DEVIEW
2019

‘고품질 영상의 안정적 송출'
네이버 모바일 라이브 스트리밍

김정명
동영상플랫폼
NAVER

大 라이브 스트리밍의 시대

DEVIEW
2019

라이브의 한 축이 된 모바일 라이브

DEVIEW
2019

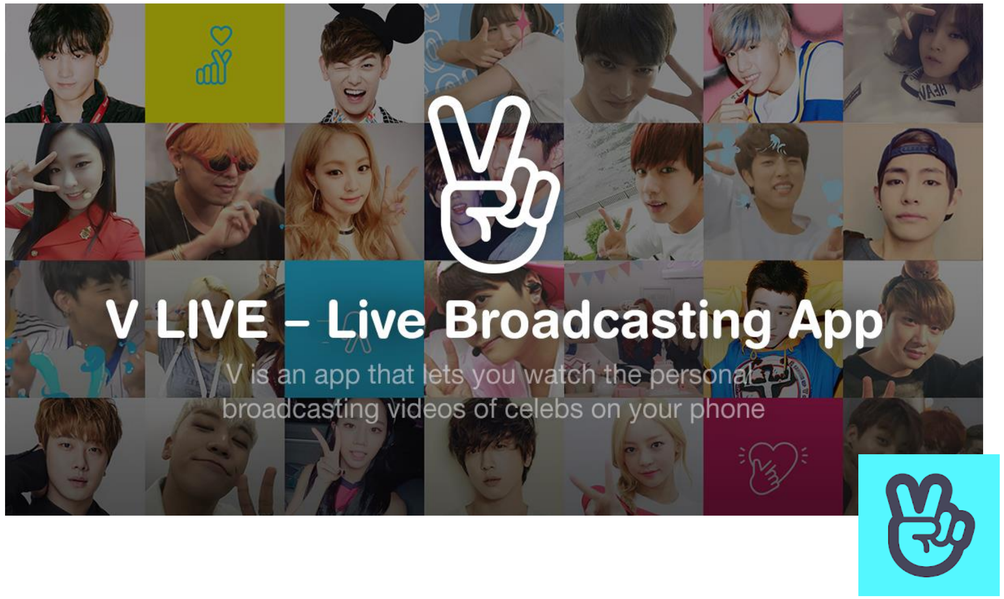
&

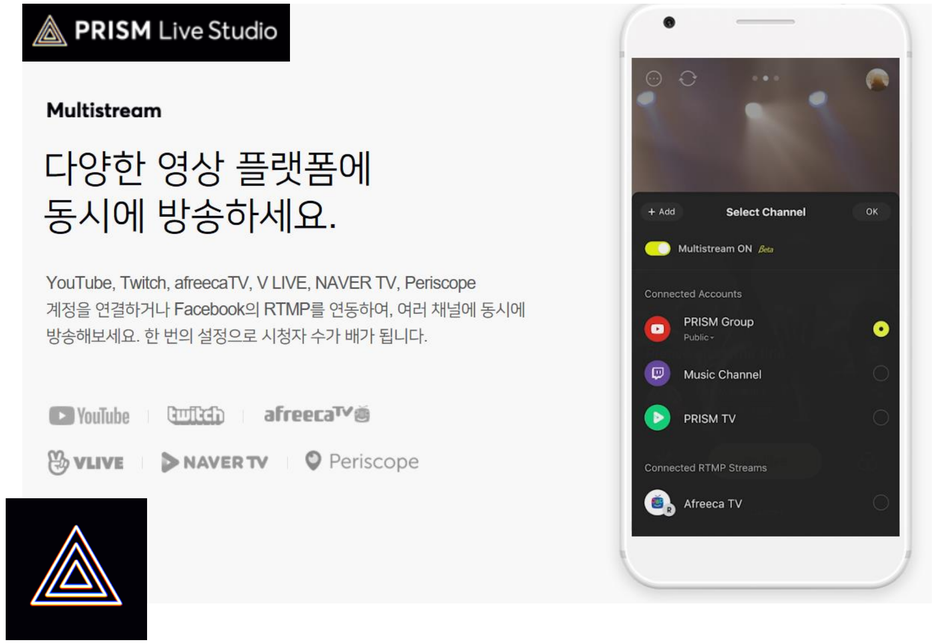
글로벌 라이브 스트리밍 서비스

DEVIEW
2019

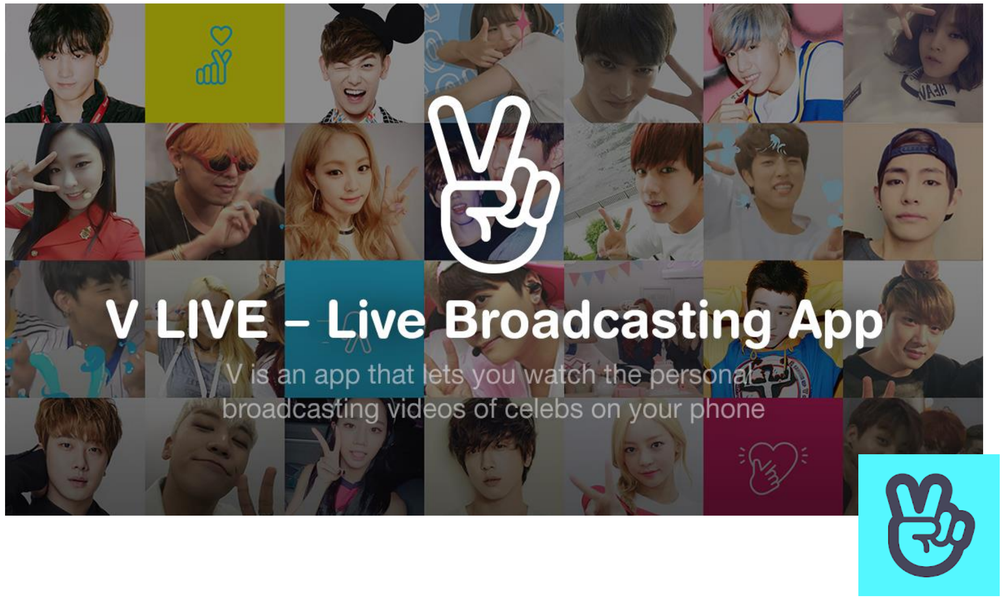
그리고 네이버

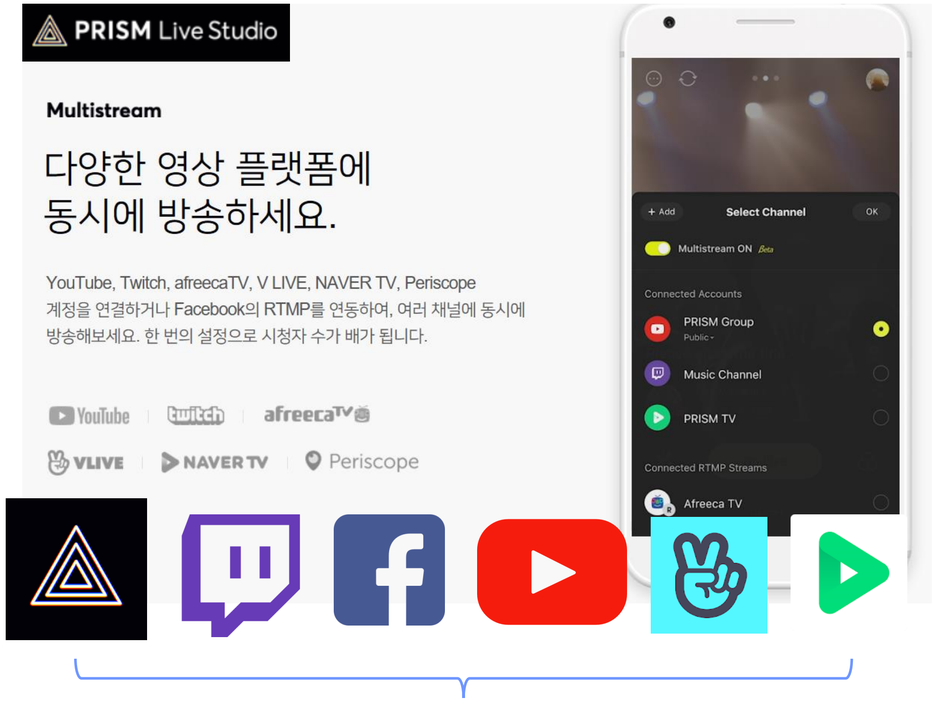
DEVIEW
2019





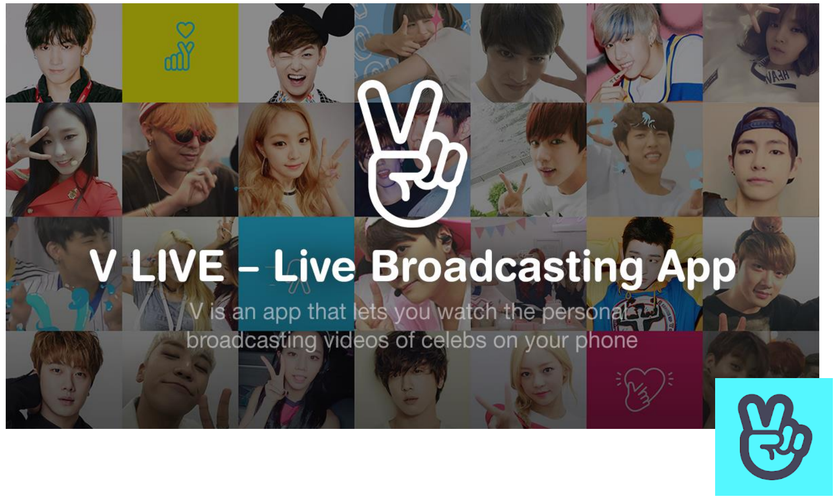
그리고 네이버

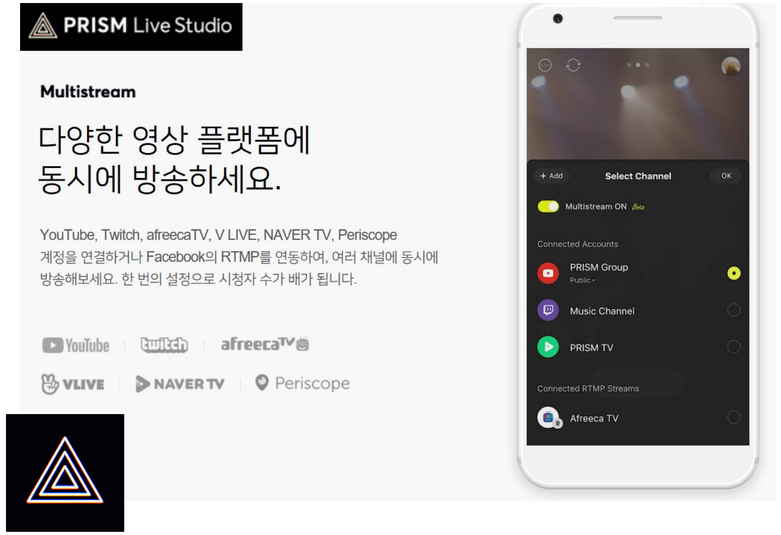




One Live - MultiStream 동시 송출!

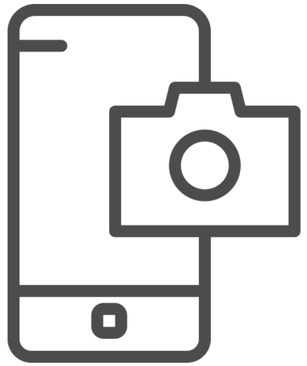
그리고 네이버



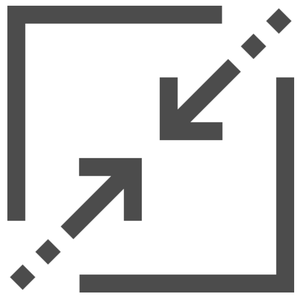


PRISM MOBILE LIVE SDK

사전 교양과목 이수



Camera
Capture



Encoding
H.264 / H.265
AAC

Streaming
RTMP / RTMPS

20분만에 만들어보는 라이브 방송 앱 (https://tv.naver.com/v/9329804/list/486582)

약간의 아쉬움

PRISM MOBILE LIVE SDK

공개하고, 모두와 다 같이 사용하고 싶습니다만 …

세 가지 실무적 주제

PRISM MOBILE LIVE SDK

모바일 라이브 스트리밍에 적합한 '권장 비트레이트 자동 검출 도구' 연구/개발

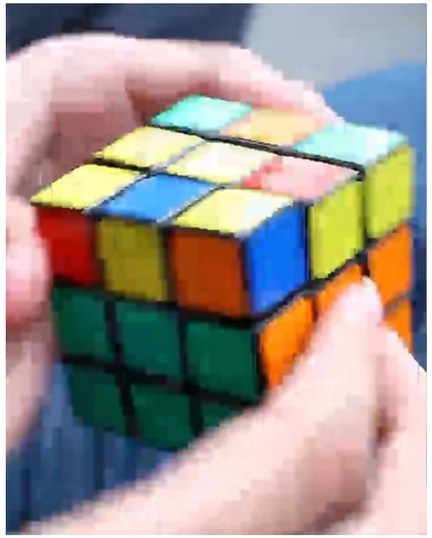
어디서나 원활한 송출을 가능하게 하는 'Adaptive Bitrate Publish'

다양한 Android 기기에서의 '안정적 동작 보장 노하우'

DEVIEW
2019

영상 품질의 장
권장 비트레이트 도출 자동화 도구

좋은 화질, 당연한 이야기



저화질

그러나 작은 크기

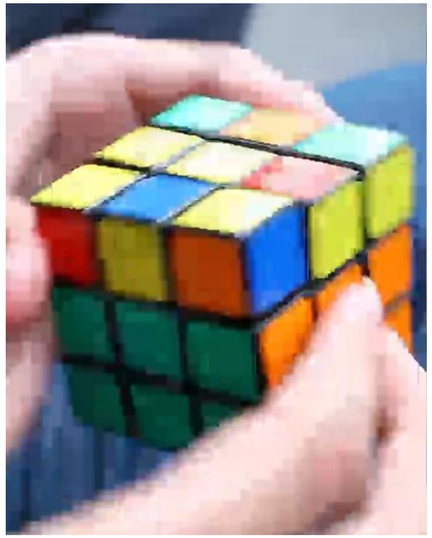
낮은 네트워크 부하

고화질

그러나 큰 크기

높은 네트워크 부하

모바일이란 이름의 무게



네트워크 품질의 상시 보장이 어려운 모바일 환경 고려

화질 대 크기 비가 우수한 권장 비트레이트의 결정이 필요

여기서 문제

H.264 + 1080p + CBR + High Profile

송출 시작 권장 비트레이트는 얼마로 결정하면 될까요?

여기서 문제

H.264 + 1080p + CBR + High Profile

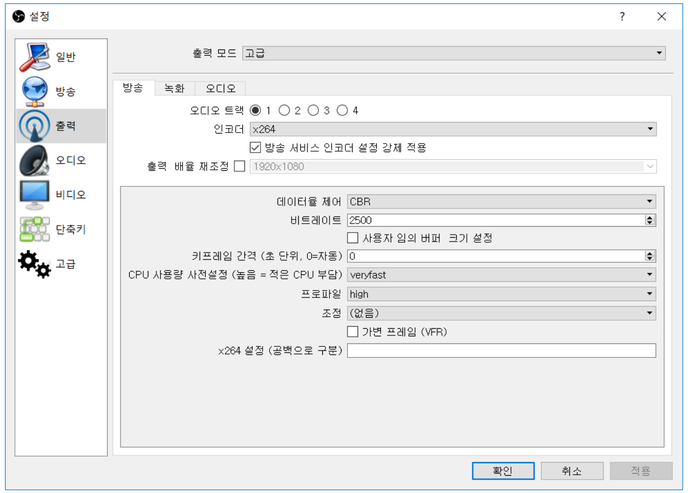
송출 시작 권장 비트레이트는 얼마로 결정하면 될까요?

그러게요..

전통적 비트레이트 결정의 방법

전통적 비트레이트 결정의 방법

DEVIEW 2019



	MOS	PSNR	SSIM	SSIMPLUS	VMAF
Predictive value	Best	Relatively low	Slightly higher	Much better	Much better
Basis	Subjective	Math	Math	Math + ML	Math + ML
Reference	Can be	Yes	Yes	Yes	Yes
Scoring	1–5	0–100	0–1	0–100	0–100
No artifact threshold	NA	45 dB	0.99	100	93
Artifacts likely present	NA	35 dB	0.5	NA	NA
Interpreting scores					
Excellent	5	45+	0.99 +	80–100	80–100
Good	4	38	0.95–0.99	60–80	60–80
Fair	3	30	0.88–0.98	40–60	40–60
Poor	2	24	0.50–.088	20–40	20–40
Bad	1	< 15	< .5	< 20	< 20
Just-noticeable difference	NA	NA	NA	NA	6
Device ratings	No	No	No	Multiple	Standard, Phone, 4K
Ownership	Open source	Open source	Open source	Proprietary	Open source

https://www.streamingmedia.com/Articles/ReadArticle.aspx?ArticleID=130675

조건이 다양하다

우리는 주로 셀럽들만 사용하고, 고해상도 위주인데요?
해외 라이브는 360p 도 고려해야 해요.

우리는 누가 어떤 해상도로 어떻게 쓸지 아무도 모릅니다.
그러니까 해상도 별로 다 결정해주세요.

