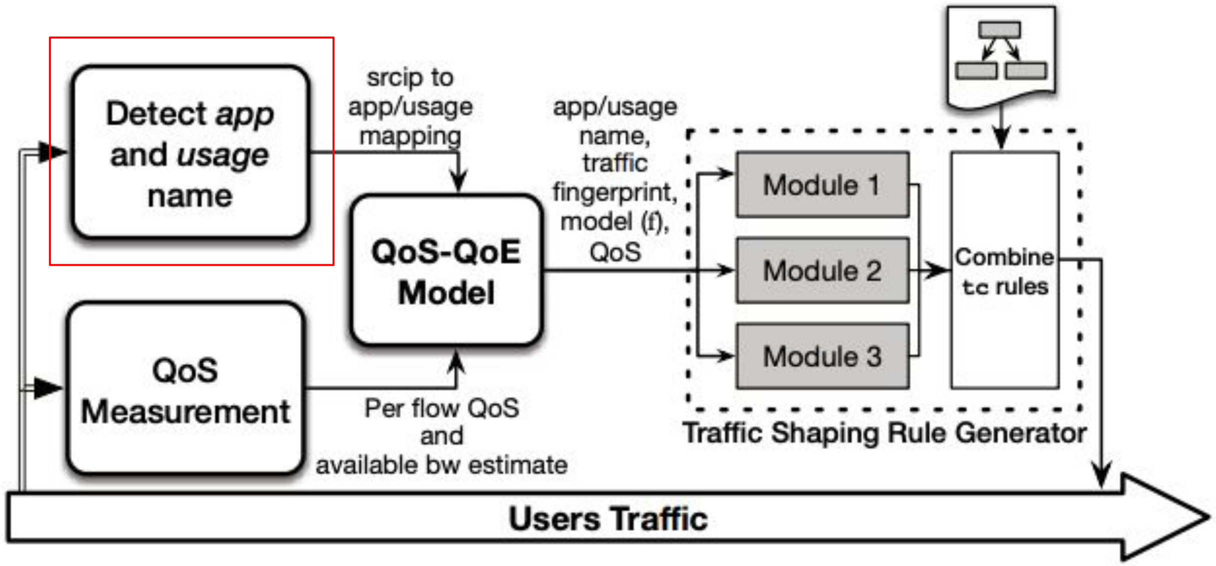
# Progressive Slicing for Application Identification in Application-Specific Network Slicing

IEEE Globecom 2020

Takamitsu Iwai, Akihiro Nakao
University of Tokyo

# Application-Specific Traffic Control

With the diversity of mobile apps, application slicing is proposed to isolate and control traffic for each app with different QoS requirements.

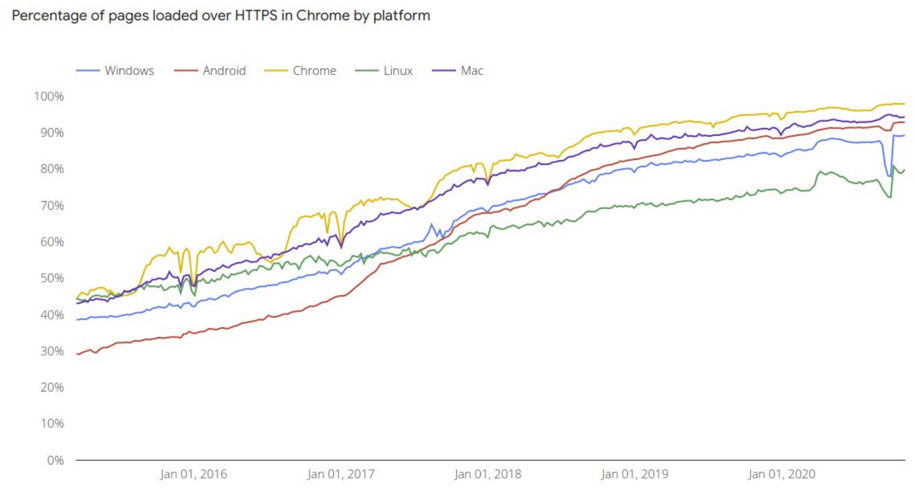

Detect *app* and *usage* name
srcip to app/usage mapping
QoS-QoE Model
app/usage name, traffic fingerprint, model (f), QoS
Module 1
Module 2
Module 3
Combine tc rules
Traffic Shaping Rule Generator
QoS Measurement
Per flow QoS and available bw estimate
Users Traffic

Nikravesh, Ashkan, et al. "QoE Inference and Improvement Without End-Host Control." *2018 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2018.

## Difficulties of Application Identification

Encryption traffic is increasing rapidly.
→it's not easy to identify apps from flows.

Encryption is under consideration for SNI and DNS as well as increased TLS traffic
→We can't identify the app from the protocol.

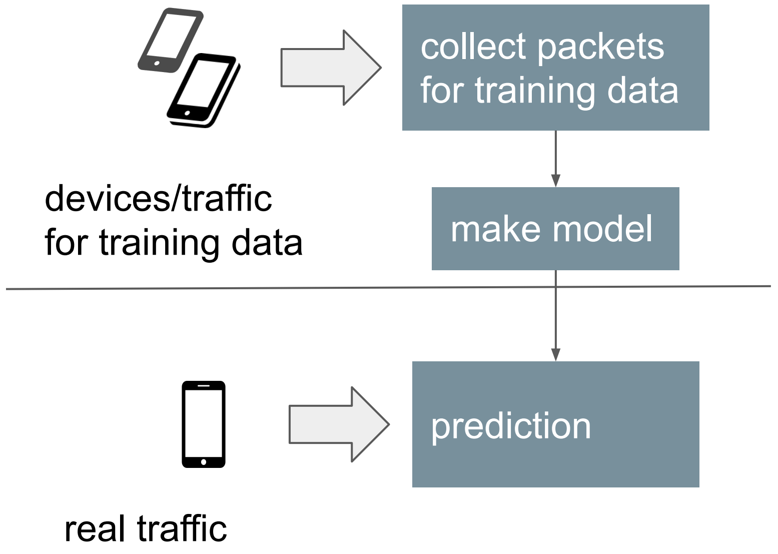


https://transparencyreport.google.com/https/overview

# Application Identification Using Machine Learning

Use statistics of the flows to identify app without any references of packet data.

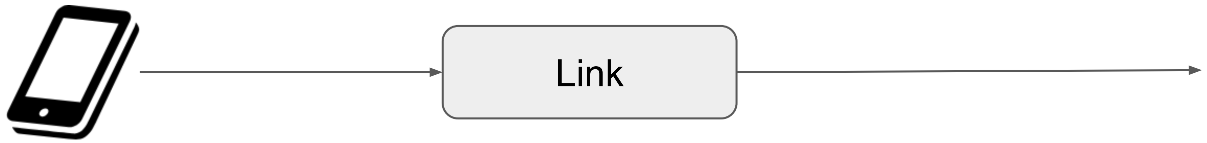
Existing application identification research do not sufficiently discuss the impact of classification delay and misclassification on the application QoE.


collect packets for training data
devices/traffic for training data
make model
prediction
real traffic

# Problem 1. classification delay

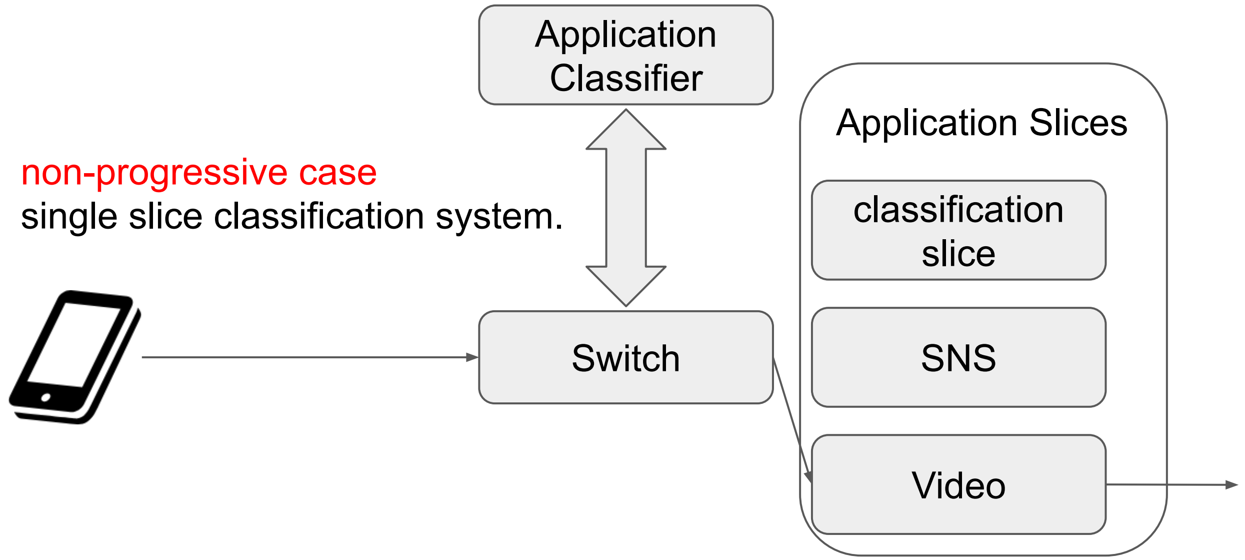
To control traffic based on the application,
we need app classifier and app slices.

