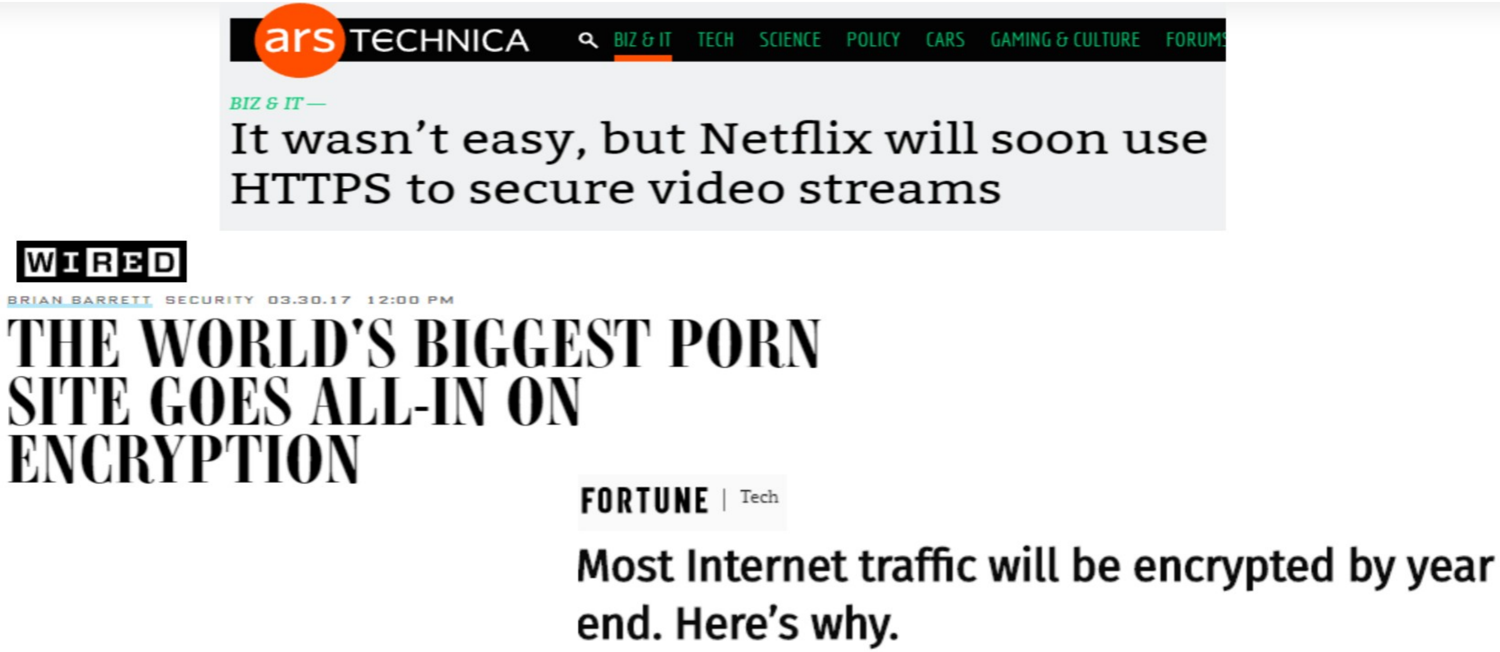
Beauty and the Burst : Remote Identification of Encrypted Video Streams

Roei Schuster, Vitaly Shmatikov, Eran Tromer

- USENIX Security 2017 -

Key idea of the paper :

- Goal of the paper : « Can an attacker know what video is the victim watching? »
- Nowadays , all traffic is encrypted with HTTPS.



BIZ & IT —

It wasn't easy, but Netflix will soon use HTTPS to secure video streams

WIRED

BRIAN BARRETT SECURITY 03.30.17 12:00 PM

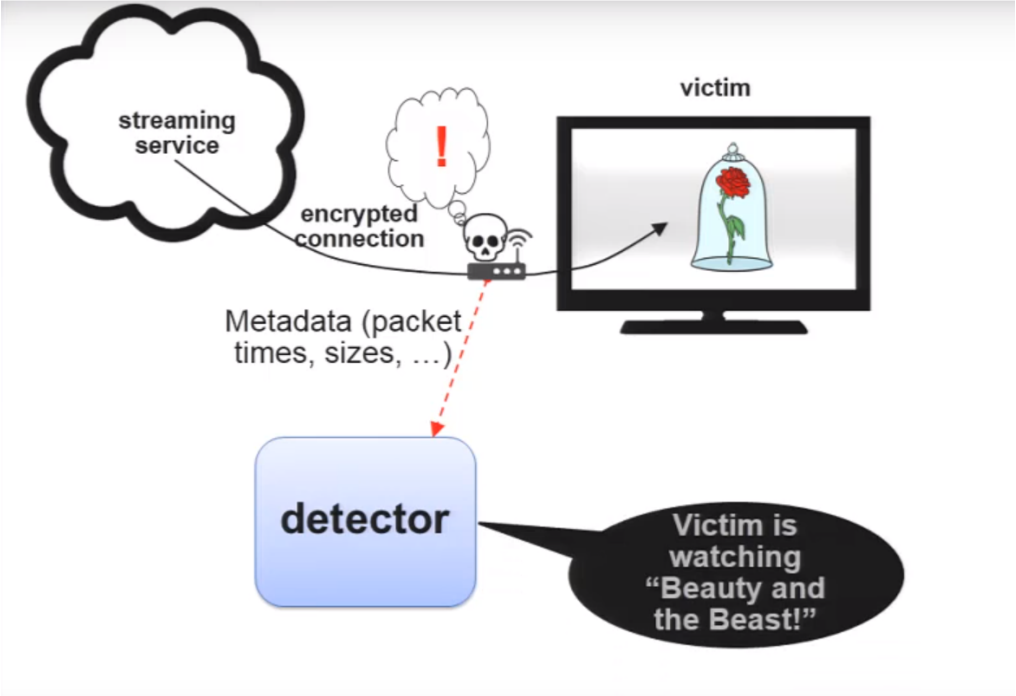
THE WORLD'S BIGGEST PORN SITE GOES ALL-IN ON ENCRYPTION

FORTUNE | Tech

Most Internet traffic will be encrypted by year end. Here's why.

Key idea of the paper :

- Authors solution : Analyze the traffic to identify the video


streaming service
victim
!
encrypted connection
Metadata (packet times, sizes, ...)
detector
Victim is watching "Beauty and the Beast!"

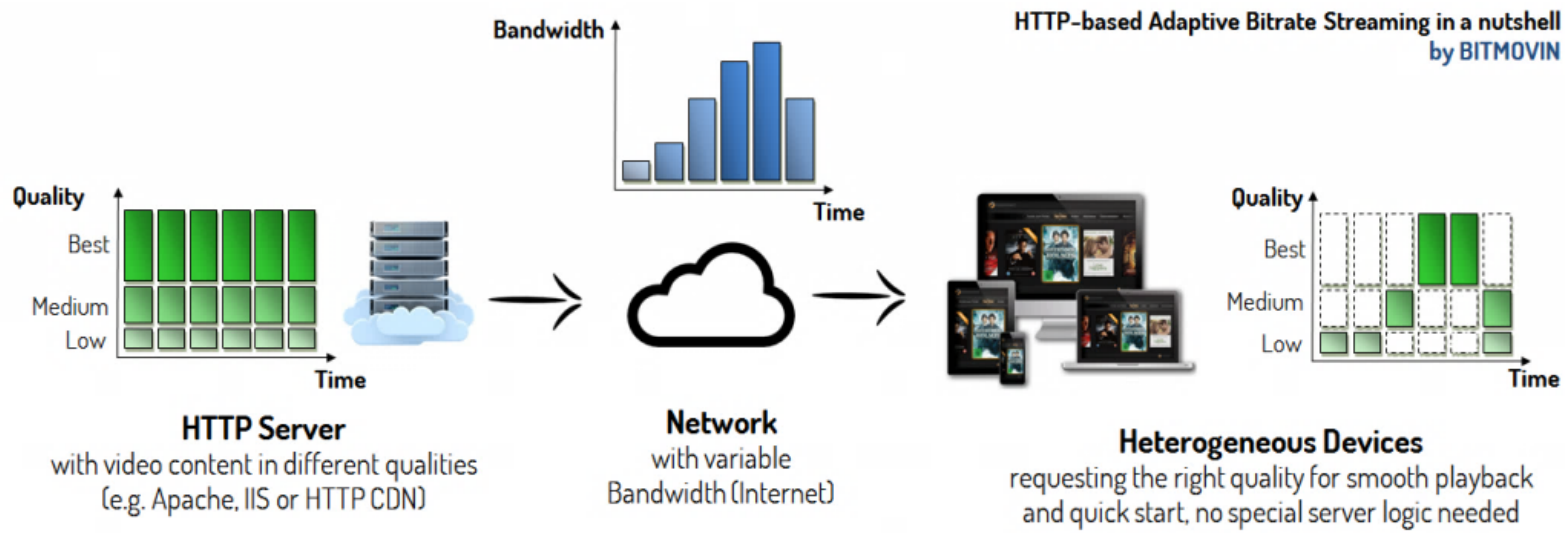
How can the attacker monitor the traffic ?
How can we identify the video with traffic analysis ?

Summary of the paper :

- Problem :
  - Video traffic is encrypted and is hard to identify.
  - MPEG-DASH (video streaming technology) leaks information related to the video and leaks its identity.
- Contributions :
  - Encrypted streaming traffic has a fingerprint related to each video.
  - New video identification method using a CNN architecture.
  - Shows that video identification does not require direct access to the network of the victim
- Results :
  - Detection of Youtube content with a precision of 99% and 0 false positives (0,988 recall)
  - Detection of Netflix content with a precision of 98% and a 0,0005 false positive rate (0,93 recall)