우리의 목표

개발자답게, 실험과 결과, 근거를 통해 결정하자

= '영상 품질 비교를 토대로 한, 권장 비트레이트 결정 도구'를 개발하자

우리의 목표

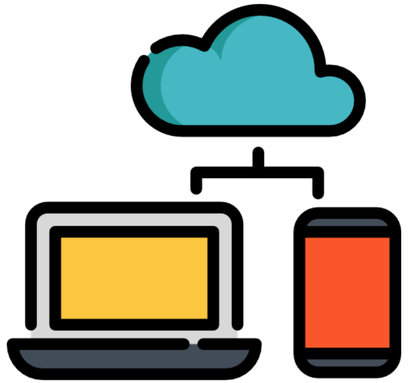
Resolution

Complexity

Codec

Profile

...



우리의 목표

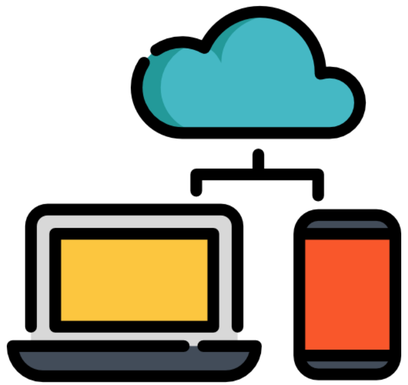
Resolution

Complexity

Codec

Profile

...



다양한 품질비교 결과
권장 비트레이트
도출

몇 가지 추가 고려사항

당연하다고 여겨지는 사실에 대한 의심

- 같은 영상, Codec, Bitrate, Profile 일 경우 단말별로 화질 차이가 있을까?
- 인코딩 테스트는 실제 모바일 단말에서 수행한다

몇 가지 추가 고려사항

당연하다고 여겨지는 사실에 대한 의심

- 같은 영상, Codec, Bitrate, Profile 일 경우 단말별로 화질 차이가 있을까?
- 인코딩 테스트는 실제 모바일 단말에서 수행한다

영상 종류에 따른 비트레이트-품질 간 관계 비교

- 서로 다른 복잡도의 영상별 비트레이트 투자 대비 화질 향상률 측정

몇 가지 추가 고려사항

당연하다고 여겨지는 사실에 대한 의심

- 같은 영상, Codec, Bitrate, Profile 일 경우 단말별로 화질 차이가 있을까?
- 인코딩 테스트는 실제 모바일 단말에서 수행한다

영상 종류에 따른 비트레이트-품질 간 관계 비교

- 서로 다른 복잡도의 영상별 비트레이트 투자 대비 화질 향상률 측정

권장 비트레이트 와 동시에, 체감적으로 유의미한 min/max 값 도출

- 비트레이트를 동적으로 변경하는 Adaptive Bitrate Publish 동작 구간 설정 근거로 활용

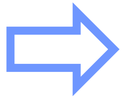
테스트 도구 시나리오

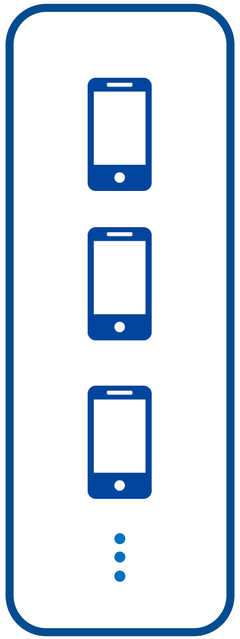
해상도, 복잡도 별
영상 원본

- 1080p YUV Low-complex, High-complex (10~15sec) ~2GB
- 720p YUV Low-complex, High-complex (10~15sec) ~4GB

테스트 도구 시나리오

해상도, 복잡도 별
영상 원본





nMobile 이용
단말 동시 트랜스코딩

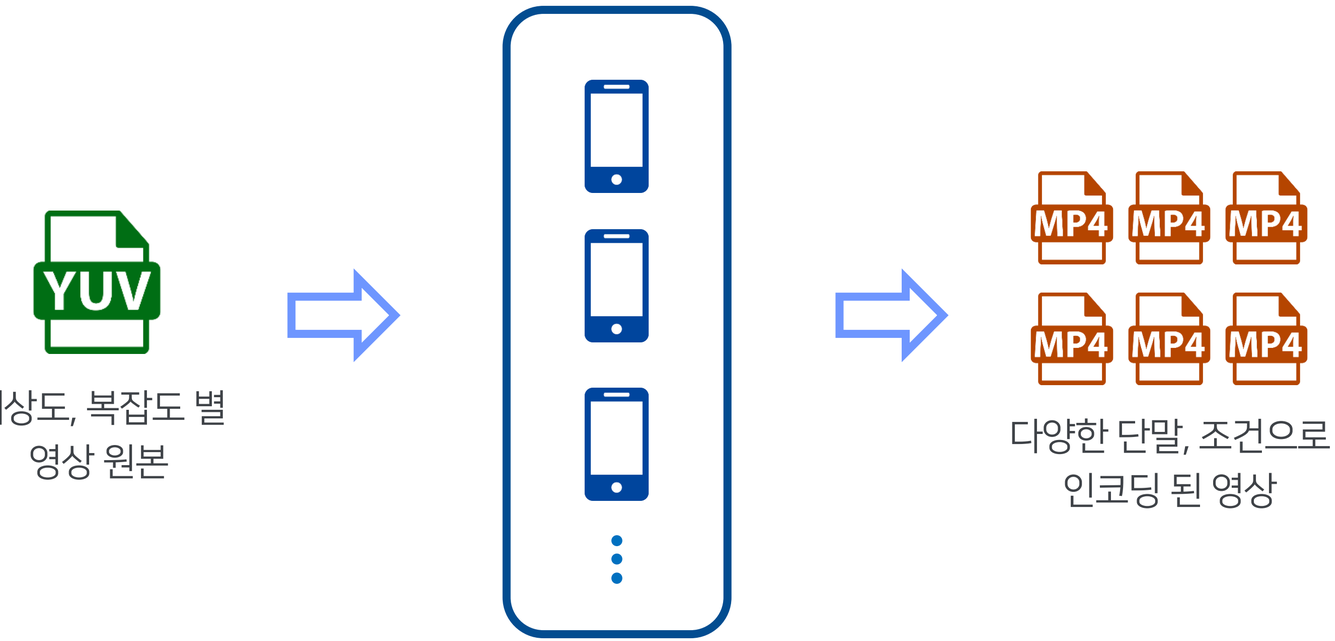
nMobile?

AWS Device Farm

Firebase Test Lab

유사한 성격의 네이버 사내 서비스

테스트 도구 시나리오

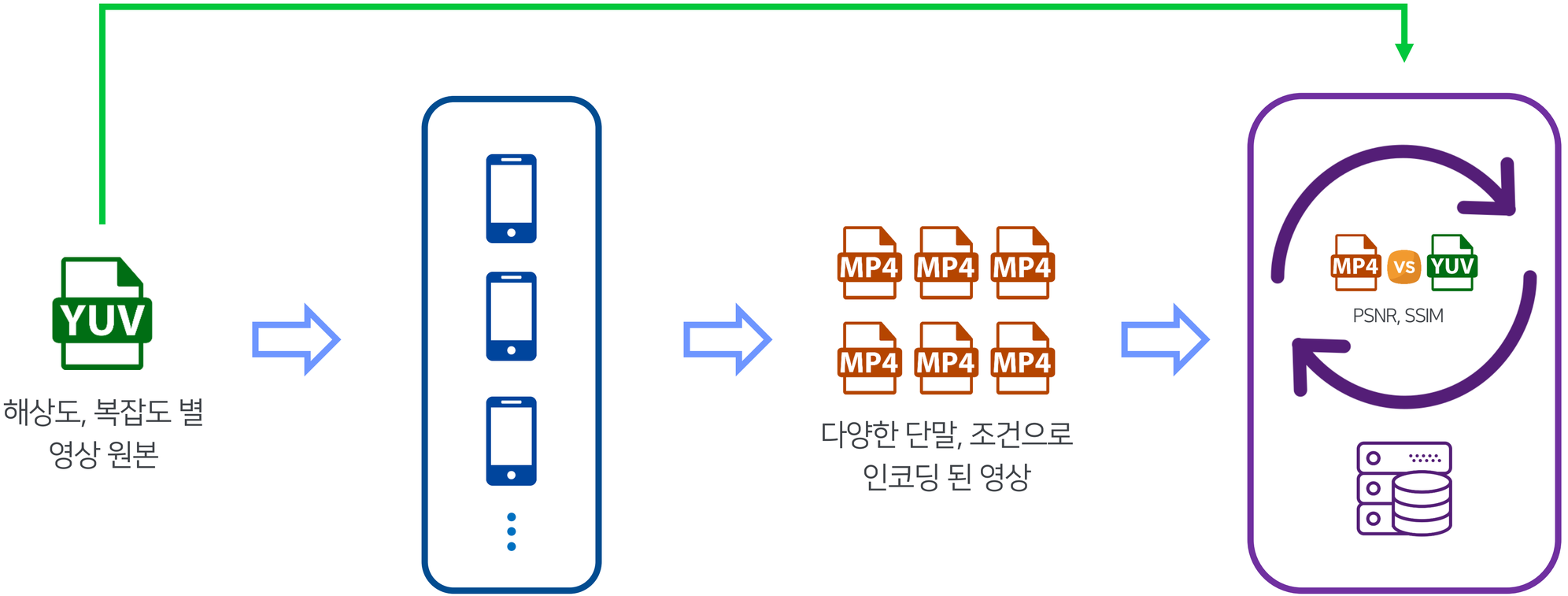


해상도, 복잡도 별
영상 원본

nMobile 이용
단말 동시 트랜스코딩

다양한 단말, 조건으로
인코딩 된 영상

테스트 도구 시나리오



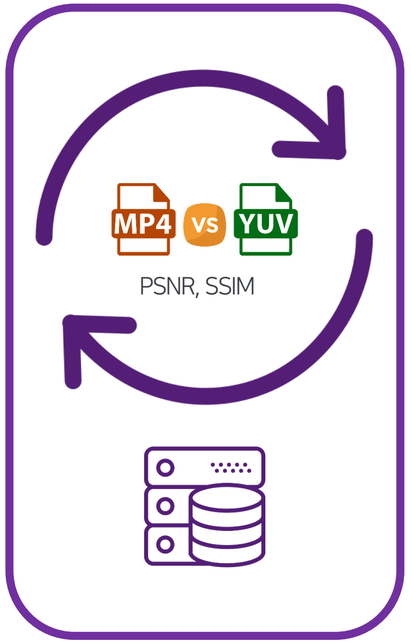
해상도, 복잡도 별
영상 원본

nMobile 이용
단말 동시 트랜스코딩

다양한 단말, 조건으로
인코딩 된 영상

영상 간 품질 비교
결과 도출

비교측정 방법



측정 방법

- 원본 영상과 비트레이트별 인코딩 영상의 QoE Metric 측정
- 1sec 단위로 탐색 하며, PSNR, SSIM 측정 비교
- Video Bitrate (500Kbps ~ 2Mbps), bitrate step = 500Kbps

인코딩 패러미터

- H.264 (High Profile 3.1 Level, GOP 1sec, 30FPS)

QoE Metric

DEVIEW 2019

인코딩 품질 비교와 평가를 위한 QoE metric 선정

1. 최대신호 대 잡음비 비교를 위한 PSNR

$$MSE = \frac{1}{m\,n}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

The PSNR (in dB) is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \\ &= 20 \cdot \log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \\ &= 20 \cdot \log_{10}(MAX_I) - 10 \cdot \log_{10}(MSE) \end{aligned}$$

2. 원본 영상 대비 구조적 유사도 측정을 위한 SSIM

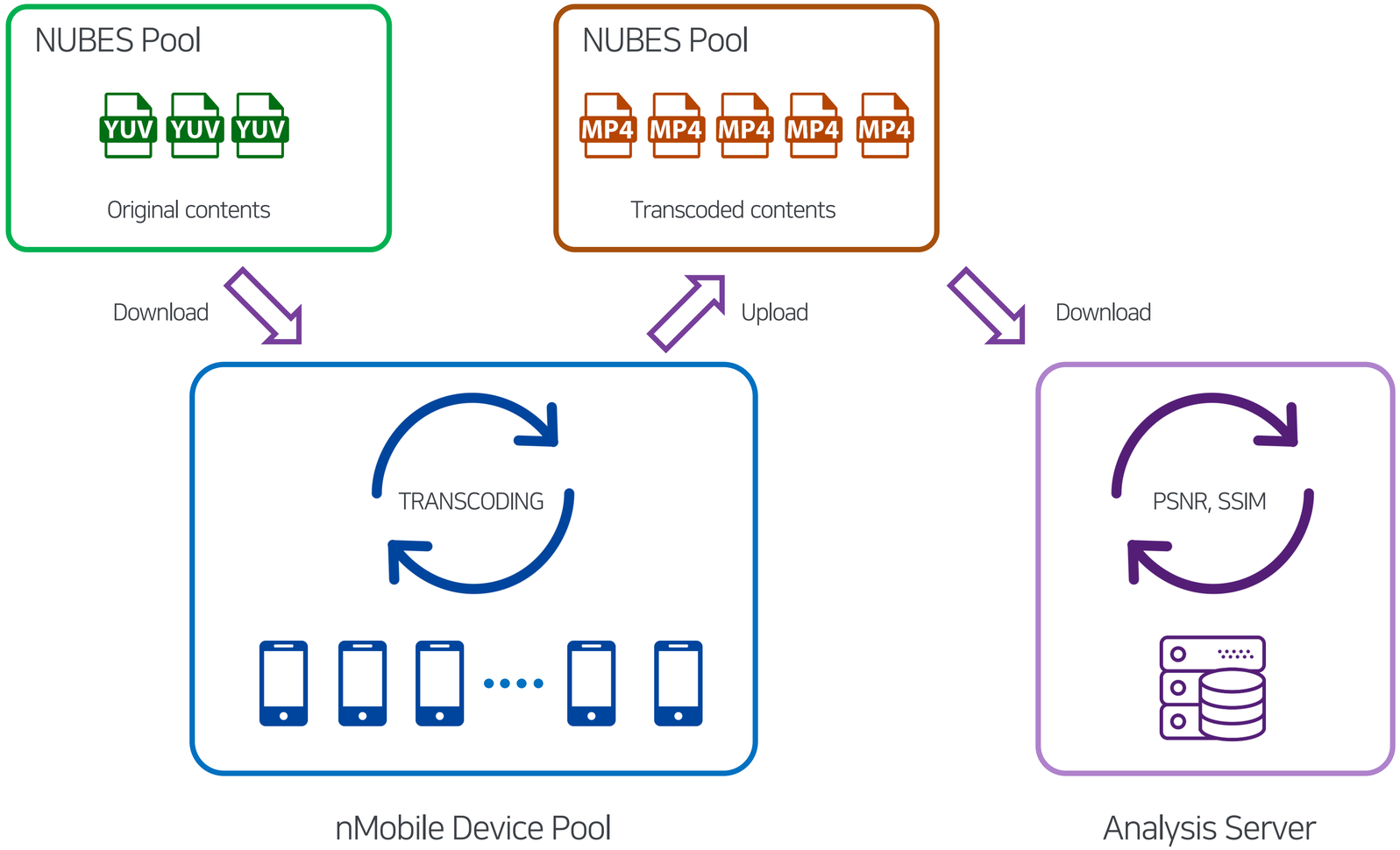
The SSIM index is calculated on various windows of an image. The measure between two windows x and y of common size *N*×*N* is:[3]

$$\mathrm{SSIM}(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

with:

- μ_x the average of x;
- μ_y the average of y;
- σ_x^2 the variance of x;
- σ_y^2 the variance of y;
- σ_{xy} the covariance of x and y;
- $c_1 = (k_1 L)^2$, $c_2 = (k_2 L)^2$ two variables to stabilize the division with weak denominator;
- L the dynamic range of the pixel-values (typically this is $2^{\#bits\ per\ pixel} - 1$);
- $k_1 = 0.01$ and $k_2 = 0.03$ by default.

완성 도구 구조도


NUBES Pool
YUV
YUV
YUV
Original contents
NUBES Pool
MP4
MP4
MP4
MP4
MP4
Transcoded contents
Download
Upload
Download
TRANSCODING
PSNR, SSIM
nMobile Device Pool
Analysis Server

만들었으니 마음껏 써보자!