non-classified case
nothing to do with the flows



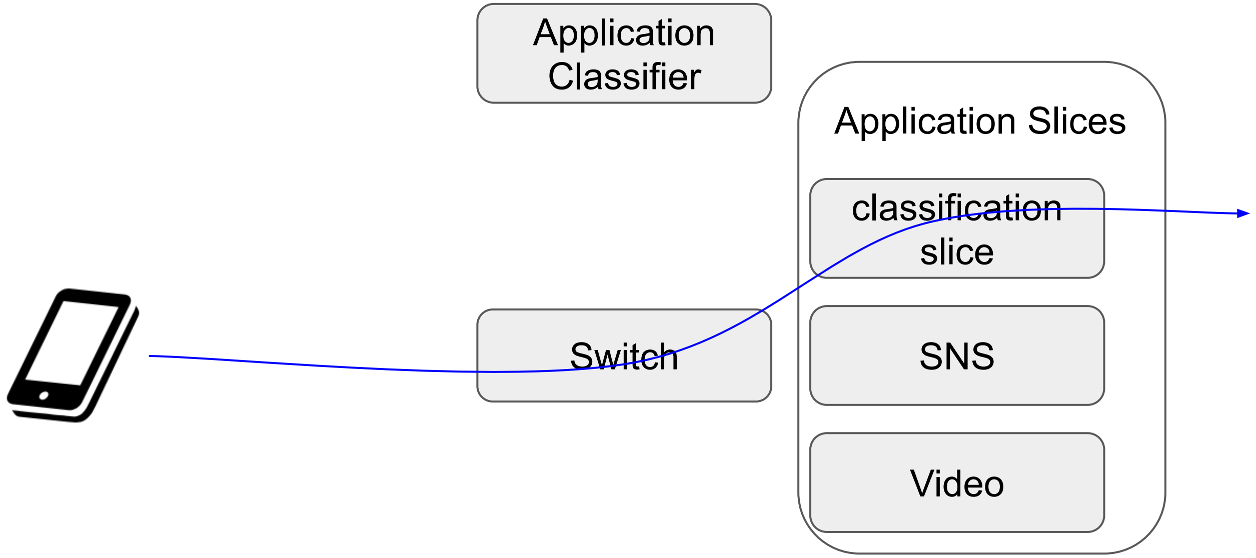
# Problem 1. classification delay

We need **classification slice** in classification.



## Problem 1. classification delay

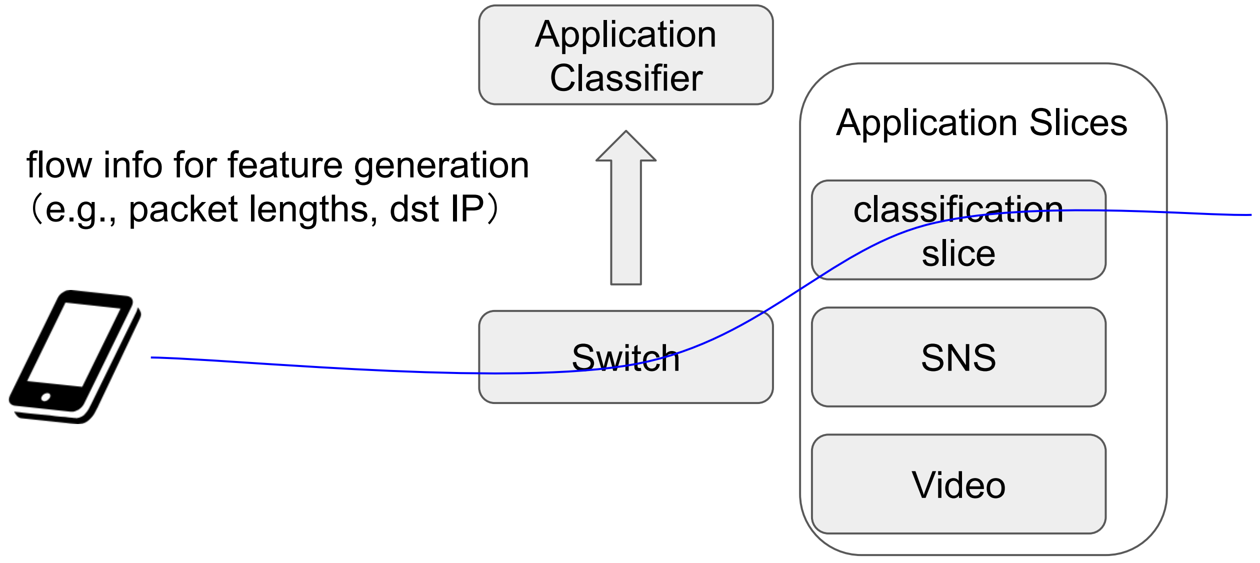
We need **classification slice** in classification.



When new flow is coming

# Problem 1. classification delay

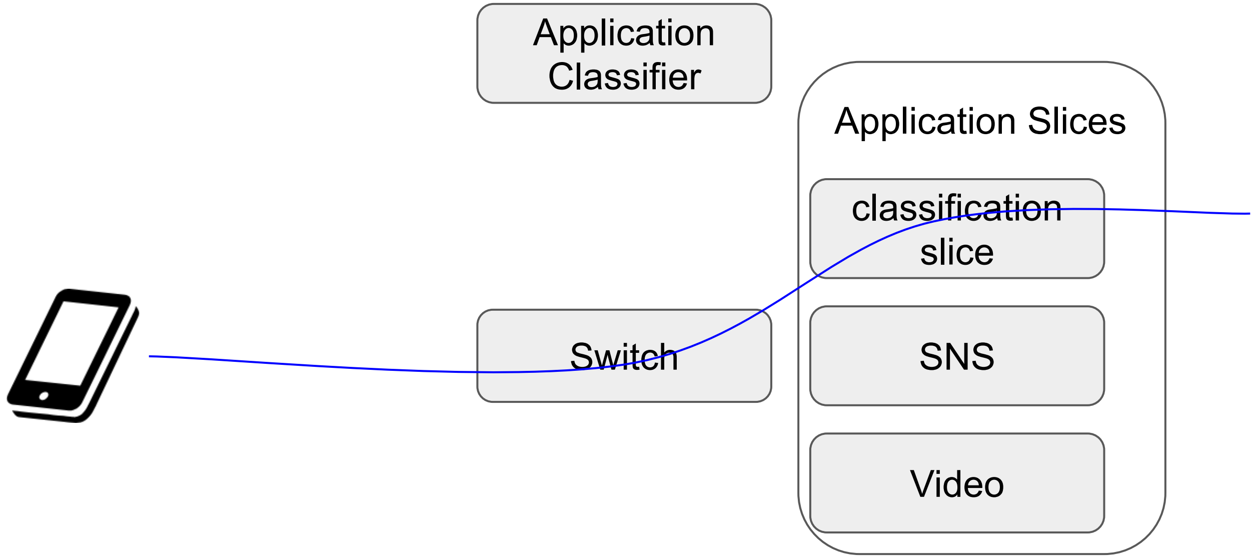
We need **classification slice** to wait for the flow in classification and feature generation.



collected the sufficient number of pkts for classification

## Problem 1. classification delay

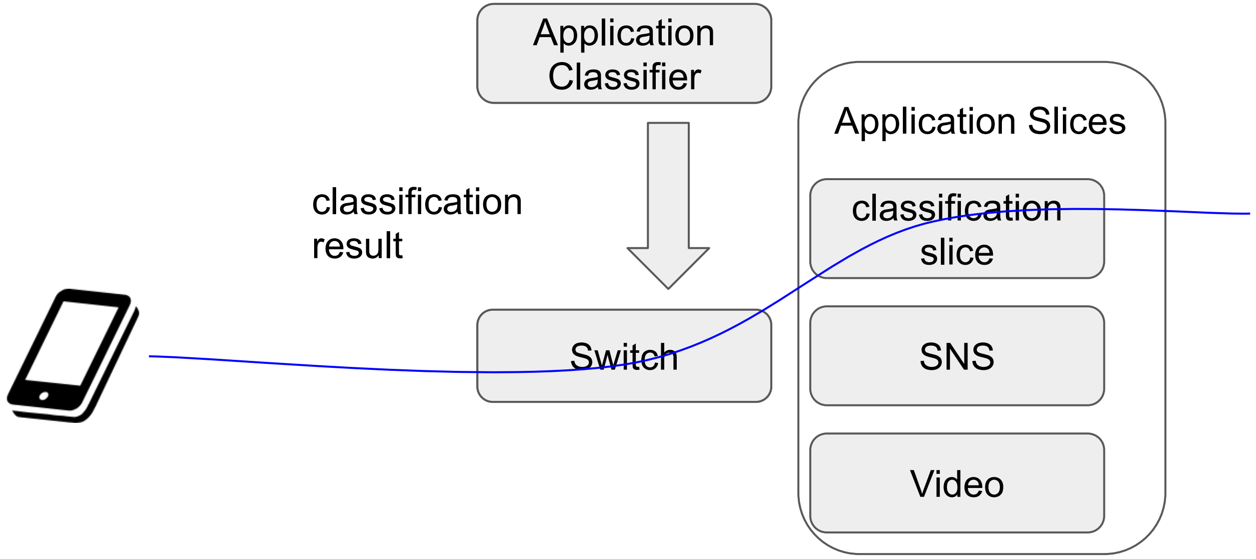
We need **classification slice** to wait for the flow in classification and feature generation.



Wait for classification result

## Problem 1. classification delay

We need **classification slice** to wait for the flow in classification and feature generation.



assigned to the video slice after the classification

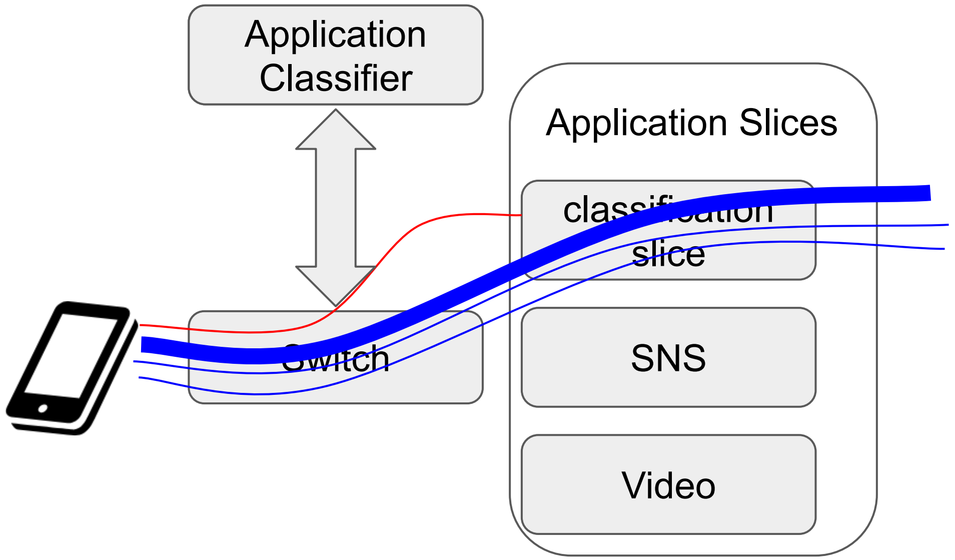
## Problem 1. classification delay

We need classification slice to wait for the flow in classification and feature generation.

