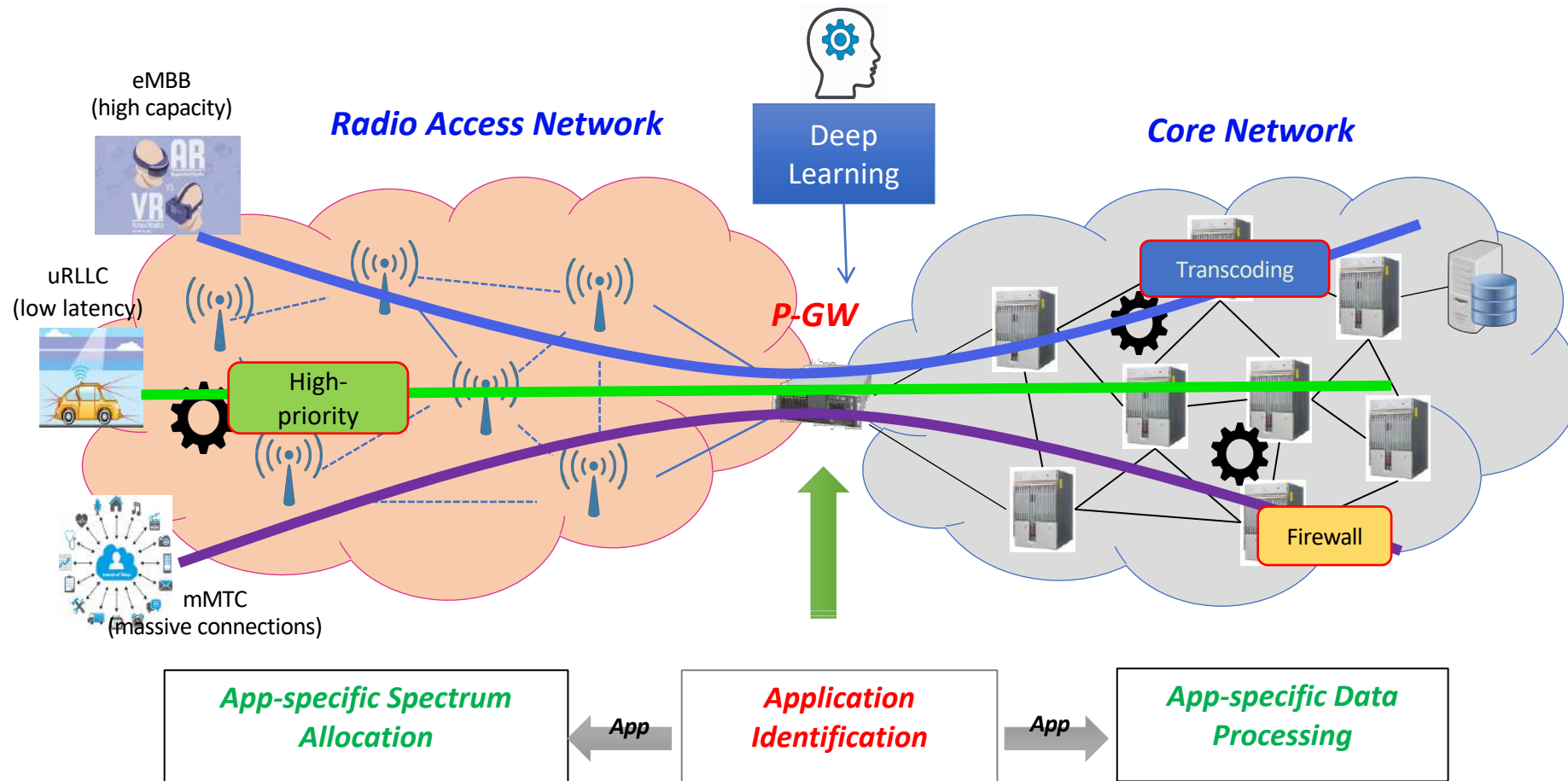
# With Network Slicing on Per-App Basis



## 5G! Pagoda's View on Future Mobile Infrastructure 5G/IoT Service Centric Network Slicing Control and Operations over Multi-Domains and Multi-Technologies

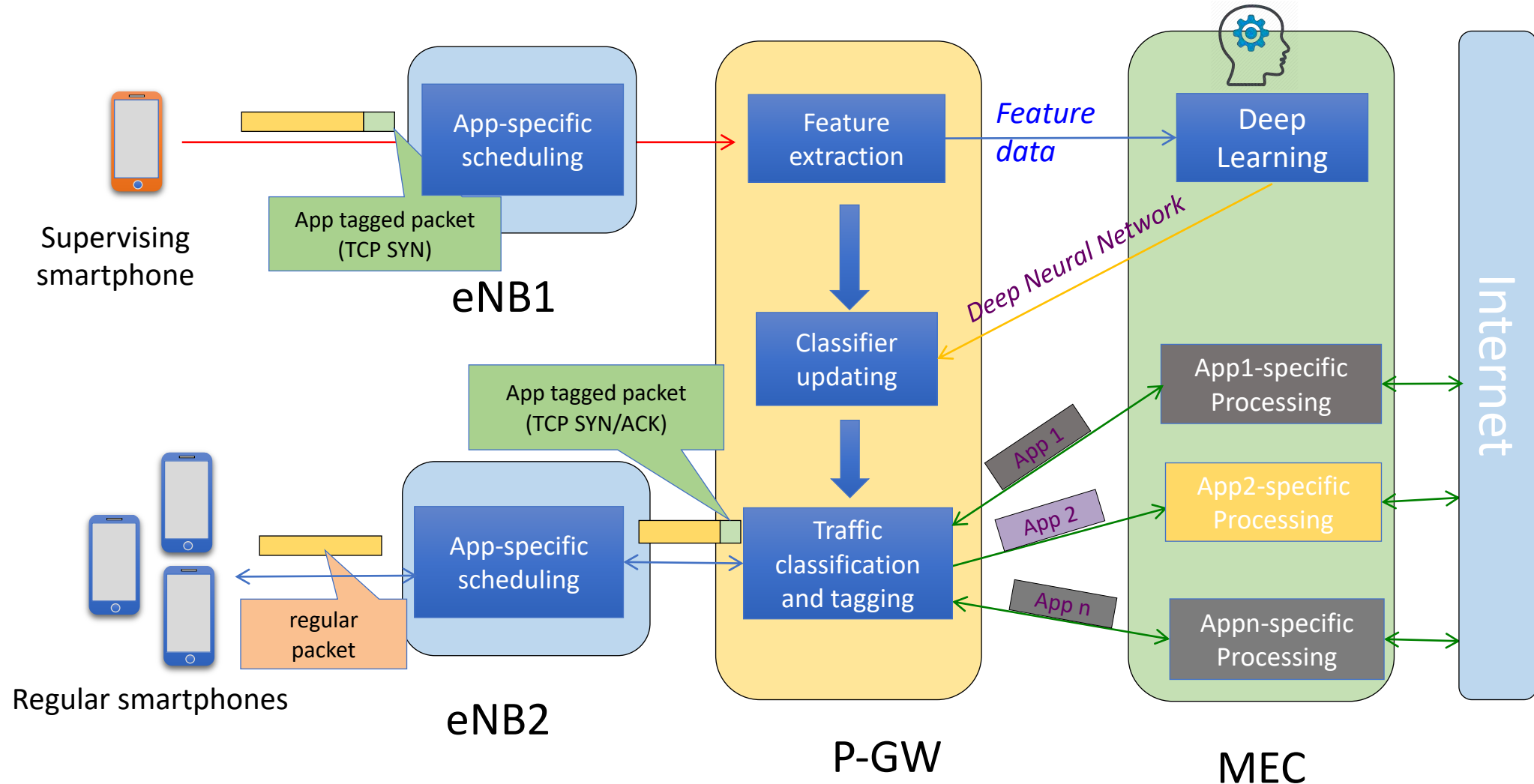


# Per-Application Network Slicing for Mobile Networks



- P-GW is the best point to perform application identification since all the traffic go through it
- P-GW can convey its identified app-info to both RAN and CN.

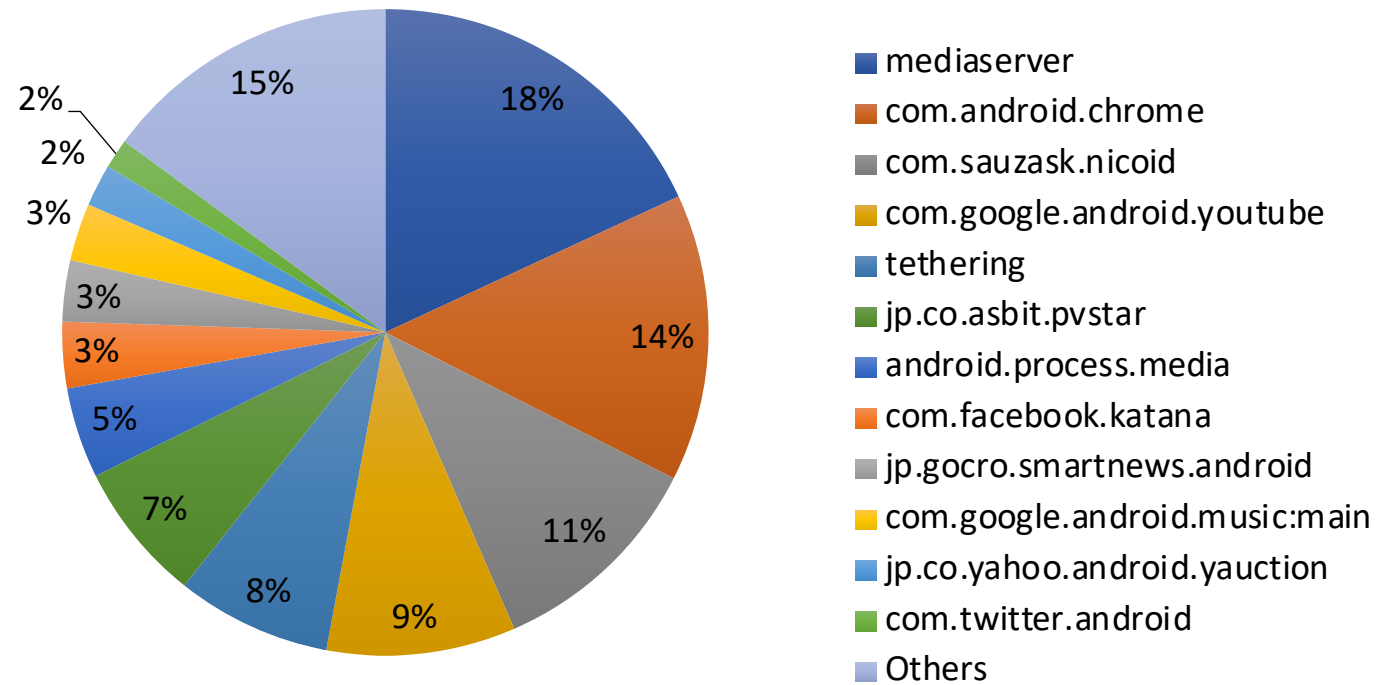
# (Deep) Machine Learning-based App-Slicing Architecture



## ***In-Network Deep Learning at P-GW:***

- UL: classifies traffic to different app-specific MEC for processing
- DL: tags acknowledgement packets (e.g., zero-payload SYN/ACK) from different application with app-info and sends to eNB for app-specific RB scheduling

# Breakdown of Real MVNO Traffic



- Observations

- Video Streaming (43%)

- mediaserver, com.sauzask.nicoid, com.google.android.youtube, android.process.media

- Web browsing (14%)

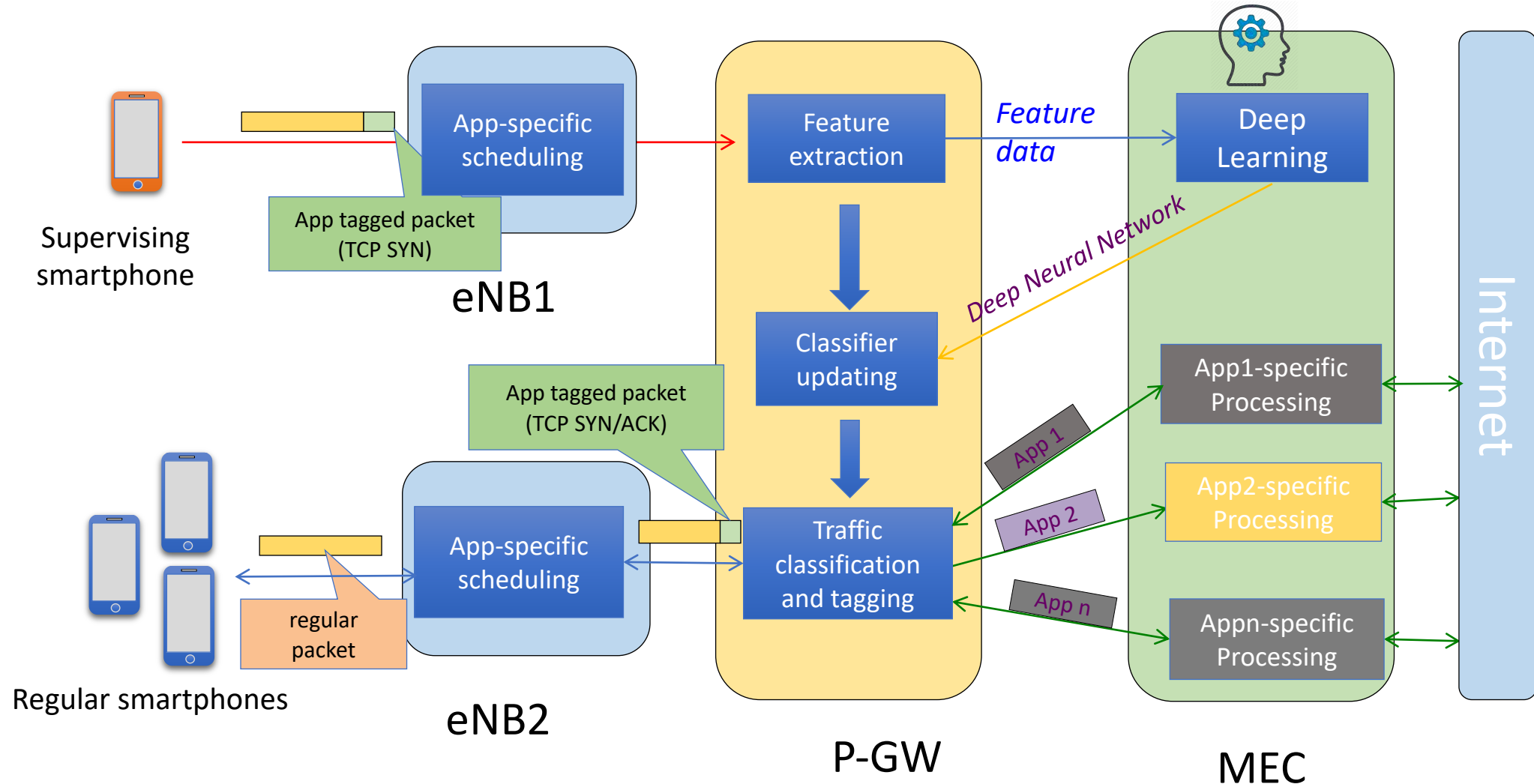
- com.android.chrome

- Tethering (7.8%)

- Social networks (4.8%)

- com.facebook.katana, com.twitter.android

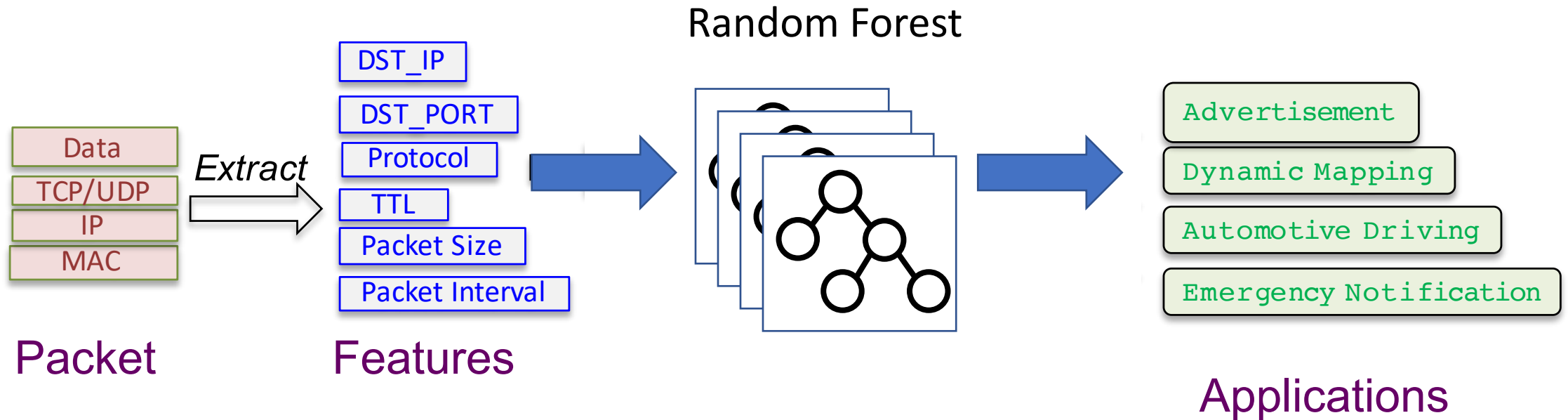
# (Deep) Machine Learning-based App-Slicing Architecture