DEVIEW
2019

우선 다양한 조건의 품질 비교 실험부터 무제한으로 즐겨보자

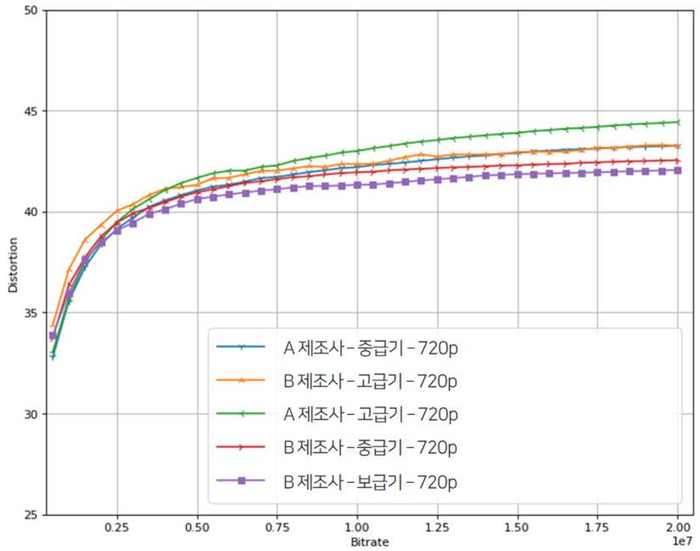
첫 번째 원초적 궁금증

DEVIEW
2019

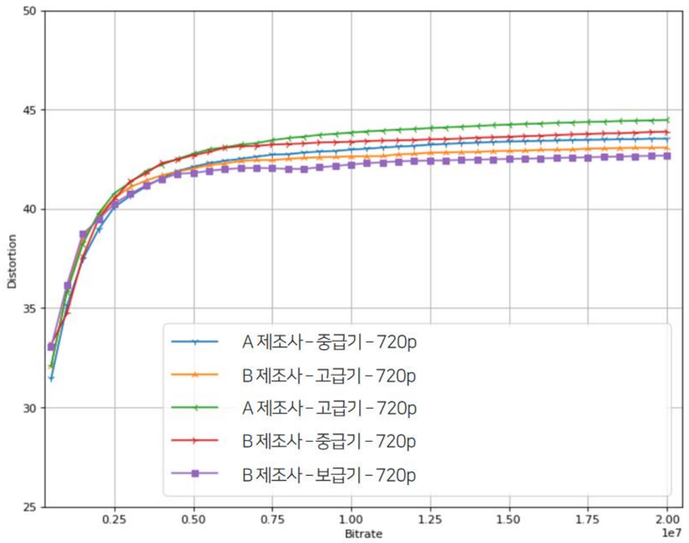
동일 영상, 동일 인코딩 조건에서, 모바일 단말간의 인코딩 화질 차이가 있을까?

이미지 출처 : https://newatlas.com/best-smartphones-specs-features-comparison-2017/49418/

Device 별 품질 차이 - 720p PSNR



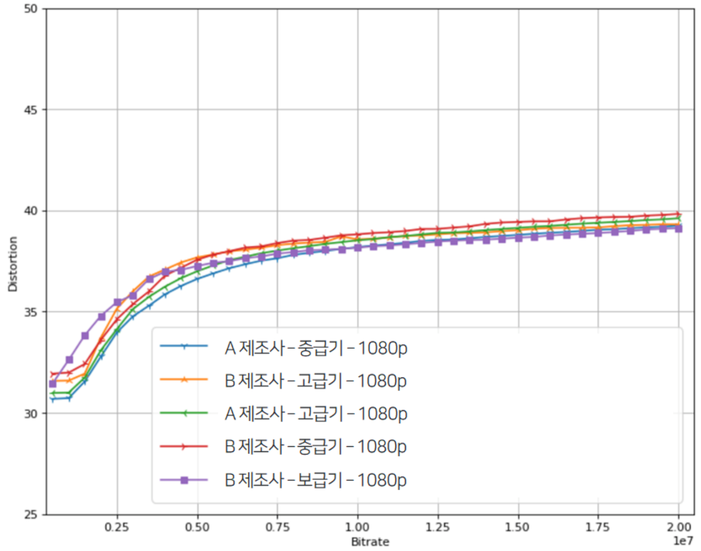
Low-Complex



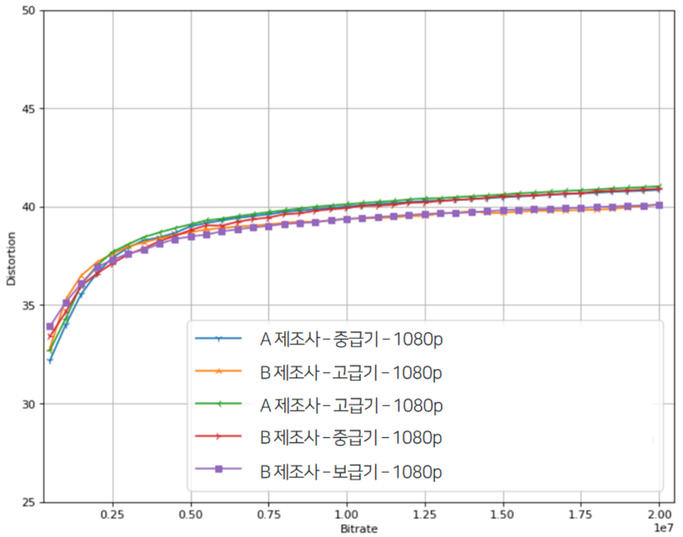
High-Complex

1. Device 간 PSNR 은 약간의 차이는 있으나 크지 않다
2. Complexity 가 높은 영상일 수록 차이가 더 적은 편

Device 별 품질 차이 - 1080p PSNR



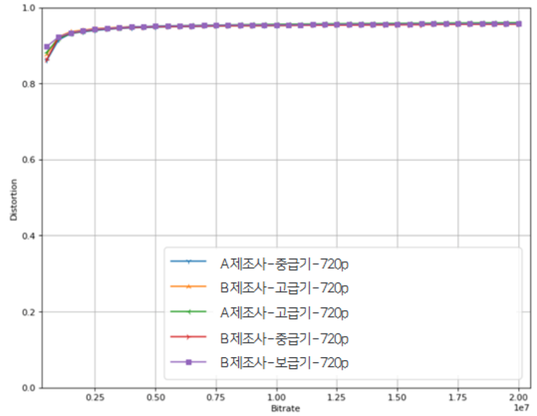
Low-Complex

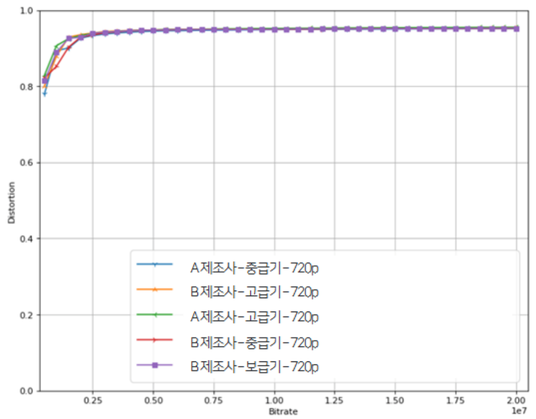


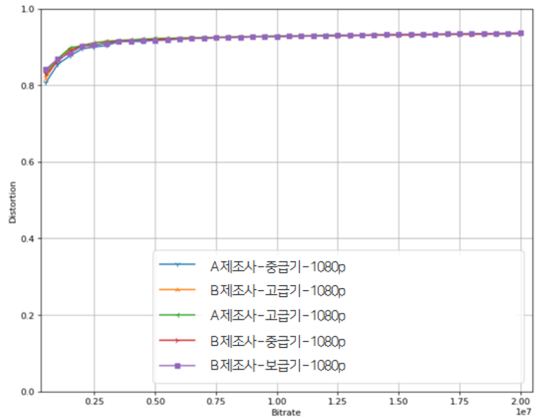
High-Complex

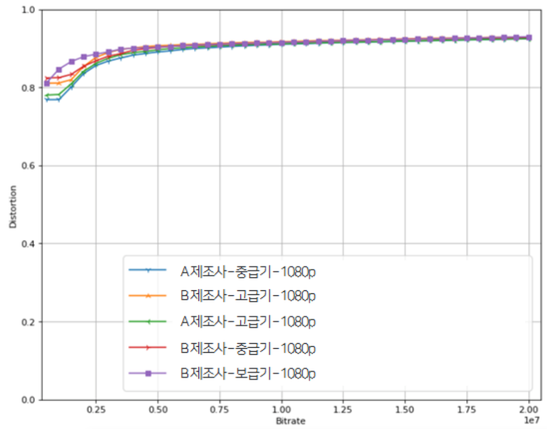
720p와 비교하면, 오히려 단말 간 품질 차이가 더 줄어든다

Device 별 품질 차이 – SSIM









Low-Complex

High-Complex

첫 번째 결론

동일 영상, 동일 인코딩 조건에서, 모바일 단말간의 인코딩 화질 차이가 있을까?

약간의 차이는 있으나, 큰 의미를 두기는 어렵다

두 번째 검증

Complex scenes with motion and textures require higher bitrates than scenes with less movement.

동일 인코딩 조건에서, 서로 다른 복잡도를 지닌 영상 간의 결과물 품질 차이는?

이미지 출처 : https://bitmovin.com/per-title-encoding

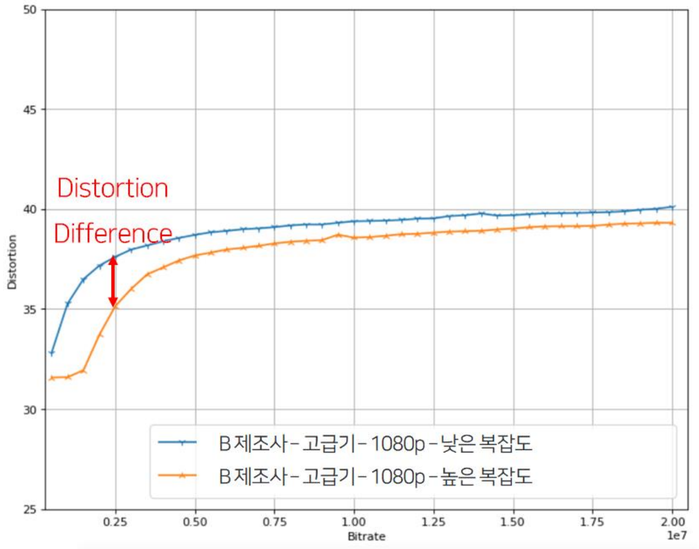
두 번째 검증

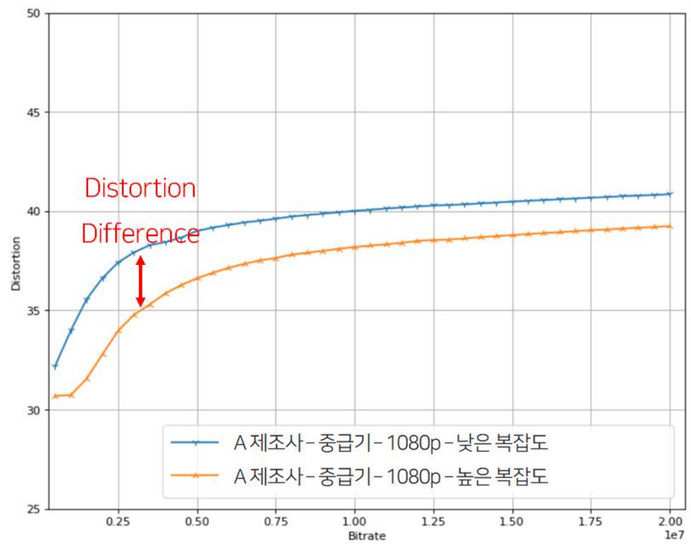
Complex scenes with motion and textures require higher bitrates than scenes with less movement.

복잡도에 따른 차이는 당연히 존재하고, 이론과 실험을 통해 검증 된 사실

권장 비트레이트 결정에 있어, 복잡도가 미치는 영향의 정도 결정

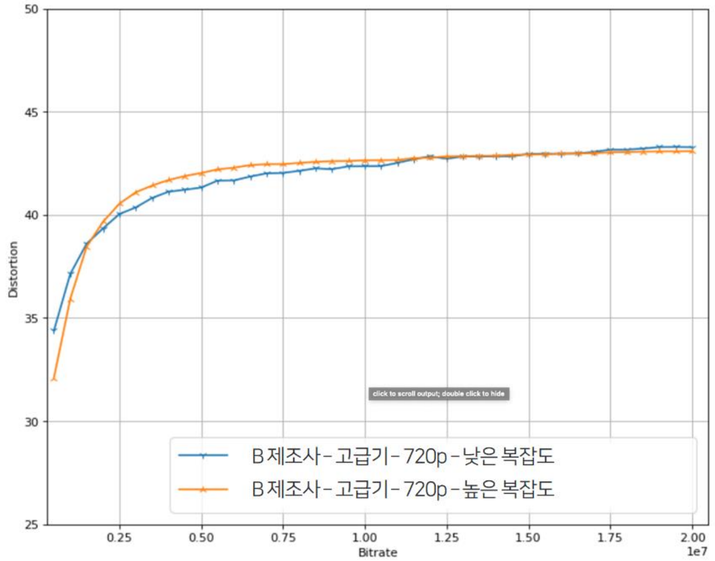
Complexity 별 품질 차이 – 1080p

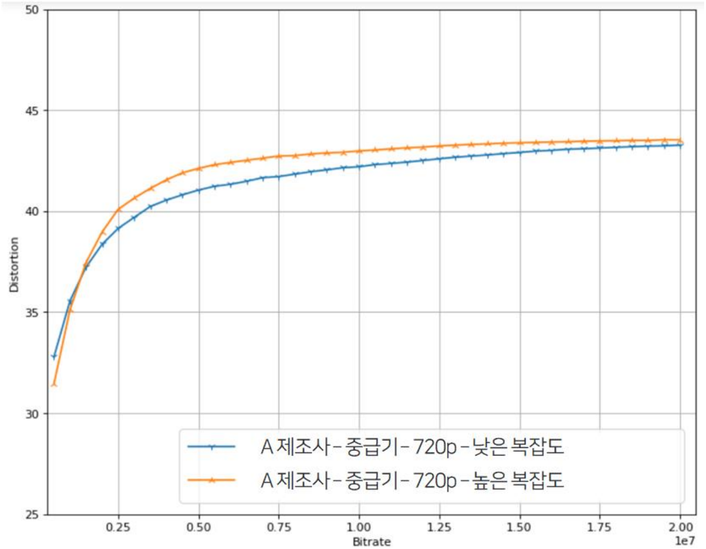




영상 복잡도에 따른 PSNR 차이는 상당히 큰 편이다 (> 2dB)

Complexity 별 품질 차이 – 720p





720p는 1080p 에 비해 PSNR 편차가 크지 않다

두 번째 검증 결과

Complex scenes with motion and textures require higher bitrates than scenes with less movement.

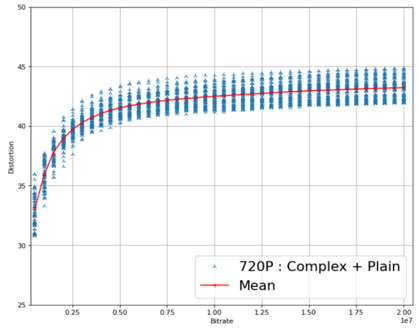
동일 인코딩 조건에서, 서로 다른 복잡도를 지닌 영상 간의 결과물 품질 차이는?