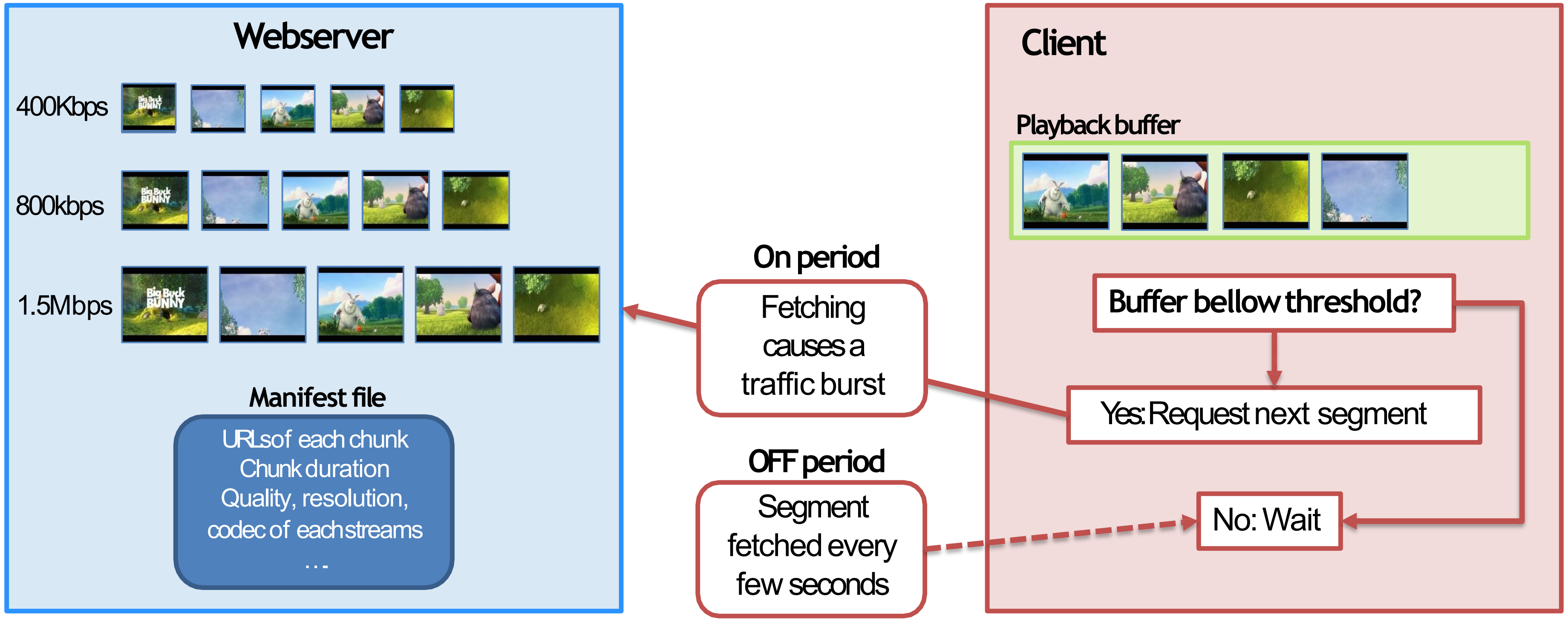
What is MPEG-DASH ?

- An adaptive bitrate streaming technique that enables high quality streaming of media content over the Internet delivered from conventional HTTP web servers.
- How does it work ?


HTTP-based Adaptive Bitrate Streaming in a nutshell
by BITMOVIN
Bandwidth
Time
Quality
Best
Medium
Low
Time
HTTP Server
with video content in different qualities
(e.g. Apache, IIS or HTTP CDN)
Network
with variable
Bandwidth (Internet)
Quality
Best
Medium
Low
Time
Heterogeneous Devices
requesting the right quality for smooth playback
and quick start, no special server logic needed

The principal of the video identification :

- Initial burst to fill the playback buffer
- On and off periods (fetching data from server) which causes the burst pattern.



Webserver

400Kbps

800kbps

1.5Mbps

Manifest file

URLsof each chunk
Chunk duration
Quality, resolution,
codec of eachstreams
…

On period

Fetching causes a traffic burst

OFF period

Segment fetched every few seconds

Client

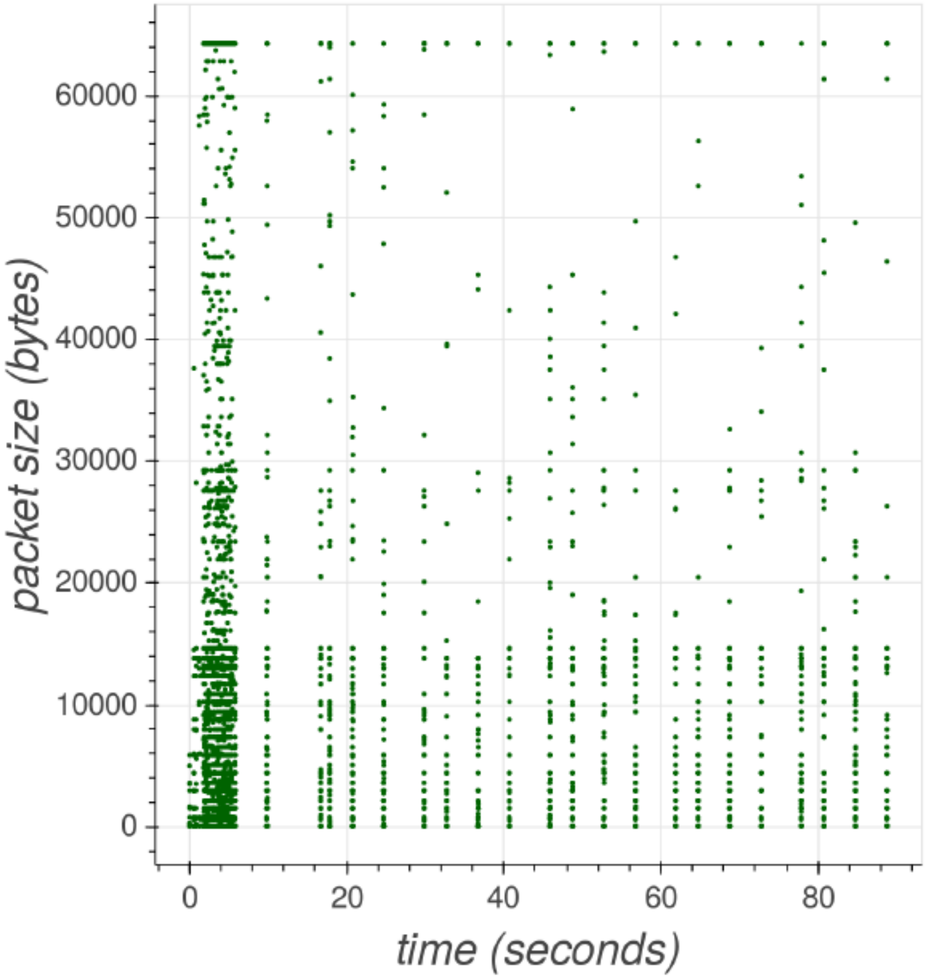
Playback buffer

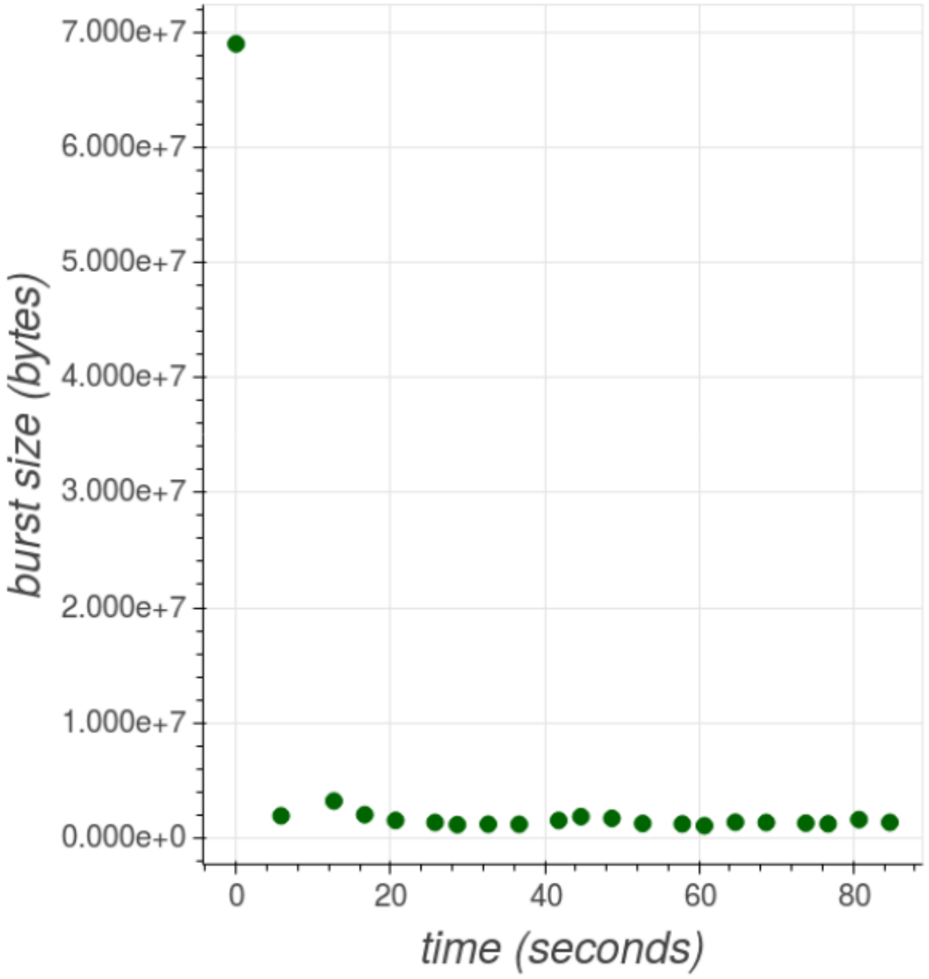
Buffer bellow threshold?

Yes:Request next segment

No: Wait

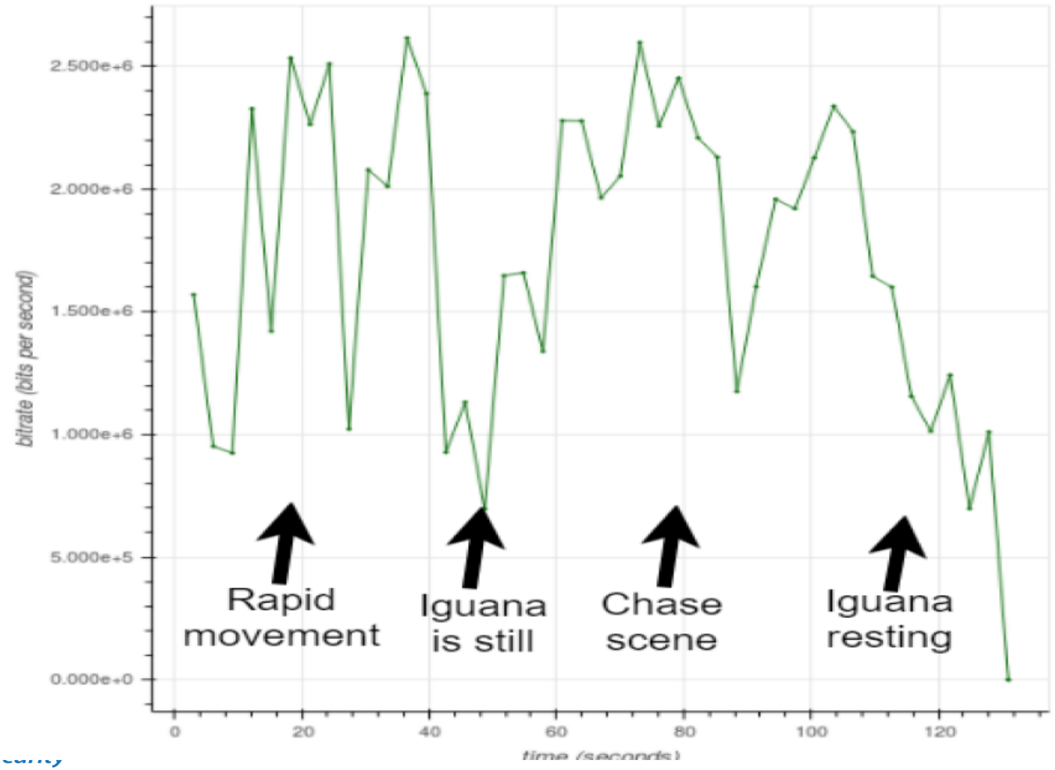
Variable burst size :