**Congestion on the classification slice degrades User's QoE due to the classification delay.**

Video apps determine video quality based on the available bandwidth.
→congestion of classification slice negatively impacts QoE.

Delays occur due to congestion caused by waiting for classifications on multiple flows



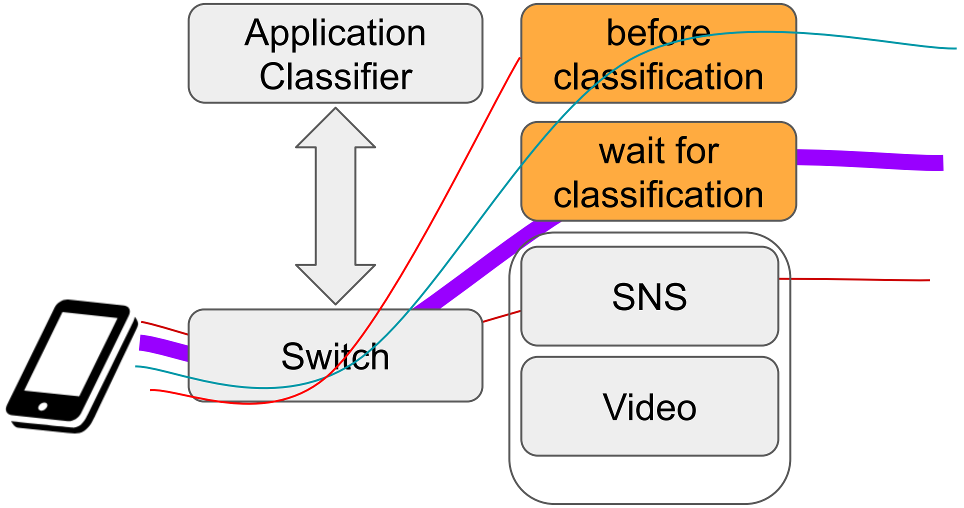
# Problem 1. classification delay

We need classification slice because we need to wait for the flow in classification and feature generation.

**Congestion on the classification slice degrades User's QoE due to the classification delay.**

Video apps determine video quality based on the available bandwidth.
→congestion of classification slice negatively impacts QoE.

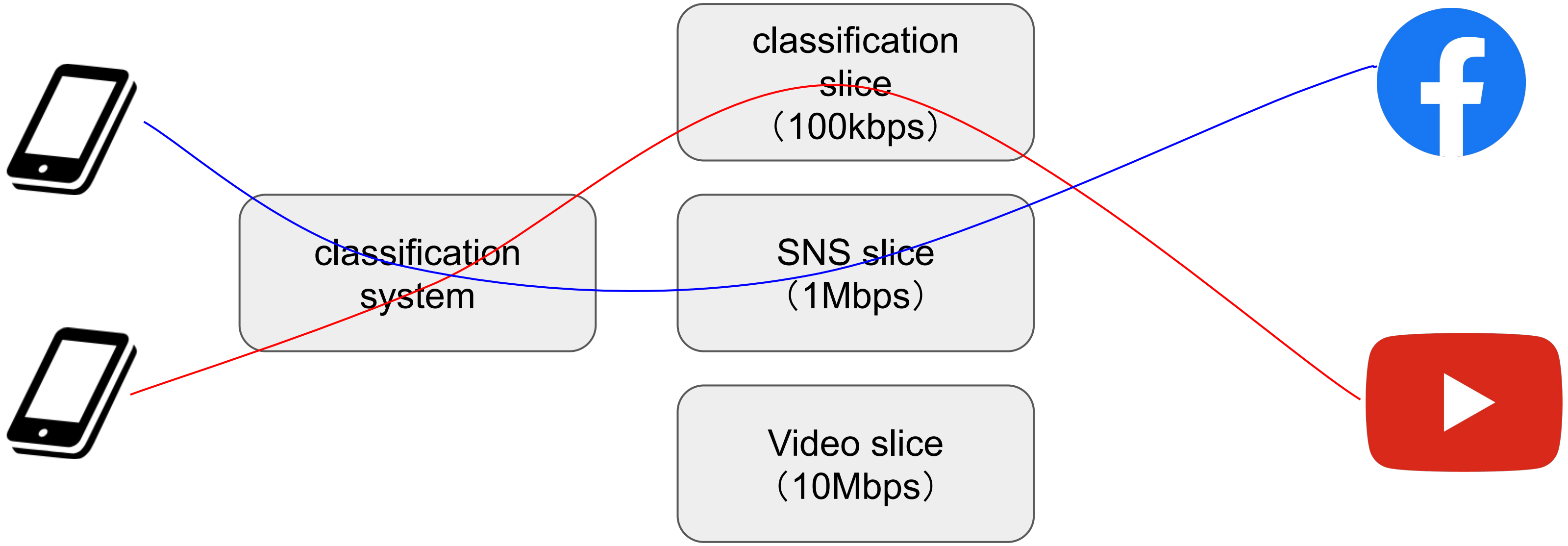
Avoid congestion by switching slices according to the classification status



progressive slicing
proposed slicing architecture

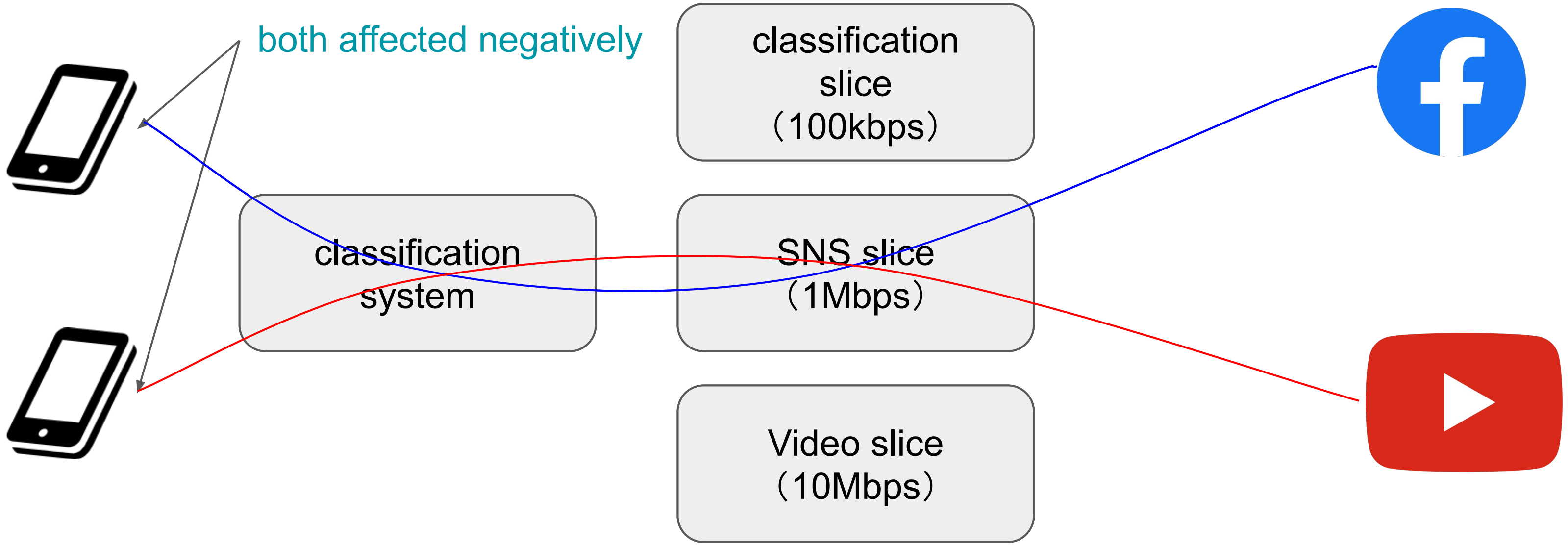
# Problem 2. misclassification

Since app slices have different QoS, if flows are assigned into a different slice due to misclassification, the QoE of both the target app flows and the misclassified flows may decrease.