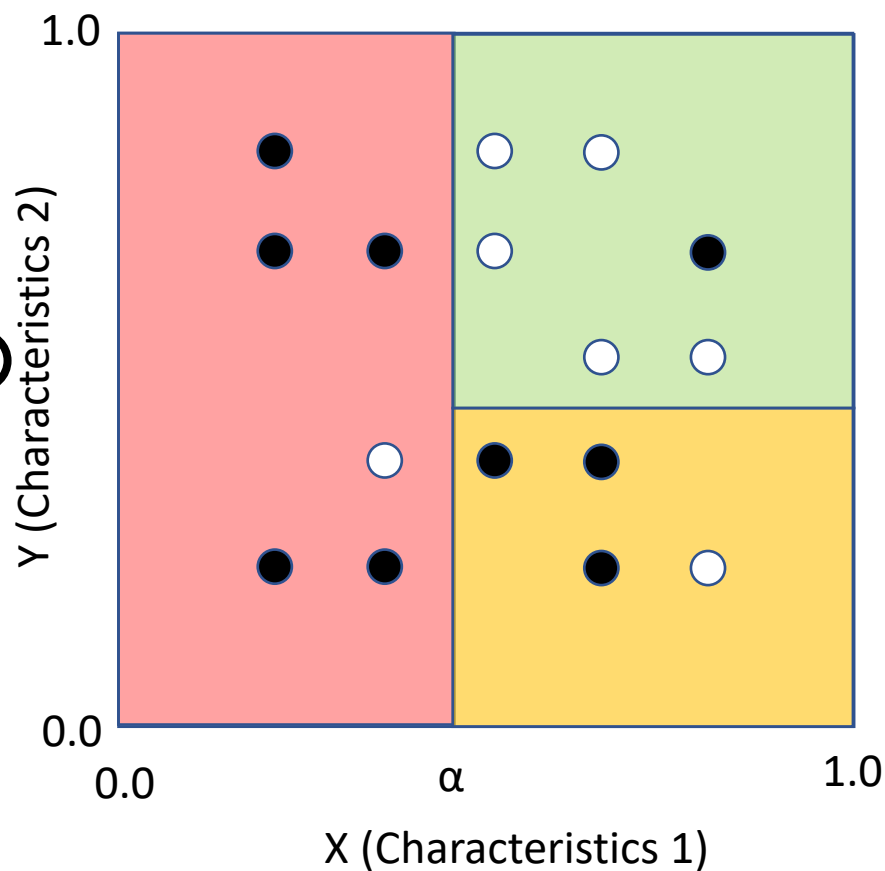
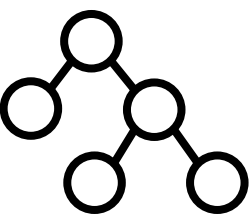
## ***In-Network Deep Learning at P-GW:***

- UL: classifies traffic to different app-specific MEC for processing
- DL: tags acknowledgement packets (e.g., zero-payload SYN/ACK) from different application with app-info and sends to eNB for app-specific RB scheduling

# Random Forest

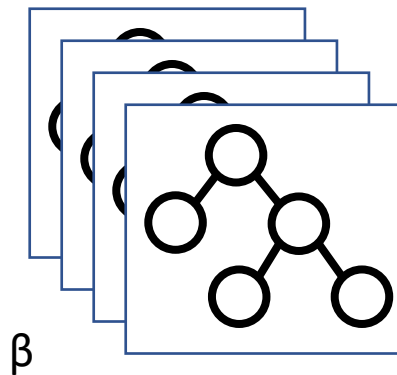


# Decision Tree

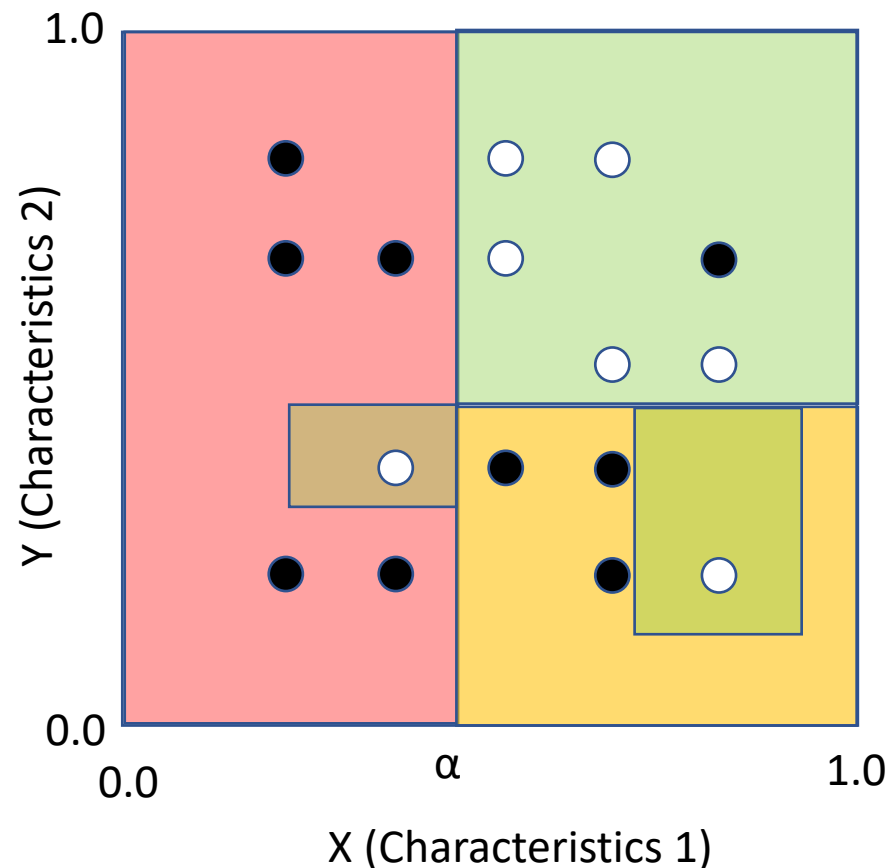


Classification Accuracy ~ 80%

# Ensemble



# Random Forest



Classification Accuracy ~ 94% <sup>24</sup>



# Ensemble Learning

Vote among Weak Students

10 Questions  
Correct Answers



Score (Accuracy)

Student1



60%

Student2



60%

Student3



70%

Vote Result

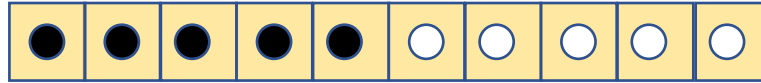


90%!!

# Ensemble Learning

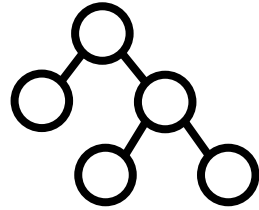
Vote among Weak Classifier (Deterministic Tree)

10 Questions  
Correct Answers



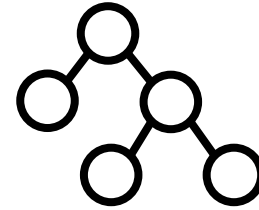
Accuracy

Decision Tree1



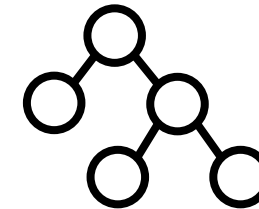
60%

Decision Tree2



60%

Decision Tree3



70%

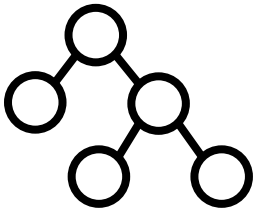
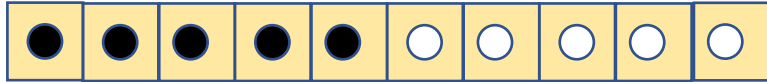
Ensemble Learning



90%!!

## Bad Choice of Weak Classifiers

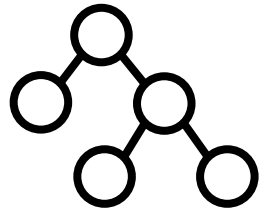
Correct Answer



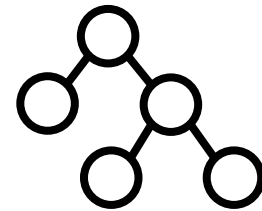
Score



60%



60%



70%

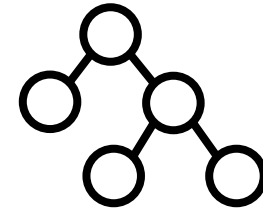
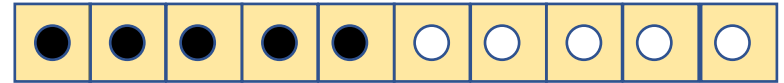
Vote Result



60%

## Good Choice of Weak Classifiers

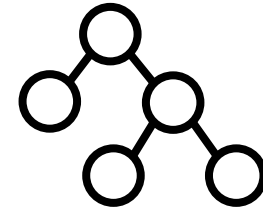
Correct Answer



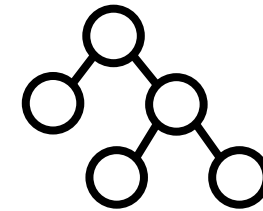
Score



60%



60%



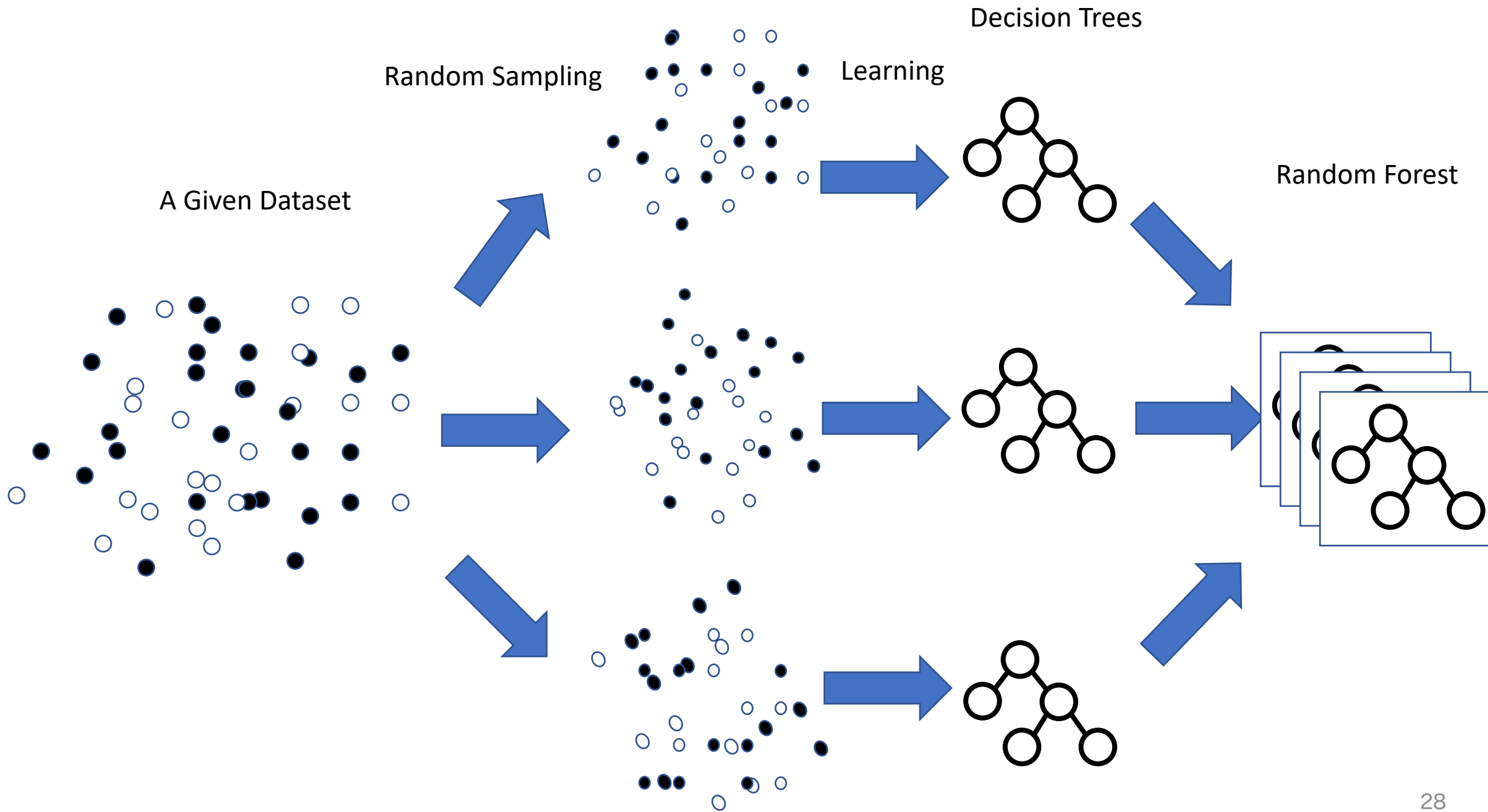
70%

Vote Result



90%!!