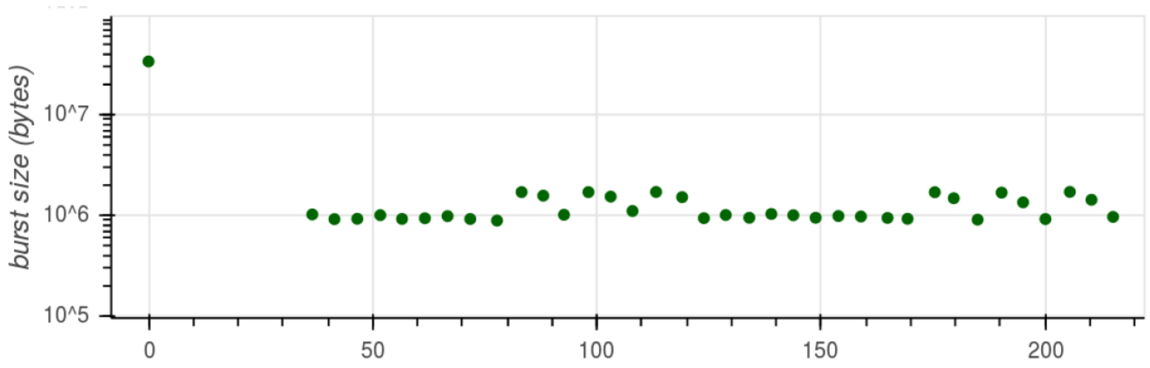




Variable bitrate encoding :

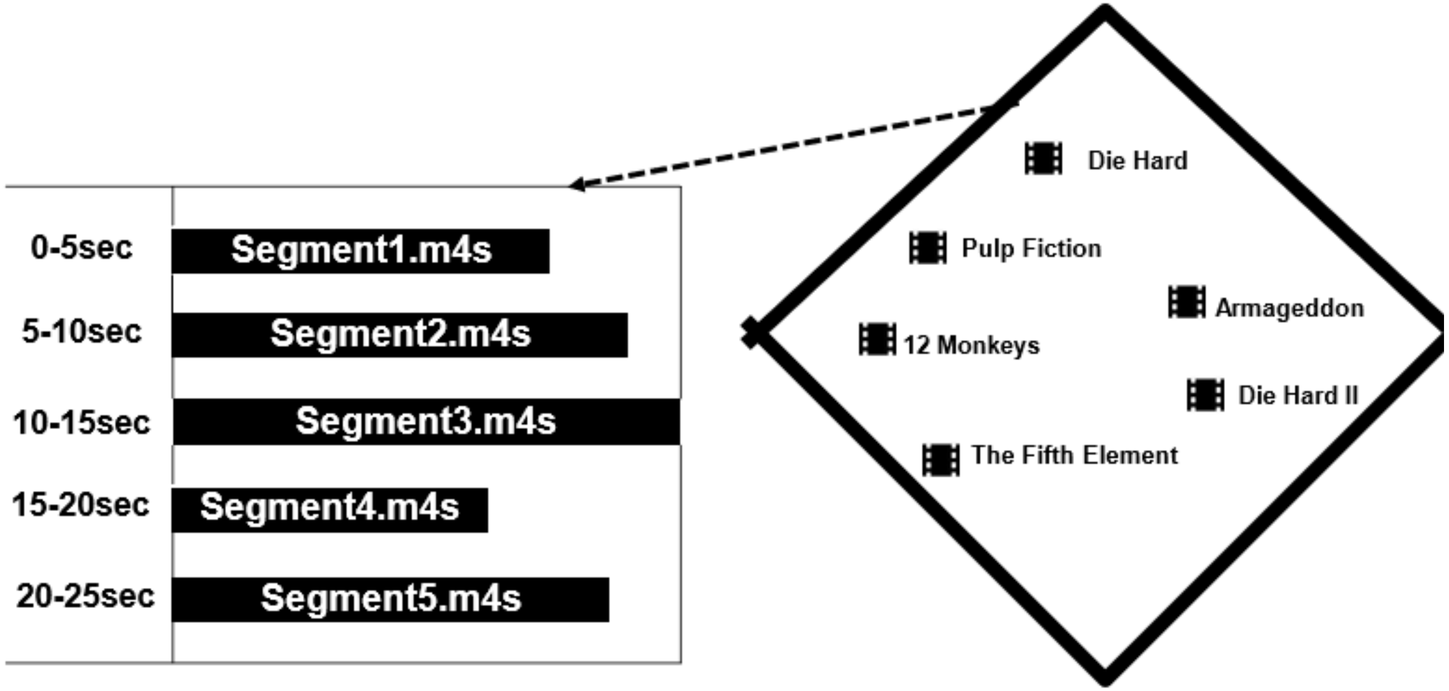
- Different video segments require different amount of bytes to encode:
  - Action : high bitrate
  - Slow scene : low bitrate





Variable bitrate = Variable segment size :

- Each video stream is different : not the same content.
- Each segment has different motion level and intensity.
- Every segment is different.



From a leak to a fingerprint :

- Can we link a burst pattern to a video ?
- Does the burst pattern uniquely characterize a video ?
- Is it possible to learn a title's burst pattern ?

We need diversity and consistency !

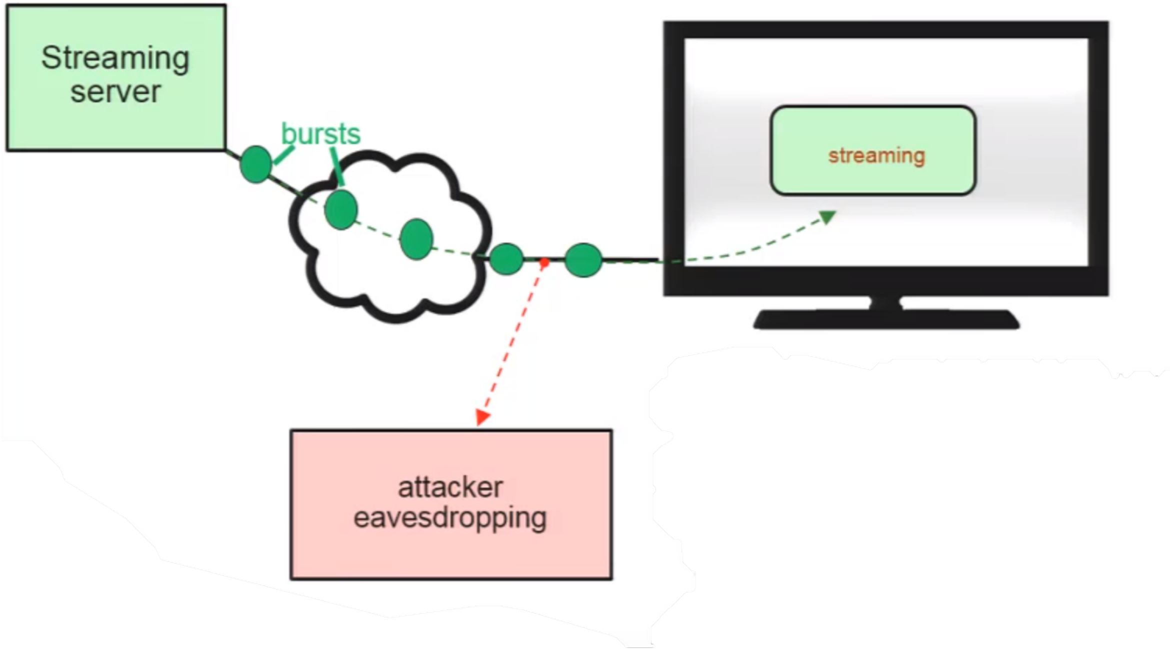
Attack overview :

- Creating detectors :
  - Gathering phase :
    - Attacker streams the video and captures the traffic
  - Training Phase :
    - Detector for each segmented video he wants to identify
- Applying the detectors :
  - Attacker measures the victim's network traffic
    - On-Path attack
    - Off-Path attack
  - Video traffic easily recognizable because of coarse-grained features
  - Apply detectors and determine the video title.

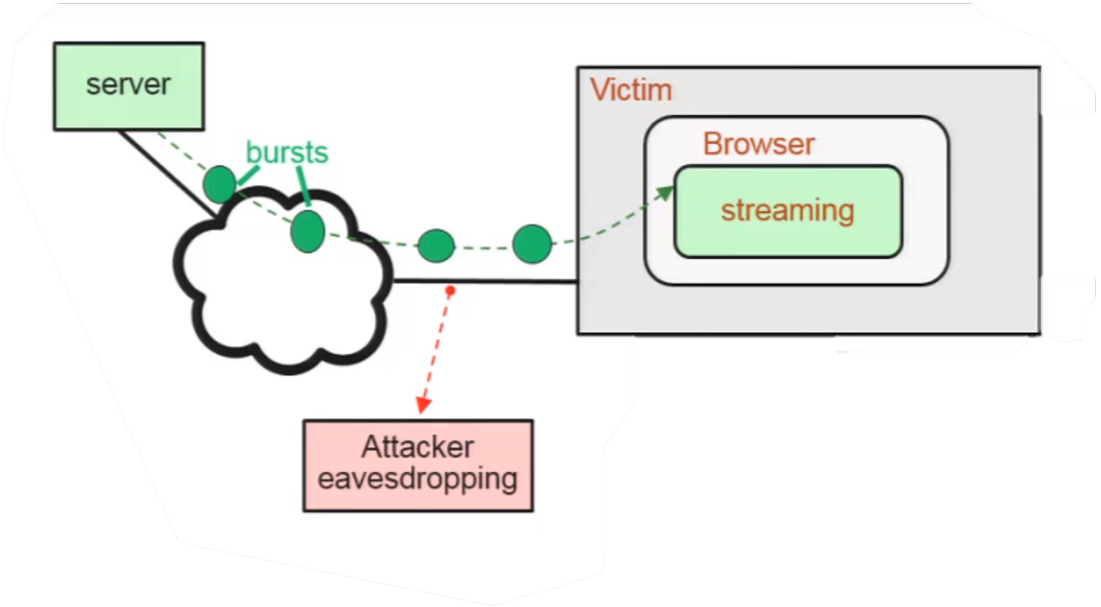
Attack scenario :

- How can the attacker monitor and measure the victim's traffic ?