1080p 에서는 품질 차이가 크며, 권장 비트레이트 결정시에 고려되어야 한다

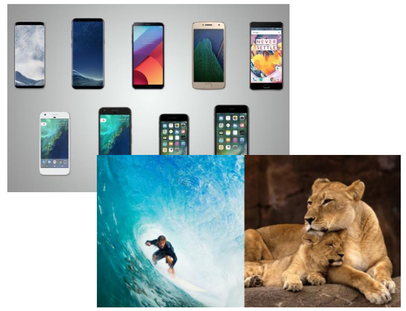
권장 비트레이트 도출

DEVIEW 2019

다양한 조건을 고려해서 권장 비트레이트를 도출하자



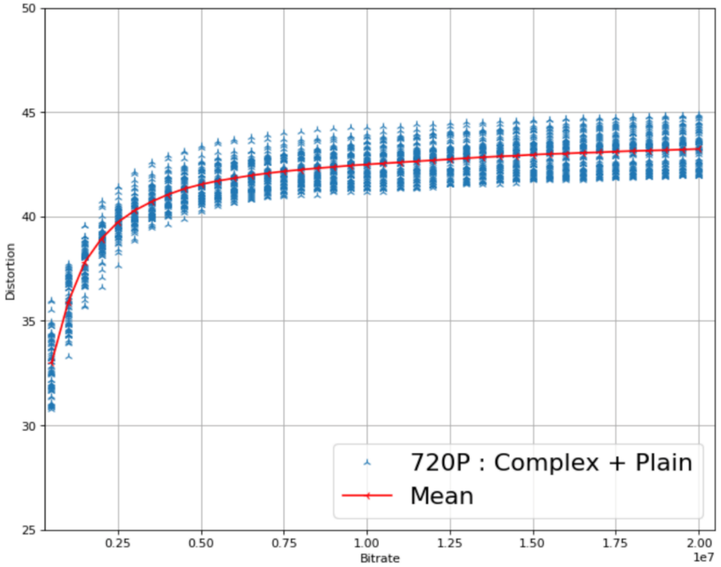
다양한 단말, 조건에
따른 테스트 결과

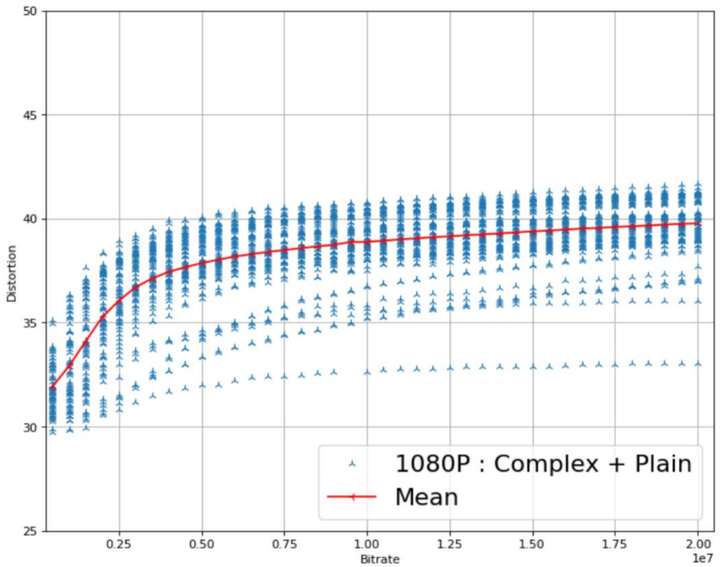


복잡도, 단말별 검증 결과

서비스, 지역, 사용자 특성

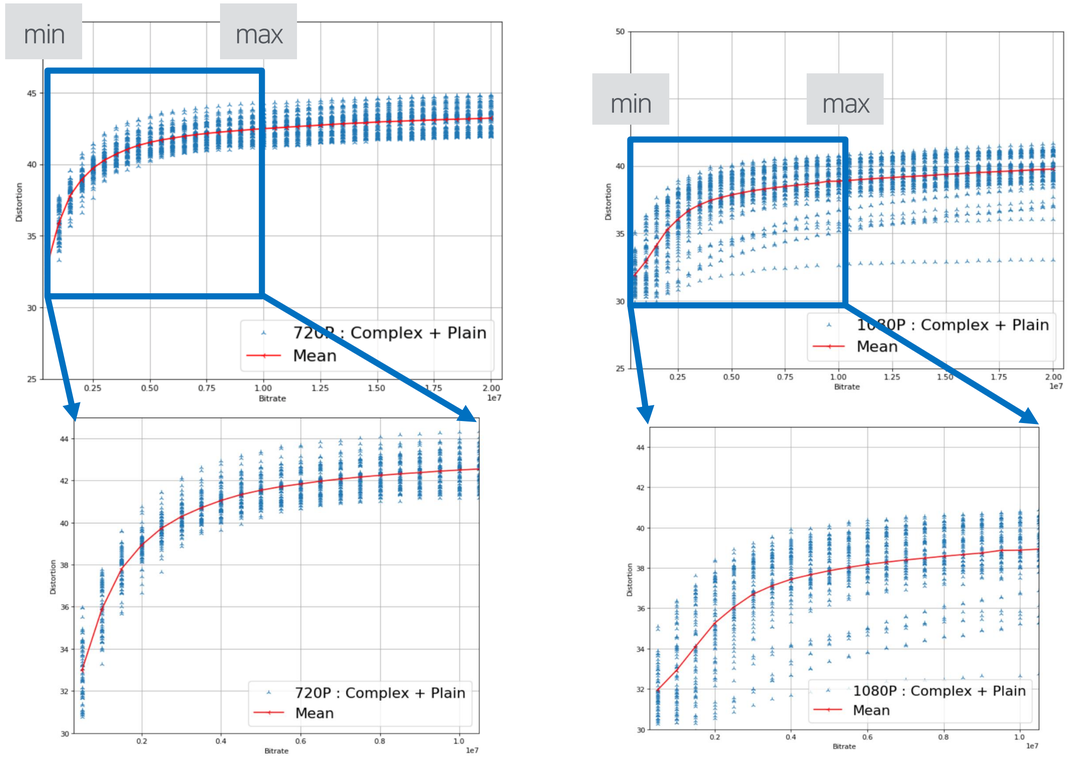
권장 비트레이트 도출





1. 테스트 결과 값들을 토대로, 서비스 특성과 목표에 따른 적합한 로그 함수 기반의 모델 디자인, 근사

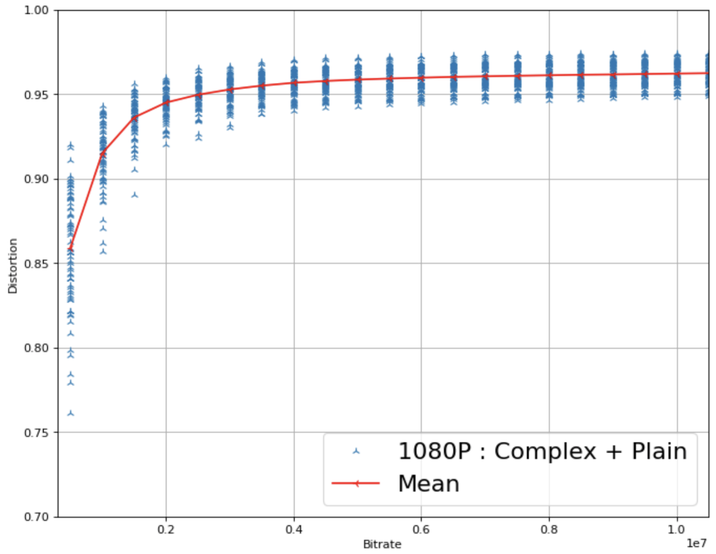
권장 비트레이트 도출



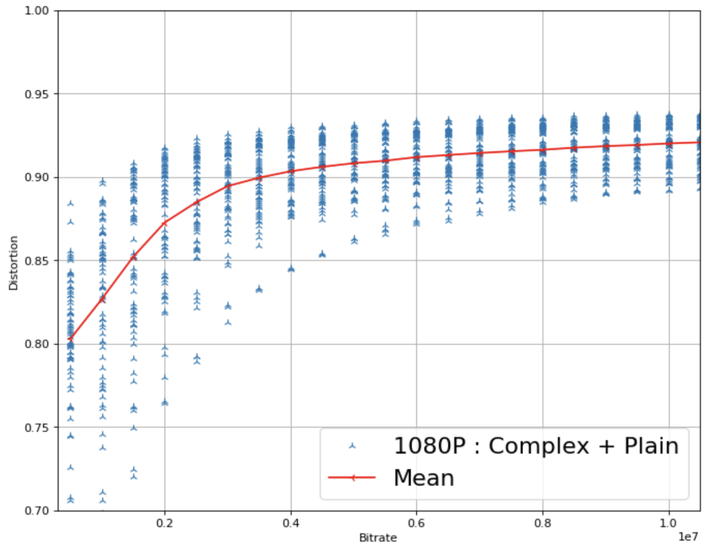
2. 비트레이트 투자 대비 화질 향상량이 충분한 구간을 결정

= Adaptive Bitrate Publish 의 min/max 구간

권장 비트레이트 도출



720p

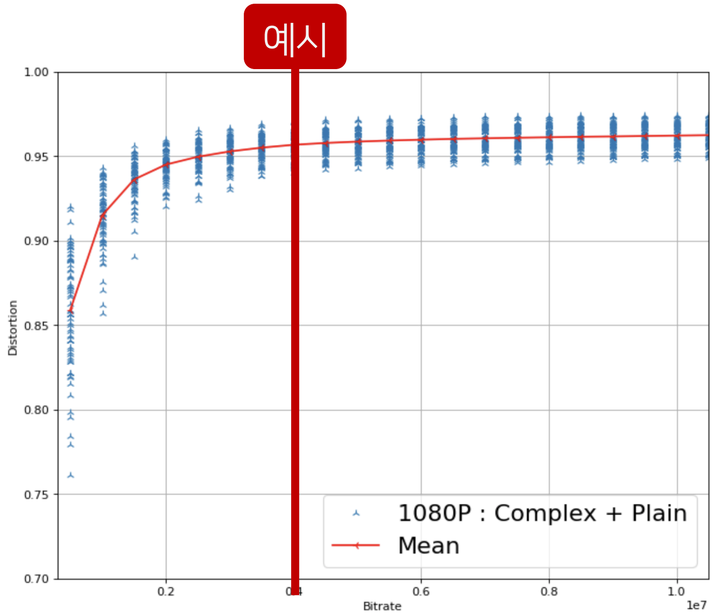


1080p

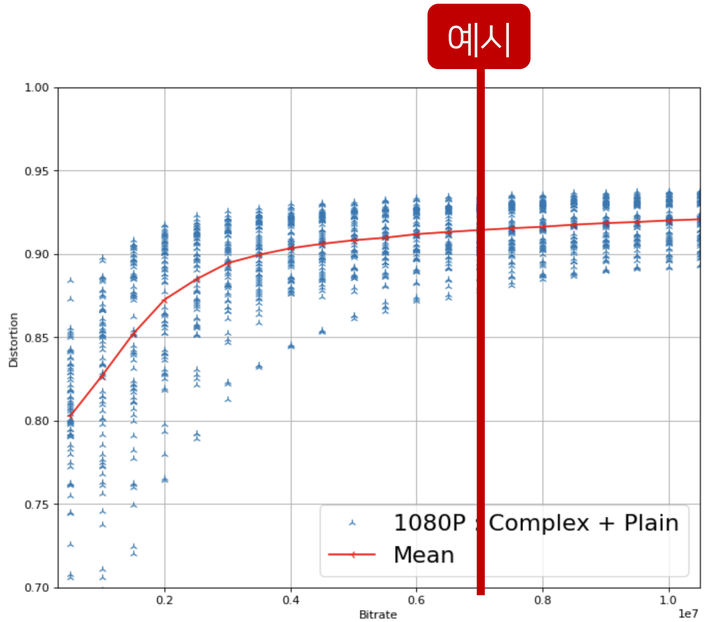
3. 우수한 품질을 보장하는 것으로 알려진 QoE Metric 값과

비트레이트 투자 대비 화질 상승률을 기준으로 권장 비트레이트 결정

권장 비트레이트 도출



720p



1080p

3. 우수한 품질을 보장하는 것으로 알려진 QoE Metric 값과

비트레이트 투자 대비 화질 상승률을 기준으로 권장 비트레이트 결정

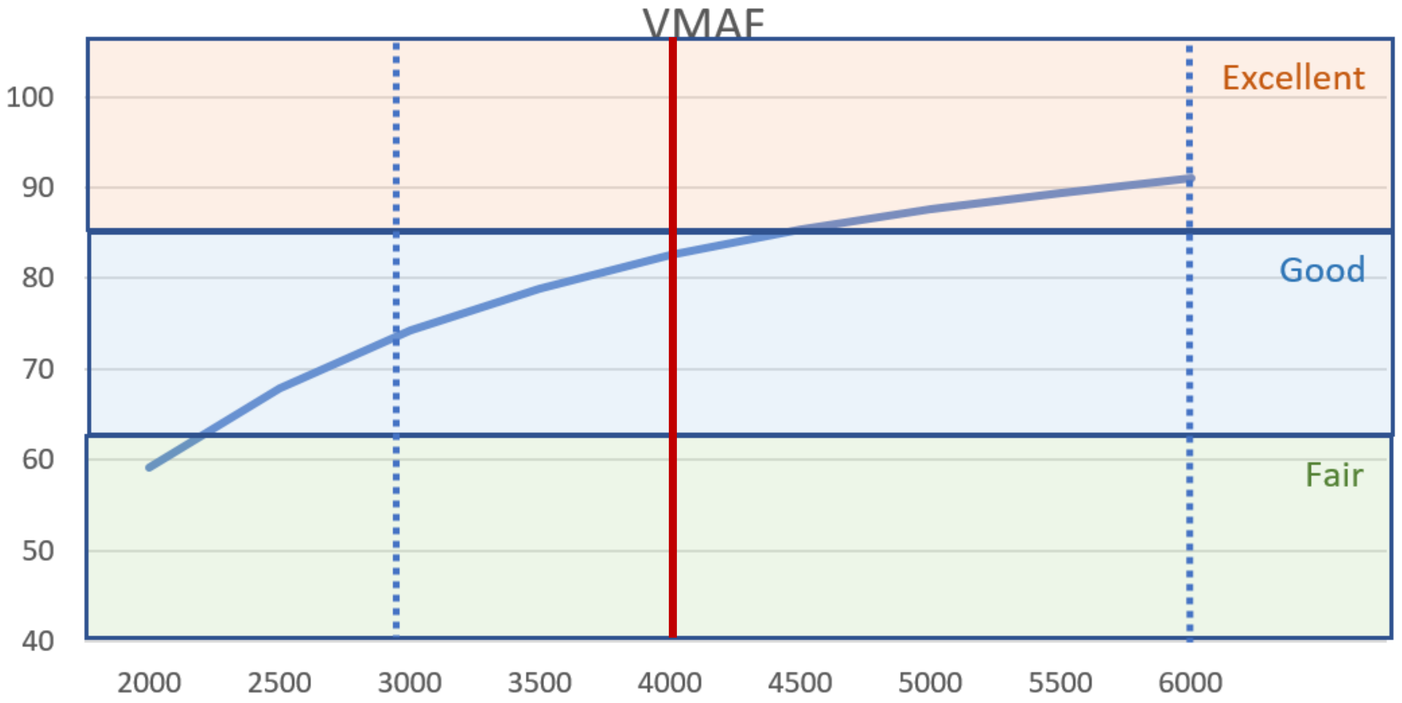
PSNR, SSIM 의 한계

PSNR, SSIM 과 같은 산술적 비교로는 '체감 품질' 을 계량하기 어렵다

https://github.com/Netflix/vmaf

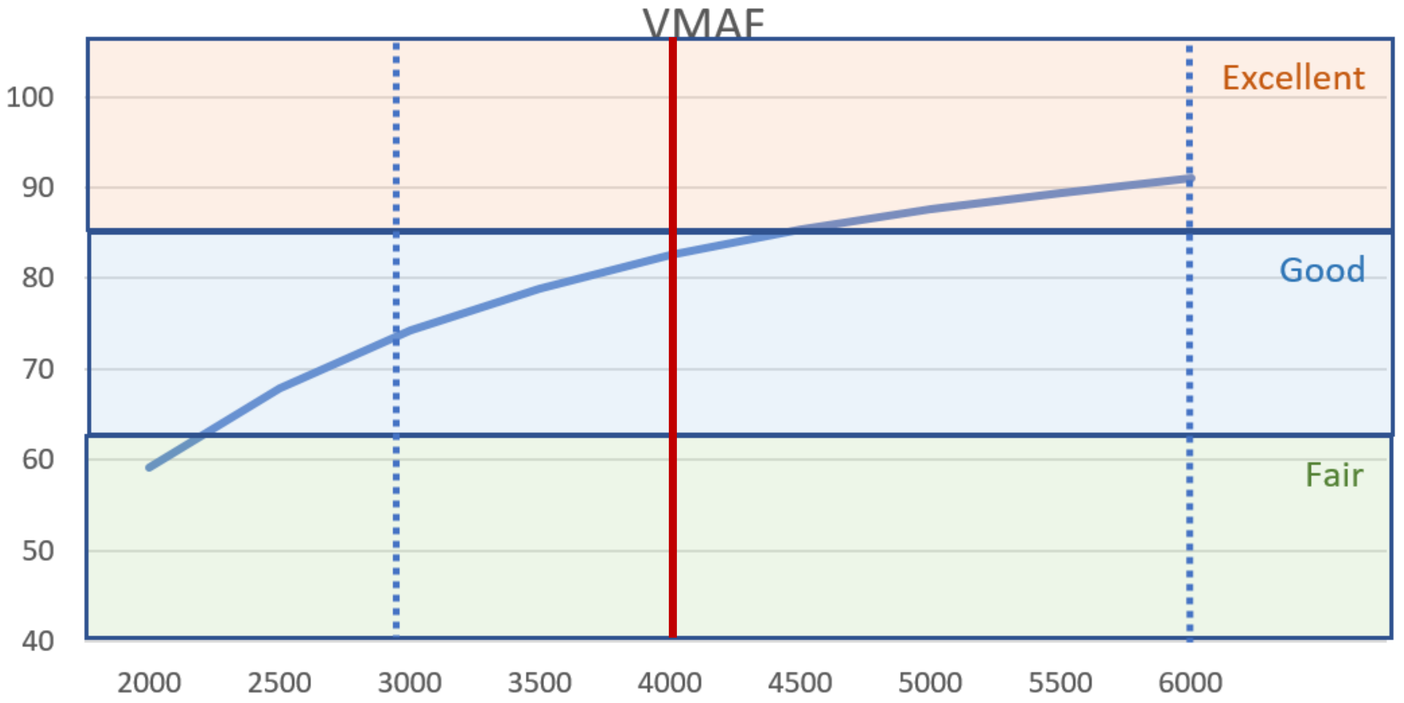
VMAF is a perceptual video quality assessment algorithm developed by Netflix. VMAF Development Kit (VDK) is a software package that contains the VMAF algorithm implementation, as well as a set of tools that allows a user to train and test a custom VMAF model.