Uncorrelated Weak Classifiers Produces Better Classification

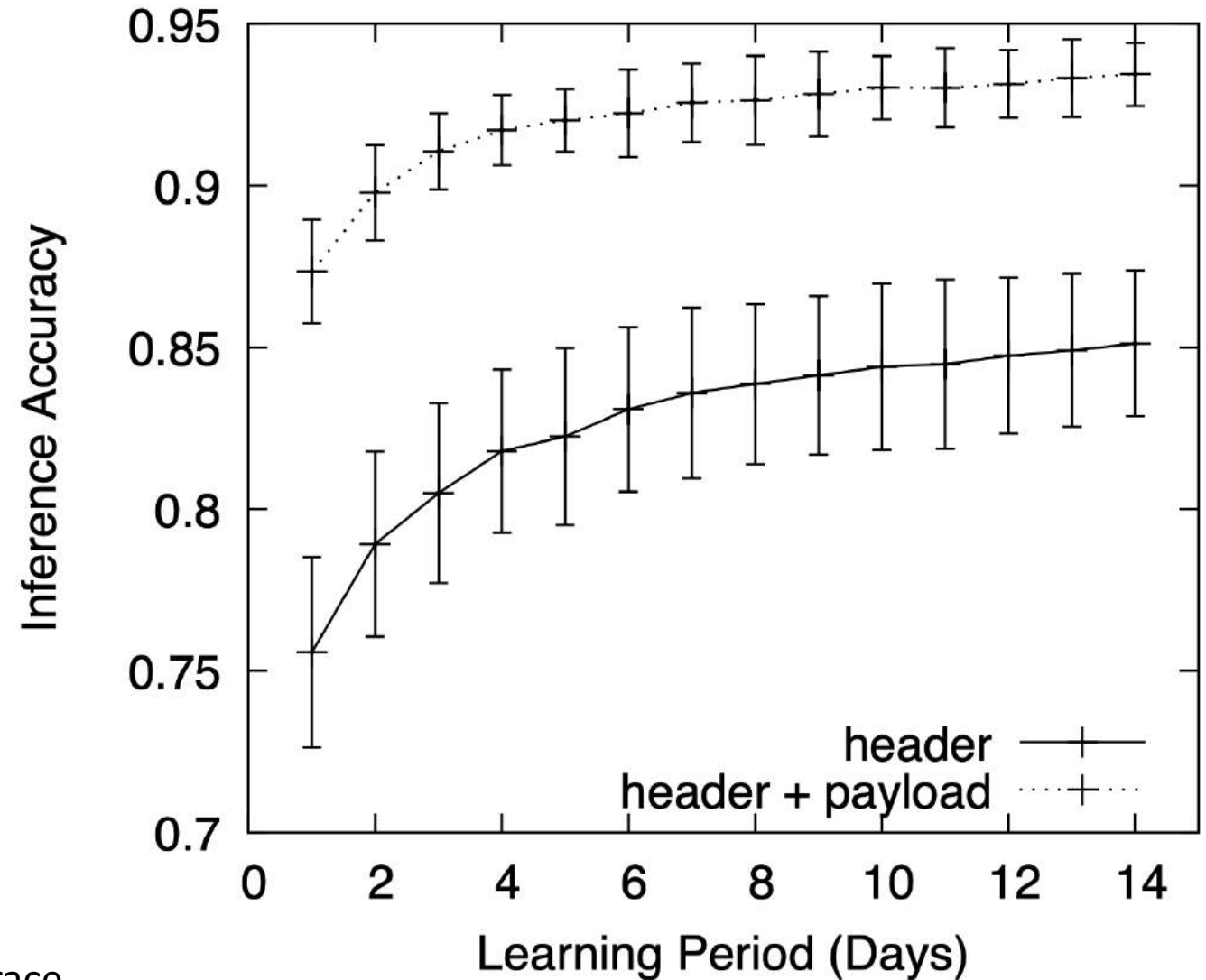


## 20 Features of Traffic Flows

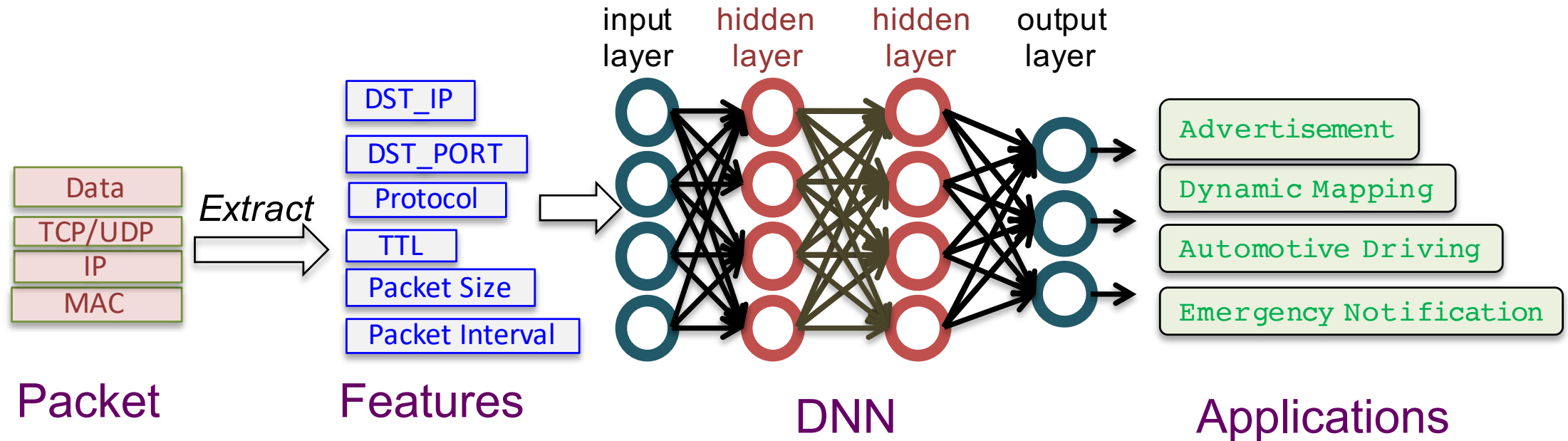
Destination IP Address
Destination Port Number
Source Port Number
average of the src→dst packet lengths
variance of the src→dst packet lengths
maximum of the src→dst packet lengths
average of the src←dst packet lengths
variance of the src←dst packet lengths
maximum of the src←dst packet lengths
average of the src→dst packet arrival
variance of the src→dst packet arrival
average of the src←dst packet arrival
variance of the src←dst packet arrival
source window-size in SYN packet
destination window-size in SYN packet
ratio of the ACK-PSH flags in the all TCP
dst→src packet count/ src←dst packet

Yellow cells: feature designed based on the trace.  
white cells: designed based on existing research.

## Application Classification Accuracy

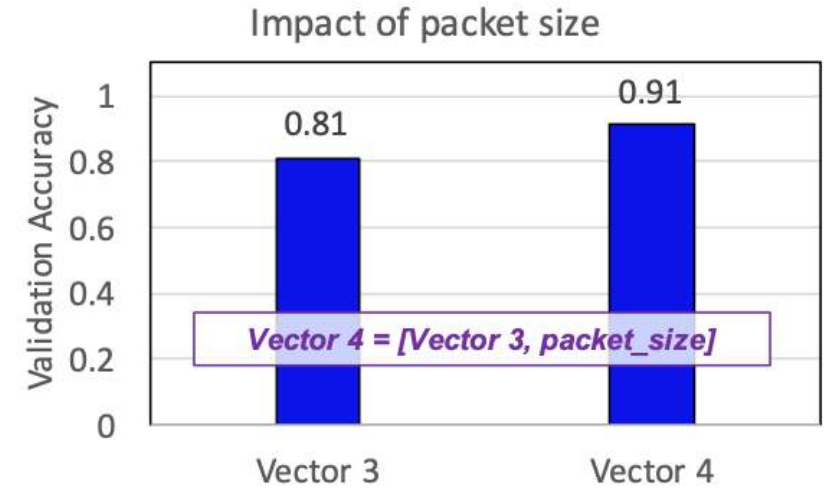
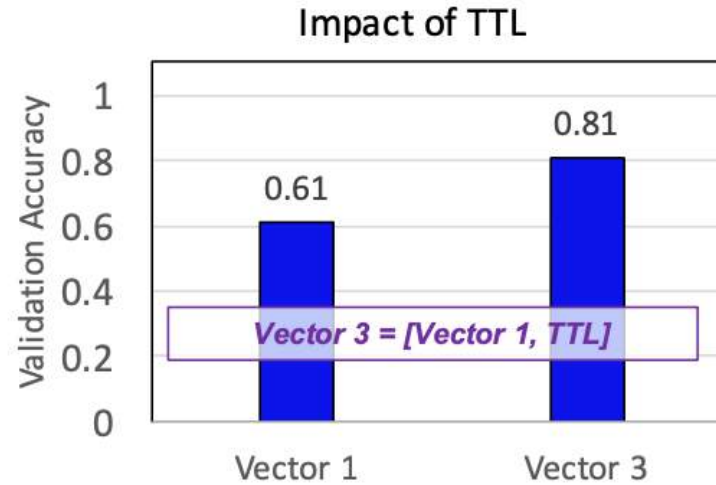
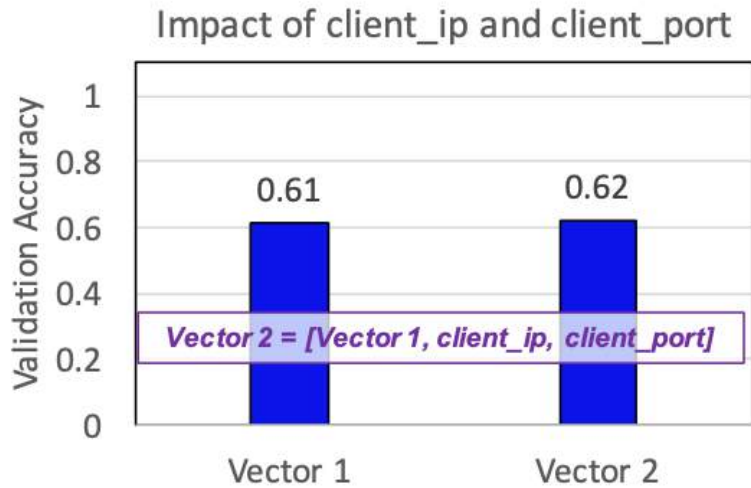


# Deep Learning



- Step 1: The system extracts **feature vectors** from packets and feed the vectorized features into the input layer of DNN (deep neural network)
- Step 2: Training model DNN is defined with an input layer, multiple fully connected hidden layers, and an output layer. Each hidden layer is a feed-forward neural network.
- Step 3: Output layer is built with **softmax regression mode** and its output is a probability vector over applications. The packet is identified as the application with highest probability.

# Selection of Feature Vectors



Vector 1 = <server<sup>(a)</sup>\_ip, server\_port, proto>

(b)

(c)

- (a) Feature <client\_ip, client\_port> has almost no impact on the identification accuracy.
- (b) Feature **TTL** has a great impact on the identification accuracy because TTL is a metric of distance from the application server to the P-GW. The distance is application-specific.
- (c) Feature **packet\_size** is also a useful feature because client and server need to exchange information during connection establishment. The size of exchange information is application-specific.

# Summary:

1. For 5G and Beyond 5G, fine-grained network slicing becomes important as network requirement varies significantly per application. Advanced traffic classification using machine learning (ML) without privacy violation is a viable use case to demonstrate the power of ML.
2. In-Network Machine Learning is powerful means to derive useful high-level information especially in 5G and beyond 5G
  - Traffic User Data
  - Network Operational Data
  - Human behavior Data (Usage of UE, Applications, etc)
3. Tangible example use cases such as traffic classification are simple but suitable and attractive for education purpose (lower barrier to entry to ML/AI application to networking)

Such examples would accelerate research and education on the subject.



# References

- Akihiro Nakao and Ping Du, “Toward In-Network Deep Machine Learning for Identifying Mobile Applications and Enabling Application Specific Network Slicing”, IEICE Transactions E101-B, No.7, pp.1536-1543, 2018.
- Ping Du, Akihiro Nakao, Zhaoxia Sun, Lei Zhong and Ryokichi Onishi, “Deep Learning-based C/U Plane Separation Architecture for Automotive Edge Computing”, The Fourth ACM/IEEE Symposium on Edge Computing (SEC), 2019.
- Ping Du and Akihiro Nakao, "Deep Learning-based Application Specific RAN Slicing for Mobile Networks", IEEE International Conference on Cloud Networking (CloudNet), 2018.
- Takamitsu Iwai, Akihiro Nakao (University of Tokyo), Adaptive Mobile Application Identification Through In-Network Machine Learning, APNOMS 2016

ITUEvents

Machine Learning for Wireless LANs +  
Japan Challenge Introduction  
ITU-ML5G-PS-031, ITU-ML5G-PS-032  
29 July 2020

ITU  
**AI/ML in 5G**  
Challenge

*Applying machine learning in  
communication networks*

ai5gchallenge@itu.int

Register [here](#)  
Join us on [Slack](#)

Sponsors



ZTE

Organizer



ITUEvents

Machine Learning for Wireless LANs +  
Japan Challenge Introduction  
ITU-ML5G-PS-031, ITU-ML5G-PS-032  
29 July 2020

ITU  
**AI/ML in 5G**  
Challenge

*Applying machine learning in  
communication networks*

ai5gchallenge@itu.int

Register [here](#)  
Join us on [Slack](#)

Sponsors



Organizer



# Machine Learning for Wireless LANs

Koji Yamamoto

Graduate School of Informatics, Kyoto University

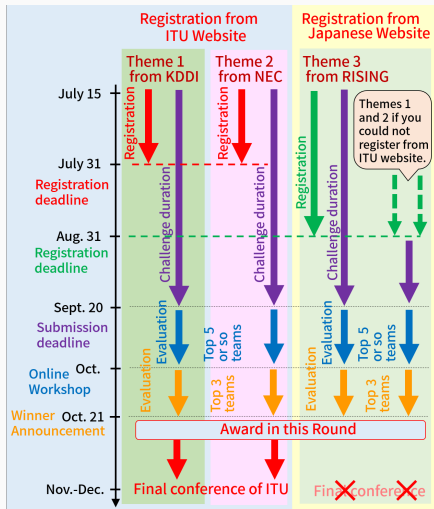
2020-07-29



- ▶ Location estimation from Wi-Fi RSSI (Received Signal Strength Indicator)
- ▶ Japan round only  
Not eligible for final conference

## This lecture talk

- ▶ Our applications of deep supervised learning and reinforcement learning
- ▶ For microwave and mmWave WLANs
- ▶ Deep (supervised) learning in Part I
- ▶ Deep reinforcement learning in Part II

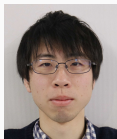


## Deep Learning for mmWave WLANs

[Nishio+2019] T. Nishio, H. Okamoto, K. Nakashima, Y. Koda, K. Yamamoto, M. Morikura, Y. Asai, and R. Miyatake, "Proactive received power prediction using machine learning and depth images for mmWave networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, Nov. 2019



T. Nishio



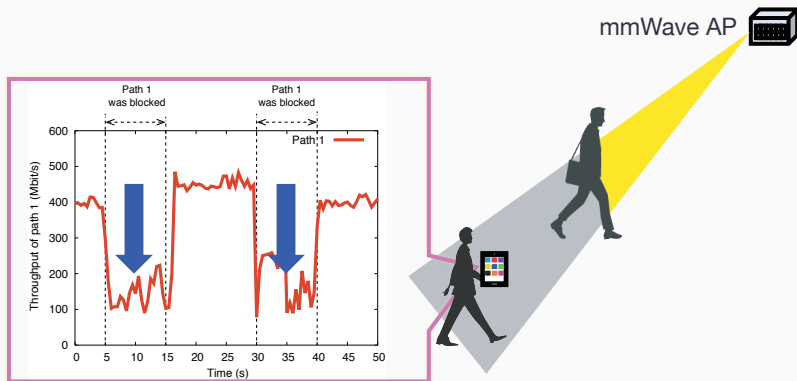
Y. Koda

# Human body blocking in mmWave communications

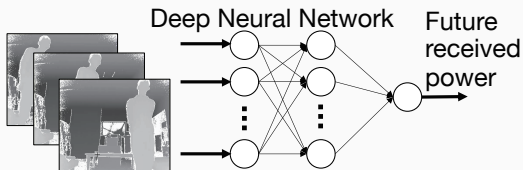
5G — 28 GHz band

IEEE 802.11ad/ay — 60 GHz band

- ▶ Beyond Gbit/s communications using bandwidth of 2.16 GHz or more (IEEE 802.11ad)
- ▶ Strong attenuation (15 dB–) when human blocks line-of-sight

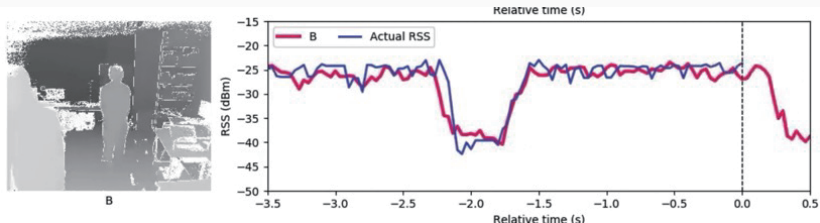


Key idea: Deep learning and camera images



Prediction:

Received power in 500 ms ahead is accurately predicted only from camera images





(1) Received power prediction  
(Regression)

(2) Linear regression

Input  $x$

Camera images



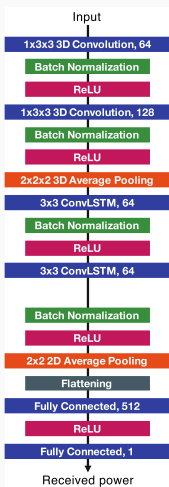
$$x \in \mathbb{R}^1$$

Output  $y$

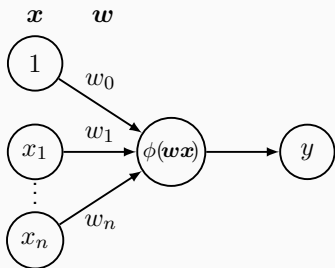
Future received power

$$y \in \mathbb{R}^1$$

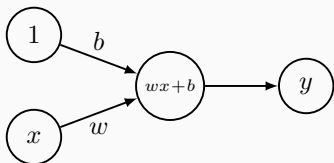
NNs



Simple perceptron (NN) with  
linear activation function



$$y = \phi \left( \sum_{i=1}^n w_i x_i + w_0 \right) = \phi(\mathbf{w}\mathbf{x})$$



Linear activation function

$$\phi(x) = x$$

$$y = wx + b$$

```
class Net(torch.nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = torch.nn.Linear(1, 1)
    def forward(self, x):
        x = self.fc1(x)
        return x
```

(1) Received power prediction

$$\text{Future power} = f^{(1)}\left(\begin{array}{c} \text{Image 1} \\ \text{Image 2} \\ \text{Image 3} \\ \text{Image 4} \end{array}\right)$$