On path :

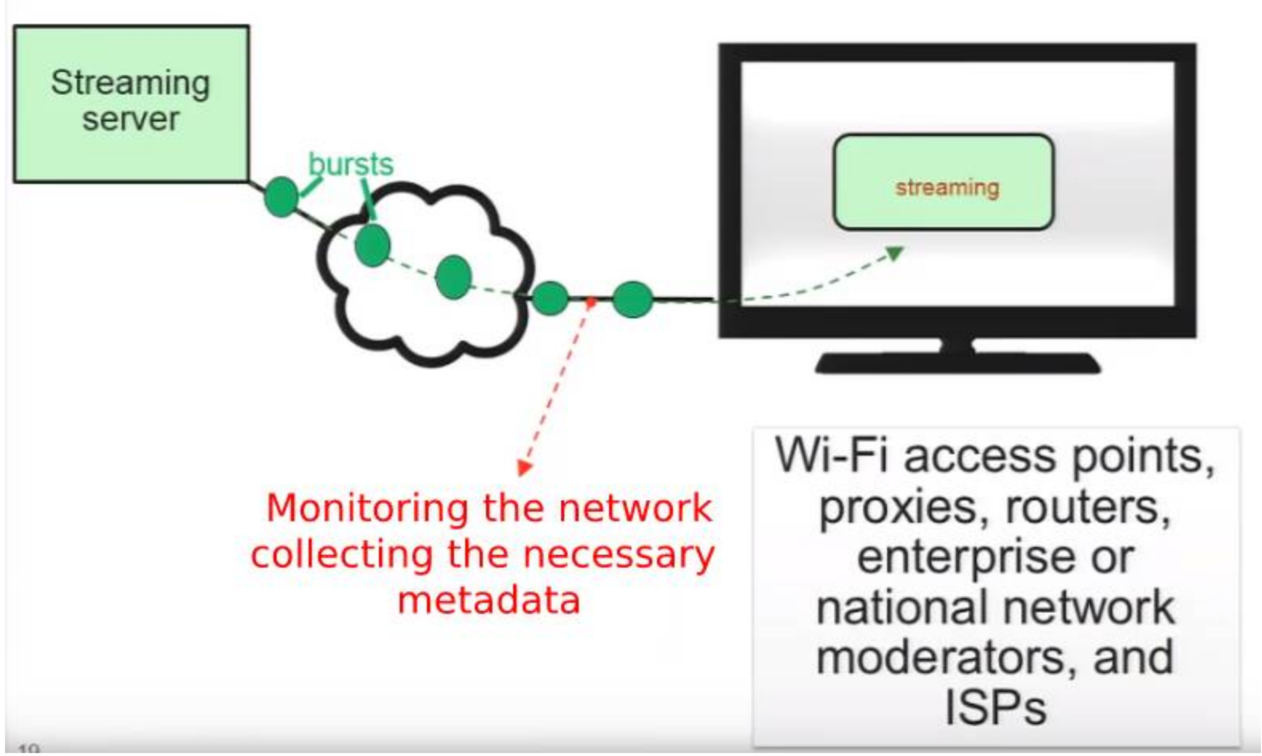


Off-path :



Scenario 1 : On-path attack

- Attacker has on-path access to the victims network at the network layer or transportation layer



Monitoring the network collecting the necessary metadata

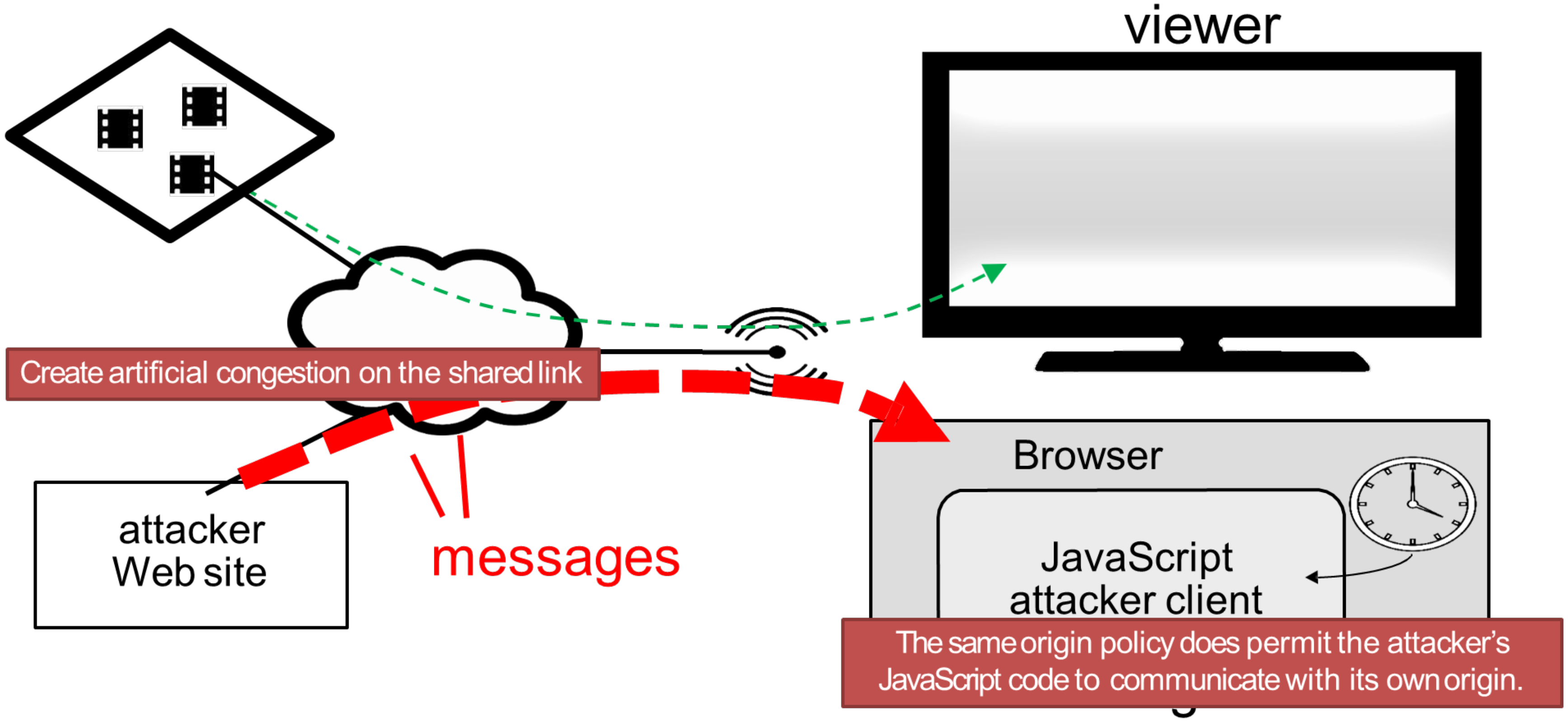
Wi-Fi access points, proxies, routers, enterprise or national network moderators, and ISPs

Scenario 2 : Off-path attack

- Two types of off-path attack :
  - Cross-device :
    - Attack is running on the same network (e.g 2 computers using the same access point)
    - Execute JavaScript in the different machine on the same local network
  - Cross-site :
    - Attack is running on the same device but in a different tab of the browser or an other browser
    - Execute JavaScript in the victim's Web browser in different tab / embedded ad

- Why are those scenario's possible ?
  - Attacker and the victim share the same limited resource
  - Possible to manipulate this shared link to create congestion

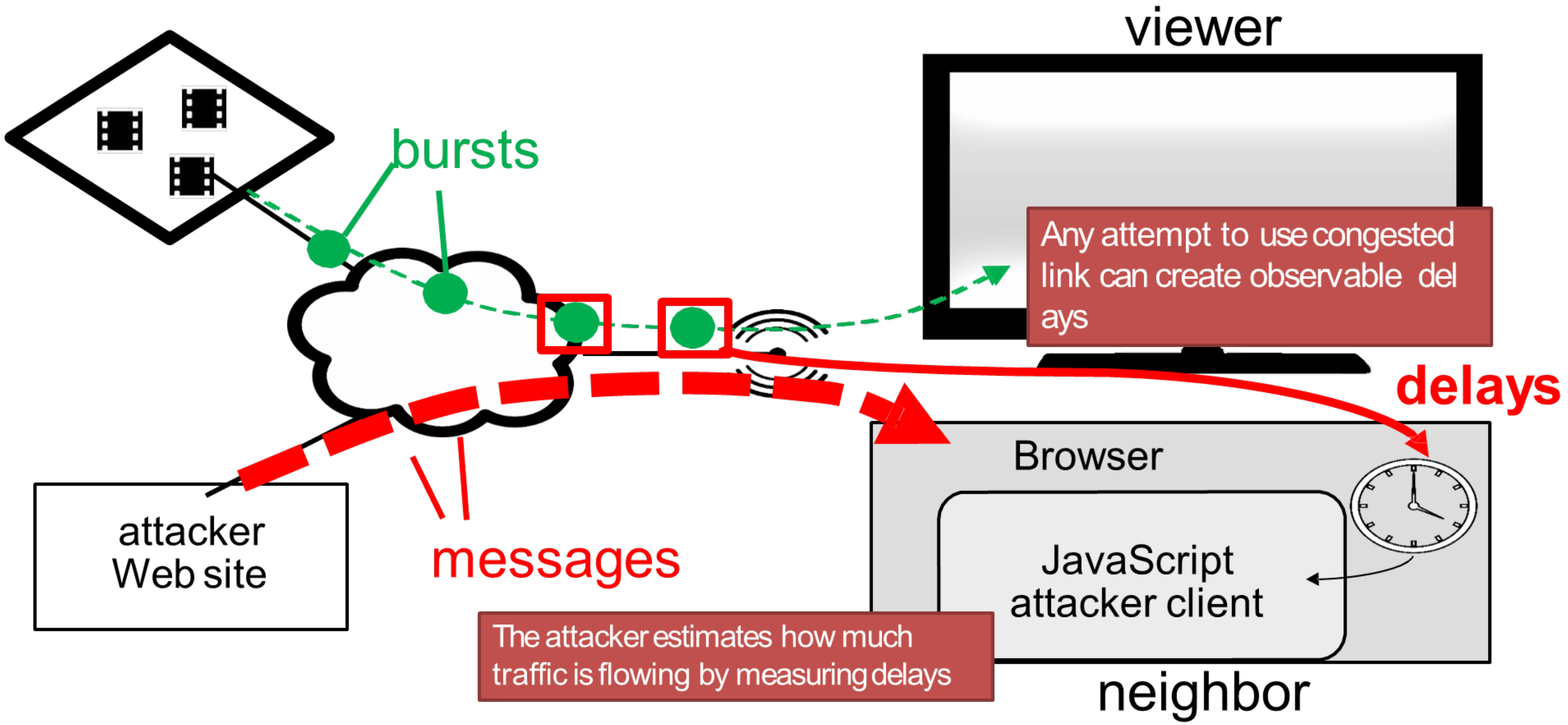
Scenario 2 : Off-path attack



Create artificial congestion on the shared link

The same origin policy does permit the attacker's JavaScript code to communicate with its own origin.