H.264 720p VMAF 추가 검증



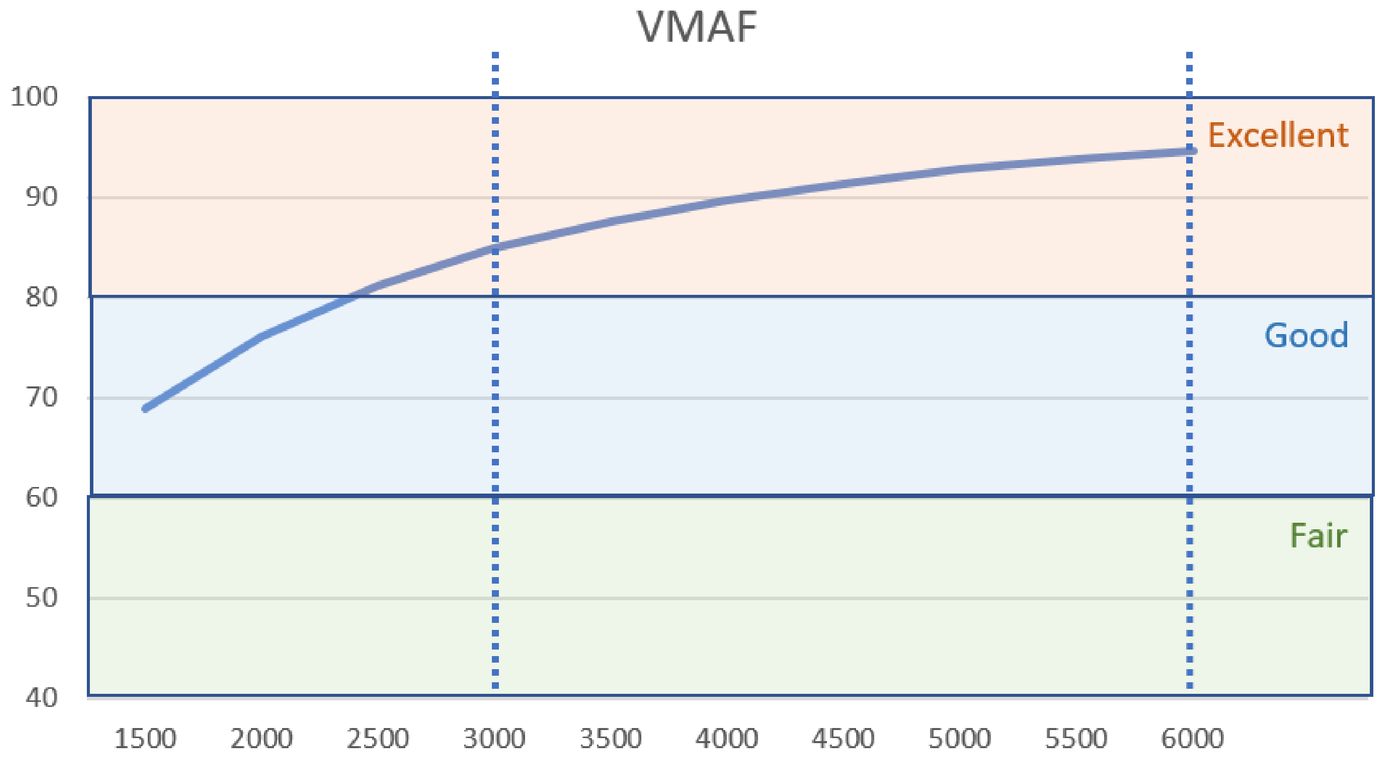
Bitrate(kbps)	VMAF
2000	59.1107
2500	67.959517
3000	74.214315
3500	78.915293
4000	82.524579
4500	85.3576
5000	87.618399
5500	89.468072
6000	91.00941

H.264 720p VMAF 추가 검증



	VMAF
Predictive value	Much better
Basis	Math + ML
Reference	Yes
Scoring	0–100
No artifact threshold	93
Artifacts likely present	NA
Interpreting scores	
Excellent	80–100
Good	60–80
Fair	40–60
Poor	20–40
Bad	< 20
Just-noticeable difference	6
Device ratings	Standard, Phone, 4K
Ownership	Open source

별책부록 H.265 (HEVC)



Bitrate(kbps)	VMAF
1500	68.928233
2000	76.204665
2500	81.291183
3000	84.97686
3500	87.753544
4000	89.868598
4500	91.528826
5000	92.855763
5500	93.92815
6000	94.809386

결과

Codec, Profile, Encoding Parameter 등에 따른 영상의 품질 변화를 확인할 수 있는 ‘자동화 테스트 도구 확보‘

PRISM, V LIVE 및 다양한 네이버 서비스의 상황과 조건에 따라 지속적으로 검증, 업데이트

여기서 문제

H.264 + 1080p + CBR + High Profile

권장 비트레이트는 얼마로 결정하면 될까요?

테스트 도구 실행 후, 답변 드리겠습니다!

DEVIEW
2019

송출 품질의 장
불안정한 네트워크에서의 안정적 송출
Adaptive Bitrate Publish

출시 당시 예상한 우리의 미래

DEVIEW
2019

파편화, 배터리, 카메라 해상도, 네트워크, 국가, 지역 …

현실

출시, 운영 결과. 당연히 예상한 그대로 발생합니다 ^^;;