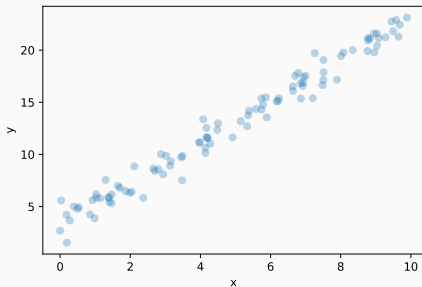
Training  $f^{(1)} = \text{NN}$  by labeled data set

(2) Linear regression

$$y = f^{(2)}(x) = wx + b$$

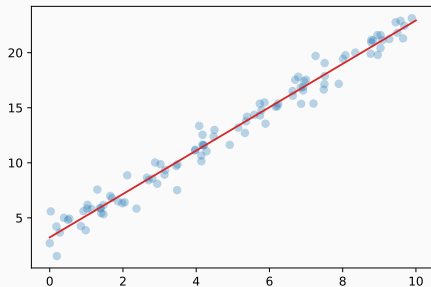
Training  $f^{(2)}$ , i.e.,  $w$  and  $b$  by labeled data set  $(x_i, y_i)$

$$y_i = 2x_i + 3 + \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$



 Open in Colab

```
optimizer = torch.optim.SGD(net.parameters(), lr=0.01)
criterion = torch.nn.MSELoss()
(Training)
print(net.fc1.weight) # tensor([[2.0332]], requires_grad=True)
print(net.fc1.bias)   # tensor([2.8885], requires_grad=True)
```



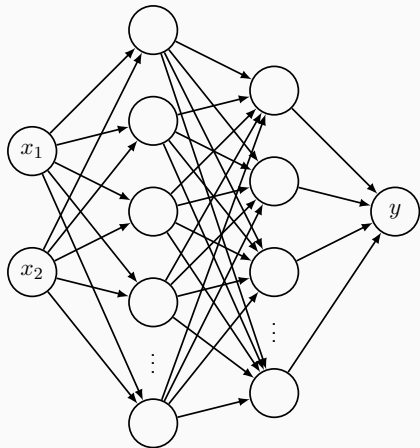
---

$y = 2x + 3 + \epsilon$ : Labeled data  $(x_i, y_i)$

$y = wx + b$ : Neural network

Minimum mean-squared error estimation in linear regression





## Multiple inputs $\rightarrow$ Deep networks



```
class Net(torch.nn.Module):  
    def __init__(self):  
        super(Net, self).__init__()  
        self.fc1 = torch.nn.Linear(2, 64)  
        self.fc2 = torch.nn.Linear(64, 32)  
        self.fc3 = torch.nn.Linear(32, 1)  
    def forward(self, x):  
        x=torch.nn.functional.relu(self.fc1(x))  
        x=torch.nn.functional.relu(self.fc2(x))  
        x=self.fc3(x)  
        return x
```

$$f(x) = \text{ReLU}(x) := \max\{0, x\}$$

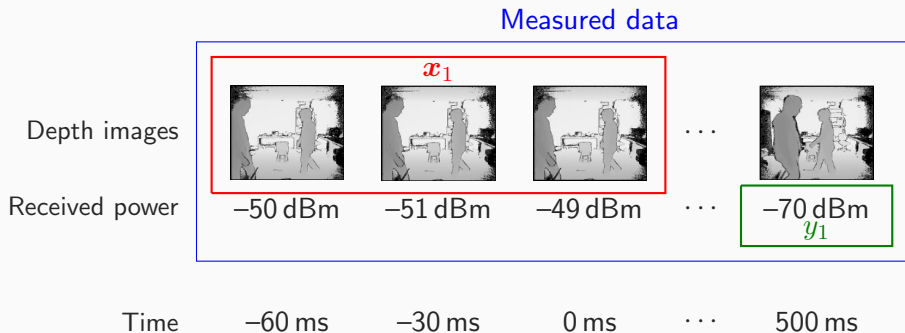
# Measured dataset for future received power prediction

Depth images				...	
Received power	-50 dBm	-51 dBm	-49 dBm	...	-70 dBm
Time	-60 ms	-30 ms	0 ms	...	500 ms

By using this dataset, how can we train the NN  $f^{(1)}$ , i.e.,  $f^{(1)}$ ?



$$\text{Future received power} = f^{(1)}\left(\left(\begin{array}{c} \text{Depth image 1} \\ \text{Depth image 2} \\ \text{Depth image 3} \\ \dots \\ \text{Depth image 5} \end{array}\right)\right)$$



$$y = f^{(1)}(x)$$



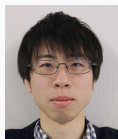
# Deep Reinforcement Learning for WLANs

[Nakashima+2020] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep reinforcement learning-based channel allocation for wireless LANs with graph convolutional networks," *IEEE Access*, vol. 8, Feb. 2020



S. Kamiya

[Koda+2020] Y. Koda, K. Nakashima, K. Yamamoto, T. Nishio, and M. Morikura, "Handover management for mmWave networks with proactive performance prediction using camera images and deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, Feb. 2020

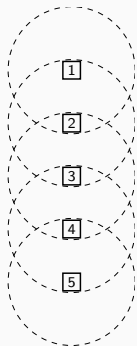


Y. Koda

Optimal channel allocation?

Criterion: Aggregated throughput

Number of channels: 2

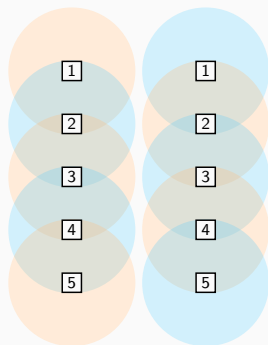
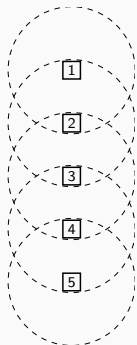


Framework: Combinatorial optimization (NP-hard)

Optimal channel allocation?

Criterion: Aggregated throughput

Number of channels: 2

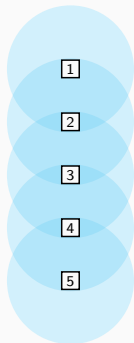


Framework: Combinatorial optimization (NP-hard)


Different problem setting:

- ▶ Finding the minimal *sequence*
- ▶ Only one AP can change its channel at a given time
- ▶ Throughput can be observed only after channel allocation

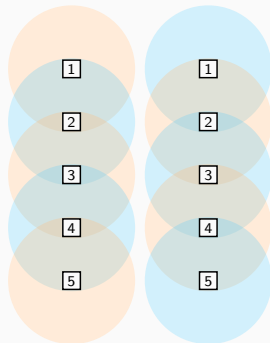
Initial state



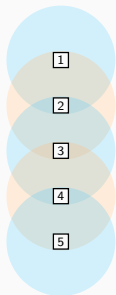
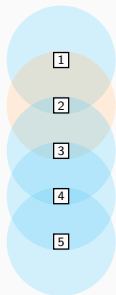
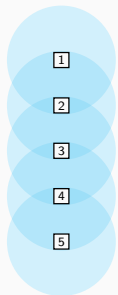
We would like to  
find the minimum  
sequence.



Optimal states

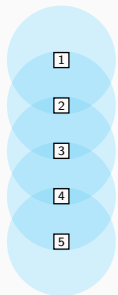


# Channel Allocation Sequence

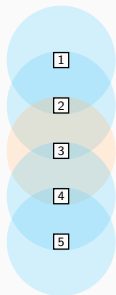


←

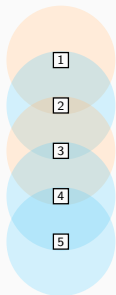
We want to acquire this sequence based on observed throughput  
Criterion?



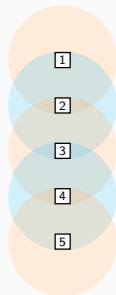
$t = 0$



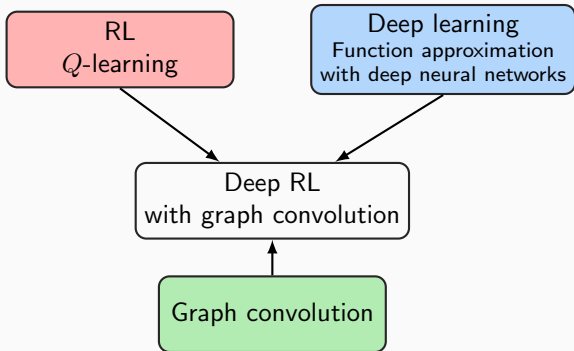
$t = 1$

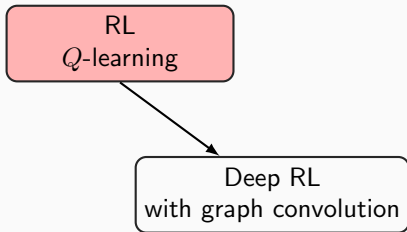


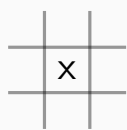
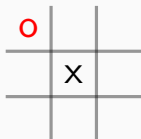
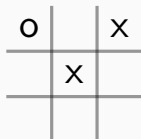
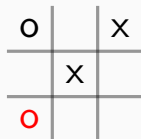
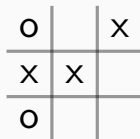
$t = 2$



$t = 3$





State  $S_0$ Action  $A_0$ State  $S_1$ Action  $A_1$ State  $S_2$ 

The second player observes state  $S_t$  and takes action  $A_t$ ,  
then observes the next state  $S_{t+1}$ ,

and gets the reward  $R_{t+1}$  — win or loss.

### Problem:

Without knowing the rule of the game, i.e.,  
only based on the observed sequence  $(S_t, A_t, R_{t+1}, S_{t+1})$ ,  
we determine the appropriate action.

### Approach:

Reinforcement learning — (Tabular) Q-learning



Action-value function:

$Q : \text{State} \times \text{Action} \mapsto \text{Expected reward}$

State	Action								
	A	B	C	D	E	F	G	H	I
⋮									
o   x									
—   x									
								o	
o     x									
—   x									
								o	
o									
x   x									
o									
x   x									
⋮									

A	B	C
D	E	F
G	H	I

Updating the value of Q table according to observed sequence  
 $(S_t, A_t, R_{t+1}, S_{t+1})$ .

 Open in Colab

### Number of actions:

$$3 \times 3 = 9 \text{ cells}$$

### Number of states:

$$3^9 = 19683$$

blank, o, or x for each cell

### Number of elements in Q table:

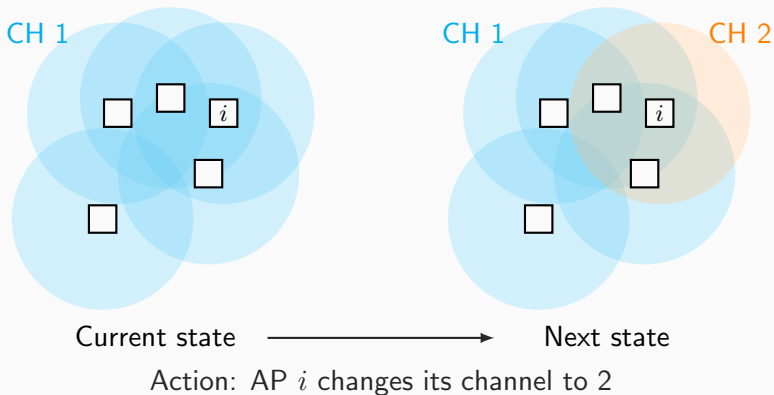
$9 \times 19683$  even in this simple problem

To estimate  $Q^*(s, a)$ , every state-action pair should be visited

When the number of states is huge, training is infeasible — State explosion

## Markov decision process

- ▶ Agent: Centralized controller of all APs
- ▶ State: Channels and contention graph of APs
- ▶ Action: Channel selection of one AP
- ▶ Reward: Throughput



	Case 1	Case 2
Number of APs	4	10
Number of links between APs	$\binom{4}{2} = 6$	$\binom{10}{2} = 45$
Number of link states	$2^6$	$2^{45}$
Number of channels	2	3
Number of channel states	$2^4$	$3^{10}$
Number of states	$2^6 \cdot 2^4 = 1024$	$2^{45} \cdot 3^{10} = 2 \cdot 10^{18}$

## Tabular Q-learning

$$Q_{t+1}(S_t, A_t) \doteq Q_t(S_t, A_t) + \alpha_t \delta_{t+1}(Q_t)$$
$$\delta_{t+1}(Q_t) \doteq R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_t(S_{t+1}, a') - Q_t(S_t, A_t)$$

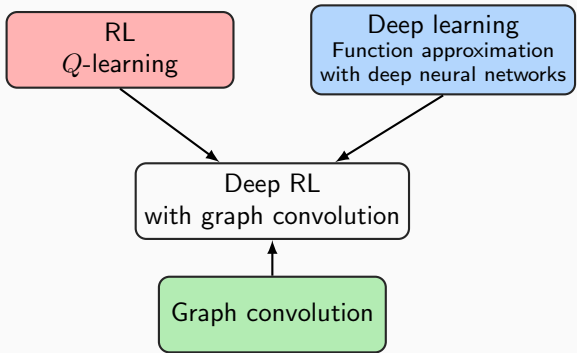
## Q-learning with function approximation

Extension to function approximation with a parameterized function  $Q_{\theta}$ ,  
 $\theta \in \mathbb{R}^d$

$$\theta_{t+1} \doteq \theta_t + \alpha_t \delta_{t+1}(Q_{\theta_t}) \nabla_{\theta} Q_{\theta_t}(S_t, A_t)$$
$$\delta_{t+1}(Q_{\theta_t}) \doteq R_{t+1} + \gamma \max_{a' \in \mathcal{A}} Q_{\theta_t}(S_{t+1}, a') - Q_{\theta_t}(S_t, A_t)$$

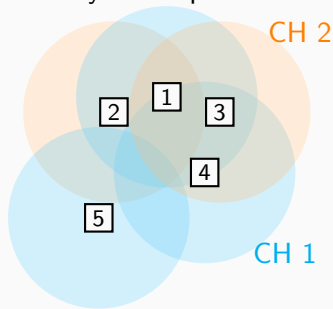
## Deep reinforcement learning

Q-learning with function approximation using deep neural networks

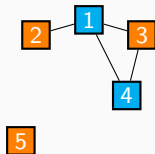


**State:** Adjacency of APs and selected channel

Physical expression



Graph expression



Tabular expression

5x7 matrix

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \begin{array}{l} \text{Ch 1} \\ \text{Ch 2} \end{array}$$

**Feature extraction (pre-processing):**

---

Graph convolutional networks

Convolutional neural network (CNN)

---

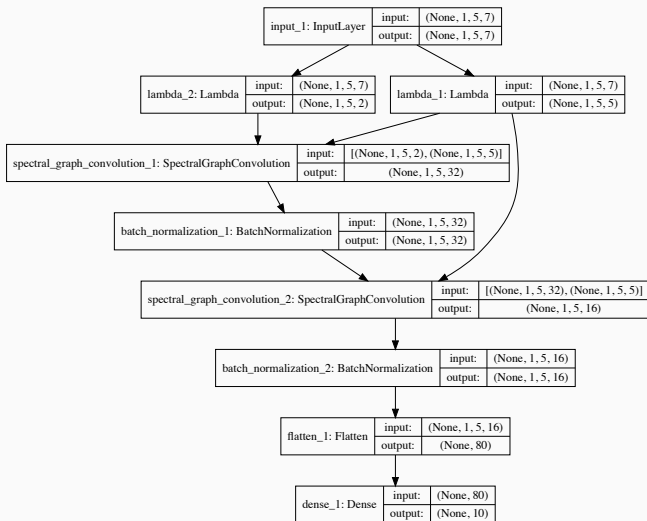
Feature extraction for graphs

Feature extraction for images

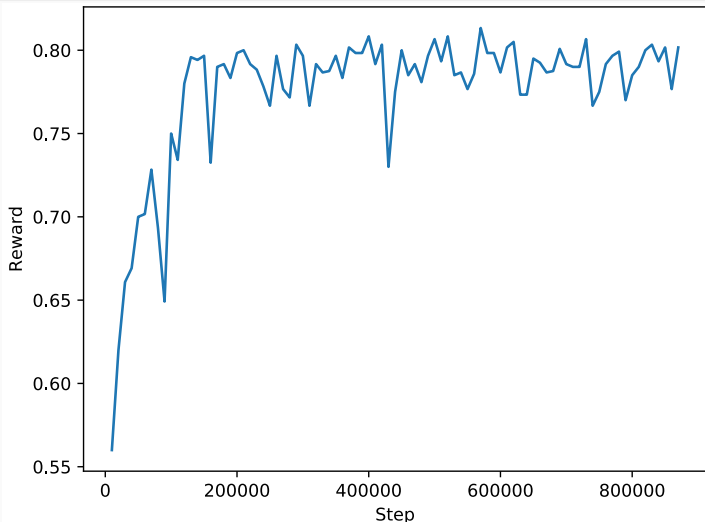
---



## State: 5x7 matrix



$Q$ -values for 10 actions (5 APs  $\times$  2 CHs)