Scenario 2 : Off-path attack

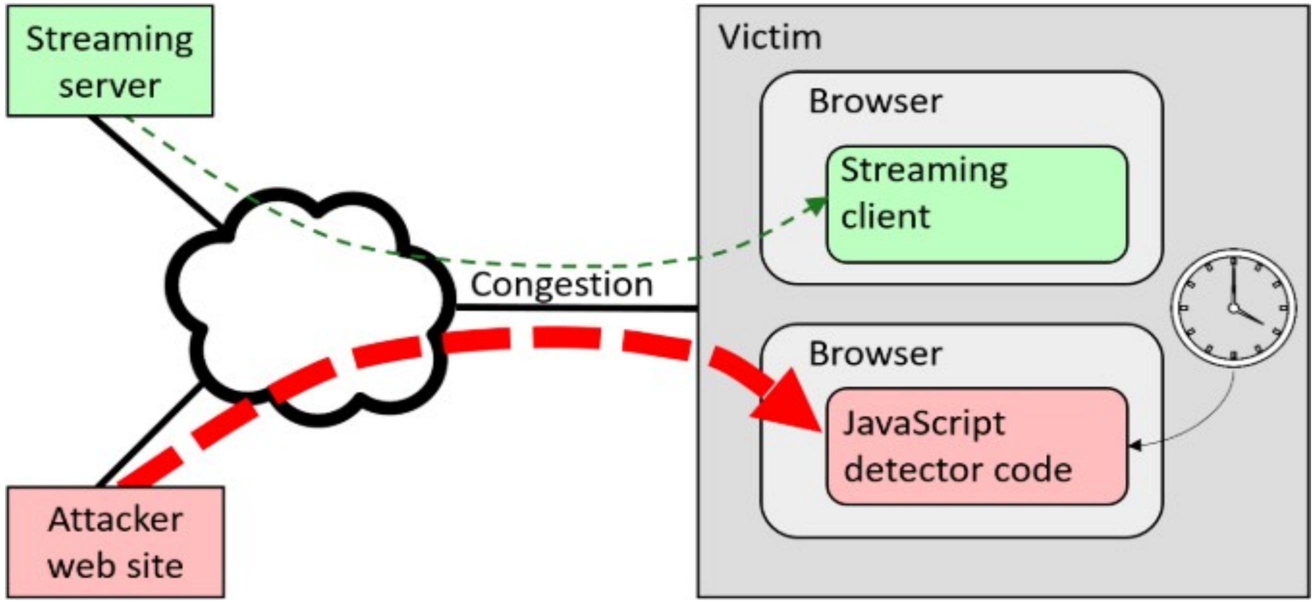


Any attempt to use congested link can create observable del ays

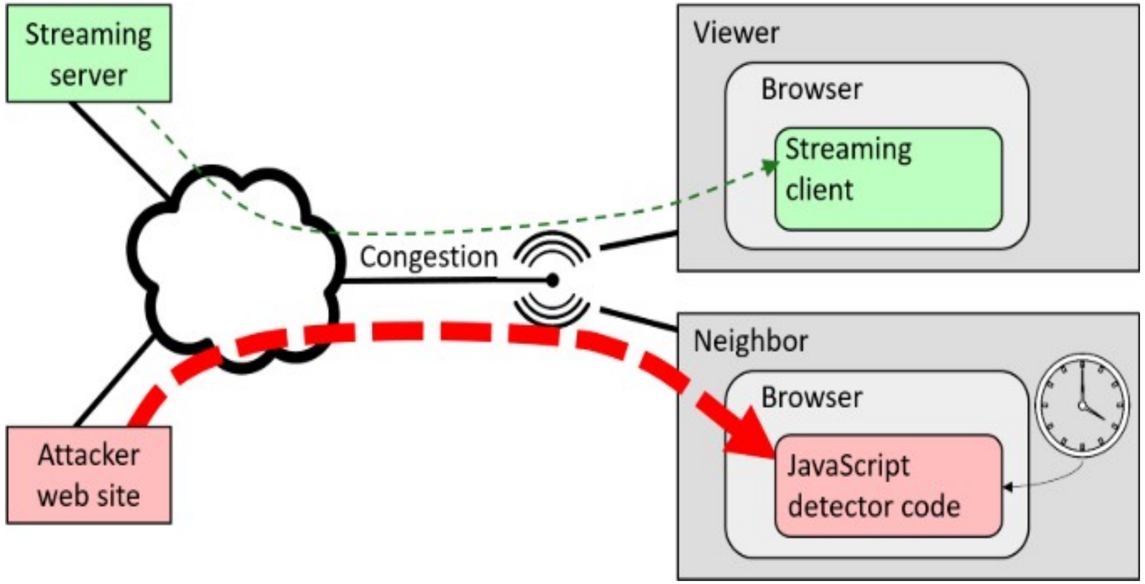
The attacker estimates how much traffic is flowing by measuring delays

Scenario 2 : Off-Path

Cross-Site :



Cross-device :



Theoretical experiments :

- Fingerprinting algorithm :

$$\alpha(v) \equiv (\underbrace{v_1, \ldots v_k}_{\text{Absolute magnitudes of segment sizes}}, \underbrace{v_2 - v_1, \ldots v_k - v_{k-1}}_{\text{Variability pattern}})$$

- Modelling the attacker :
  - Training period :
    - TS = $\{t_1, t_2, \ldots, t_n\}$ where t is the size of a burst.
    - s^m=mean(TS) (m is the video)
    - $\alpha(s^m)$ is the attacker's fingerprint of the m video
  - Attacking period :
    - $\alpha(t)$ is the traceprint
    - If $||\alpha(t)-\alpha(s^m)||_1 \leq B$, where B = 3.5 Mbytes , the victim is watching the m video

Theoretical experiments :

- Attacker's recall (true positive rate):
  - They estimated the error by lower-bounding the probability
  - Evaluation with traces from 100 streams of the same video from Youtube

$$\Pr_{t \leftarrow T^m}\left[\|\alpha(t) - \alpha(s^m)\|_1 \leq B/7\right] \geq 1 - 10^{-12}$$

  - The distance is small between the attacker's traceprint and the video's fingerprint which implies a very high recall.

Theoretical experiments :

- Attacker's precision :
  - Possible misclassification if 2 fingerprints are too close ?
  - From a pool of 3558 YouTube videos, they extracted the ones with the V variable segment:
    - Overall Bitrate > 100 KBps
    - Half of the size difference between segments >110 KB.
    - V : 671 , no fingerprints in V are 2B-Close in the L1 Norm (Manhattan distance)

Why those values ?

  - Almost 20% of the dataset is different and have its own fingerprint, misclassification is unlikely.

  - What about the rest of the dataset ?

Theoretical experiments :

- Does the burst pattern uniquely characterize a video ?

➔ Empiric measures on 3500 YouTube videos.

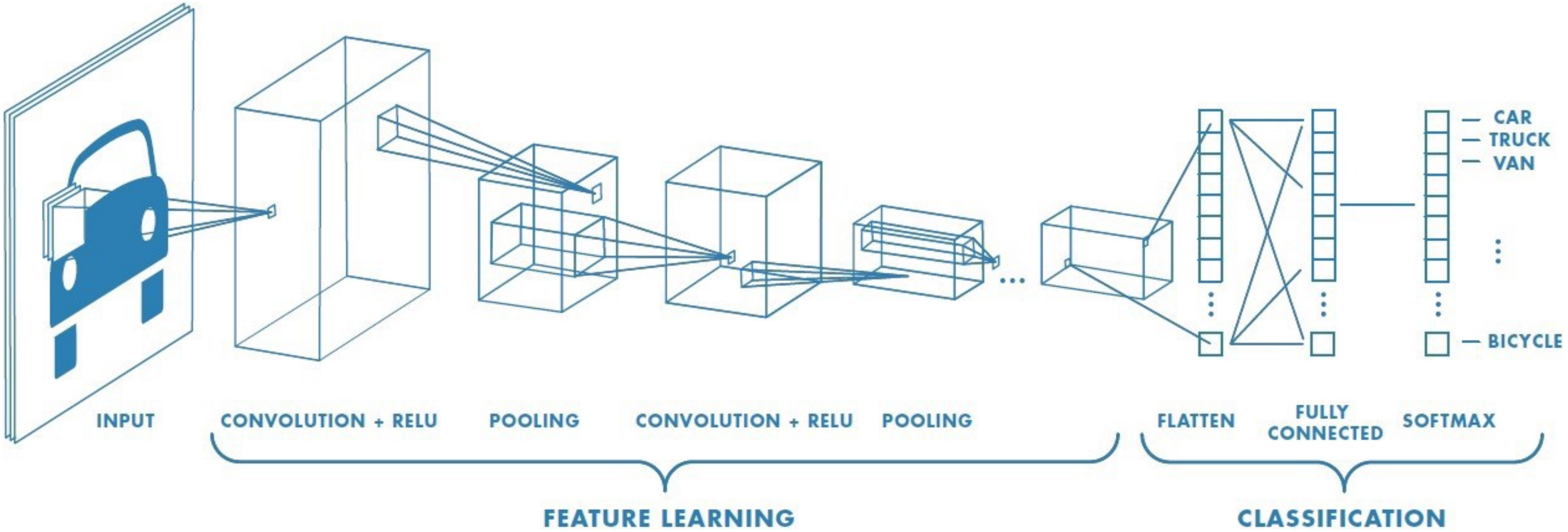
- Is it possible to learn a title’s burst pattern ?

➔ Empirically evaluated the attacker’s measurement error bound.