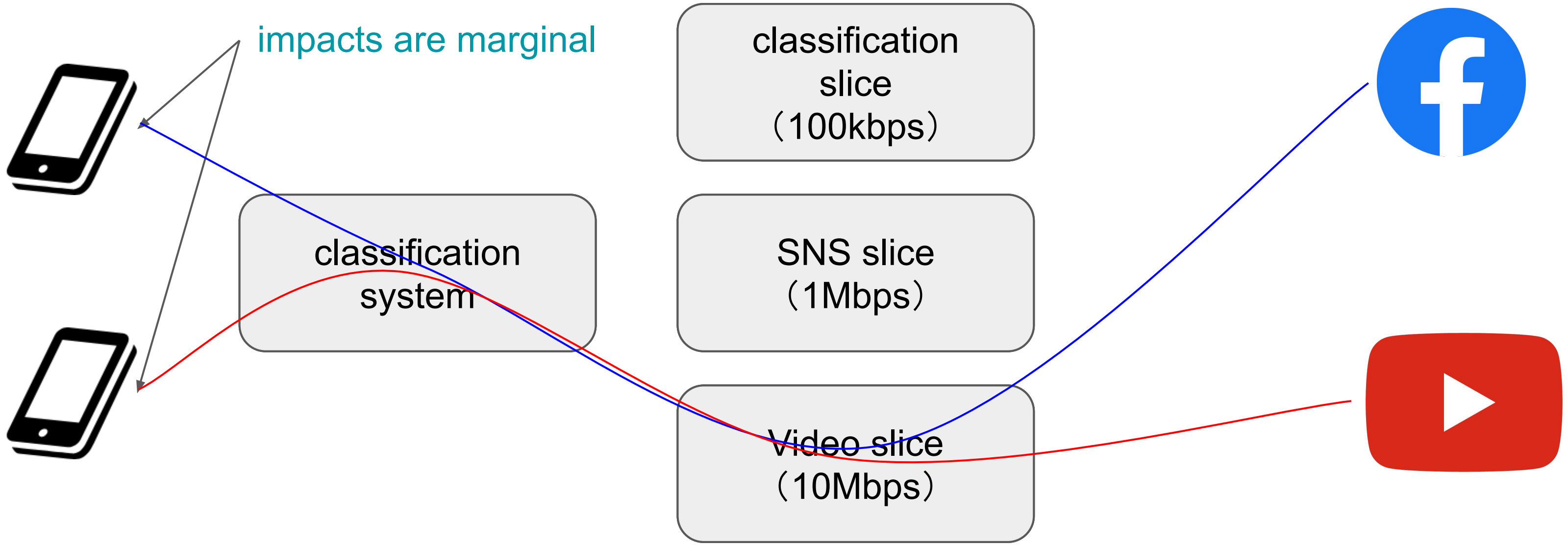
## Problem 2. misclassification

If the Youtube flow is misclassified and assigned to the SNS slice,
users who want to watch Youtube will not benefit from it
and the SNS slice will be congested.



# Problem 2. misclassification

The impact of Facebook's flow into the video slice is likely to be marginal.
We need to take into account the differences between apps.

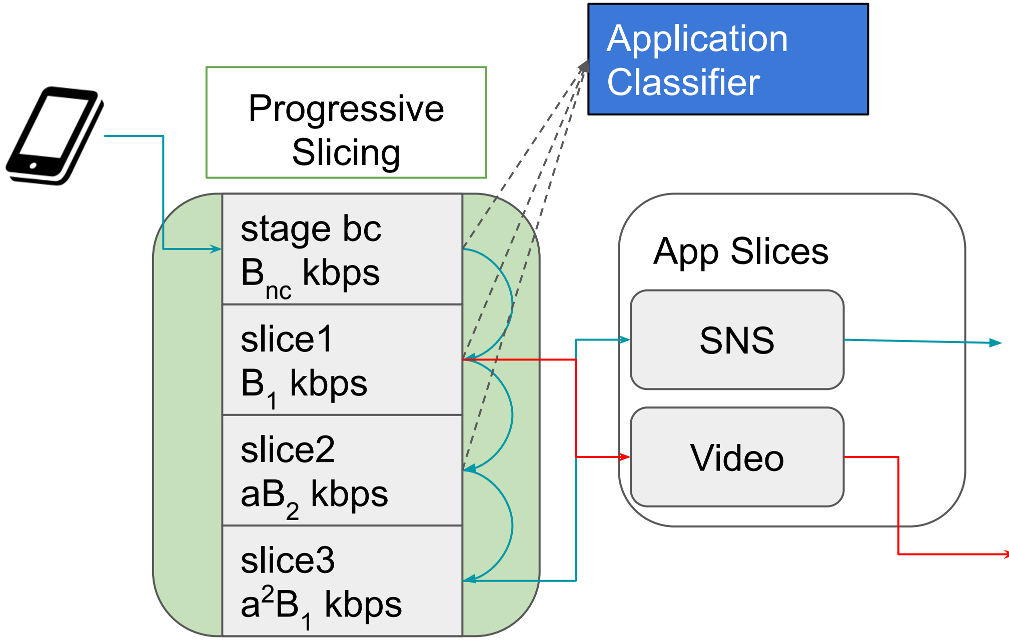


## Proposed Method: Progressive Slicing

Multiple classifications with multiple slices

Assigning flows to different slices during data collection and waiting for classification

Multiple classification allows us to decide where to allocate the flows according to the QoS of the app slice and the multiple results.



# Research Objective and Contributions

Problem Statement：

1. Classification delay causes severe degradation in QoE.
2. Single slice classification does not provide the flexibility to assign slices according to the misclassification and the QoS of app slices.

Proposed Method：

Progressive Slicing with flow isolation and multiple classification based on classification status.

Contributions：

We evaluate progressive slicing on the user scenarios in real applications.

For QoE metrics affected by classification delays (start-up time and search time),
we can reduce the QoE degradation compared to a single-slice application classification system.

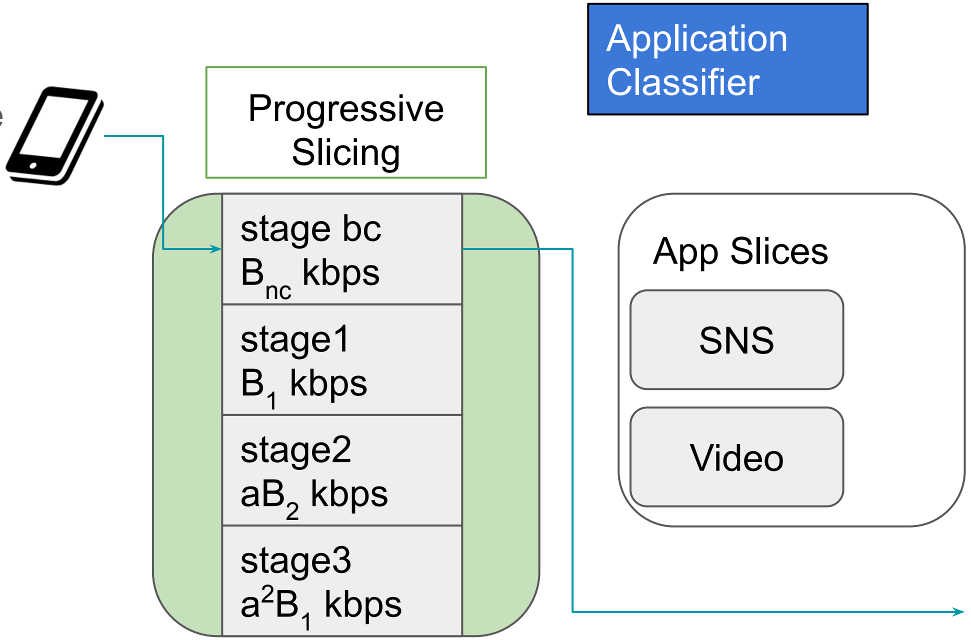
# Progressive Slicing

We assign the flow to the data collection slice (stage bc) until enough packets for the features are collected.

When collection is finished, we send a query to the classifier and assigns the flow to stage 1.
Slice-switching avoids classification delay.

The classifier returns a confidence score.

Confidence score exceeds the app slice threshold.
→ Assign the flow to the appropriate app slice.
Otherwise.
→ Move to stage 2



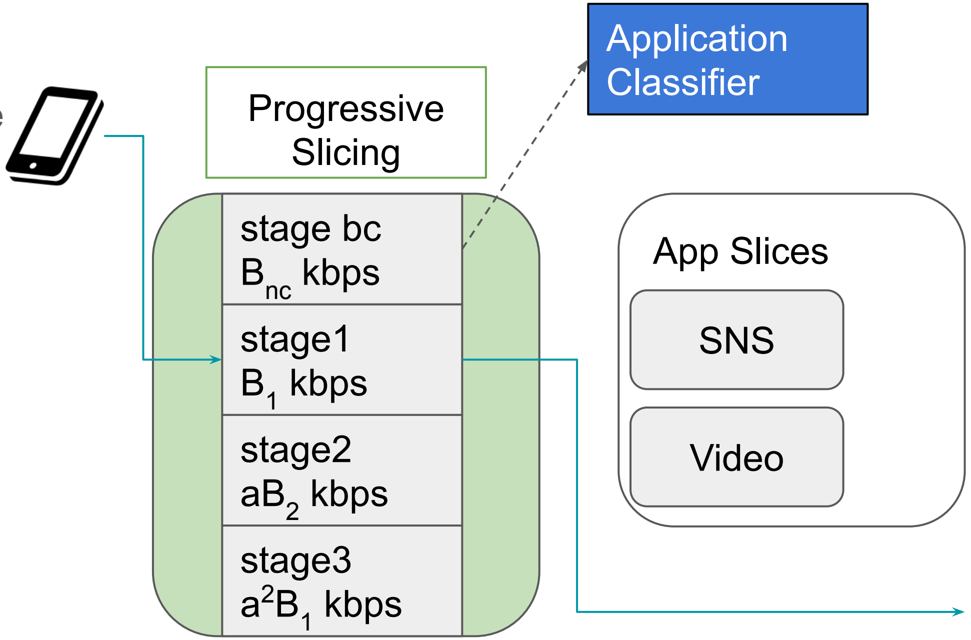
## Progressive Slicing

We assign the flow to the data collection slice (stage bc) until enough packets for the features are collected.

When collection is finished, we send a query to the classifier and assigns the flow to stage 1.
Slice-switching avoids classification delay.

The classifier returns a confidence score.