스트리머들이 나가기 시작했다

DEVIEW
2019

집 밖으로 나가더니

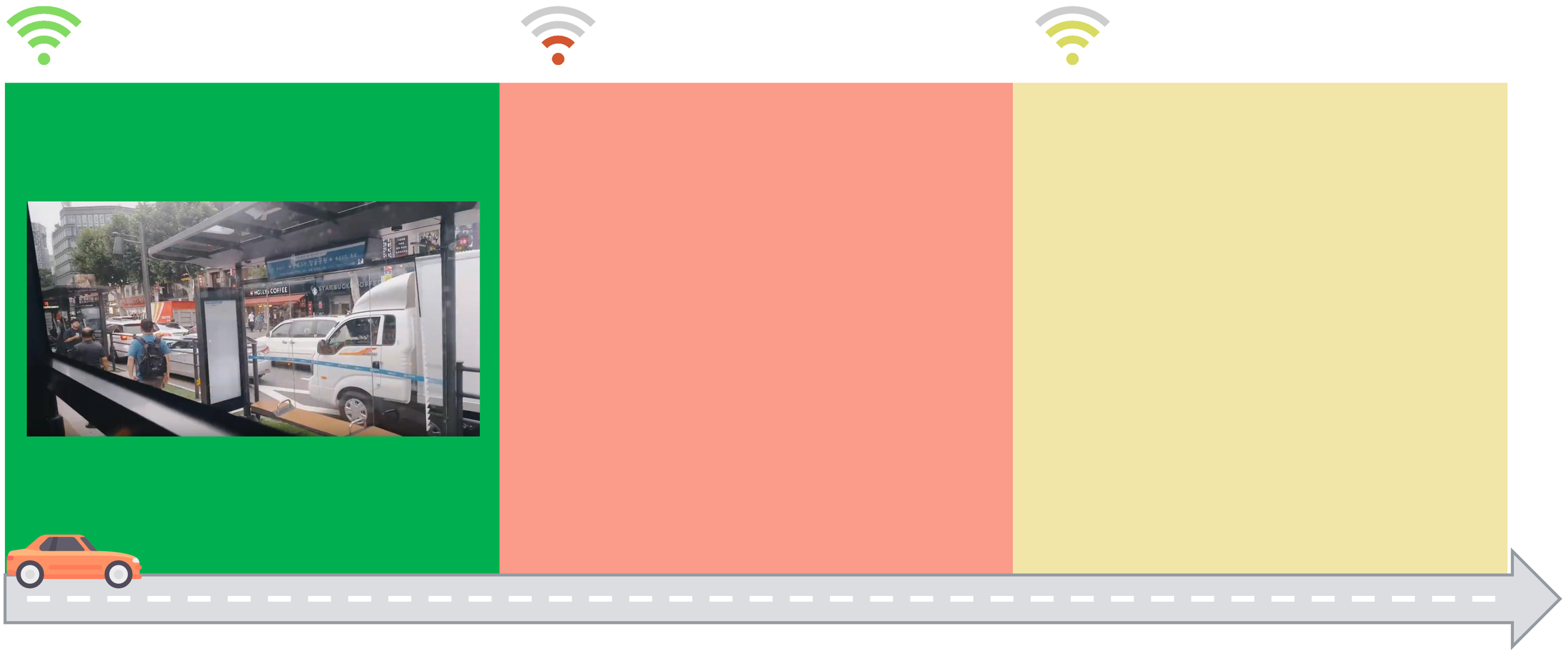
스트리머들이 나가기 시작했다

집 밖으로 나가더니

이젠 나라 밖으로도 나간다

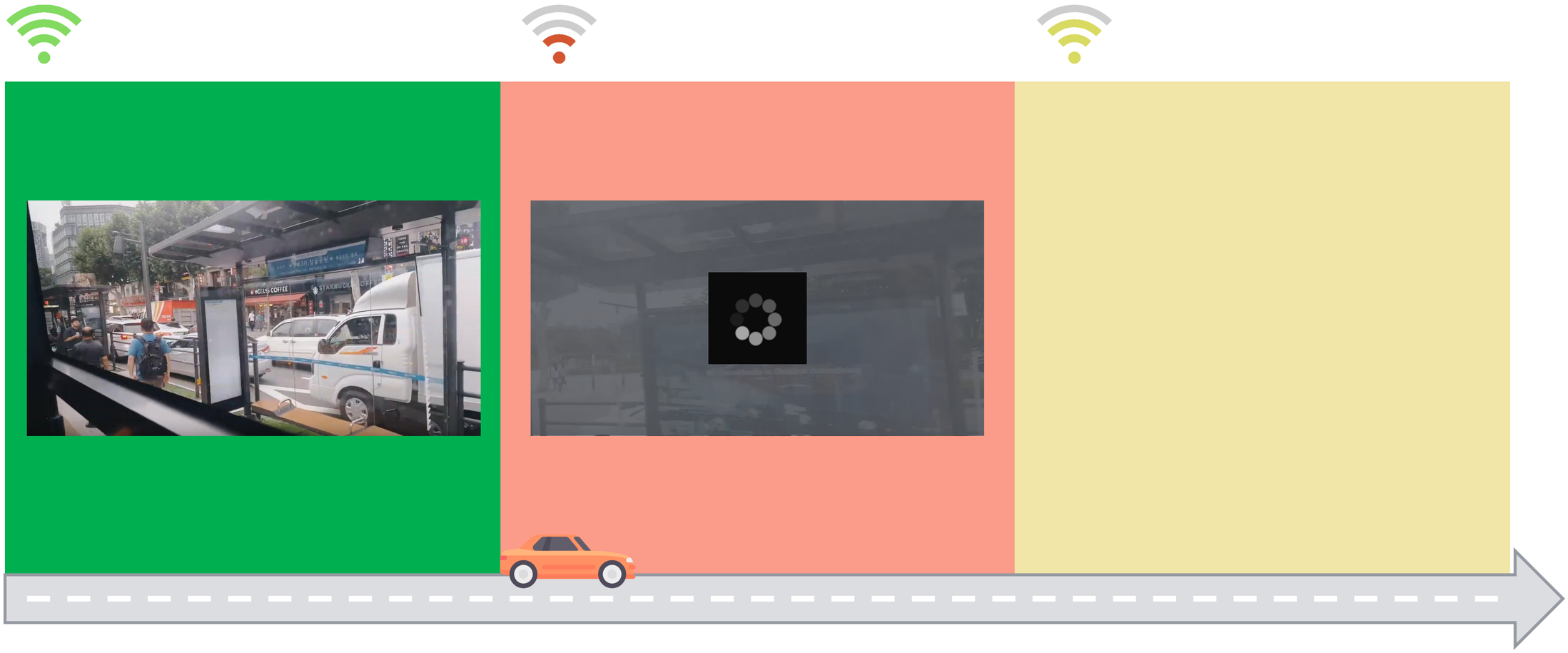
어디에서나 라이브가 잘 되는 것이 아니다

DEVIEW
2019



어디에서나 라이브가 잘 되는 것이 아니다

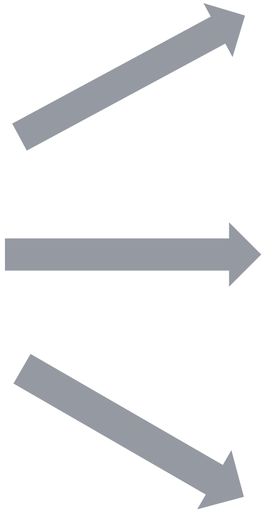
DEVIEW
2019



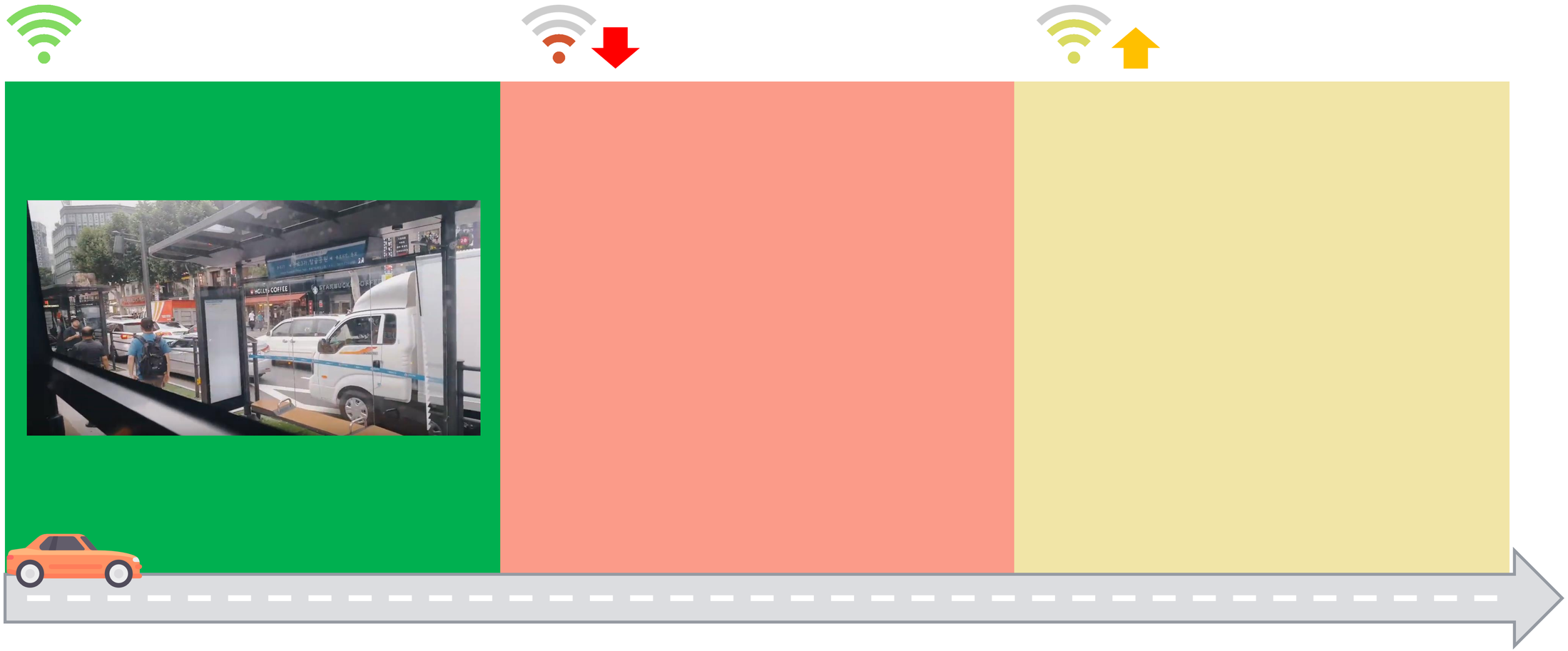
버퍼링의 파급효과가 다르다

Streamer

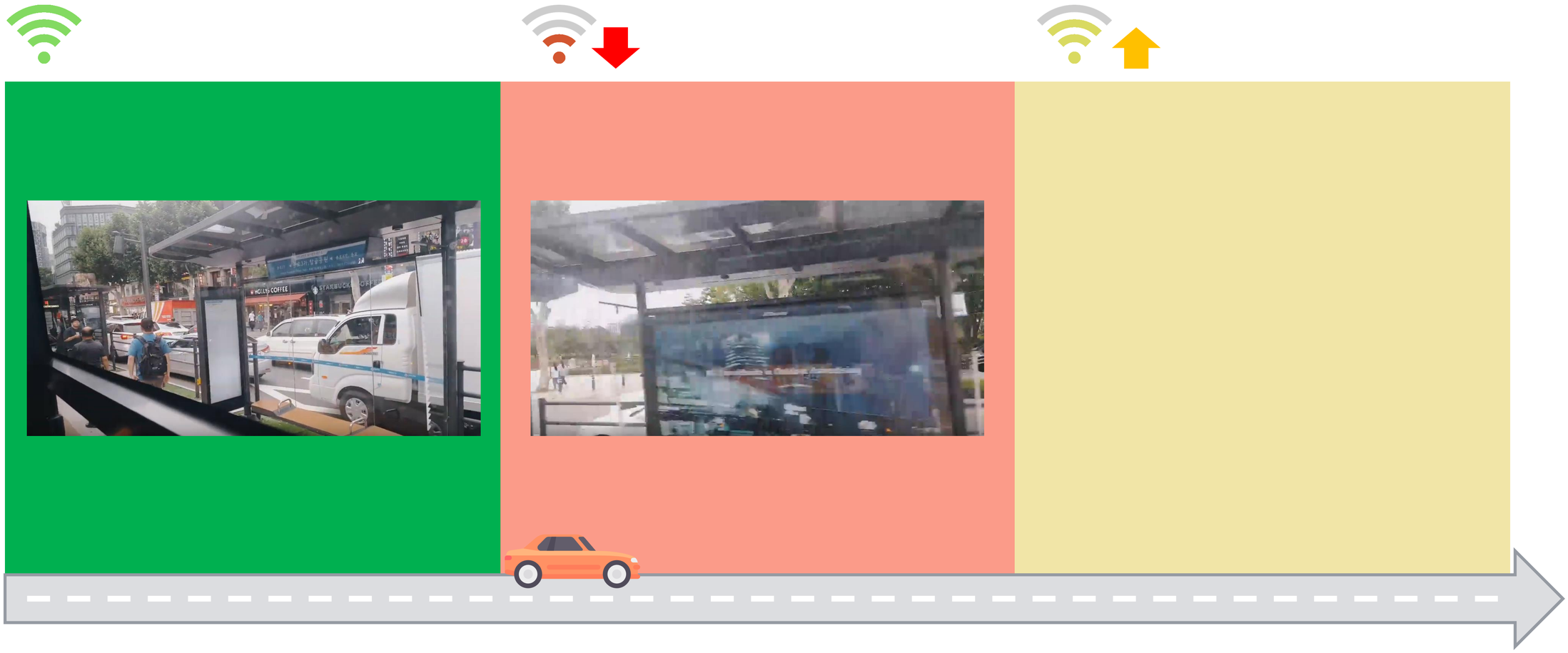
동영상 공급자



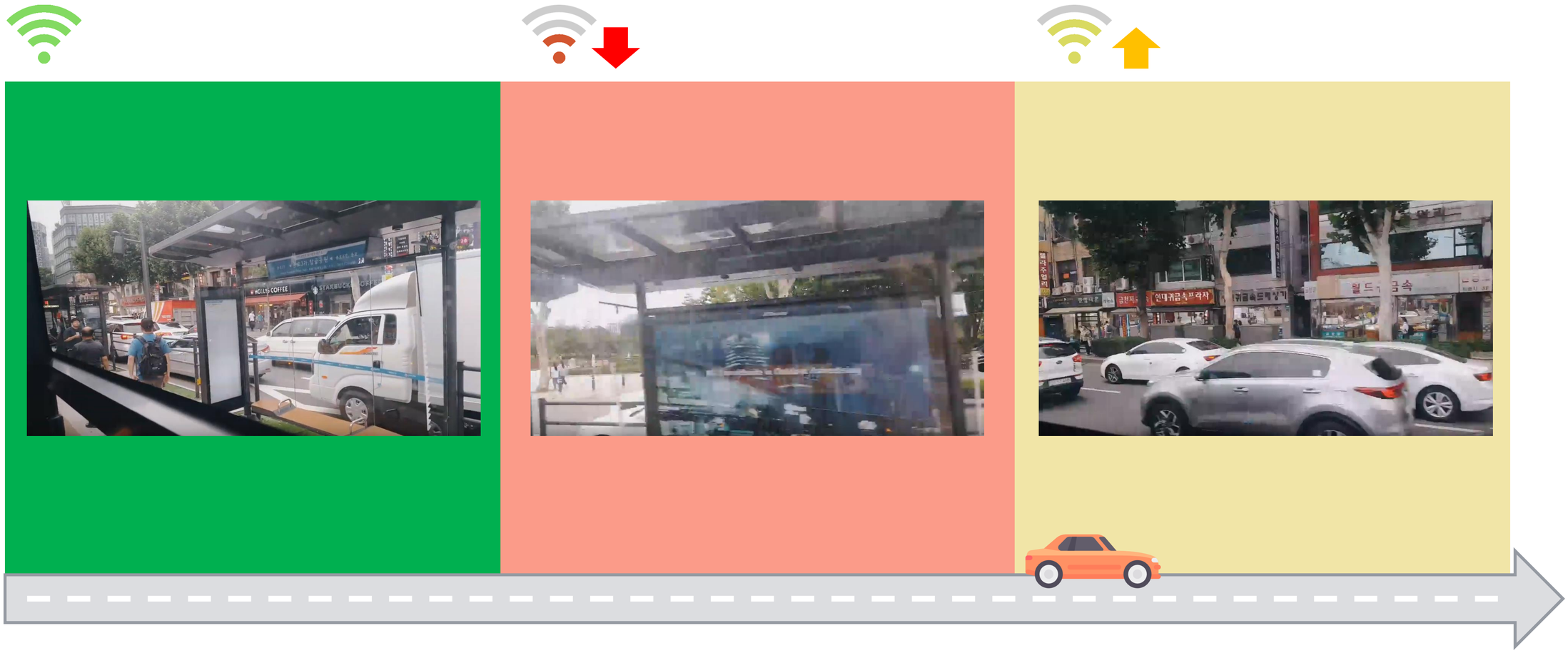
비트레이트가 변화해야 한다



비트레이트가 변화해야 한다



비트레이트가 변화해야 한다



네트워크 품질 평가의 어려움

Round Trip Time (RTT) 을 이용한 Throughput 측정

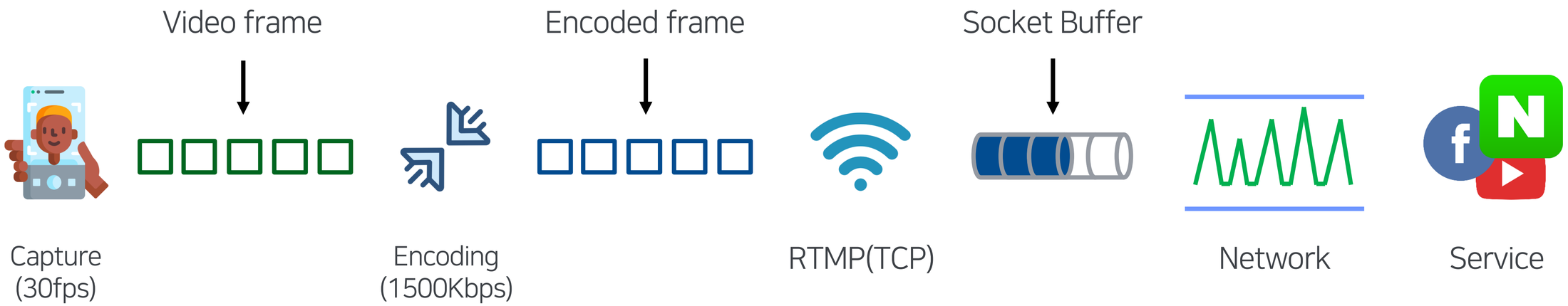
잘 알려진 전통적 계산식

TCP Throughput = TCP Buffer size (window size) / Round Trip Time

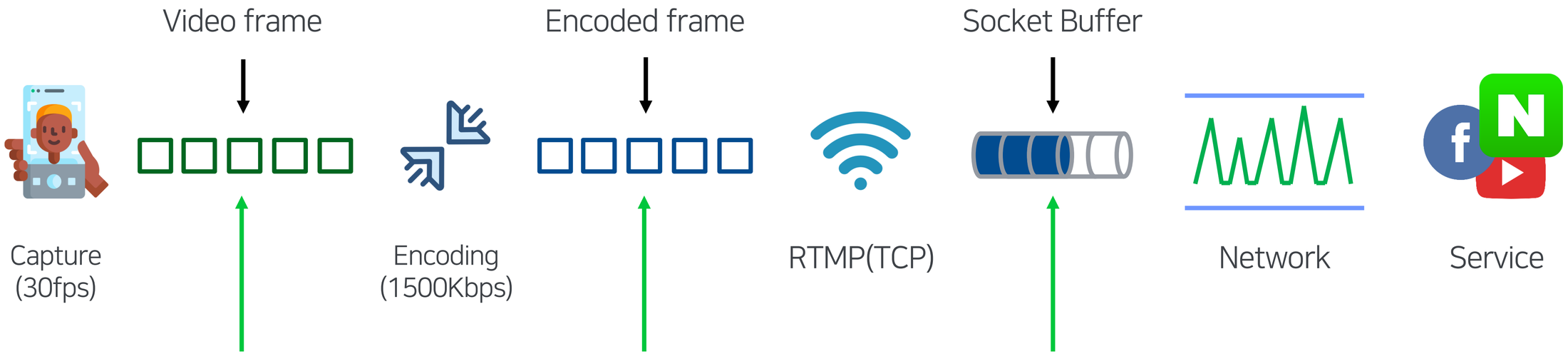
RTMP Protocol 송출 패킷을 토대로 실시간 측정 해 보면?

- RTT가 부정확하게 획득되는 경우가 발생한다
- 측정된 Throughput 이 알려진/통제된 maximum bandwidth 와 큰 차이가 있는 경우가 생긴다
- 즉각 예상되는 원인들? Flow control, slow start, congestion control 등..

실시간 RTMP Throughput 측정의 어려움

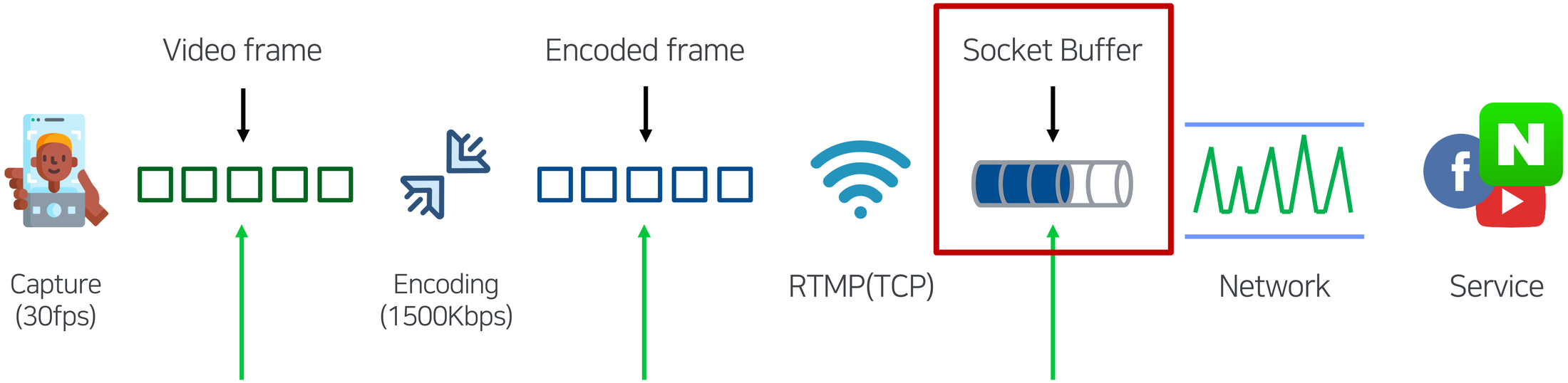


실시간 RTMP Throughput 측정의 어려움



Huang, Te-Yuan, et al. "Confused, timid, and unstable: picking a video streaming rate is hard."
Proceedings of the 2012 internet measurement conference. ACM, 2012.

실시간 RTMP Throughput 측정의 어려움



1. 경우에 따라 불연속적일 수 있는 미디어 프레임 획득

2. 작은 전송 데이터 양 인코딩 데이터의 불규칙적 크기

3. 전송 데이터 적재 이후 측정 불가능

실시간 RTMP Throughput 측정의 어려움

DEVIEW 2019

작지만 견고한 전송 버퍼

적재-전송 간격 적음

정밀한 Throughput 측정 가능

전송량 제약의 여지 있음

VS

크고 멋진 전송 버퍼

전송량 제약에서 자유롭지만

적재-전송 간극이 크다

Throughput 정밀측정 어려움

Adaptive Bitrate Publish

RTMP Protocol의 특성에 적합한 Adaptive Bitrate Publish 개발

Round Trip Time, TCP Throughput 이슈에 영향을 받지 않는 네트워크 측정의 방법

'RTMP 전송 중 실시간으로' 네트워크 품질 측정

비트레이트를 낮출때는 누구보다도 빠르게, 높일 때는 상대적으로 천천히, 조심스럽게

특허 등록번호 10-1937247

실시간 라이브 환경에서 버퍼기반 BandWidth Estimation 및 ABP(Adaptive Bitrate Publish) 적용 방법

ABP 기대 동작 양상


2.5
2
1.5
1
0.5
0
Bitrate (Mbps)
30
60
90
120
150
180
210
240
300
330
360
390
Time (Sec)
Network
Bitrate

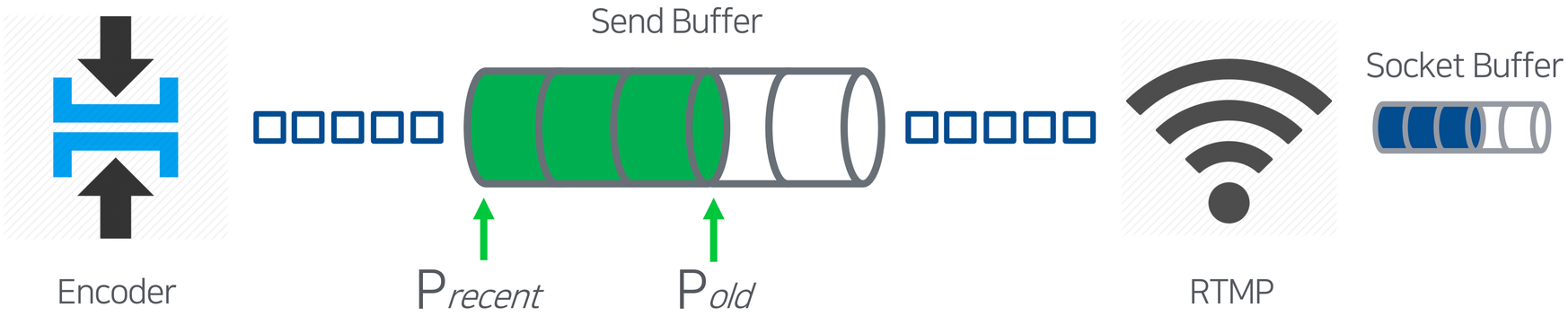
Simple Idea

Throughput 측정이 어렵다면, '다른 기준(=Buffer)' 으로 판단, 측정한다

기존 연구된 Buffer 기반의 Adaptive Bitrate estimation 을 '라이브 송출' 에 사용할 수 있도록 개선

Kevin Spiteri, Rahul Urgaonkar, Ramesh K. Sitaraman:
BOLA: Near-optimal bitrate adaptation for online videos. INFOCOM 2016: 1-9

Simple Idea



P = presentation time

$$\text{Buffer Duration} = P_{recent} - P_{old}$$

Buffer Duration > *Down Threshold Duration* = Bitrate Down

Buffer Duration < *Up Threshold Duration* = Bitrate Up

Bitrate Down Action

Bitrate Down 은 버퍼링을 일으킬 수 있으므로, 조심해서 실행해야 한다

네트워크 상태가 급격히 나빠질 때, Bitrate Down 이 늦으면 여지없이 버퍼링

Bitrate Down 실행 시, 하락폭이 충분치 않아도 여지없는 버퍼링

가중치를 적용한 비트레이트 하락폭 계산 예시

Adaptive Bitrate Streaming 을 적용한 Player 들은 제각기 다양한 하락폭 계산식 사용

하락폭 결정시에는 일반적으로, 환경에 따라 실험, 결정된 임의의 가중치(weight) 등을 사용

NewBitrate = CurrentBitrate * (1.0 – BufferDuration / (Interval * IterationCount)) * weight

Bitrate Up Action

Bitrate Up 또한 버퍼링을 일으킬 수 있으므로, 조심해서 실행해야 한다

가용 대역을 초과하여 상승하면 버퍼링, 측정에 따른 상승 여부 결정시에도 신중한 판단이 필요

단순히 비트레이트 상승 조건 부합의 횟수를 1회가 아닌 'n' 회 등으로 다중 검사

이전 시점에 발생한 Action 들을 고려한다

부합 횟수 검사로 'n' 을 결정하기 위한 전략이 필요하다

현 시점 바로 전에 3회 연속 Bitrate Up 이 일어났다면? -> Bitrate Up 조건 완화? -> 'n' 의 축소!?

= 이와 같은 '시나리오' 에 따른 조건 다변화가 필요하다

조건 결정 시나리오

시나리오는 경우에 따라 수정되거나 추가 삭제되어야 하지 않을까?

지역, 사용자, 서비스 별로 항상 동일한 시나리오가 적용되어야 할까?

시나리오의 변경이 있을 때 마다 코드를 수정해야 할까?

결국 만들어야 할 운명

시나리오는 경우에 따라 수정되거나 추가 삭제되어야 하지 않을까?

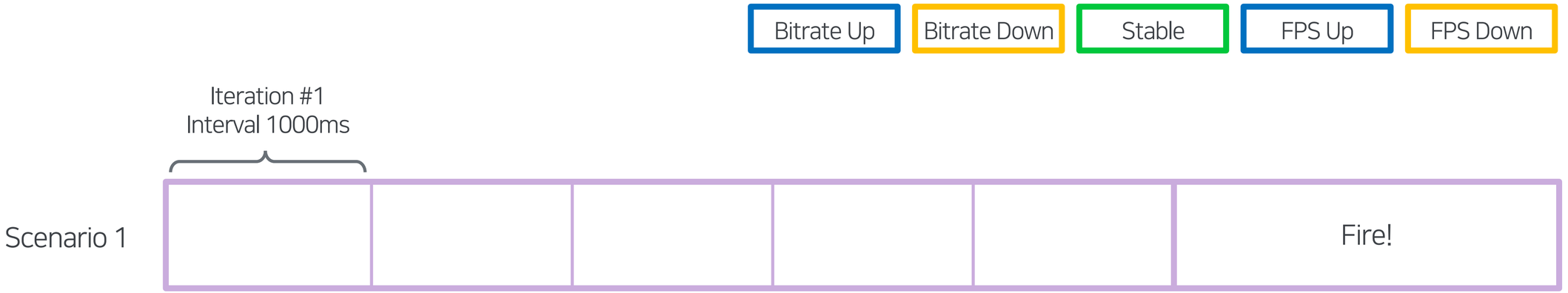
지역, 사용자, 서비스 별로 항상 동일한 시나리오가 적용되어야 할까?

시나리오의 변경이 있을 때 마다 코드를 수정해야 할까?