Convolutional Neural Networks :

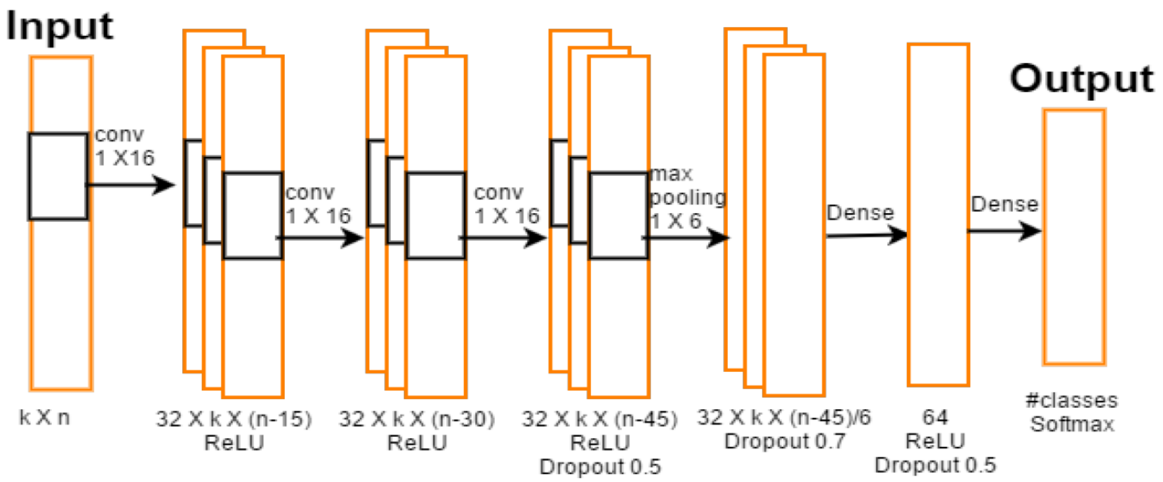
- Very good at learning high-level concepts that are hard to express formally
- Structure :
  - Convolutional layer : Feature extractor
    - Use multiple convolution filters over inputs
    - Convolution filter + Activation function (ReLU)
  - Pooling (subsampling) layer
    - Max/Average pooling
    - Make the representations smaller
- Advantages :
  - Reduce the number of parameters
  - Learn local features

Convolutional Neural Networks :



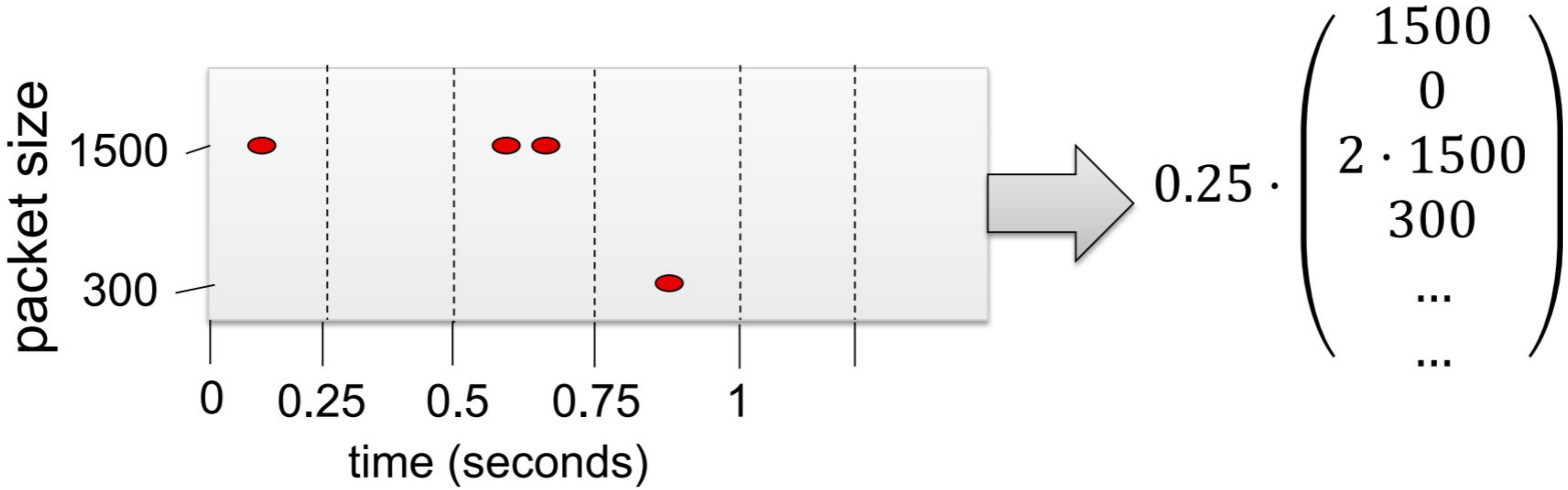
CNN Detectors :

- Advantage of using a CNN detector :
  - Robust : can operate on noisy and coarse measurements
  - Agnostic to protocol-specific attributes (e.g., QUIC vs. TLS)
  - Can learn features other than burst patterns
    - Arrival patterns of individual packets
  - Can use multiple session representations, train on all at once
  - Produce representations of local features or temporally local in a time series



CNN Detectors :

- Features used for training CNN detectors :
  - Down/Up/All bytes per second
  - Down/Up/All packets per second
  - Down/Up/All packets length
- Unified vectors : sampled every 0,25 second
- Averaging over 0,25 seconds intervals



CNN Detectors

- Construction of the dataset :
  - Automated Capture:
    - One chrome browser per title + service-specific “rewind”
    - Used WireShark’s tshark : Amazon, Netflix and Vimeo ➔ TLS protocol
      Youtube ➔ TLS or QUIC protocol
  - Feature Extraction:
    - TCP flow with the greatest amount of bits.
    - Flow attributes : down/up/all Bytes per seconds , down/up/all packets per seconds, down/up/all average packet length.
    - Uniformly sized vectors : aggregated the series into 0.25 seconds chunks by averaging over 0.25 seconds interval

CNN Detectors

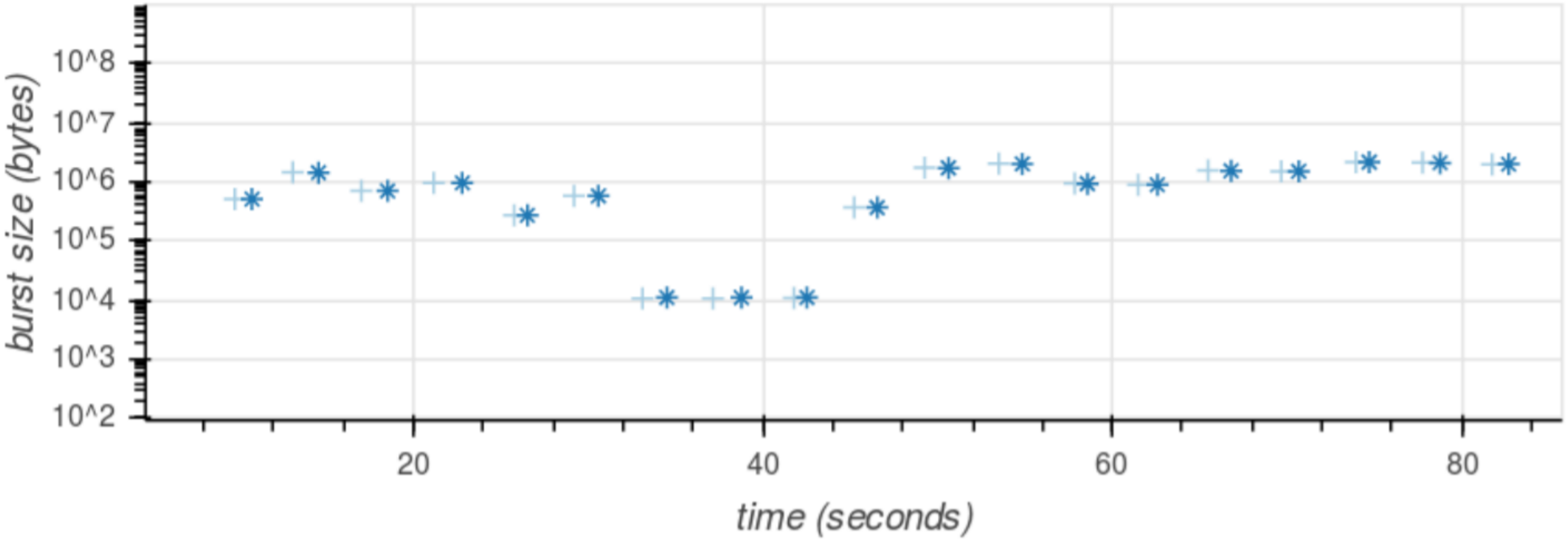
- Dataset used :

	Videos	Sessions	Time per session (seconds)	Classes	Accuracy
Netflix	100	100	60	100	98,5%
Amazon	10	100	90	10	92,5%
Vimeo	10	100	60	10	98,6%
Youtube	18	100	180	18	99,5%
	3500	1	180	1	

Why 10 or 100 classes ?

Was identification the target of the paper or detection ?

Network-Agnostic attack :



Reservoir Dogs

- Highly correlated burst patterns !

+ = Campus network

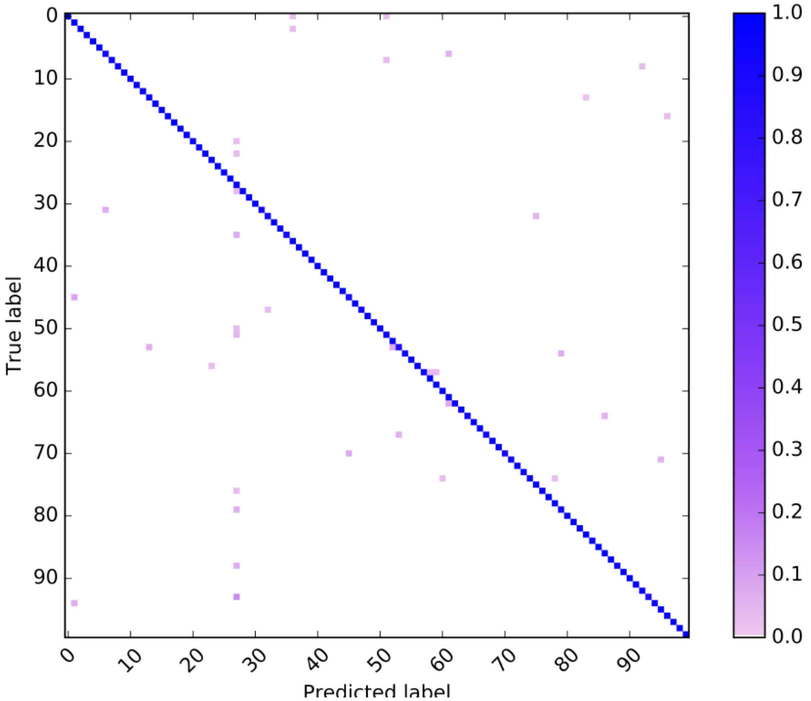
* = Home network

Results

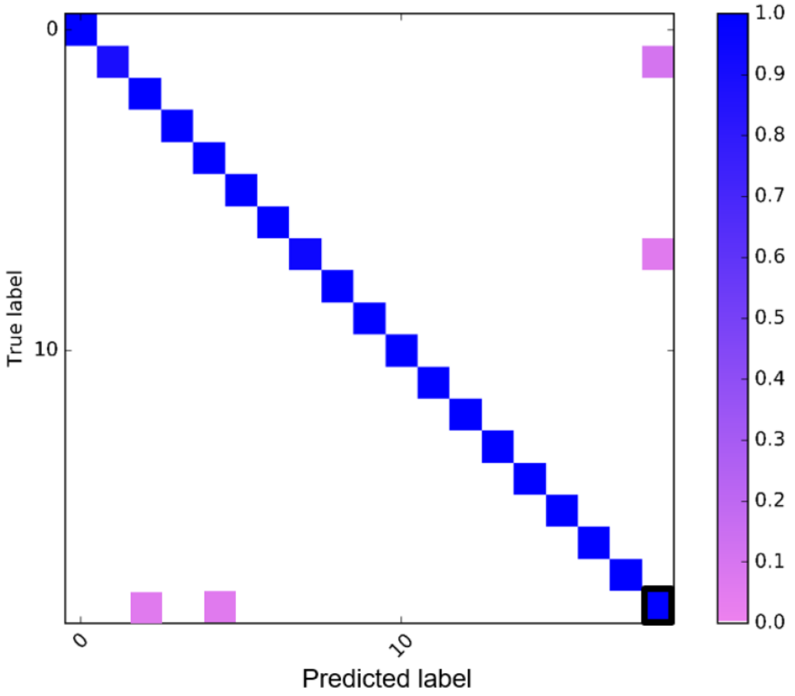
CNN Detectors :