Confidence score exceeds the app slice threshold.
→ Assign the flow to the appropriate app slice.
Otherwise.
→ Move to stage 2



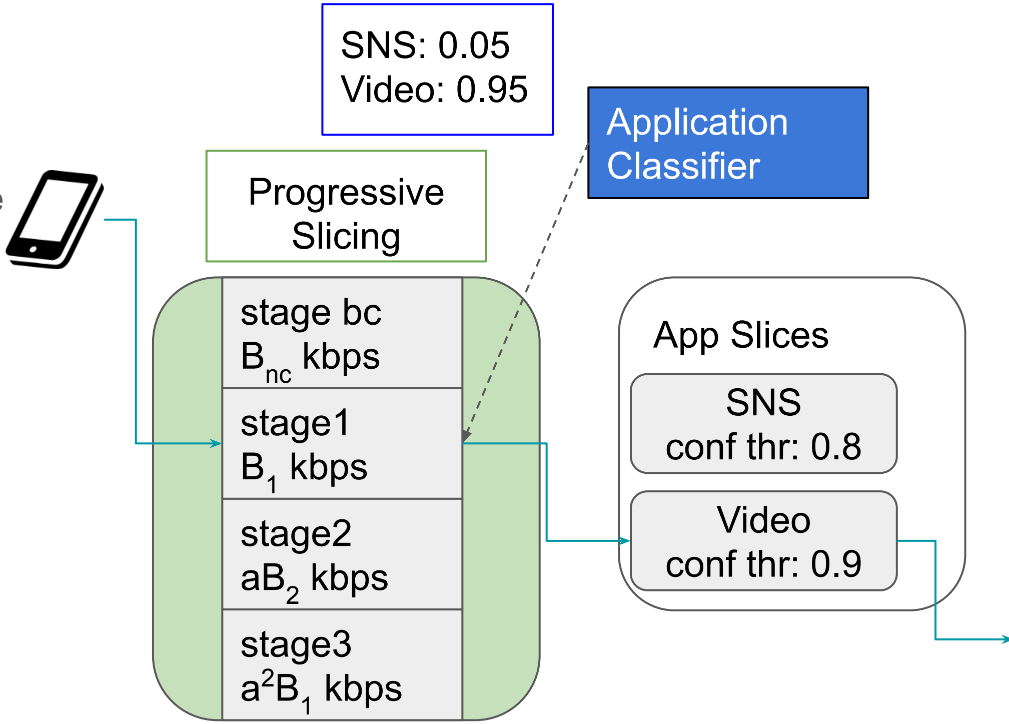
# Progressive Slicing

We assign the flow to the data collection slice (stage bc) until enough packets for the features are collected.

When collection is finished, we send a query to the classifier and assigns the flow to stage 1.
Slice-switching avoids classification delay.

The classifier returns a confidence score.

Confidence score exceeds the app slice threshold.
→ Assign the flow to the appropriate app slice.
Otherwise.
→ Move to stage 2



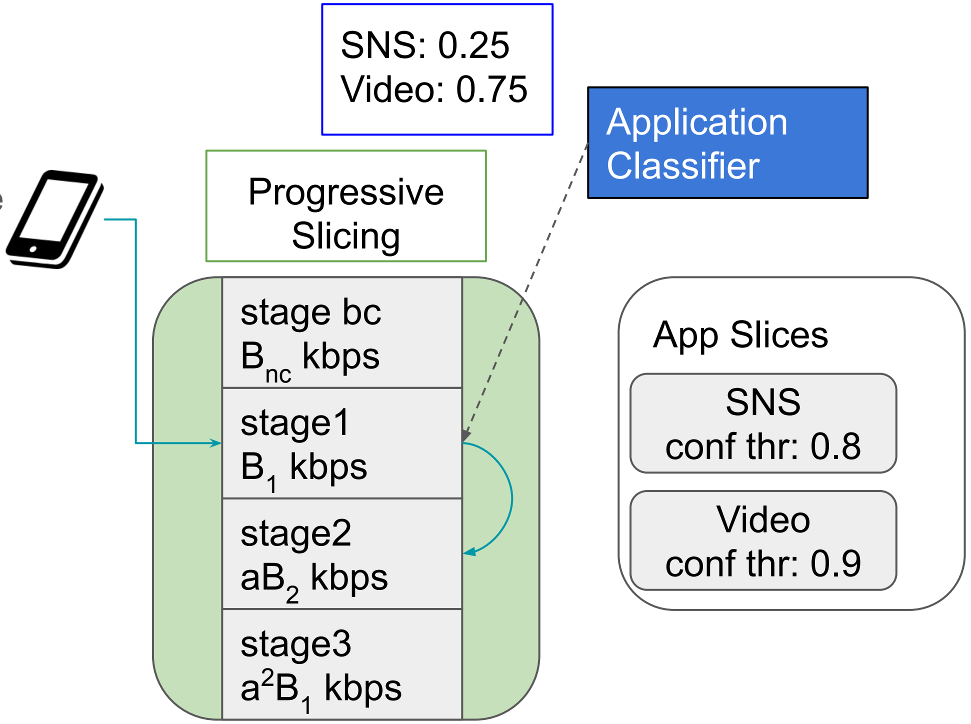
# Progressive Slicing

We assign the flow to the data collection slice (stage bc) until enough packets for the features are collected.

When collection is finished, we send a query to the classifier and assigns the flow to stage 1.
Slice-switching avoids classification delay.

The classifier returns a confidence score.

Confidence score exceeds the app slice threshold.
→ Assign the flow to the appropriate app slice.
Otherwise.
→ Move to stage 2



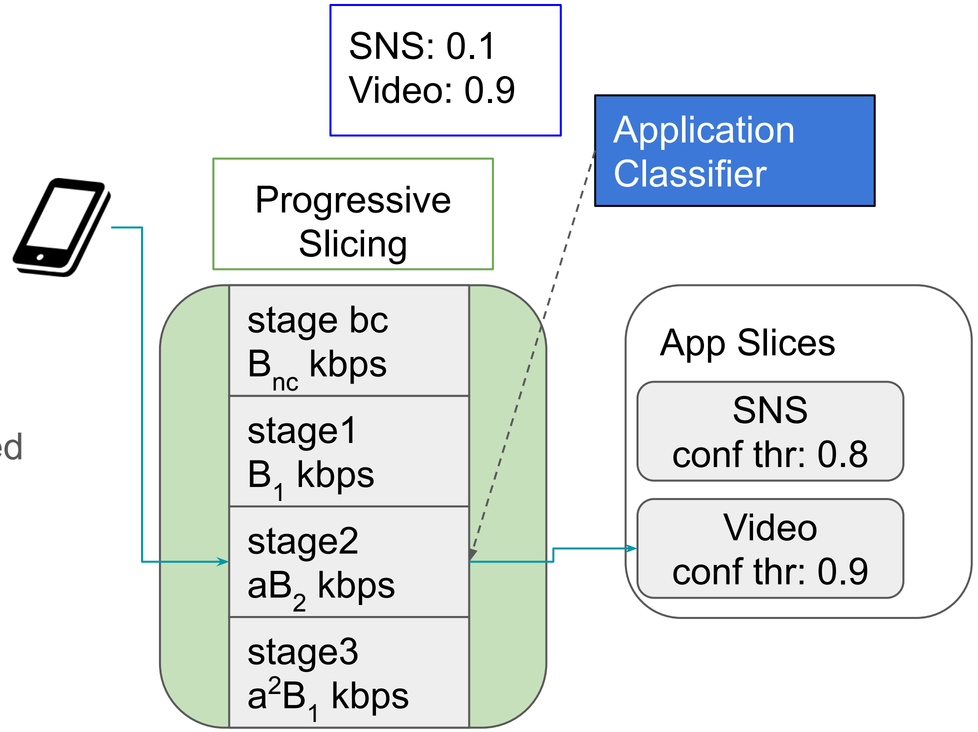
# Progressive Slicing

Update the confidence of the flow based on the second classification and the first classification.

$$C_{app} \leftarrow \mathbf{1 - (1 - C_2)(1 - C_1)}$$

$(1 - C)$: the prob that it is not a target app.
$\rightarrow (1 - C_2)(1 - C_1)$: the prob that the flow is classified as the target at least once

Reduce the negative impact by biased assigning



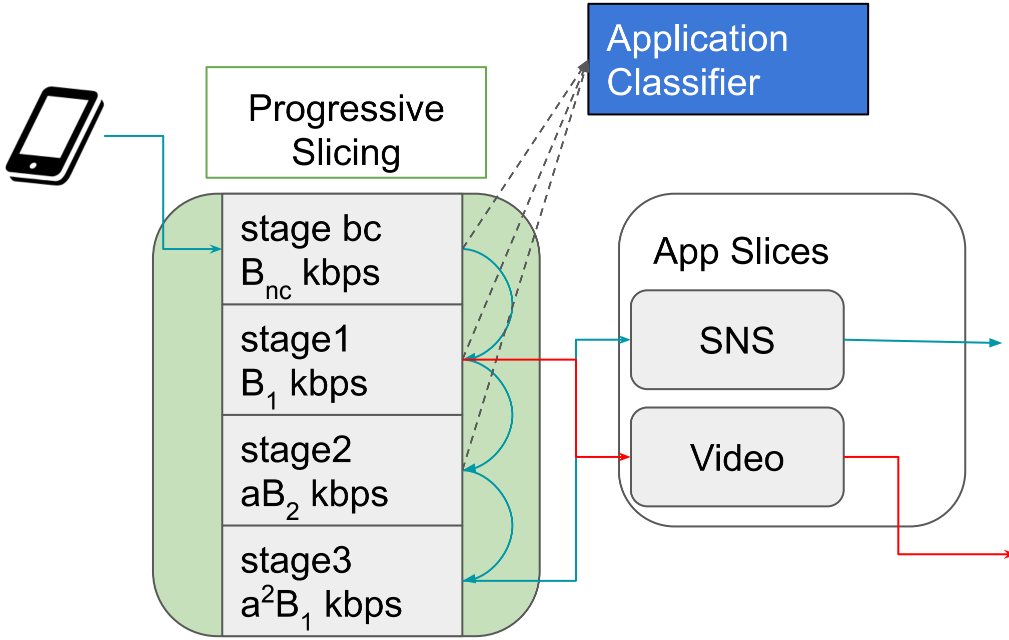
1st clf
SNS: 0.25
Video: 0.75

updated confidence
SNS: 1 - (1 - 0.1) * (1 - 0.25) = 0.325
Video: 1 - (1 - 0.9) * (1 - 0.75) = **0.975**

## Progressive Slicing

We allocate small bandwidth to the stage 1 slice and exponentially increased bandwidth to the consequent slices.
$\rightarrow B_1, aB_1, a^2B_1$……

We take a different allocation $B_{nc}$, since taking a small bandwidth for data collection will cause congestion.