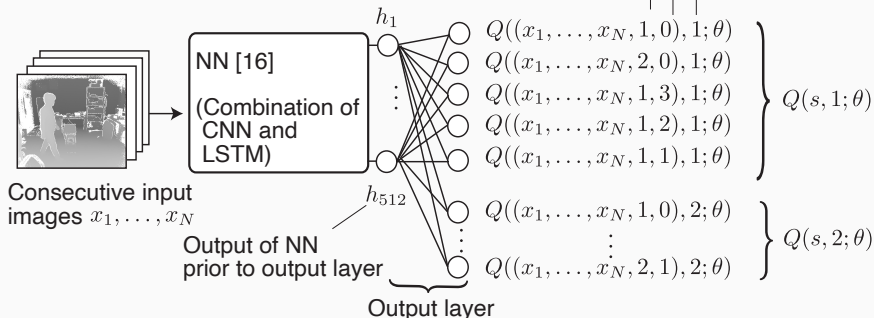
- ▶ Five APs are located uniformly and randomly
- ▶ Reward: The minimum individual throughput of five APs

- ▶ By using input images, determine one BS from two candidate BSs
- ▶ The output of NNs is Q-value for input images

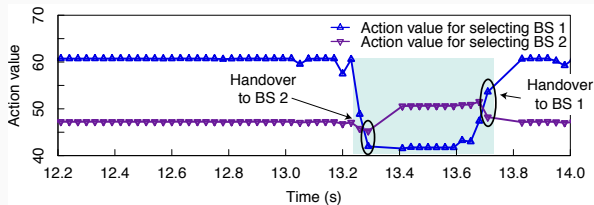
$c$  : Remaining time step until handover process is completed

$j$  : Index of associated BS

$a$  : Action

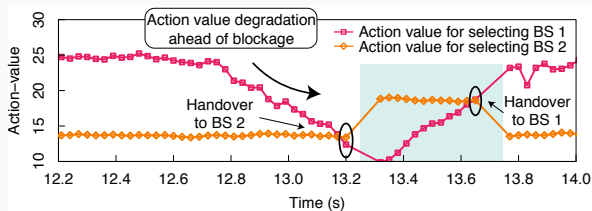


## Q-value selecting BSs 1 and 2



Without camera images

- ▶ Because the received power degrades sharply, handover decision is conducted after degradation in general



With camera images

- ▶ Usage of camera images extends the state space, and thus proactive handover is enabled

## Part I

- ▶ Deep NNs are functions.
- ▶ Deep learning successfully predict future received power only from past image sequences.

## Part II

- ▶ Reinforcement learning is used to acquire the *optimal sequence* to maximize the total reward.
- ▶ Deep neural networks are used for function approximation of  $Q$  function (deep RL).

- [Koda+2020] Y. Koda, K. Nakashima, K. Yamamoto, T. Nishio, and M. Morikura, "Handover management for mmWave networks with proactive performance prediction using camera images and deep reinforcement learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, Feb. 2020.
- [Liew+2010] S. C. Liew, C. H. Kai, H. C. Leung, and P. Wong, "Back-of-the-envelope computation of throughput distributions in CSMA wireless networks," *IEEE Trans. Mobile Comput.*, vol. 9, no. 9, Sep. 2010.
- [Nakashima+2020] K. Nakashima, S. Kamiya, K. Ohtsu, K. Yamamoto, T. Nishio, and M. Morikura, "Deep reinforcement learning-based channel allocation for wireless LANs with graph convolutional networks," *IEEE Access*, vol. 8, Feb. 2020.
- [Nishio+2019] T. Nishio, H. Okamoto, K. Nakashima, Y. Koda, K. Yamamoto, M. Morikura, Y. Asai, and R. Miyatake, "Proactive received power prediction using machine learning and depth images for mmWave networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 11, Nov. 2019.

ITUEvents

Demonstration of machine learning function  
orchestrator (MLFO) via reference implementations  
(ITU-ML5G-PS-024)

Shagufta Henna, LYIT, 31 July 2020

ITU  
**AI/ML in 5G**  
Challenge

*Applying machine learning in  
communication networks*

ai5gchallenge@itu.int

Register [here](#)  
Join us on [Slack](#)

Sponsors



Organizer



# **Analysis on route information failure in IP core networks by NFV-based test environment.**

ITU-ML5G-PS-(KDDI)



- 1. Data set**
- 2. How to use the data set**
- 3. Network configuration for evaluation**
- 4. Use cases of analysis**
- 5. Submission**

# 1. Dataset

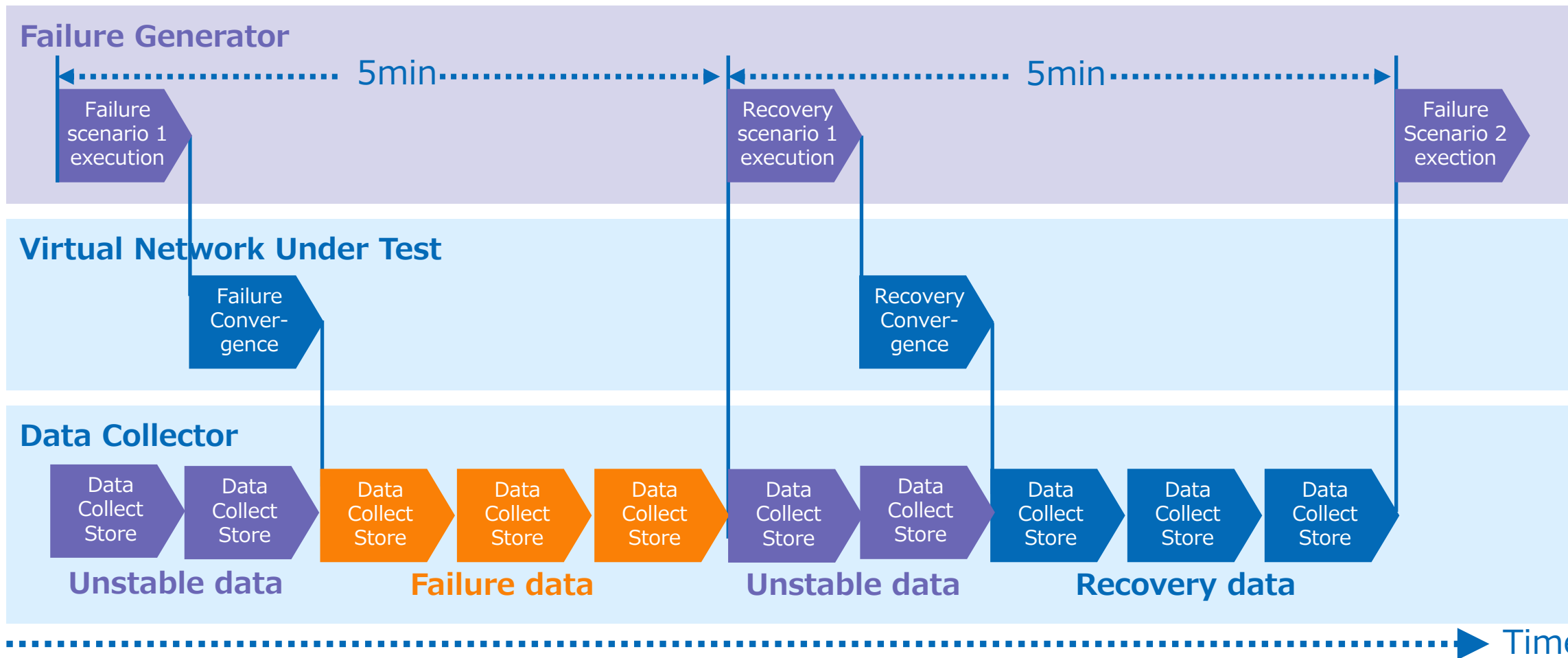
Four types of data sets for learning and evaluation are provided to participants as follows.

Category	File Name	Description	Data format	Time zone (1)
Label	Failure-management	Event date (failure and recovery) and event types, which are listed along the time series	json	UTC
Data	Virtual-Infrastructure	Performance monitoring data sets on instances and virtual network functions gathered from openstack ceilometer, which are listed along the time series	json	JST
	Physical-Infrastructure	Performance monitoring data sets gathered from the physical server under openstack, which are listed along the time series	json	JST
	Network-Device	Performance monitoring information and BGP route information gathered from NEs under the virtual IP network, which are listed along the time series	json	JST

(1) The time zone differs between those of the label and the data due to the system configuration. 9 hours difference exists between UTC and JST.

# 1. Data collection principles

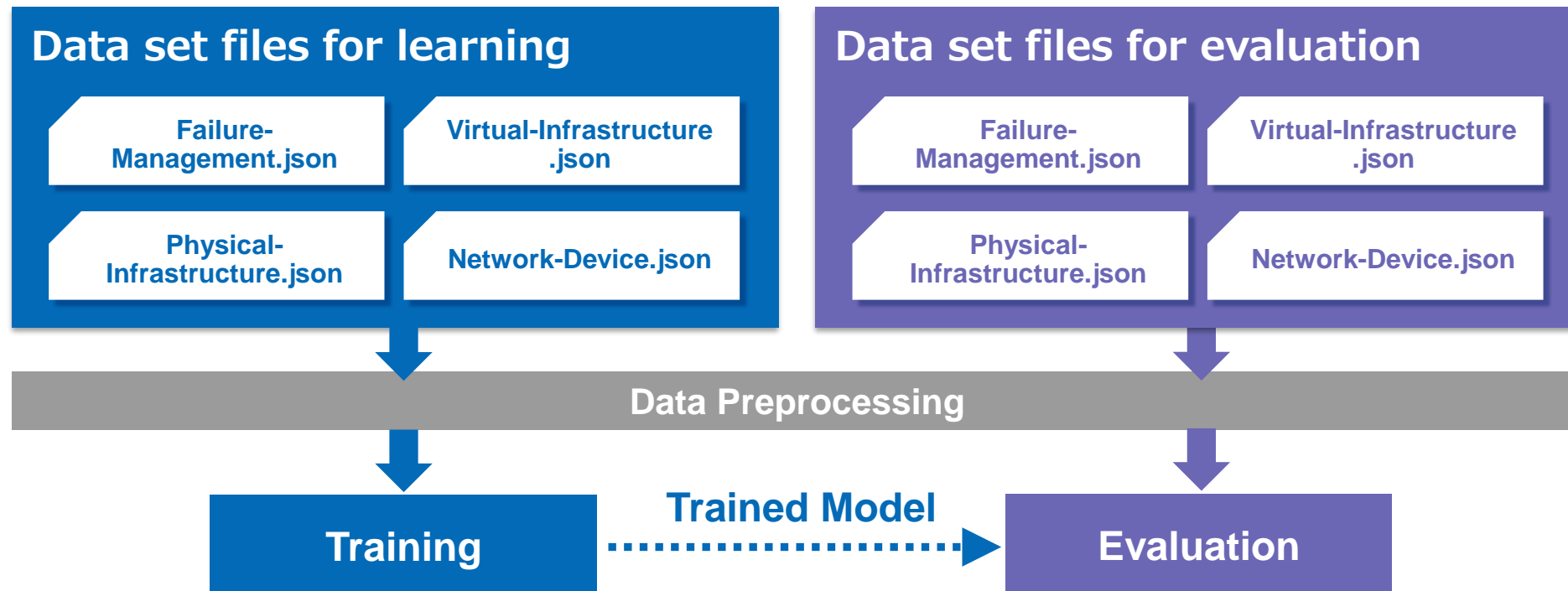
In order to create data sets, the data collector was developed to collect and store data sets every minute from the network. Once a failure is intentionally caused and recovered, the network indicates a failure or normal status after a period of transition, corresponding to failure data (orange arrows) and recovery data (blue arrows). The period of transition depends on a failure scenario and enough guard time is desired to be considered. The time interval between a failure and a recovery is 5 min.



## 2. How to use the dataset

Two types of dataset files for learning and evaluation are provided to participants. The dataset files for learning can be used for training AI models, and the dataset files for evaluation can be used for evaluating performance of the trained model.

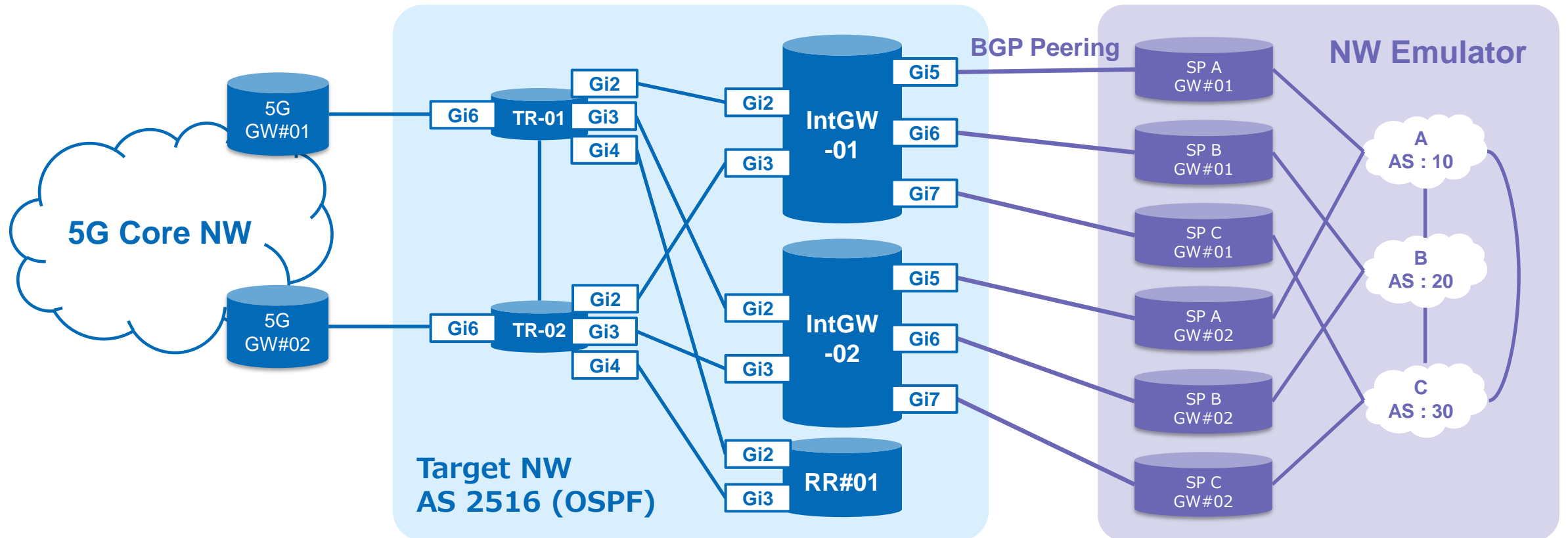
The dataset files for learning corresponds to use cases when all failure scenarios (shown in Chapter 4) are comprehensively invoked at all possible failure points. The dataset files for evaluation corresponds to the case when a combination of a failure scenario and a failure point is randomly and limitedly generated.