- Confusion matrix :

Netflix :



YouTube:

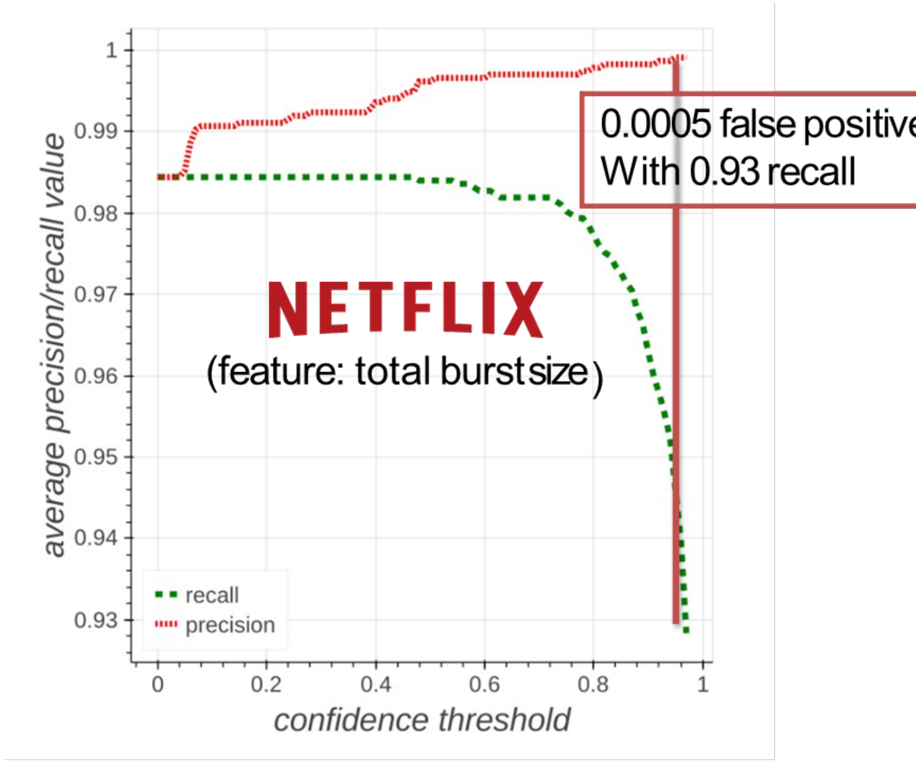


Web Security & Privacy Lab

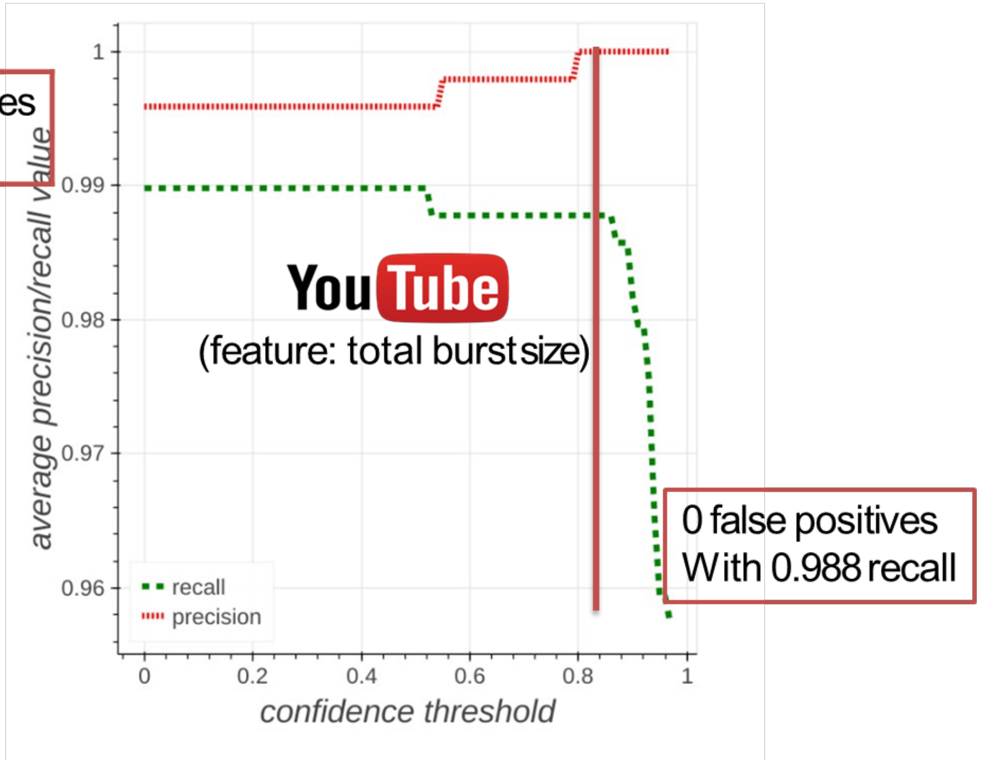
KAIST

CNN Detectors :

- Confidence threshold :

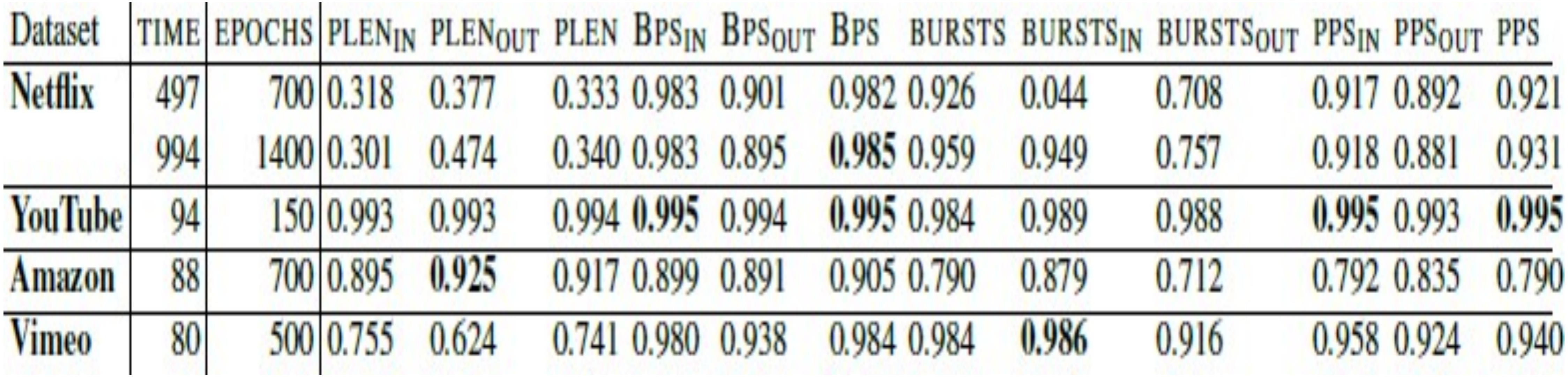


0.0005 false positives
With 0.93 recall



0 false positives
With 0.988 recall

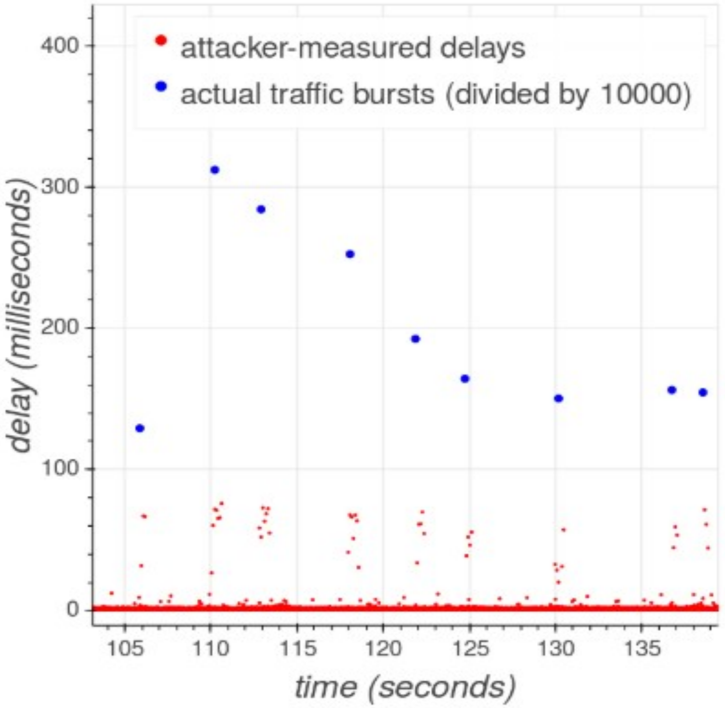
Accuracy on various features :

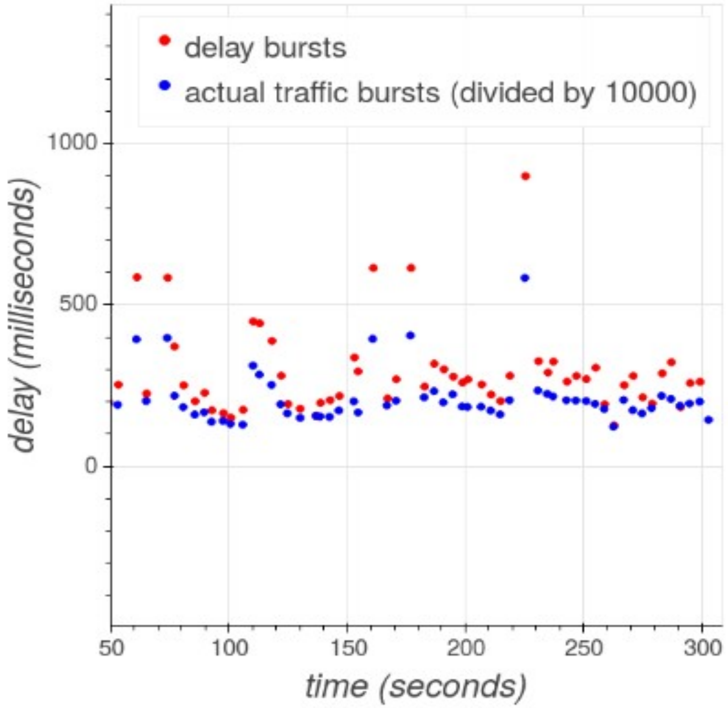


Dataset	TIME	EPOCHS	PLEN_{IN}	PLEN_{OUT}	PLEN	BPS_{IN}	BPS_{OUT}	BPS	BURSTS	$\text{BURSTS}_{\text{IN}}$	$\text{BURSTS}_{\text{OUT}}$	PPS_{IN}	PPS_{OUT}	PPS
Netflix	497	700	0.318	0.377	0.333	0.983	0.901	0.982	0.926	0.044	0.708	0.917	0.892	0.921
	994	1400	0.301	0.474	0.340	0.983	0.895	**0.985**	0.959	0.949	0.757	0.918	0.881	0.931
YouTube	94	150	0.993	0.993	0.994	**0.995**	0.994	**0.995**	0.984	0.989	0.988	**0.995**	0.993	**0.995**
Amazon	88	700	0.895	**0.925**	0.917	0.899	0.891	0.905	0.790	0.879	0.712	0.792	0.835	0.790
Vimeo	80	500	0.755	0.624	0.741	0.980	0.938	0.984	0.984	**0.986**	0.916	0.958	0.924	0.940