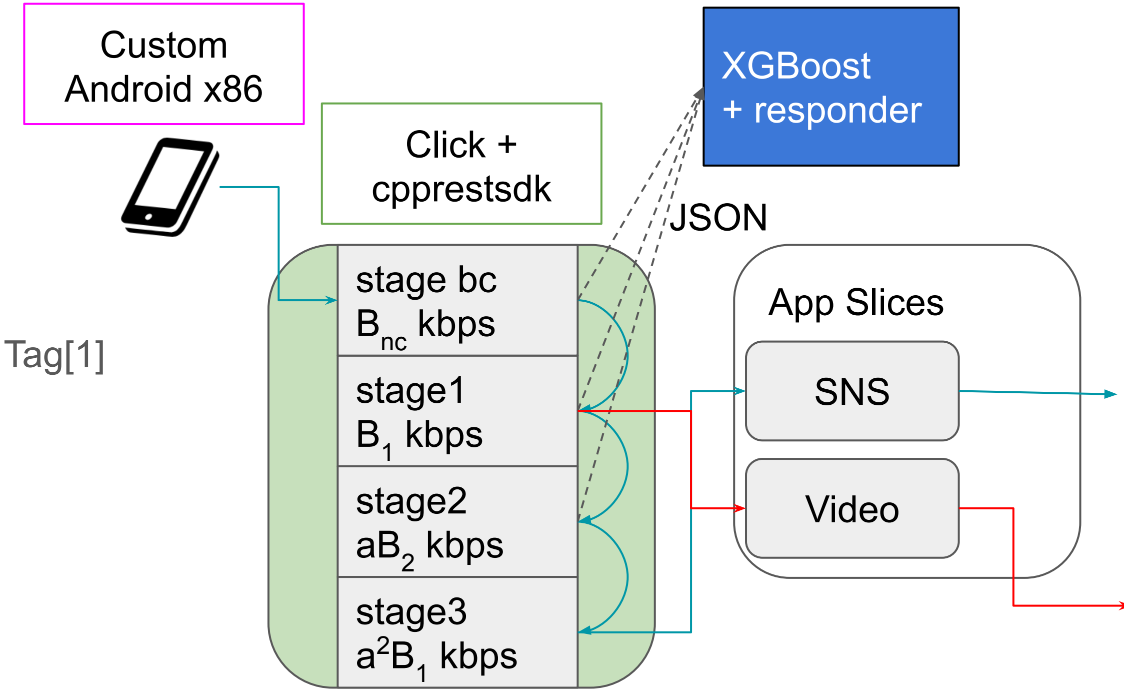


# Implementation

Progressive Slicing:
Click modular router + cpprestsdk

Application Classifier: responder+XGBoost

We modify Android x86 to send Application Tag[1]
→add process infos to TCP SYN
to make accurate ground-truth.

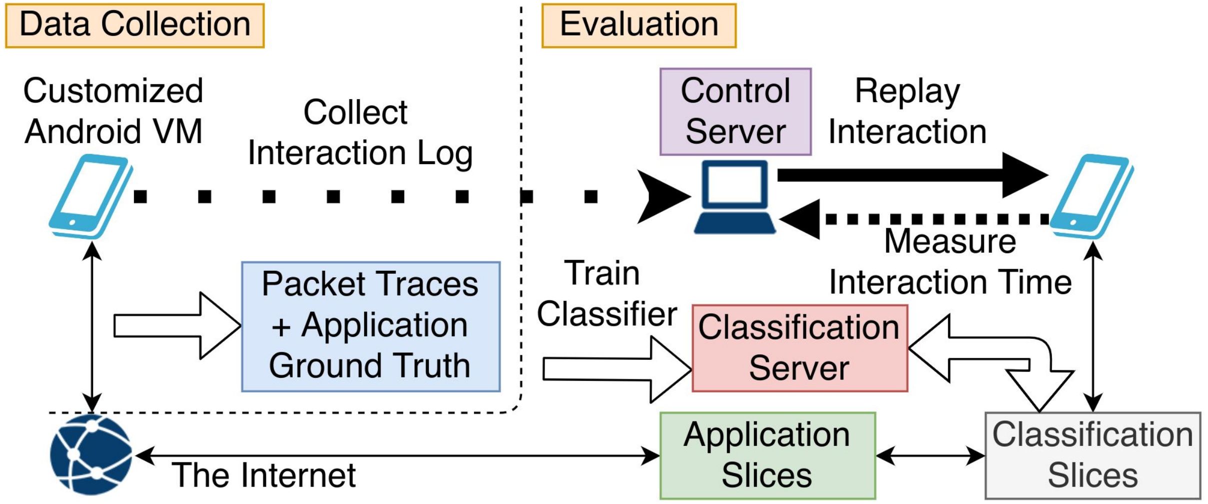


[1] Nakao, Akihiro, Ping Du, and Takamitsu Iwai. "Application specific slicing for MVNO through software-defined data plane enhancing SDN." *IEICE Transactions on Communications* 98.11 (2015): 2111-2120.

# Evaluation Flow

Modify View class in Android SDK used in QoEBox[1]
→We can record users' interaction.

We make scenario based on obtained interaction record.



[1] Nikravesh, Ashkan, et al. "QoE Inference and Improvement Without End-Host Control." *2018 IEEE/ACM Symposium on Edge Computing*. IEEE, 2018.

# Application Classifier

We collect the flows from VMs for training data.

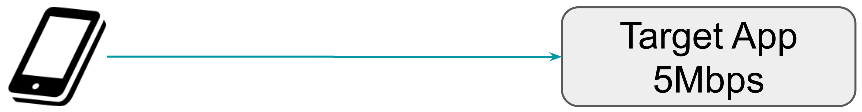
we use statistics on destination IP and packet length as features.
← packet arrival intervals are not useful due to the different bandwidth slices.

| Use Cases | Search on Twitter and BBC | ExoPlayer Video Streaming | Search on Twitter under ExoPlayer Traffic |
|---|---|---|---|
| features | dstIP, mean of packet length (both directions) | mean, std, median, skew, nunique of packet lengths (both directions) | dstIP, mean, std, median, skew, nunique of packet length (both directions) |
| #features | 6 | 8 | 12 |
| clf time per flow (ms) | 5.4 ± 6.5 | 12 ± 5.0 | 13 ± 6.5 |

# Baselines

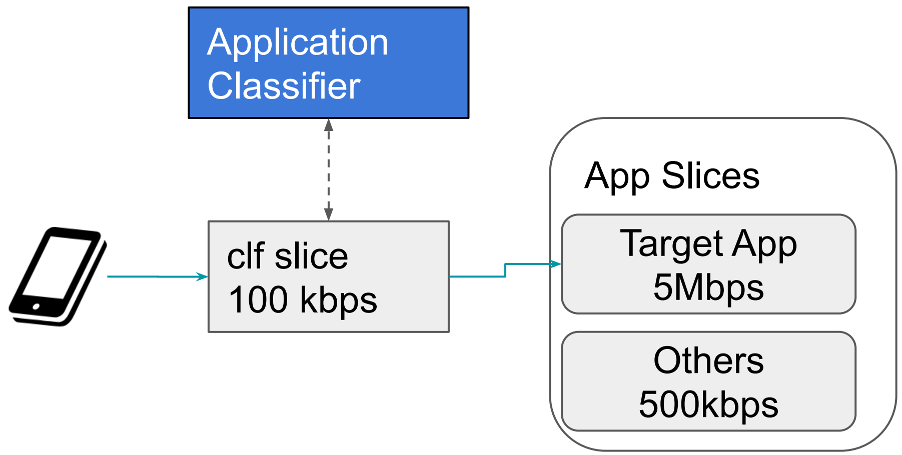
**ideal / non-classified**

All the flows are assigned to the 5Mbps slice.



**non-progressive**

single slice classification system.
We compare the progressive slicing to this.

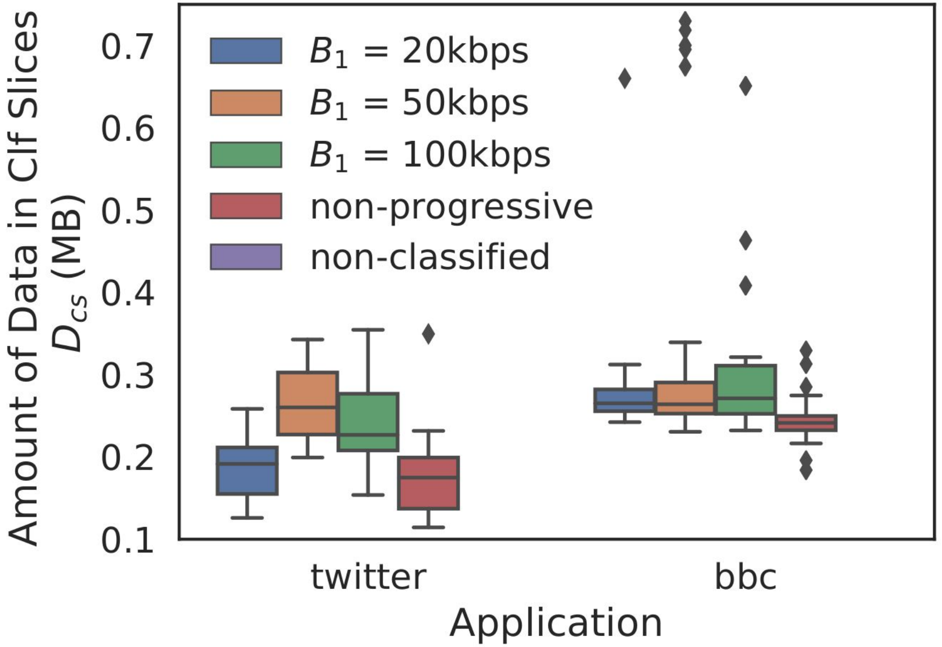


# Search on Twitter and BBC

The median value of non-progressive is 3%, while the median value of B1 is 4.1%, which is 1.4 times higher when $B_1$ is 100 kbps.

The impact of the increased number of classification slice is marginal.