⇨ 자유로운 시나리오 생성/수정/삭제가 가능하고, 서비스/목적 별 적용이 가능한 기능 개발

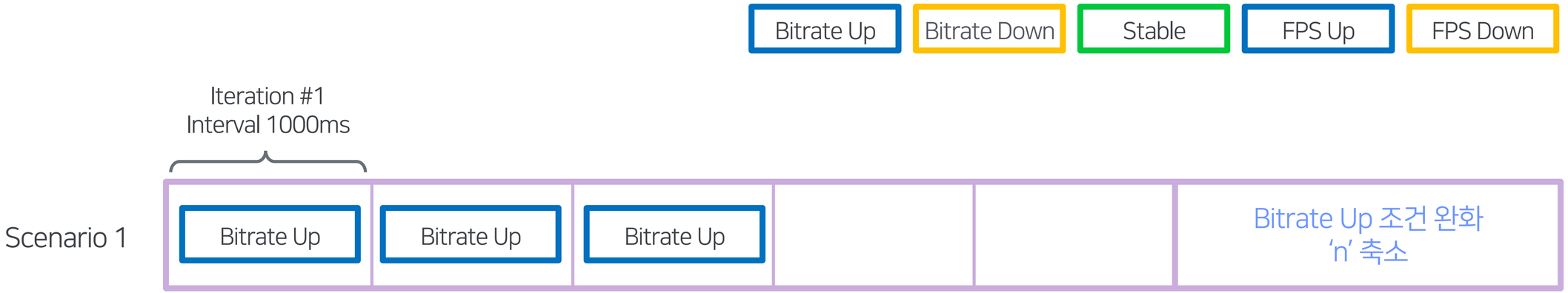
ABP Action

시나리오 예시



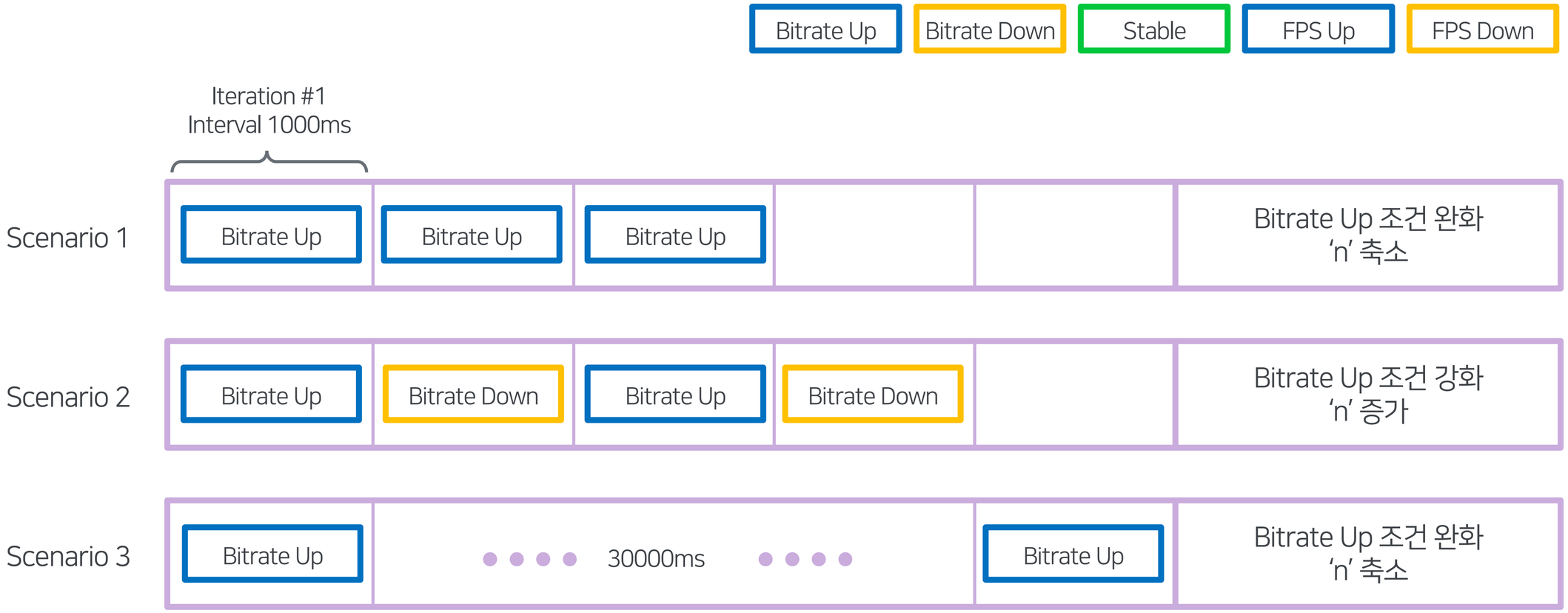
미리 지정된 시나리오대로 Action 이 일어나면 예약된 작업을 Fire!

상태 연계 시나리오



1000ms 간격으로 3회의 Bitrate Up Action 이 발생하면 Bitrate Up 조건 완화

자유로운 추가/삭제/수정


Bitrate Up
Bitrate Down
Stable
FPS Up
FPS Down
Iteration #1
Interval 1000ms
Scenario 1
Bitrate Up
Bitrate Up
Bitrate Up
Bitrate Up 조건 완화
'n' 축소
Scenario 2
Bitrate Up
Bitrate Down
Bitrate Up
Bitrate Down
Bitrate Up 조건 강화
'n' 증가
Scenario 3
Bitrate Up
30000ms
Bitrate Up
Bitrate Up 조건 완화
'n' 축소

환경별 상수 탐색 적용

실험적으로 탐색, 결정해야 하는 다양한 상수들이 존재한다

Interval, Up/Down ThresholdBufferDuration, BitrateDownWeight 등

지역, 국가, 서비스에 기인하여 각기 다를 수 있으며, 개별적인 테스트와 탐색의 과정 필요

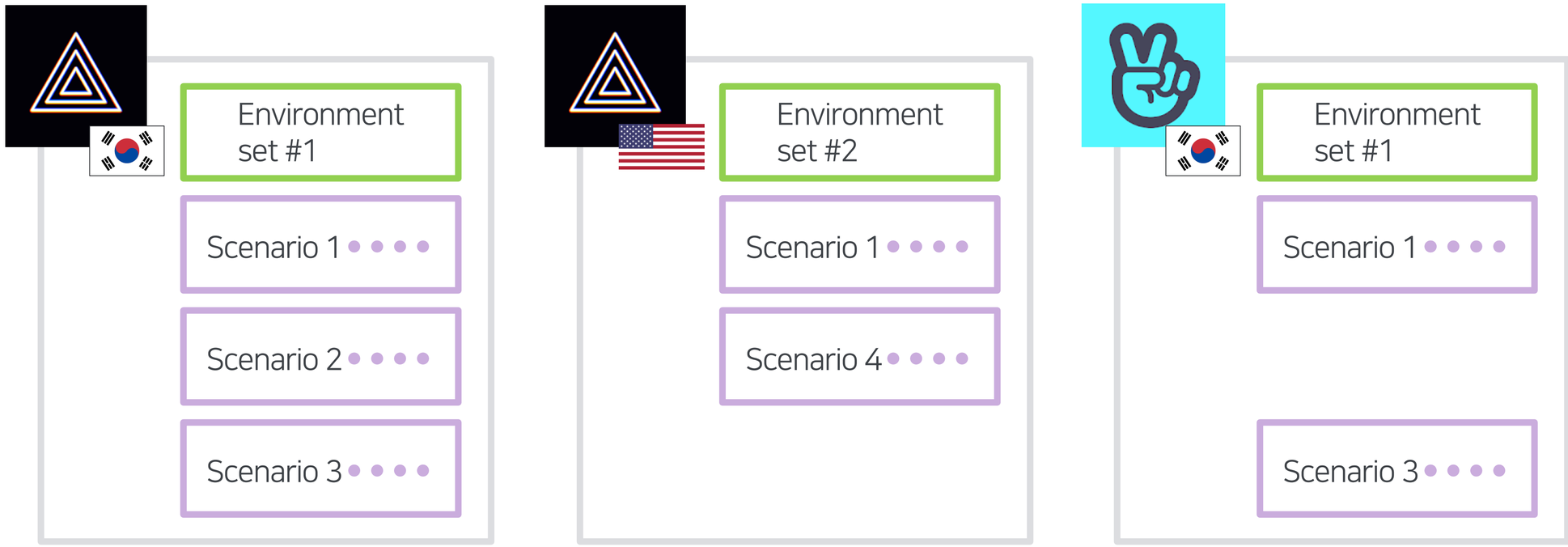
PRISM Pacman a.k.a. Network Emulator!

지역, 국가별 네트워크를 '캡쳐' 하여, 실험환경에서 재현할 수 있는 네트워크 에뮬레이터 개발과 활용

개념과 구성, 소개는 아래 링크에서 ^^;

https://d2.naver.com/helloworld/7847943

서비스/환경 개별 시나리오 적용



지역, 국가, 서비스 별로 적합한 독자적 정책 수립과 운용 계획 중

비트레이트 변경

이제 실제로 비트레이트를 변경해보자!

ABS 비트레이트 변동 시나리오

각기 다른 화질의 개별적인 Stream 중에서 선택

ABP 비트레이트 변동 시나리오

라이브 송출자
(streamer)

동영상 공급자

단일 스트림 내에서 지속적으로 비트레이트 변경

Android MediaCodec

DEVIEW 2019

PARAMETER_KEY_VIDEO_BITRATE

Added in API level 19

```
public static final String PARAMETER_KEY_VIDEO_BITRATE
```

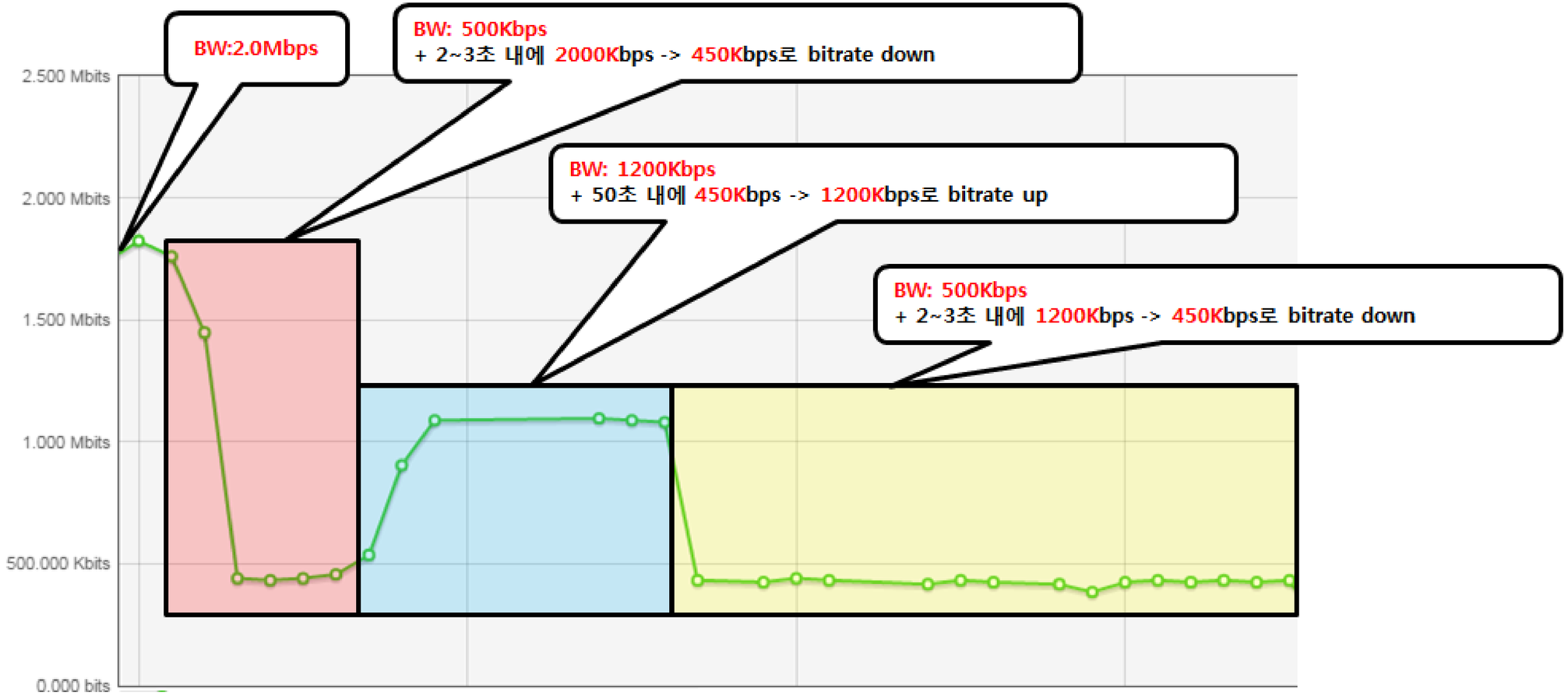
Change a video encoder's target bitrate on the fly. The value is an Integer object containing the new bitrate in bps.

See also:

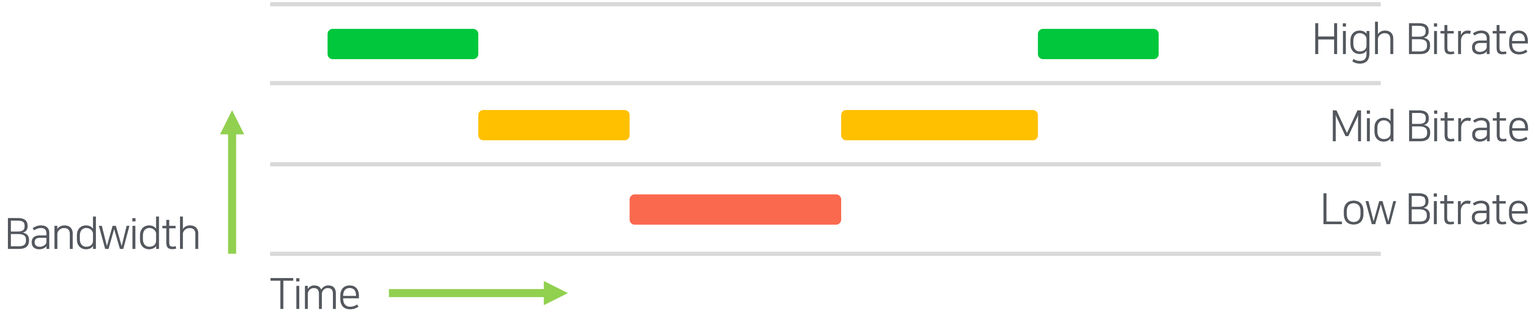
`setParameters(Bundle)`

Constant Value: "video-bitrate"

결과


BW:2.0Mbps
BW: 500Kbps
+ 2~3초 내에 2000Kbps -> 450Kbps로 bitrate down
BW: 1200Kbps
+ 50초 내에 450Kbps -> 1200Kbps로 bitrate up
BW: 500Kbps
+ 2~3초 내에 1200Kbps -> 450Kbps로 bitrate down
2.500 Mbits
2.000 Mbits
1.500 Mbits
1.000 Mbits
500.000 Kbits
0.000 bits

결과



결과

ABP 는 LIVE SDK가 적용된 모든 네이버 서비스가 사용 중

V LIVE, PRISM LIVE STUDIO, WAV, BAND 등

해외 송출, 글로벌 서비스 등에 안정적으로 대응 중

개선 과제

Cellular 환경의 QoS 시나리오 대응, 극단적인 약전개 상황에서의 송출 보장

‘Pensieve’ 와 같은 Deep Learning 도입 을 통한 개선 연구

H. Mao, R. Netravali, and M. Alizadeh.
Neural adaptive bitrate streaming with pensive. SIGCOMM, 2017.

DEVIEW
2019

안정성의 장
Android MediaCodec 사용량 제한 이슈

모든 기기에서 다 잘되는 것이 아니다

DEVIEW
2019

제한된 사용자

단말 권장 가능

송출 환경 가이드 가능

모든 기기에서 다 잘되는 것이 아니다

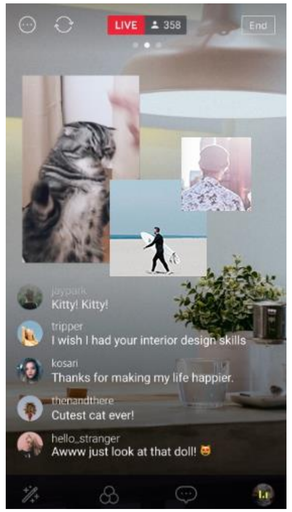
제한된 사용자
단말 권장 가능
송출 환경 가이드 가능

PRISM MOBILE
LIVE SDK

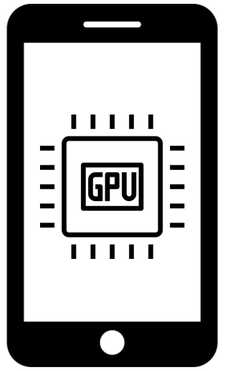
글로벌, 사용자 다양화
수많은 단말
송출 환경 다양화

문제들이 살살 나온다

기능의 다양화
저사양 성능저하



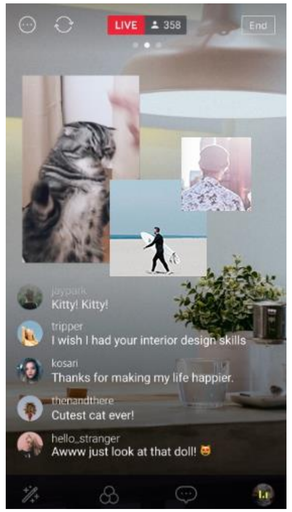
코덱을 다수 사용하는 기능
원인을 알 수 없는 간헐적 크래시



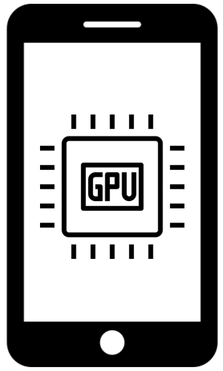
병렬처리 적용
특정 GPU에서의 크래시

문제들이 살살 나온다

기능의 다양화
저사양 성능저하

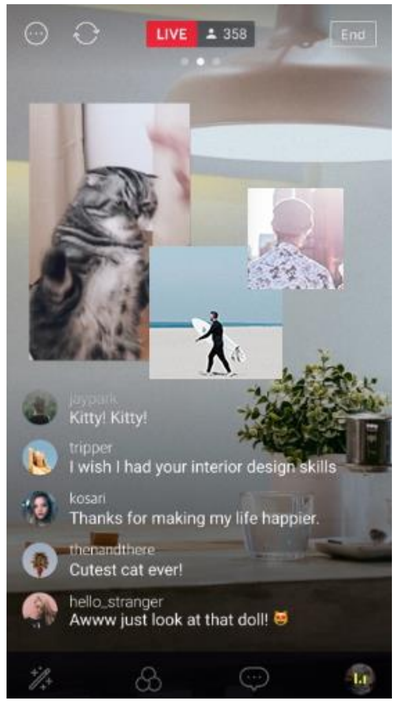


코덱을 다수 사용하는 기능
원인을 알 수 없는 간헐적 크래시



병렬처리 적용
특정 GPU에서의 크래시

간헐적이고 정체를 알 수 없는 크래시



Multi Encoding / Media Overlay 등의

코덱을 다수 사용하는 기능 수행 중, 간헐적인 크래시 보고

Exception 예시 #1

```
E/ACodec: [OMX.qcom.video.encoder.avc] ERROR(0x80001009)
E/ACodec: signalError(omxError 0x80001009, internalError -2147483648)
E/MediaCodec: Codec reported err 0x80001009, actionCode 0, while in state 6
E/ACodec: [OMX.qcom.video.encoder.avc] ERROR(0x80001009)
E/ACodec: signalError(omxError 0x80001009, internalError -2147483648)
E/MediaCodec: Codec reported err 0x80001009, actionCode 0, while in state 6
```