# 3. Network Topology

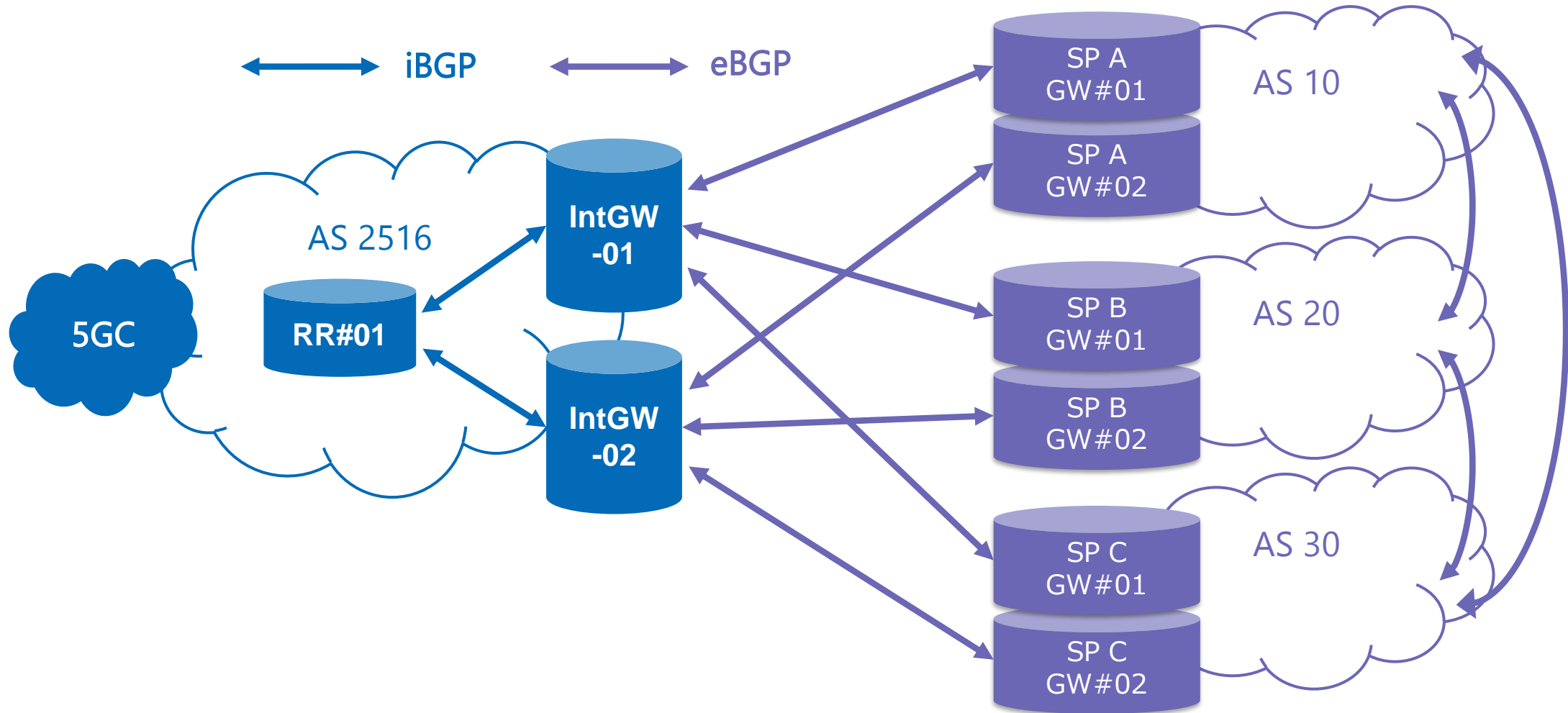
## Physical topology

The target of data collection is an IP core network, which connects the 5G core network for the Internet connectivity. The IP core network totally consists of 5 network elements, where TR-01, 02 are an IP core node, IntGW-01, 02 are an Internet gateway router peered with other SPs, and RR-01 is a route reflector sharing route information.



# 3. Network Topology

## ■ Logical Topology (BGP)

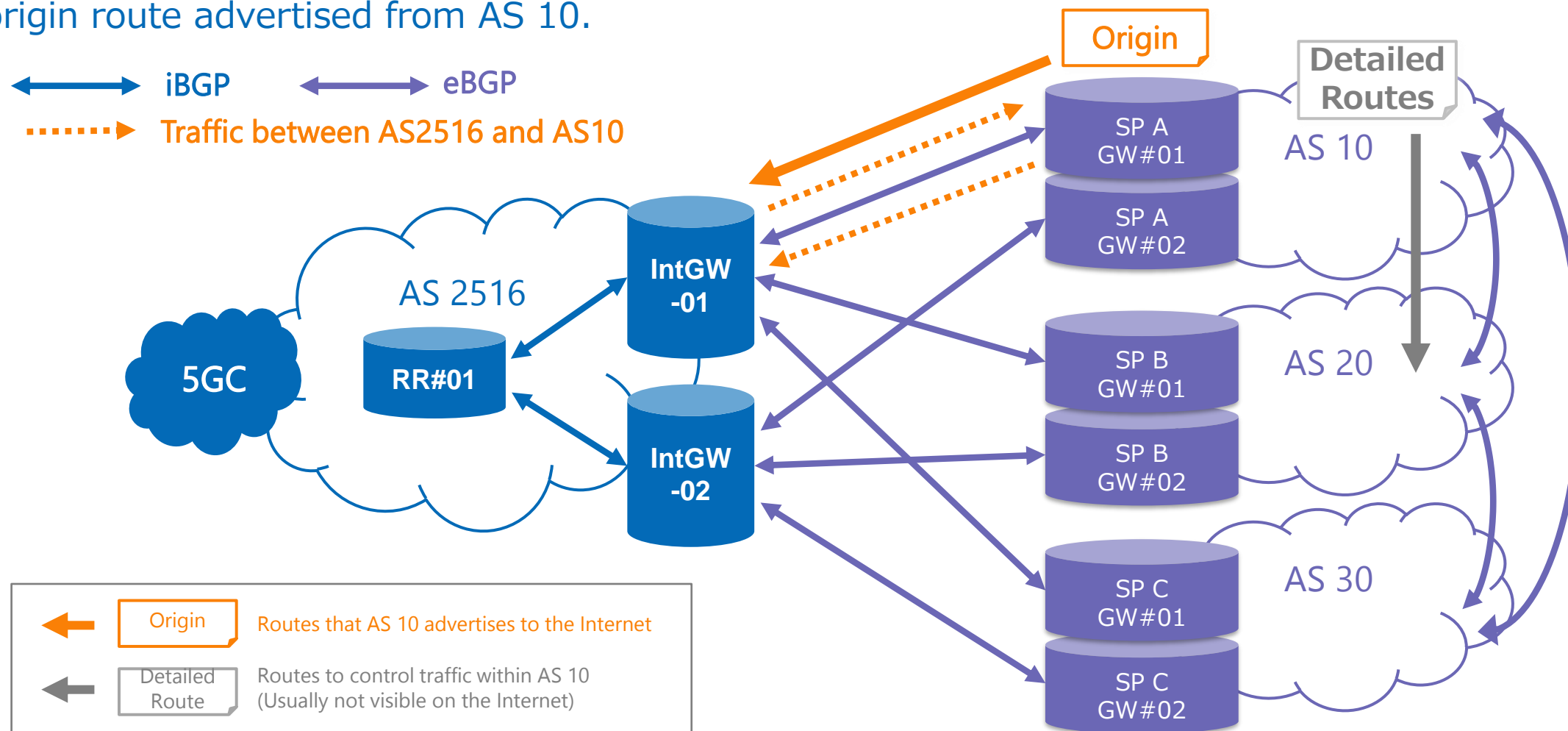


# 4. Use cases

Scenario	UC No.	Use Case Name	Description
Route Information Failure	UC1	BGP Injection	Inject the anomaly route from another SP
	UC2	BGP Hijack	Hijack the own origin route by another SP
Interface Failure	UC3	Interface Down	Cause an interface down
	UC4	Packet Loss	Cause the packet loss on an interface
	UC5	Packet Delay	Cause the delay of packets on an interface
Network Element(NE) Failure	UC6	NE Reboot	Unplanned reboot of a NE

## Route Information in a normal state

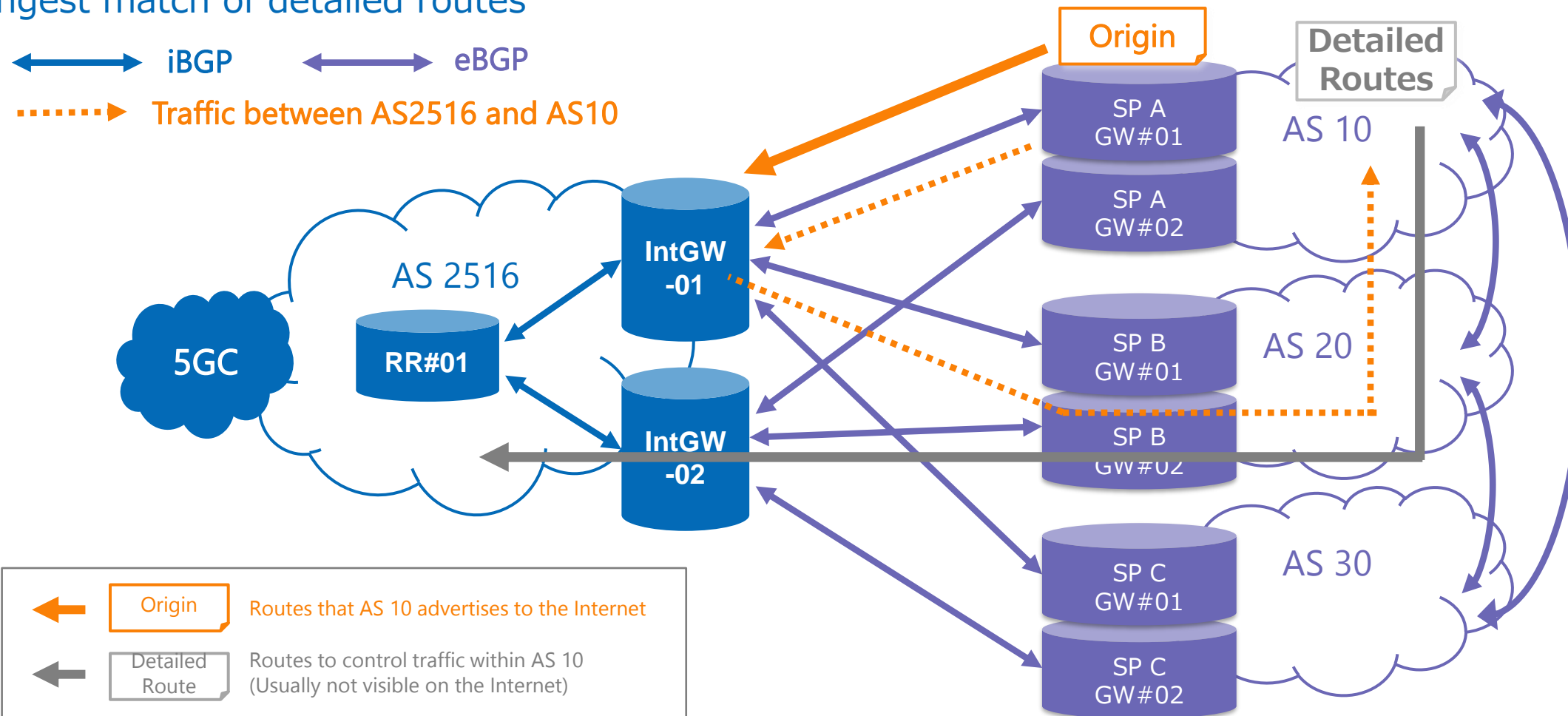
Detailed routes of AS 10 are exchanged only with AS 20 by the coordinated operation between AS 10 and AS 20, therefore the traffic between AS2516 and AS 10 are transported following the origin route advertised from AS 10.





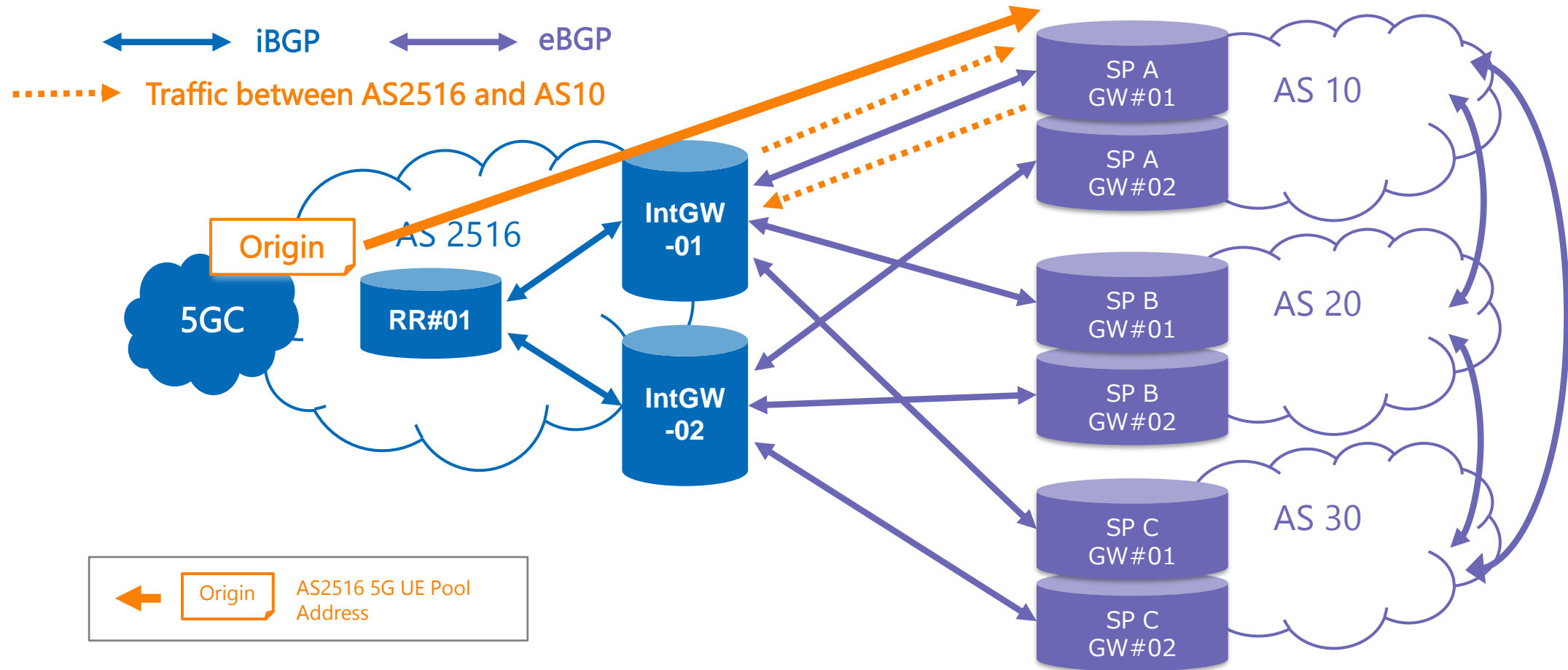
## ■ Route Information in an anomaly state

AS 20 happens to send detailed routes of AS 10 to AS 2516 due to an operational mistake. The traffic is switched from the direct route to the AS 10 to bypass route via AS 20 according to the longest match of detailed routes



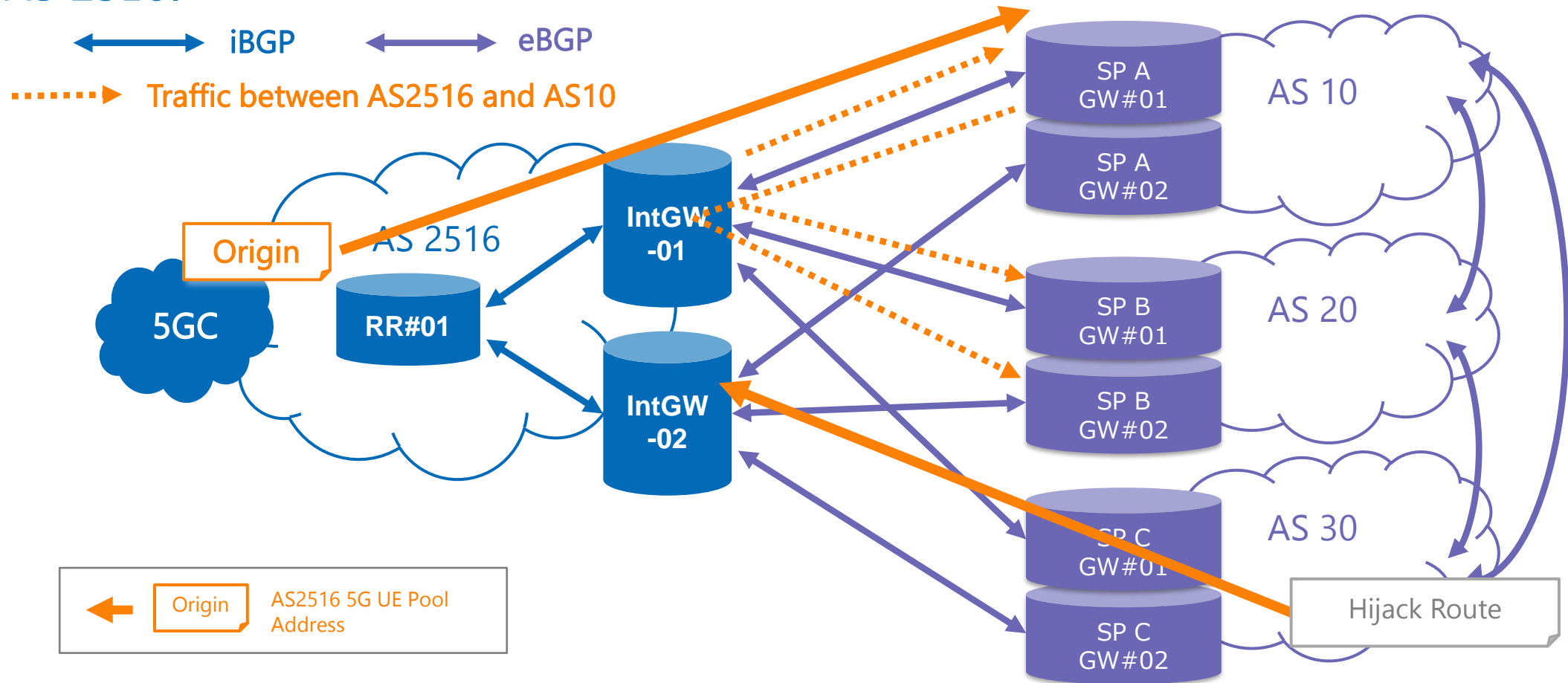
## Route Information in a normal state

The traffic between AS 2516 and AS 10 is transported according to the advertised origin routes of AS 2516.