Since progressive slices have slightly more bandwidth than the non-progressive, a relatively large amount of data is consumed in the classification slice.

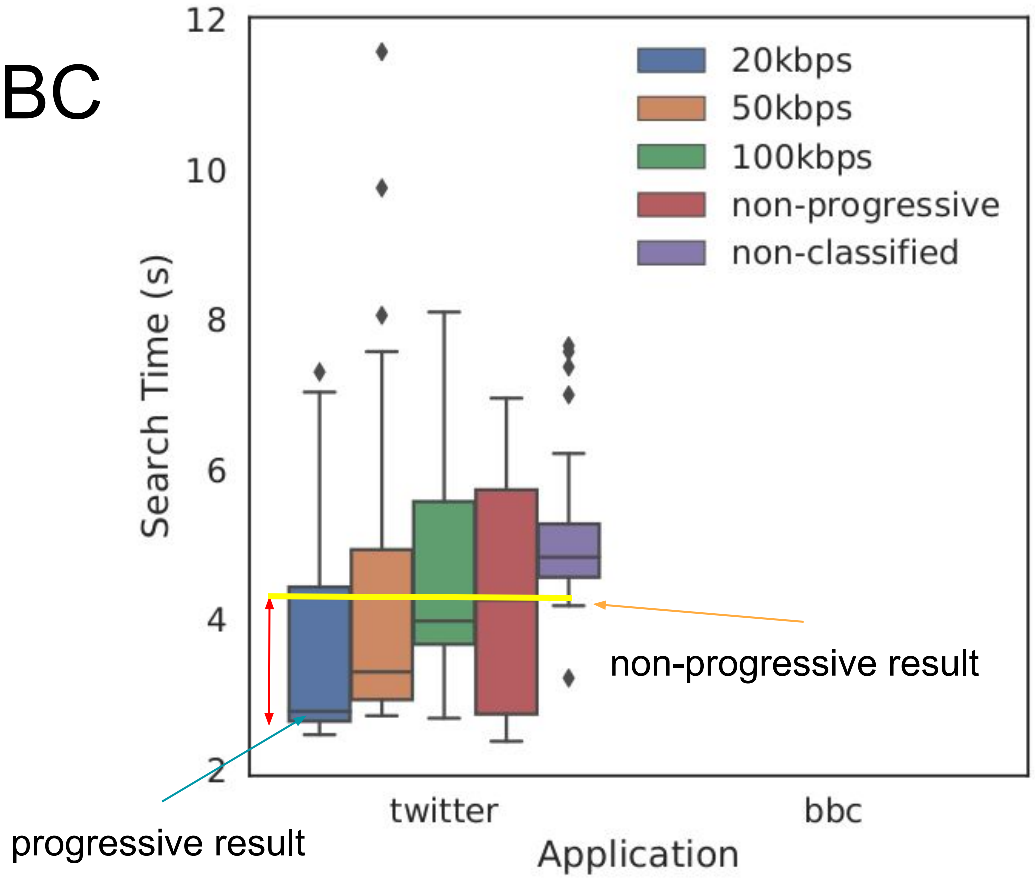


## Search on Twitter and BBC

In the Twitter search scenario,
We improve the search time at 35%.

We can reduce the degradation of user's QoE on Twitter.

There is no difference in the BBC search.
→The classification overhead is larger than the delay caused in the classification.

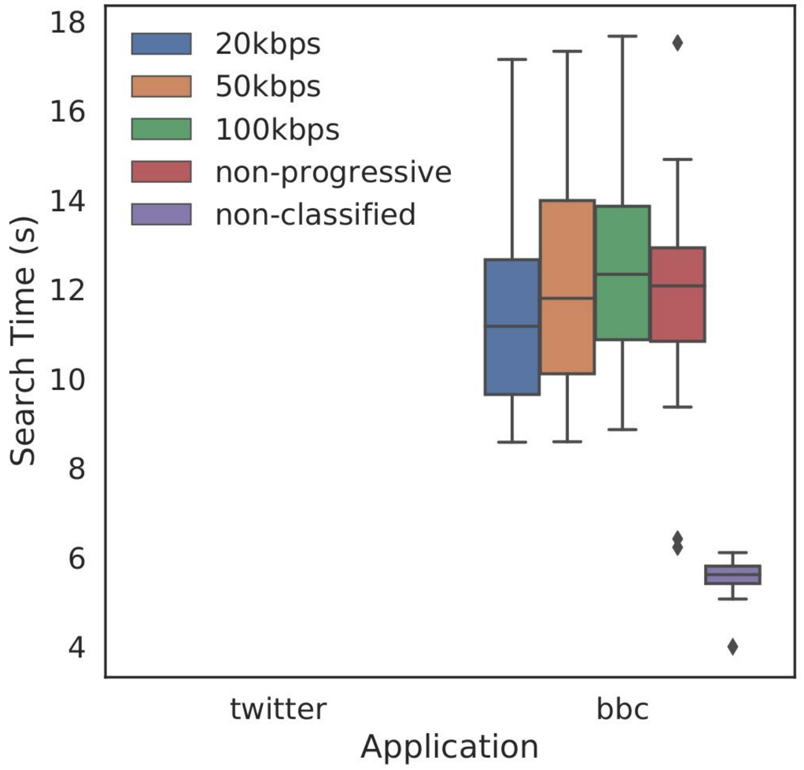


## Search on Twitter and BBC

In the Twitter search scenario,
We improve the search time at 35%.

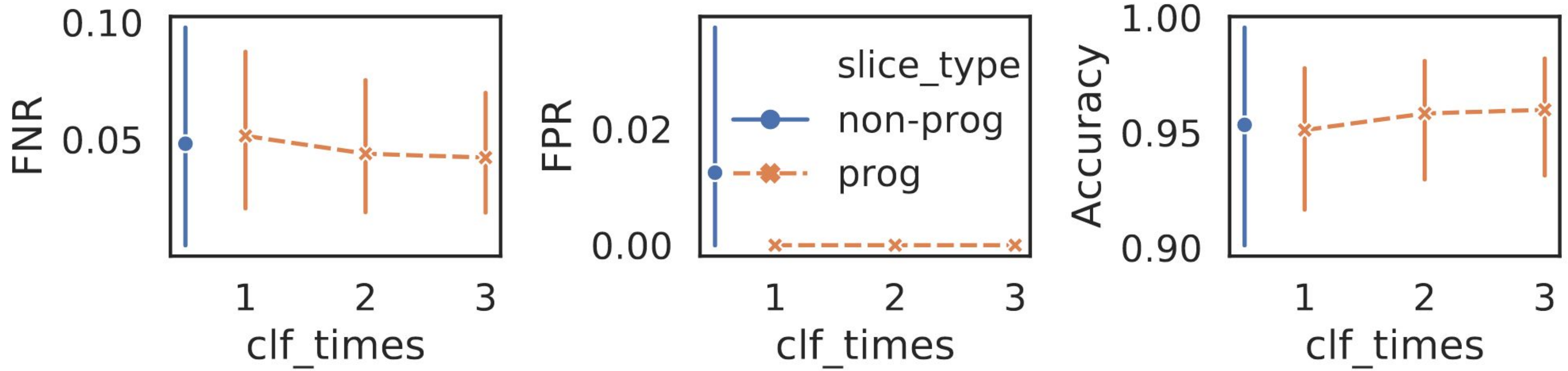
We can reduce the degradation of user's QoE on Twitter.

There is no difference in the BBC search.
→The classification overhead is larger than the delay caused in the classification.



## Search on Twitter and BBC

On BBC scenario, FNR decreases as the number of clf times increases.
→Accuracy is 0.7% better than the non-progressive mechanism
Recursive classifications improve the accuracy according to the policy.

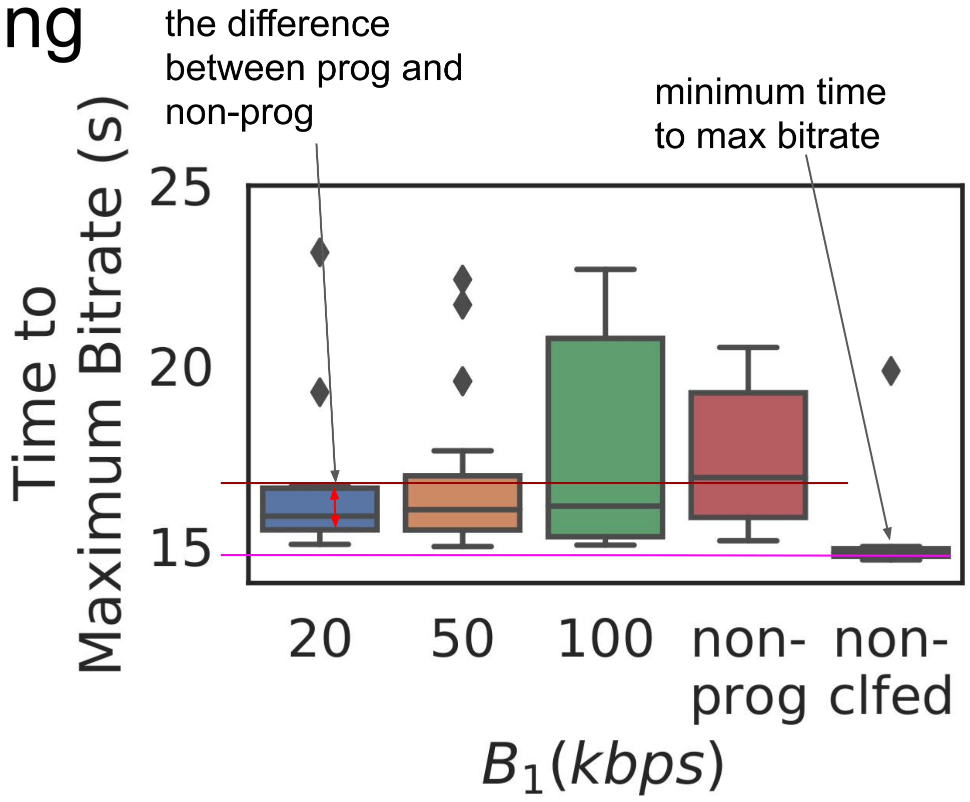


# ExoPlayer Video Streaming

We measure the time to the maximum bitrate.