이미 알려진 사례들

2

When I try to decode H.264 raw stream with MediaCodec on Note3(N9005, Android 4.3) , I get these errors:

```
12-25 19:57:40.362: E/ACodec(19827): [OMX.qcom.video.decoder.avc] ERROR(0x80001009)
12-25 19:57:40.362: E/MediaCodec(19827): Codec reported an error. (omx error 0x80001009, inter
12-25 19:57:40.362: W/System.err(19827): java.lang.IllegalStateException
12-25 19:57:40.362: W/System.err(19827): at android.media.MediaCodec.dequeueInputBuffer(Nativ
```

But the same codes work on Note3(N900) and Google Nexus 7 (th
code what I referenced to: http://developer.android.com/reference/a
P.S: the header of my H.264 raw stream looks like this:

```
-----------------------------------------------------------------
|00 00 00 01 67 ... 00 00 00 01 68 ...00 00 00 01 65(tatol 4
-----------------------------------------------------------------
```

I can attach my testing video file for you if need.

1

I implemented a video encoder by MediaCodec API to encode 2560x720 videos. I verif
several devices but I encountered a configuration problem on Samsung Note 2 (Androi
following is my code to prepare the codec instance:

```
Format = MediaFormat.createVideoFormat("video/avc", 2560, 720);
Format.setInteger(MediaFormat.KEY_BIT_RATE, (int) (2560 * 720 * 20 * 0.5f));
Format.setInteger(MediaFormat.KEY_FRAME_RATE, RefreshRate);
Format.setInteger(MediaFormat.KEY_COLOR_FORMAT, MediaCodecInfo.CodecCapabilitie
Format.setInteger(MediaFormat.KEY_I_FRAME_INTERVAL, 1);

try {
    Codec = MediaCodec.createEncoderByType("video/avc");
    Codec.configure(Format, null, null, MediaCodec.CONFIGURE_FLAG_ENCODE);
    CodecFound = true;
} catch (Exception e) {
    CodecFound = false;
    if(DEBUG) Log.e(CodecTag, "Failed to find available codec", e);
}
```

No exception is thrown during this try-catch block. But after starting codec operation, n
encoded data nor the output format is output by the encoder even if the input buffers a
("dequeueOutputBuffer" always returns -1). The following is the log after starting codec
errors are reported:

```
05-04 11:29:17.071: E/MFC_ENC(1006): SOMXVE_Init: isSEIOn : 0
05-04 11:29:17.071: E/MFC_ENC_APP(1006): SsbSipMfcEncInit]    IOCTL_MFC_ENC_INI
05-04 11:29:17.111: A/libc(1006): Fatal signal 11 (SIGSEGV) at 0x00000000 (code
05-04 11:29:19.326: E/VideoCodecInfo(1353): Failed to get track format due to c
05-04 11:29:25.606: E/ACodec(1353): OMX/mediaserver died, signalling error!
05-04 11:29:25.606: E/MediaCodec(1353): Codec reported an error. (omx error 0x8000100d, inter
05-04 11:29:25.606: E/AudioService(2462): Media server died.
05-04 11:29:25.606: E/Camera(1353): Error 100
```

1

From Grafika project, file DoubleDecodeActivity.java. I tried 3 simultaneous video(h264) decoders using MediaCodec APIs on 3 SurfaceViews. On adding 4th decoder to 4th SurfaceView to Nexus 7 with Android 5.1 CRASHES, So how many simultaneous decoders would be possible or supported.

PS. After this crash, MediaCodec doesn't work anymore. Need to restart the device to use MediaCodec.

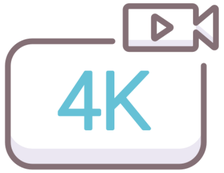
Below is the crash log. Crashes at `decoder.start()` function for 4th decoder thread.

```
com.example.app.one V/DecodeActivity: Mime: video/avc
com.example.app.one I/OMXClient: Using client-side OMX mux.
com.example.app.one V/DecodeActivity: Mime: video/avc
com.example.app.one I/OMXClient: Using client-side OMX mux.
com.example.app.one V/DecodeActivity: Mime: video/avc
com.example.app.one E/ACodec: [OMX.qcom.video.decoder.avc] storeMetaDataInBuffers failed w/ er
com.example.app.one E/ACodec: [OMX.qcom.video.decoder.avc] storeMetaDataInBuffers failed w/ er
com.example.app.one W/ACodec: do not know color format 0x7fa30c03 = 2141391875
com.example.app.one W/ACodec: do not know color format 0x7fa30c03 = 2141391875
com.example.app.one I/OMXClient: Using client-side OMX mux.
com.example.app.one V/DecodeActivity: Mime: video/avc
com.example.app.one I/OMXClient: Using client-side OMX mux.
com.example.app.one E/ACodec: [OMX.qcom.video.decoder.avc] storeMetaDataInBuffers failed w/ er
com.example.app.one E/ACodec: [OMX.qcom.video.decoder.avc] storeMetaDataInBuffers failed w/ er
com.example.app.one W/ACodec: do not know color format 0x7fa30c03 = 2141391875
com.example.app.one W/ACodec: do not know color format 0x7fa30c03 = 2141391875
com.example.app.one E/ACodec: registering GraphicBuffer 9 with OMX IL component failed: -2147
com.example.app.one V/PlayerFromFileThread: inputBuffer not available.
com.example.app.one E/ACodec: Failed to allocate buffers after transitioning to IDLE state (er
com.example.app.one E/ACodec: signalError(omxError 0x80001001, internalError -2147483648)
com.example.app.one V/PlayerFromFileThread: inputBuffer not available.
com.example.app.one E/MediaCodec: Codec reported err 0x80001001, actionCode 0, while in state
? E/ACodec: registering GraphicBuffer 4 with OMX IL component failed: -2147483648
? E/AndroidRuntime: FATAL EXCEPTION: Thread-485
                                                    Process: com.example.app.one, PID: 17143
```

기기별로 코덱 최대 사용가능량이 다르다

해당 이슈 발생 빈도가 높은 시나리오 집계

해외 단말, 저사양
보급형 기기



고해상도 영상
Overlay

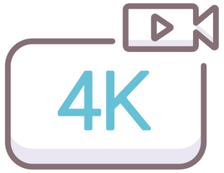
인코더를 여러 개
생성

코덱을 사용하는
다른 프로세스

기기별로 코덱 최대 사용가능량이 다르다

해당 이슈 발생 빈도가 높은 시나리오 집계

해외 단말, 저사양
보급형 기기



고해상도 영상
Overlay

인코더를 여러 개
생성

코덱을 사용하는
다른 프로세스

⇨ 인코더/디코더 동시 사용량의 제약이 있음을 추론

관련 API

getMaxSupportedInstances

Added in API level 23

```
public int getMaxSupportedInstances ()
```

Returns the max number of the supported concurrent codec instances.

This is a hint for an upper bound. Applications should not expect to successfully operate more instances than the returned value, but the actual number of concurrently operable instances may be less as it depends on the available resources at time of use.

Returns
`int`

AOSP CTS 검증


```
int getActualMax(boolean isEncoder, String name,
                 String mime, CodecCapabilities caps, int max) {
  // ...
  for (int i = 0; i < max; ++i) {
    try {
      codec = MediaCodec.createByCodecName(name);
      codec.configure(format, null, null, flag);
      codec.start();
      codecs.add(codec);
      codec = null;
    } catch {
      // ...
    } finally {
      // ...
    }
  }
}
```

CTS 작성된 코덱 생성 테스트,

생성은 가능하지만 실제로 인코딩/디코딩 시점 크래시 발생

platform/cts 저장소 master 브랜치

tests/tests/media/src/android/media/cts/MediaCodecCapabilitiesTest.java

자동화 테스트 준비

nMobile 을 이용한 물량공세 테스트

- 국내외 출시된 42개 이상의 단말에서 동시 자동화 테스트

테스트 방법

- 해상도 별 코덱 동시 반복 생성과 더미 미디어 데이터를 이용한 인코딩/디코딩 수행
- 기기의 연관 스펙 확인과 로깅 (OS Version, Display Resolution, Camera Resolution 등)

주요 검증 과제

- 해상도/FPS 와 생성 가능한 코덱 수 간의 연관 관계
- 동적 테스트 없이 기기의 최대 생성 가능 코덱 수 획득 가능 여부
- 최대 생성 가능 코덱 수 획득 동적 테스트의 서비스 시점 활용 가능 여부

테스트 결과 – 이상 현상

다수의 코덱 생성 테스트 중, 다양한 기기들에서, 다양한 문제들 발생

현상1. ‘통제할 수 없는 크래시‘ 가 발생하는 기기

현상2. 기기의 OMX Server 가 죽어버리는 기기

```
E/ACodec: OMX/mediaserver died, signalling error!
E/ACodec: signalError(omxError 0x8000100d, internalError -32)
```

현상3. OS 커널 패닉, 리부트 발생 기기

테스트 결과 – 이상 현상

다수의 코덱 생성 테스트 중, 다양한 기기들에서, 다양한 문제들 발생

현상1. ‘통제할 수 없는 크래시’ 가 발생하는 기기

현상2. 기기의 OMX Server 가 죽어버리는 기기

```
E/ACodec: OMX/mediaserver died, signalling error!
E/ACodec: signalError(omxError 0x8000100d, internalError -32)
```

현상3. OS 커널 패닉, 리부트 발생 기기

⇨ 런타임에서의 최대 생성 가능수 동적 테스트 루틴을 작성, 운용하기에는 너무 위험하다

테스트 결과

해상도, 비트레이트 별 코덱 생성 시도, 인코딩/디코딩 제외

		1280x720@30, 4500 Kbps			1280x720@30, 10000 Kbps			1920x1080@30, 20000 Kbps		
		Baseline@ L3.1	Main@ L3.1	High@ L3.1	Baseline@ L3.1	Main@ L3.1	High@ L3.1	Baseline@ L4	Main@L4	High@L4
A제조사 고급기A Andorid 8.0.0, API 26	Max	16	16	16	16	16	16	16	16	16
	actualMax	14	14	14	14	14	14	14	14	14
B제조사 중급기A Android 7.0, API 24	Max	16	16	16	16	16	16	16	16	16
	actualMax	16	16	16	16	16	16	16	16	16
A제조사 고급기B Android 8.0.0, API 26	Max	16	16	16	16	16	16	16	16	16
	actualMax	14*	14*	15*	15*	15*	15*	14*	15*	15*
B제조사 보급기A Android 6.0.1, API 23	Max	9	9	9	9	9	9	4	4	4
	actualMax	9	9	9	9	9	9	4	4	4
B제조사 보급기B Android 5.0, API 21	Max	9	9	9	9	9	9	4	4	4
	actualMax	9	9	9	9	9	9	4	4	4

* OMX/mediaserver died, signalling error!

일부 단말에서 인코딩/디코딩을 하지 않을 경우 결과 추론이 어려움, 해상도는 생성 수에 영향을 미칠 수 있음

테스트 결과

FHD, 2개의 인코더와 n개의 디코더 생성 (인코딩/디코딩 수행)

Encoder : 1920x1080@30fps, 20Mbps, High@L4, VBR Decoder : 1920x1080@30fps, 20Mbps, High@L4, VBR, 120 op-rate			
		2 enc + n-dec	n-dec
제조사 B 중급기	Android 7.0, API 24	14	16
외산제조사 A 보급기	Android 6.0.1, API 23	7*	9*
제조사 B 보급기A	Android 6.0.1, API 23	2	4
제조사 B 보급기B	Android 5.0, API 21	2	4
제조사 A 고급기A	Android 8.0.0, API 26	14*	16
제조사 A 고급기B	Android 8.0.0., API 26	12*	14

* OMX/mediaserver died, signalling error!

인코더와 디코더는 생성 가능 수에 상호 영향을 미치는 것으로 확인

테스트 결과

FHD, HD 인코더 각 1개씩 총 2개와 n 개의 UHD 디코더 생성 (인코딩/디코딩 수행)

		2 enc + n-dec (UHD)	
		1920x1080	1280x720
제조사 B 중급기	Android 7.0, API 24	3	3
외산제조사 A 보급기	Android 6.0.1, API 23	5*	7*
제조사 B 보급기A	Android 6.0.1, API 23	0	0
제조사 B 보급기B	Android 5.0, API 21	0	0
제조사 A 고급기A	Android 8.0.0, API 26	7	8***
제조사 A 고급기B	Android 8.0.0., API 26	10	10
외산제조사 B 고급기	Android 8.1.0, API 27	12**	12***

* OMX/mediaserver died, signalling error!

일부 단말에서는 UHD 디코더를 생성하지 못하는 경우 확인

자동화 테스트

DEVIEW 2019

Device Model	Android OS Version	Max. Camera Resolution	Max. Camera FPS (>=30)	Max. High Speed Camera Resolution	Max. High Speed Camera FPS	1080p AVC Enc 2 + 2160p AVC Dec	1080p AVC Enc 1 + 2160p AVC Dec	1080p AVC Enc 2 + 1080p AVC Dec	1080p AVC Enc 1 + 1080p AVC Dec
고급기A	Android 8.0.0	4032x3024	30	1280x720	240	3	3	14	16
고급기B	Android 9	4032x3024	30	1920x1080	240	3	4	16	16
고급기C	Android 9	4032x3024	30	1920x1080	240	3	4	16	16
보급기A	Android 6.0.1	1920x1080	N/A	N/A	N/A	0	0	2	3
보급기B	Android 6.0.1	3840x2160	30	1280x720	120	0	0	2	3
중급기A	Android 8.0.0	4032x3024	30	1280x720	240	1	1	7	7
중급기B	Android 7.0	4032x3024	30	1280x720	240	1	1	6	7
중급기C	Android 9	4032x3024	30	1280x720	240	3	4	16	16
고급기D	Android 9	4032x3024	30	1920x1080	240	3	4	16	16

테스트 결과

단말, 하드웨어 사양 별 인코더/디코더 생성 최대치 한계 확인

- 코덱을 사용하는 기능 구현시 한계를 고려해서 개발

테스트 결과 데이터베이스 구성, 정적으로 활용하는 방법 고려

- 다른 프로세스에서 코덱을 사용중인 경우에 대한 능동적 대응 어려움

최대 생성 가능 수를 확인할 수 있는 동적 테스트 메소드 작성 어려움

- 크래시 및 기기 리부팅과 같은 핸들링이 어려운 치명적인 문제 발생
- '최대 생성 가능 수' 검증이 아닌, '특정 기능이 필요로하는 최소한의 코덱 수' 검증 메소드 작성으로 선회

테스트 결과

단말, 하드웨어 사양 별 인코더/디코더 생성 최대치 한계 확인

- 코덱을 사용하는 기능 구현시 한계를 고려해서 개발

테스트 결과 데이터베이스 구성, 정적으로 활용하는 방법 고려

- 다른 프로세스에서 코덱을 사용중인 경우에 대한 능동적 대응 어려움

최대 생성 가능 수를 확인할 수 있는 동적 테스트 메소드 작성 어려움

- 크래시 및 기기 리부팅과 같은 핸들링이 어려운 치명적인 문제 발생
- '최대 생성 가능 수' 검증이 아닌, '특정 기능이 필요로하는 최소한의 코덱 수' 검증 메소드 작성으로 선회

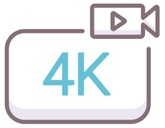
DEVIEW
2019

네이버 동영상플랫폼 모바일

네이버 동영상플랫폼 모바일

DEVIEW
2019


송출/수신

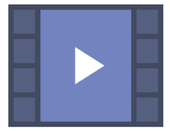


영상처리


영상합성


외부장치


AR/VR


플레이어


오디오처리

마무리

당연하다고 알려진 사실에 대한 의심과 검증의 자세

새로운 기술은 물론, 안정적인 서비스를 위한 기술적 토대와 연구 지속

동영상, 강력한 상대들이 즐비한 어려운 싸움. 개발자는 개발에 정진하며 겸허한 도전자의 자세로.