Delay-bursts and actual bursts :

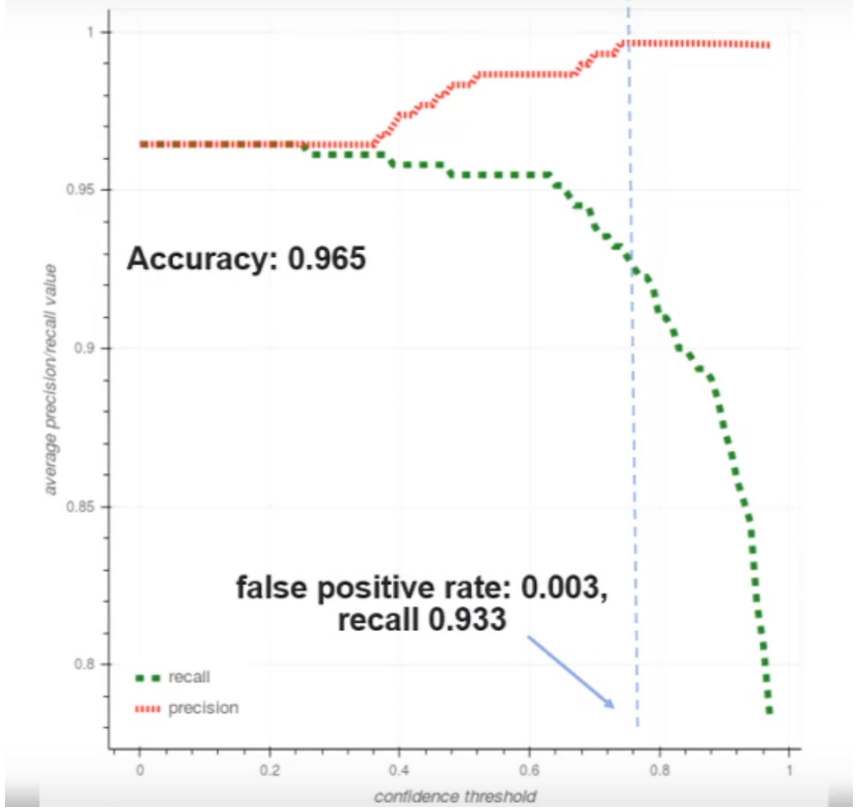
- Delay-bursts time series: the delays induced by traffic bursts
  - For each traffic burst, compute aggregate delay induced
- Use resulting time-series as input to neural network





Results of the cross-site and cross-device

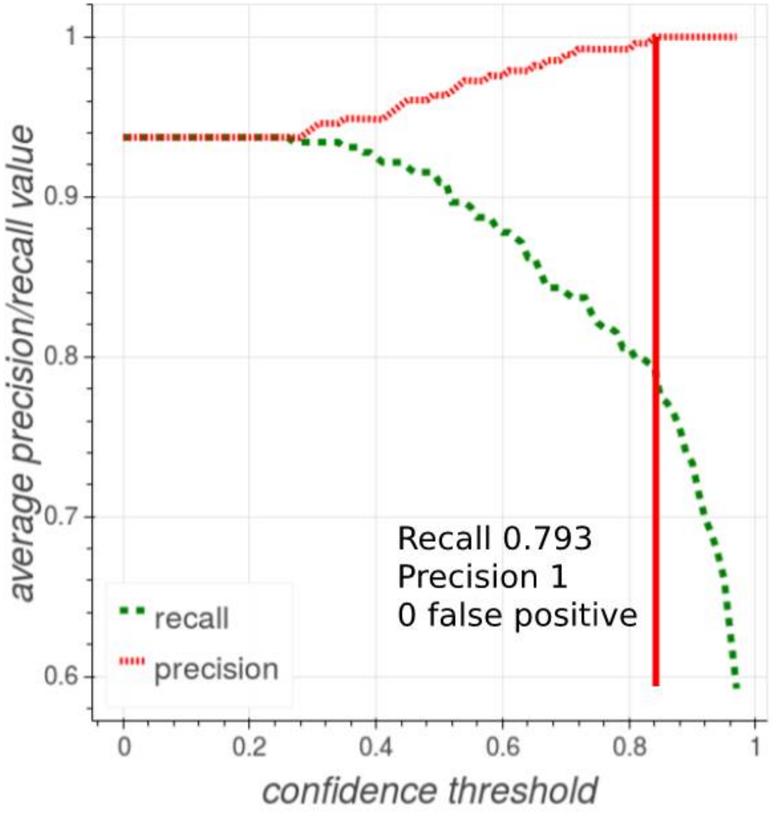
- Cross-device :



Accuracy: 0.965

false positive rate: 0.003, recall 0.933

Cross-site :



Recall 0.793
Precision 1
0 false positive

(c) Precision vs. recall of our classifier

Mitigations :

- Root cause: Modern streaming traffic characteristics
  - Title bitrate pattern is unique when sampled at few-seconds granularity
  - Fetching at segment granularity (= every few seconds)
- Solution #1: segmenting VBR video into uniformly sized segments
  - Hard to realization / duration of segments still leaks information
- Solution #2: constant bitrate?
  - Degrade QoE, network efficiency
- Solution #3: variable-size buffer (fetches equally-sized segments)?
  - Need sophisticated client logic to avoid buffering event

Limitations :

- Method used is sensible to heavy noise. (e.g concurrent usage with multiple titles being streamed)
- Off-Site attack needs a large bandwidth :
  - Cross-site: 6 KB of random data, rate of 1 per 0.001 seconds and an overall
  - Cross-device: 8KB message every 1.5ms (300 KBps), saturating the network link

  ➔Depends on location and ISP of the victim
- Adaptative streaming : variable encodings
- Re-encoded content
- Small dataset , data collection is a bottleneck

Limitations :

- Adaptative streaming :
  - Client chooses the quality of the video according to his network.
- Does the attacker consider the quality changing ?
  - Different encodings of the same content = different burst streaming
- How can the attacker know the changing of the quality ?
  - Quality selection algorithm is key research topic / property of Industry
    - Use variable features not only bandwidth, but also delay, etc…

Limitations :

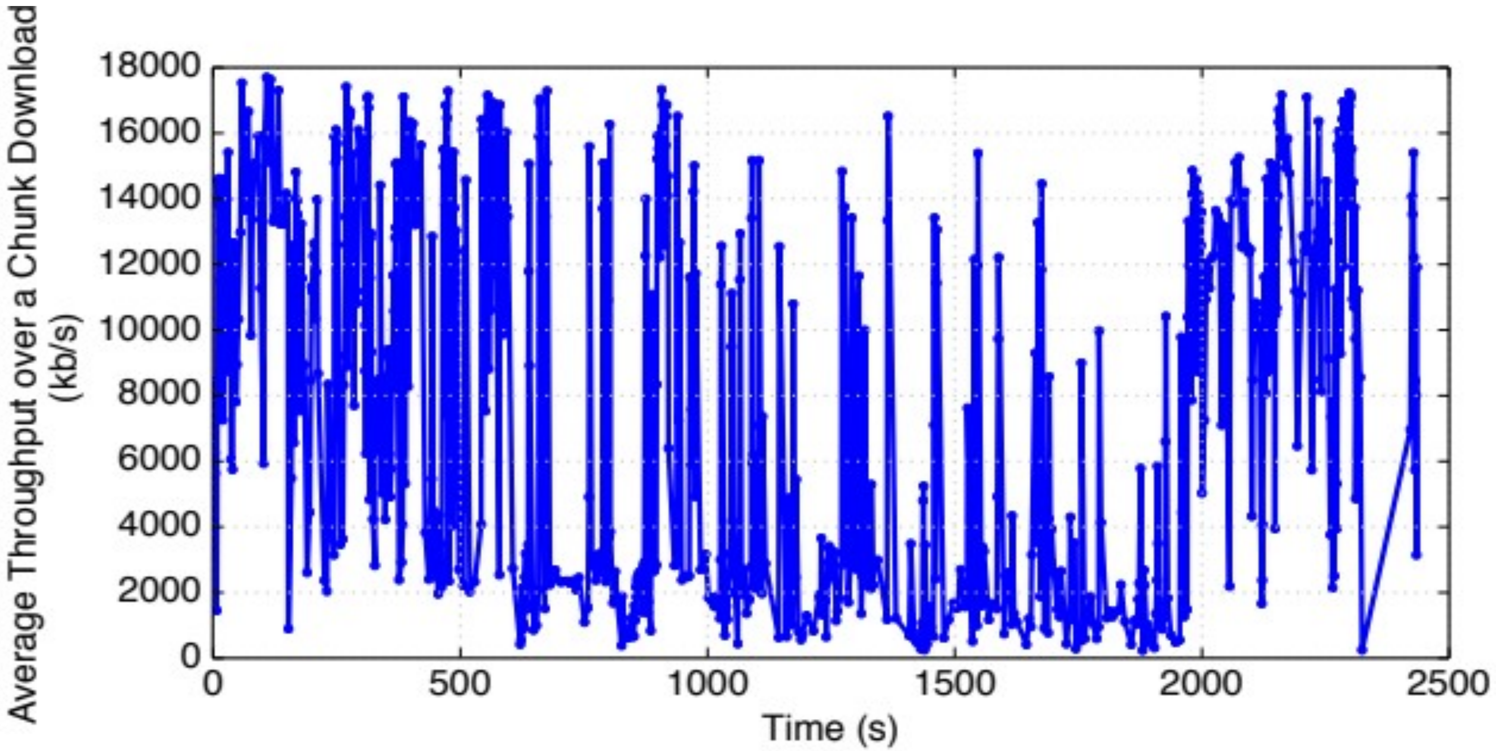


Figure 1: Video streaming clients experience highly variable end-to-end throughput.

Questions ?