*T = the difference between the non-classified median and the median of the other arch.*
progressive slicing improves T at 52%.

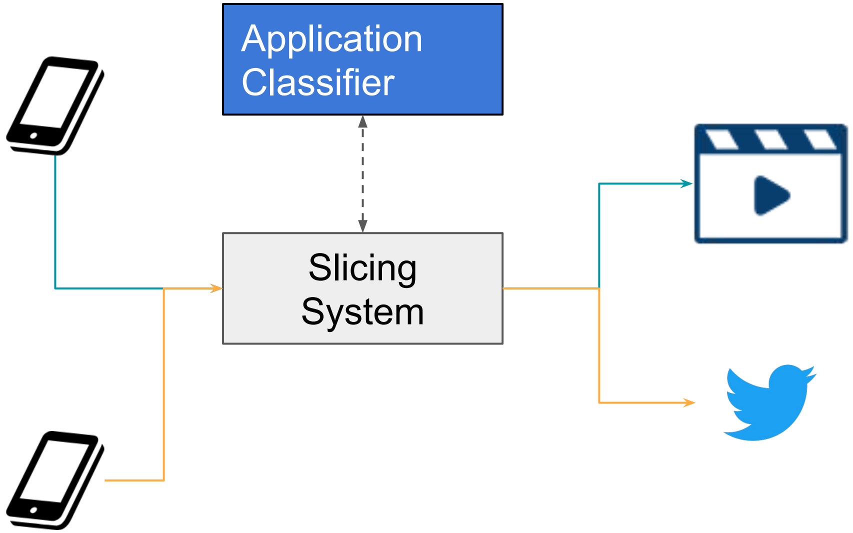
Progressive Slicing reduces the degradation of QoE on the high-bandwidth application



# Twitter Search under Video Traffic

High-bandwidth applications cause congestion of classification slice and degrades other apps' QoE.
→Investigate Twitter Search Time under Exoplayer video traffic.

Launch ExoPlayer and play video, then launch Twitter within 10 seconds.

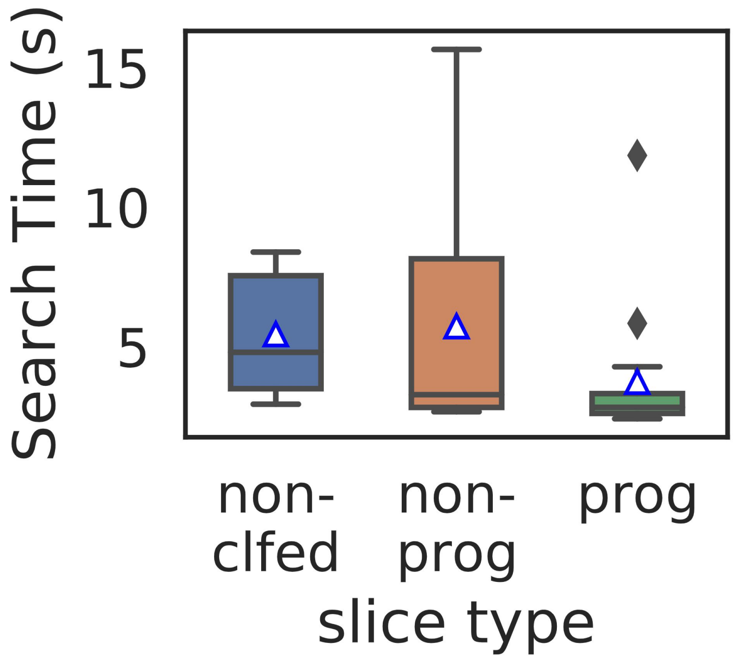


# Twitter Search under Video Traffic

High-bandwidth applications cause congestion of classification slice and degrades other apps' QoE.
→Investigate Twitter Search Time under Exoplayer video traffic.

Launch ExoPlayer and Twitter within 10 seconds.

We improve average Search Response Time at 31%.
We can reduce the tail latency significantly.

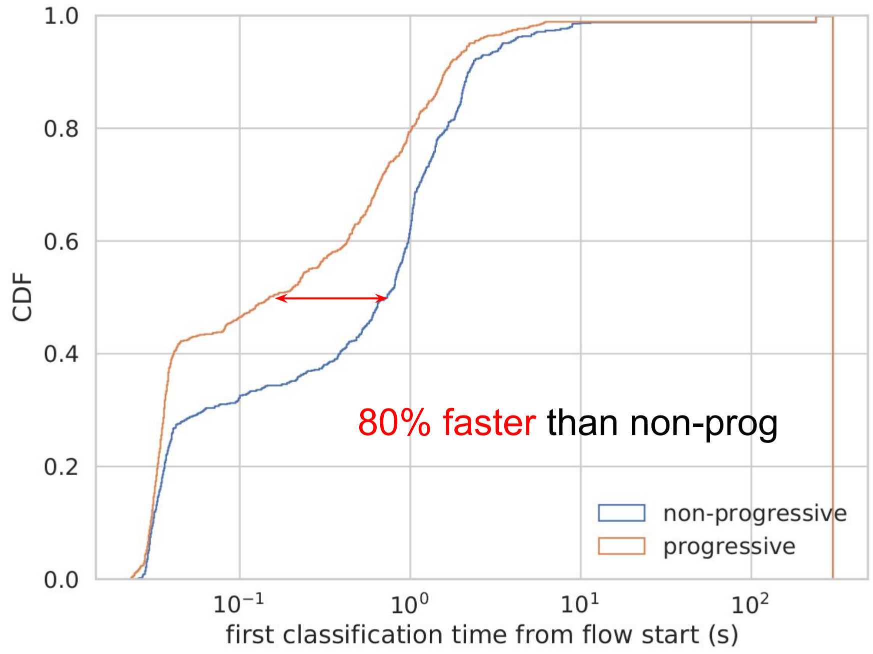


# Twitter Search under Video Traffic

High-bandwidth applications causes congestion of classification slice and degrades other apps' QoE.
→Investigate Twitter Search Time under Exoplayer video traffic.

Launch ExoPlayer and play video, then launch Twitter within 10 seconds.

median of first clf time from flow start:
non-prog: 0.74s
prog: 0.15s
→80% faster than non-prog
We can achieve faster isolation than non-prog.



## Research Objective and Contributions

Problem Statement：

1. Classification delay causes severe degradation in QoE.
2. Single slice classification does not provide the flexibility to assign slices according to the misclassification and the QoS of app slices.

Proposed Method：

Progressive Slicing with flow isolation and multiple classification based on classification status.

Contributions：

We evaluate progressive slicing on the user scenarios in real applications.

For QoE metrics affected by classification delays (start-up time and search time),

we can reduce the QoE degradation compared to a single-slice application classification system.

## Research Objective and Contributions

Proposed Method：

Progressive Slicing with flow isolation and multiple classification based on classification status.

Contributions：

We evaluate progressive slicing on the user scenarios in real applications.

For QoE metrics affected by classification delays (start-up time and search time),

we can reduce the QoE degradation compared to a single-slice application classification system.

Evaluation Results：

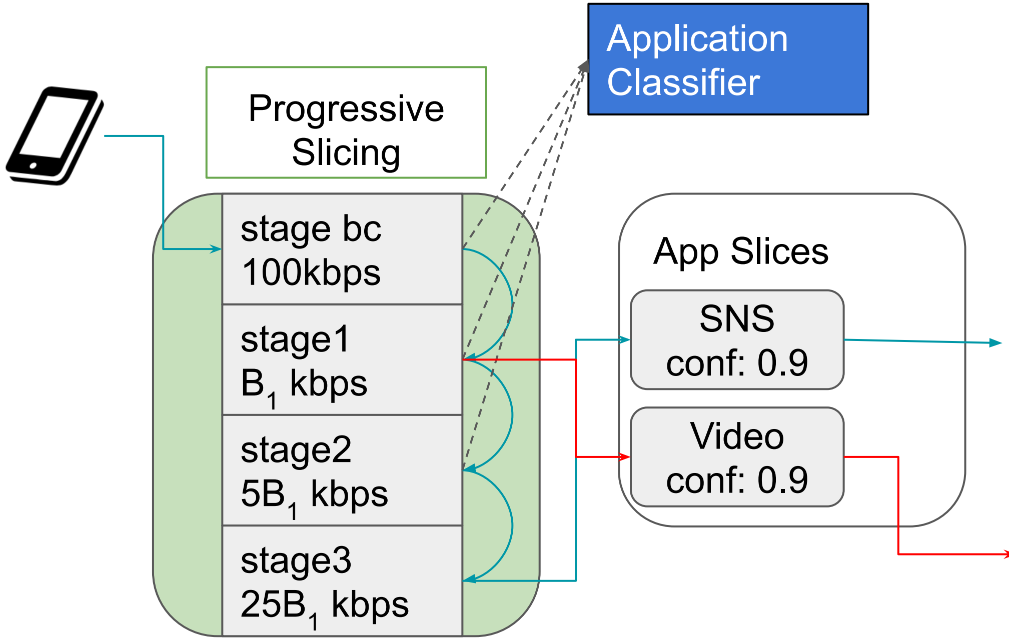
Compared to non-progressive architecture,

1. reduce the search response time on Twitter by 35%
2. reduce the delay to the maximum bitrate at 52% in the video streaming scenario
3. reduce search time at 31% when video traffic and Twitter traffic exists simultaneously

# Parameters for Evaluation

stage bc : 100kbps
a : 5
# stages : 3
confidence threshold : 0.9

We investigate
<u>response time on applications data in classification slice $D_{cs}$</u>
when changing $B_1$.



Application Classifier
Progressive Slicing
stage bc
100kbps
stage1
$B_1$ kbps
stage2
$5B_1$ kbps
stage3
$25B_1$ kbps
App Slices
SNS
conf: 0.9
Video
conf: 0.9