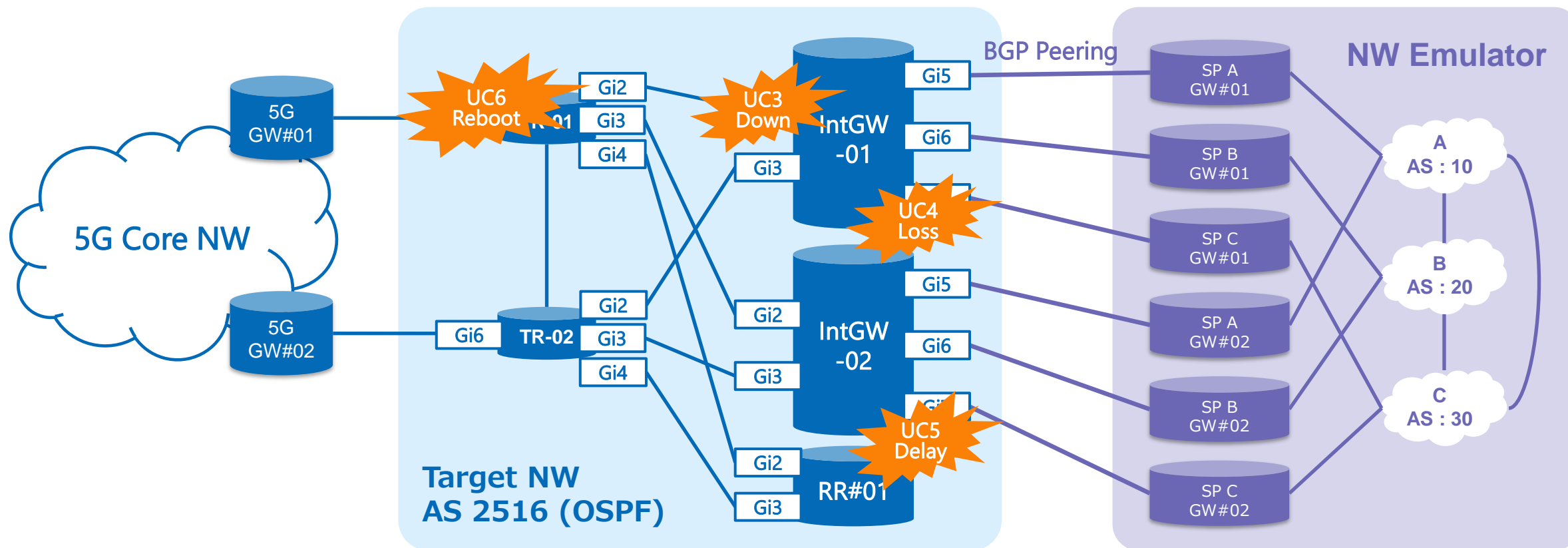
## Route Information in an anomaly state

Malicious SP C intentionally advertised a part of the IP address space allocated to 5G terminals in AS 2516 for route hijack, and caused traffic disruption from other operators to AS 2516.



## ■ Use case detail

Interface failures (packet loss, packet delay, down) and node failures (reboot) are comprehensively (for learning) and randomly (for evaluation) caused.



# failure-management sample

```
{
  "index": "failure_label_management-20200326",
  "_type": "_doc",
  "_id": "2020-03-26T13:16:29+0900",
  "_version": 1,
  "_score": null,
  "_source": {
    "scenario": "bgp-20200326-05",
    "proc": 1,
    "proc_type": 0,
    "failure_type": "ixnetwork-bgp-injection-start",
    "project": "bgpnw",
    "sp": "A",
    "original_sp": "B",
    "started_at": "2020-03-26T13:16:29.296+09:00",
    "stopped_at": "2020-03-26T13:16:38.619+09:00",
    "status": "succeeded",
    "@timestamp": "2020-03-26T13:16:29+0900"
  },
  "fields": {
    "stopped_at": [
      "2020-03-26T04:16:38.619Z"
    ],
    "started_at": [
      "2020-03-26T04:16:29.296Z"
    ]
  }
},
"sort": [
  1585196189296
]
```

Unique Key

Failure scenario

Failure execution start/end date and time

```
{
  "_index": "virtual-infrastructure-bgpnw2-20200706",
  "_type": "_doc",
  "_id": "2020-07-06T13:06:00+0900",
  "_version": 1,
  "_score": null,
  "_source": {
    "projects": [
      {
        "name": "bgpnw2"
      }
    ],
    "devices": [
      {
        "id": "bb7b904e-d3f9-4f12-bf7a-4497436d9901",
        "name": "IntGW-01",
        "flavor": {
          "id": "d7cfbc8a-0643-4587-b09a-bc47774c8c5b",
          "name": "bgp.router",
          "ram": 8192,
          "vcpus": 4,
          "disk": 0
        },
        "image": {
          "id": "4cca1723-2565-4beb-9b89-0ba2f677b726",
          "name": "csr1kv-9.16.09.04",
          "min_ram": 0,
          "min_disk": 0,
          "size": 919863296,
          "status": "active",
          "container_format": "bare",
          "file": "/v2/images/4cca1723-2565-4beb-9b89-0ba2f677b726/file",
          "disk_format": "qcow2",
          "visibility": "public",
          "created_at": "2020-01-17T02:49:37Z",
          "updated_at": "2020-01-17T02:49:43Z",
          "metrics": {
            "image-size": 919863296
          }
        }
      },
      {
        "status": "ACTIVE",
        "vm_state": "active",
        "power_state": 1,
        "progress": 0,
```

## Unique Key

```
    "availability_zone": "nova",
    "compute": "openstack-wf",
    "instance_name": "instance-00000009",
    "created": "2020-01-24T04:04:22Z",
    "updated": "2020-07-04T11:17:56Z",
    "metrics": {
      "disk-device-allocation": 499712,
      "disk-device-capacity": 499712,
      "disk-device-read-bytes": 0,
      "disk-device-read-bytes-rate": 0,
      "disk-device-read-latency": 0,
      "disk-device-read-requests": 0,
      "disk-device-read-requests-rate": 0,
      "disk-device-usage": 499712,
      "disk-device-write-bytes": 0,
      "disk-device-write-bytes-rate": 0,
      "disk-device-write-latency": 0,
      "disk-device-write-requests": 0,
      "disk-device-write-requests-rate": 0,
      "compute-instance-booting-time": 34.576838,
      "cpu": 2702579275000000,
      "cpu-delta": 104060000000,
      "cpu_util": 43.07878647215821,
      "disk-allocation": 47431680,
      "disk-capacity": 8590434304,
      "disk-ephemeral-size": 0,
      "disk-read-bytes": 229376,
      "disk-read-bytes-rate": 0,
      "disk-read-requests": 23,
      "disk-read-requests-rate": 0,
      "disk-root-size": 0,
      "disk-usage": 47489024,
      "disk-write-bytes": 2602914816,
      "disk-write-bytes-rate": 480.35484069415025,
      "disk-write-requests": 438803,
      "disk-write-requests-rate": 0.083309873273019,
      "memory": 8192,
      "memory-resident": 4016,
      "vcpus": 4
    },
    "project": "bgpnw2"
  }
}
```

```
{
  "_index": "physical-infrastructure-bgpnw-202001-1",
  "_type": "_doc",
  "_id": "2020-03-31T13:12:00+0900",
  "_version": 1,
  "_score": null,
  "_source": {
    "computes": [
      {
        "status": "enabled",
        "service": {
          "host": "compute01",
          "disabled_reason": null,
          "id": 11
        },
        "vcpus_used": 5,
        "hypervisor_type": "QEMU",
        "local_gb_used": 55,
        "vcpus": 48,
        "hypervisor_hostname": "compute01",
        "memory_mb_used": 8704,
        "memory_mb": 257446,
        "current_workload": 0,
        "state": "up",
        "host_ip": "172.16.1.241",
        "free_disk_gb": 383,
        "hypervisor_version": 2011001,
        "disk_available_least": 334,
        "local_gb": 438,
        "free_ram_mb": 248742,
        "id": 1,
        "metrics": {
          "compute-node": {
            "compute-node-cpu-frequency": 2236,
            "compute-node-cpu-idle-percent": 99,
            "compute-node-cpu-idle-time": 22622273938000000,
            "compute-node-cpu-iowait-percent": 0,
            "compute-node-cpu-iowait-time": 2332730000000,
            "compute-node-cpu-kernel-percent": 0,
            "compute-node-cpu-kernel-time": 5997980080000000,
            "compute-node-cpu-percent": 0,
            "compute-node-cpu-user-percent": 0,
            "compute-node-cpu-user-time": 3740360490000000
          },
          "hardware": {
            "hardware-cpu-load-15min": 0.13,
            "hardware-cpu-load-1min": 0.27,
            "hardware-cpu-load-5min": 0.215,
            "hardware-cpu-util": 0,
            "hardware-disk-size-total": 0,
            "hardware-disk-size-used": 0,
            "hardware-memory-buffer": 1196200,
            "hardware-memory-cached": 15348682.666666666,
            "hardware-memory-swap-avail": 2097148,
            "hardware-memory-swap-total": 2097148,
            "hardware-memory-total": 263624884,
```

## Unique Key

```
"hardware-memory-used": 26181624,
"hardware-network-ip-incoming-datagrams": 1488267445,
"hardware-network-ip-outgoing-datagrams": 2378446807,
"hardware-system_stats-cpu-idle": 99,
"hardware-system_stats-io-incoming-blocks": 248599704,
"hardware-system_stats-io-outgoing-blocks": 1200049368
},
"hardware-disk": [
  {
    "name": "_dev_sda2",
    "hardware-disk-size-total": 459924552,
    "hardware-disk-size-used": 44851038
  },
  {
    "name": "_dev_sda1",
    "hardware-disk-size-total": 459924552,
    "hardware-disk-size-used": 44851038
  }
],
"hardware-network": [],
"hardware-ipmi-fan": [
  {
    "name": "fan2a_(0x3a)",
    "hardware-ipmi-fan": 3840
  },
  {
    "name": "fan6a_(0x42)",
    "hardware-ipmi-fan": 3720
  },
  {
    "name": "fan3a_(0x3c)",
    "hardware-ipmi-fan": 3840
  },
  {
    "name": "fan4b_(0x3f)",
    "hardware-ipmi-fan": 4080
  },
  {
    "name": "fan6b_(0x43)",
    "hardware-ipmi-fan": 4200
  },
  {
    "name": "fan3b_(0x3d)",
    "hardware-ipmi-fan": 4080
  },
  {
    "name": "fan5a_(0x40)",
    "hardware-ipmi-fan": 3840
  },
  {
    "name": "fan5b_(0x41)",
    "hardware-ipmi-fan": 4080
  },
  {
    "name": "fan1b_(0x39)",
    "hardware-ipmi-fan": 4080
  },
  {

```

```
{
  "_index": "network-device-bgpnw-20200301",
  "type": "doc",
  "_id": "2020-03-01T12:01:00+0900",
  "_version": 1,
  "_score": null,
  "_source": {
    "devices": [
      {
        "name": "IntGW-01",
        "modules": {
          "openconfig-interfaces": {
            "interfaces": {
              "interface": [
                {
                  "name": "GigabitEthernet1",
                  "config": {
                    "name": "GigabitEthernet1",
                    "type": "ianaift:ethernetCsmacd",
                    "description": "ManagementIF",
                    "enabled": true
                  },
                  "state": {
                    "name": "GigabitEthernet1",
                    "type": "ianaift:ethernetCsmacd",
                    "enabled": true,
                    "ifindex": 1,
                    "admin-status": "UP",
                    "oper-status": "UP",
                    "last-change": 1580092289000921000,
                    "counters": {
                      "in-octets": 4965172089,
                      "in-unicast-pkts": 76964557,
                      "in-broadcast-pkts": 0,
                      "in-multicast-pkts": 0,
```

Unique Key

```
"in-discards": 0,
"in-errors": 0,
"in-unknown-protos": 0,
"in-fcs-errors": 0,
"out-octets": 1712733927,
"out-unicast-pkts": 152852647,
"out-broadcast-pkts": 0,
"out-multicast-pkts": 0,
"out-discards": 0,
"out-errors": 0,
"last-clear": 1580092091000706000
}
},
```



Create and train a model of AI/ML by the data set for learning and verify the performance of the derived model by the data set of evaluation in terms of anomaly detection and root cause analysis.

Submit a power point file with a pdf format indicating the results and a demonstration video showing predicting performance.

Contact information

[info\\_itu5G\\_jp@lists.cc1g.kddi-research.jp](mailto:info_itu5G_jp@lists.cc1g.kddi-research.jp)



ITU AI/ML in 5G challenge

**【ITU-ML5G-PS-031】**  
**Network State Estimation by Analyzing Raw Video Data**

2020/7/29

NEC Corporation, Japan

# Agenda

**1. Introduction**

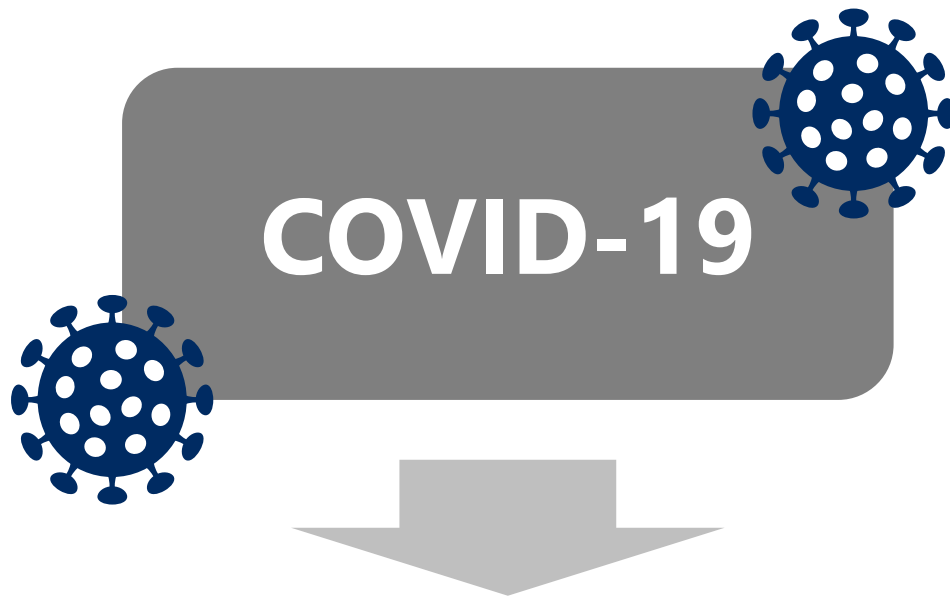
**2. Challenge**

**3. Dataset**

**4. Information**

# 1. Introduction

# Background – COVID19 pandemic



**The importance of *interactive live video streaming services*,  
e.g., telework system, is increasing!!**

# Social problem

## OTT services

Netflix

YouTube

Amazon

⋮

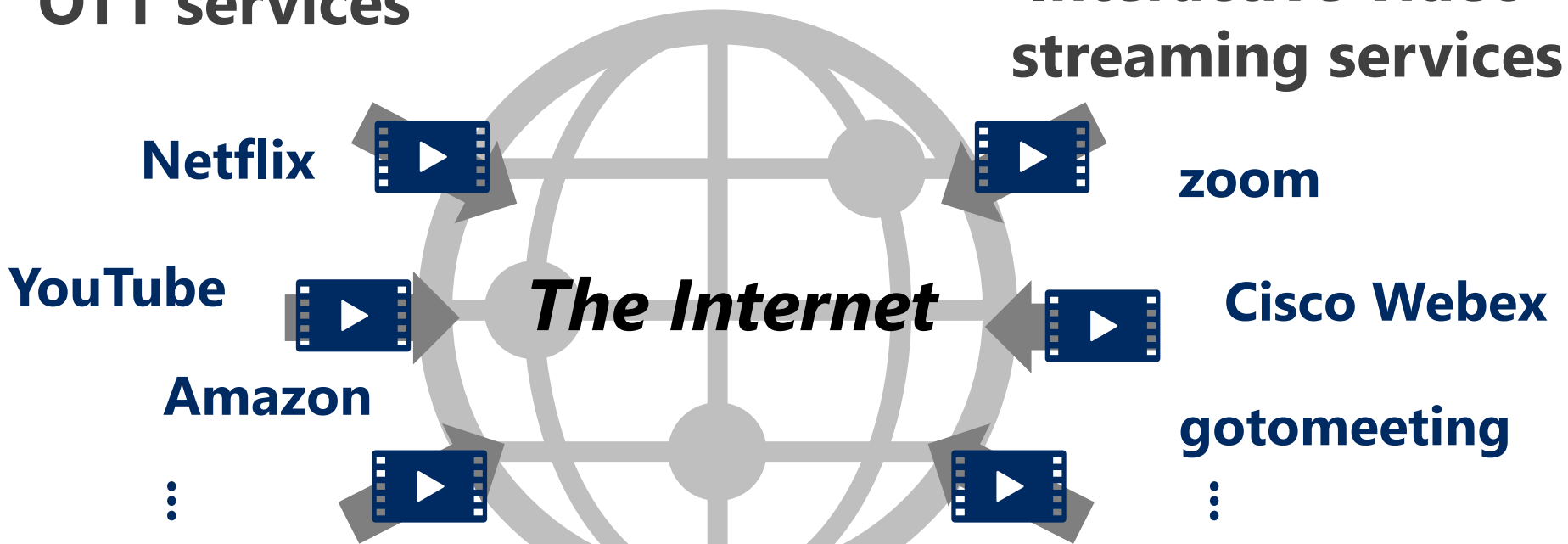
## Interactive video streaming services

zoom

Cisco Webex

gotomeeting

⋮



**For avoiding congestion in the Internet,  
video traffic reduction is required!!**

# Difference between OTT services and interactive services

## Service provided by OTT



**OTT providers,**  
e.g., Netflix and YouTube



**Standard resolution**  
**720p → 480p**

**Deliver**



**Consumers**

## Interactive services



**Consumer**

**Resolution depends on**  
***player setting***

e.g., zoom  
video volume depends on player's  
display size



**Consumer**



# Who needs to know network state?

Service provided by OTT

*OTT providers* need to know network states.

Interactive services

*Consumers* need to know network states.



*Interactive video streaming services force network state estimation for control video traffic to consumers.*

# Relationship between network state & video images

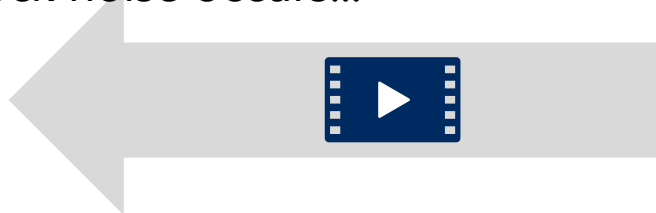
## Good network condition



## Bad network condition



Block noise occurs...



**Streaming  
server**

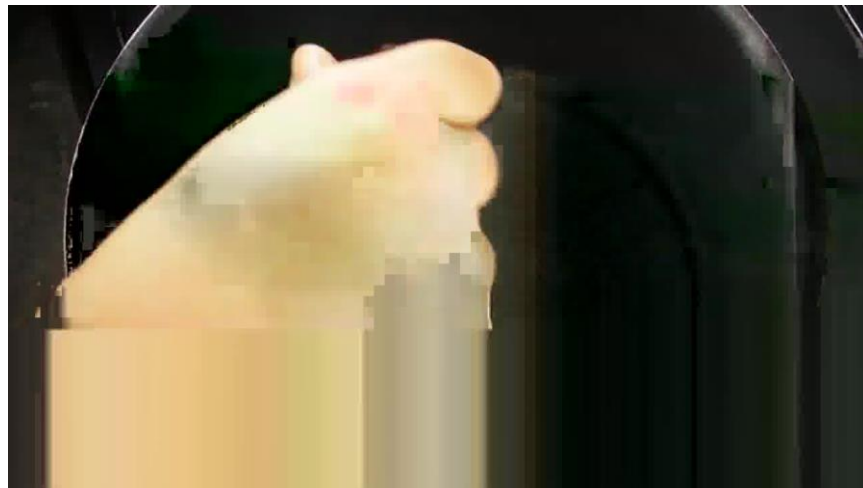
## What happens in the network???

# Example (Original vs Received)

## Original video



## Received video



### *Network state*

Throughput: 1100kbps

Loss rate: 0.1%

## Conventional approach

**KPI** is important for evaluate their (researcher's) approaches.

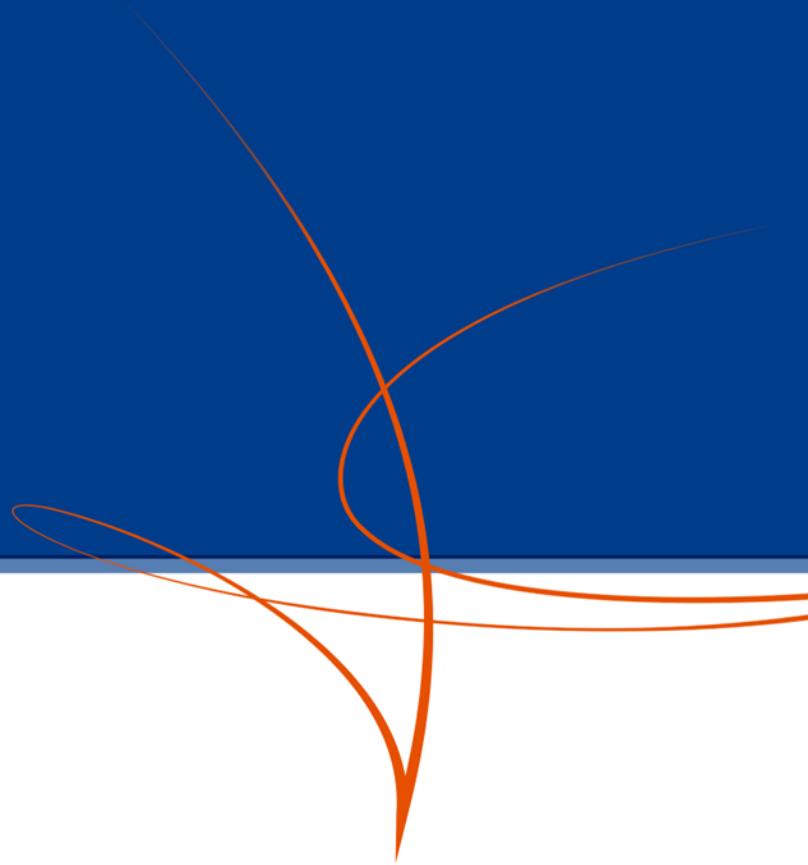
e.g., bit rate

## Practical case

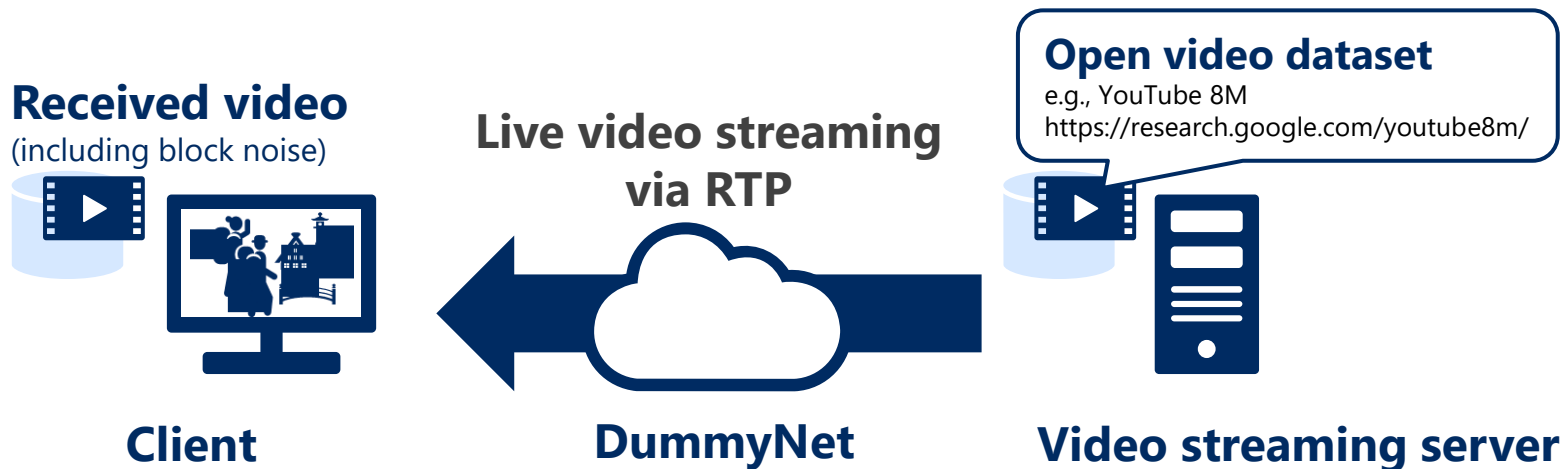
**Raw video image** is important.

**This challenge is the first step to understand relationship between raw video images and network state.**

## 2. Challenge



# Understanding network state from raw video



## Provided dataset

Original video data (.mp4)  
Received video data (.mp4)

## Task for participants

Estimate network state, i.e., throughput/loss ratio.  
Train their ML-based method by using given dataset.  
Performance measure is MAE.

# Training/test process

## Training phase

Original video (.mp4)



Received videos (.mp4)



Network condition

1100kbps, 0.001% loss



1200kbps, 0.001% loss

⋮



2000kbps, 0.001% loss



## Test phase

Original video (.mp4)



Received video (.mp4)



Input



Trained ML

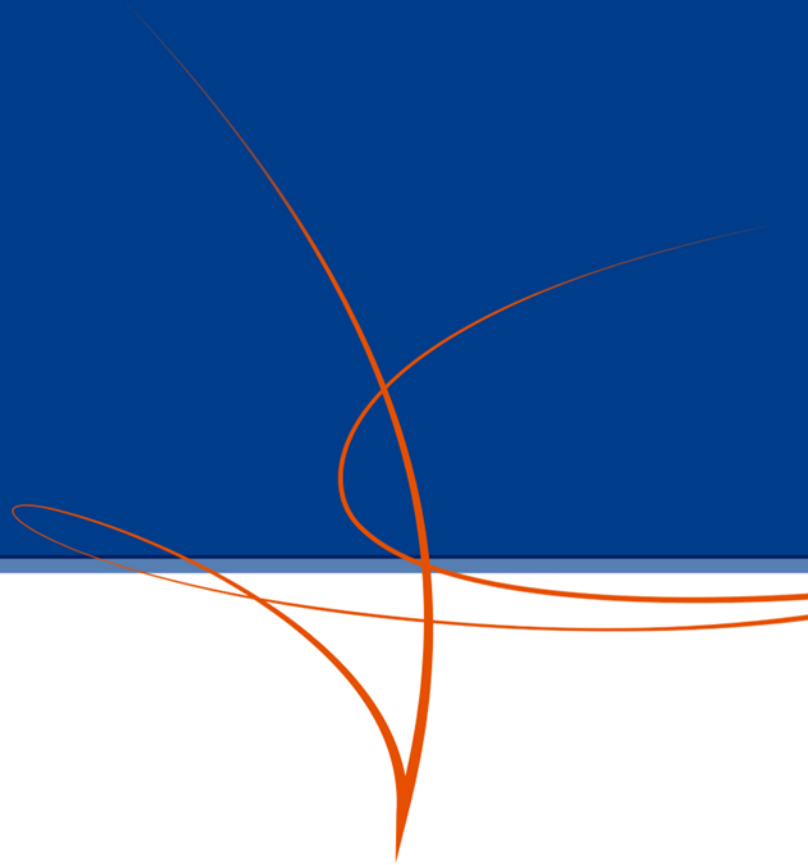
Network condition

Output



1100kbps, 0.001% loss

# 3. Dataset





## Provided dataset

- One original video data (.mp4)
- Many received video data (.mp4)

## Dummysnet configuration

- Traffic rate: from 1100kbps to 2000kbps at 100kbps intervals
- Packet loss ratio: 0.001%, 0.01%, 0.025%, 0.05%, 0.1%

### *Original video*



original.mp4

### *Received video*



videoid\_1100kbps\_001.mp4



videoid\_1200kbps\_001.mp4

•  
•  
•

•  
•  
•

# Dataset

**You can download our dataset from RISING web site.**

- [https://www.ieice.org/~rising/AI-5G/dataset/theme2-NEC/dataset\\_and\\_issue.tar.gz](https://www.ieice.org/~rising/AI-5G/dataset/theme2-NEC/dataset_and_issue.tar.gz)

**Dataset (dataset\_and\_issue.tar.gz, 24GB) includes following files.**

- **dataset**

- **original: original videos for training**
- **received: received videos named <video\_id>\_<bandwidth>\_<loss ratio>.mp4 for training**

- **issue**

- **original: original videos for network state estimation**
- **received: received videos for network state estimation**

- **README.md**

# Performance measure

For each of throughput and loss ratio, MAE is calculated as performance measure. (n is the number of test videos)

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |\text{Estimation} - \text{Answer}|$$

## **\*\* IMPORTANT \*\* - Submission**

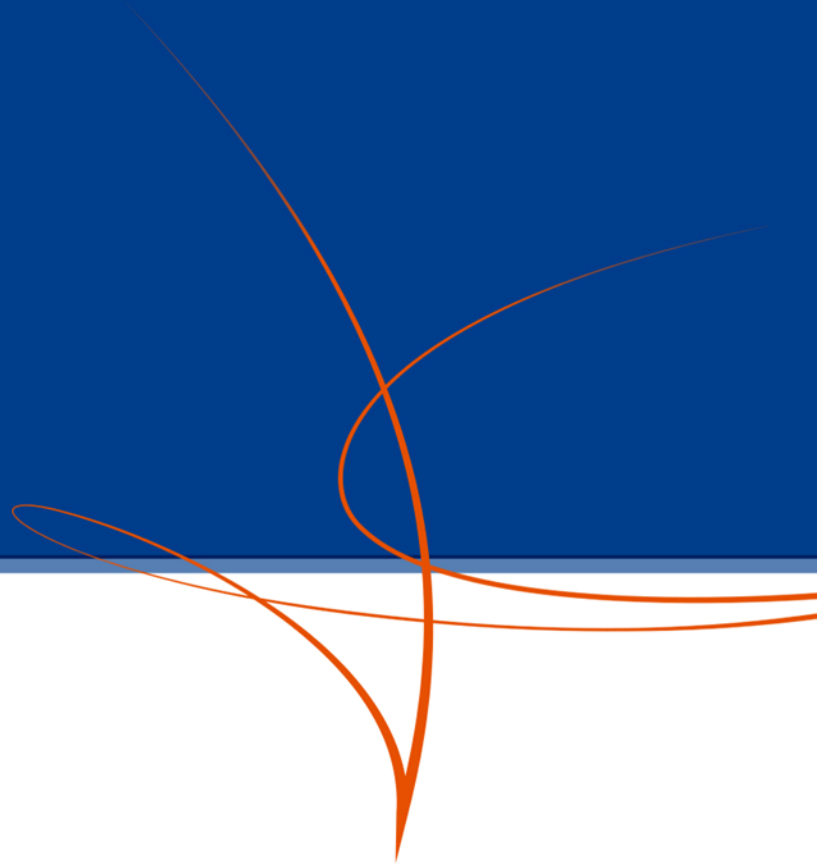
**Participants need to submit only *report*.**

- **Report includes the following items at least.**
  1. **Explanation of your method/approach**
  2. **Evaluation results (MAE, See “Evaluation criteria”) for provided data set**
  3. **Consideration**
- **Report format: A4 size, pdf, 4 pages at most.**

**All submissions will be evaluated in terms of**

1. **Performance measure (MAE)**
2. **Technical excellence**

# 4. Information



## Detailed information

- <https://www.ieice.org/~rising/AI-5G/>
- *Updated problem statement is shown in the web page!!*

## Contact by e-mail

- [5gc@nakao-lab.org](mailto:5gc@nakao-lab.org) or [rising-itu-support@mail.ieice.org](mailto:rising-itu-support@mail.ieice.org)
- Subject of E-mail has to be [ITUML5G-PS-031] or [ITU-JP-Theme2].

 **Orchestrating** a brighter world

**NEC**