Inferring Streaming Video Quality from Encrypted Traffic

Francesco Bronzino

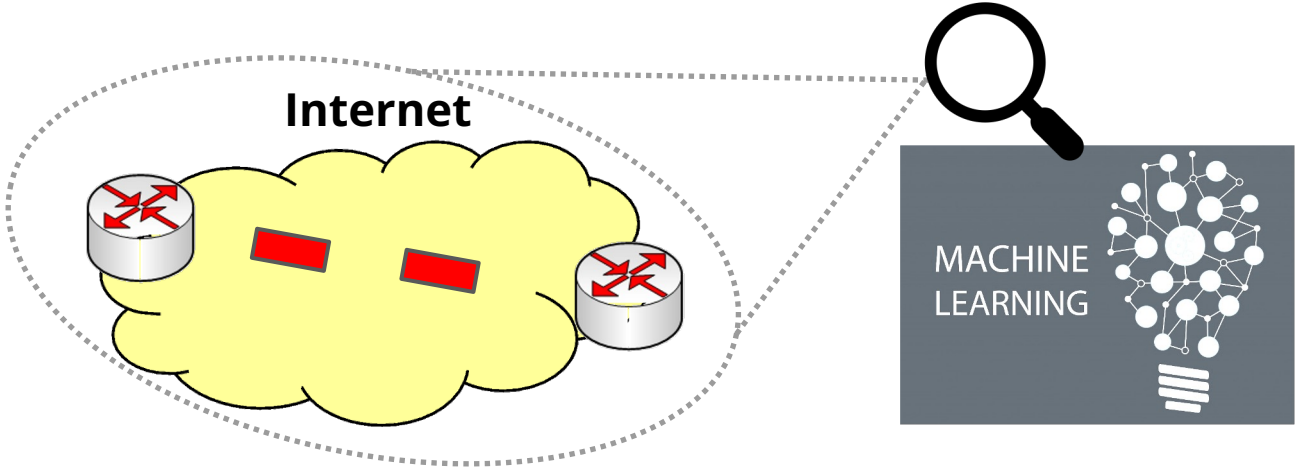
Associate Professor, ENS Lyon
francesco.bronzino@ens-lyon.fr

Overview

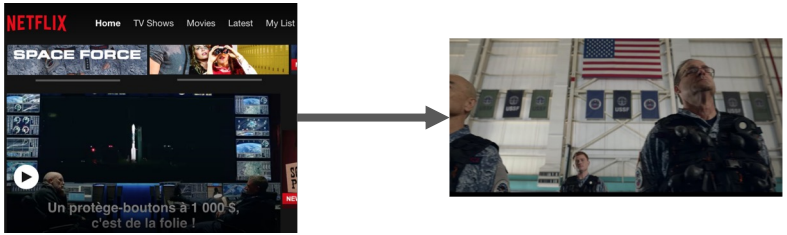
- **Goal:** Learn the end-to-end process required for developing a quality inference model (for video) from encrypted traffic
- **Structure:** ~1.5h seminar + exercises
- **Important:** Ask as many questions as needed!

What is network inference?

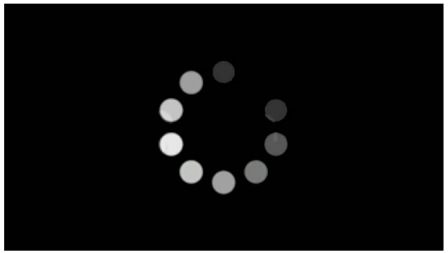
Map network traffic characteristics to network events



Video quality metrics

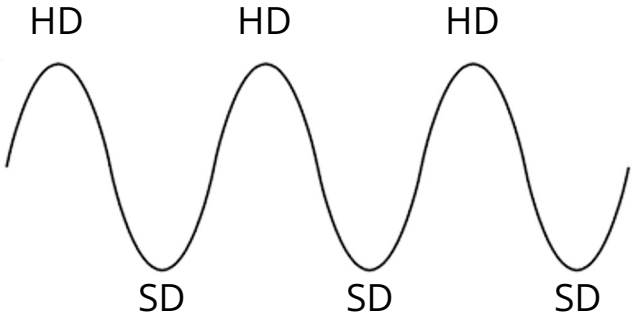


Startup delay



Rebuffering

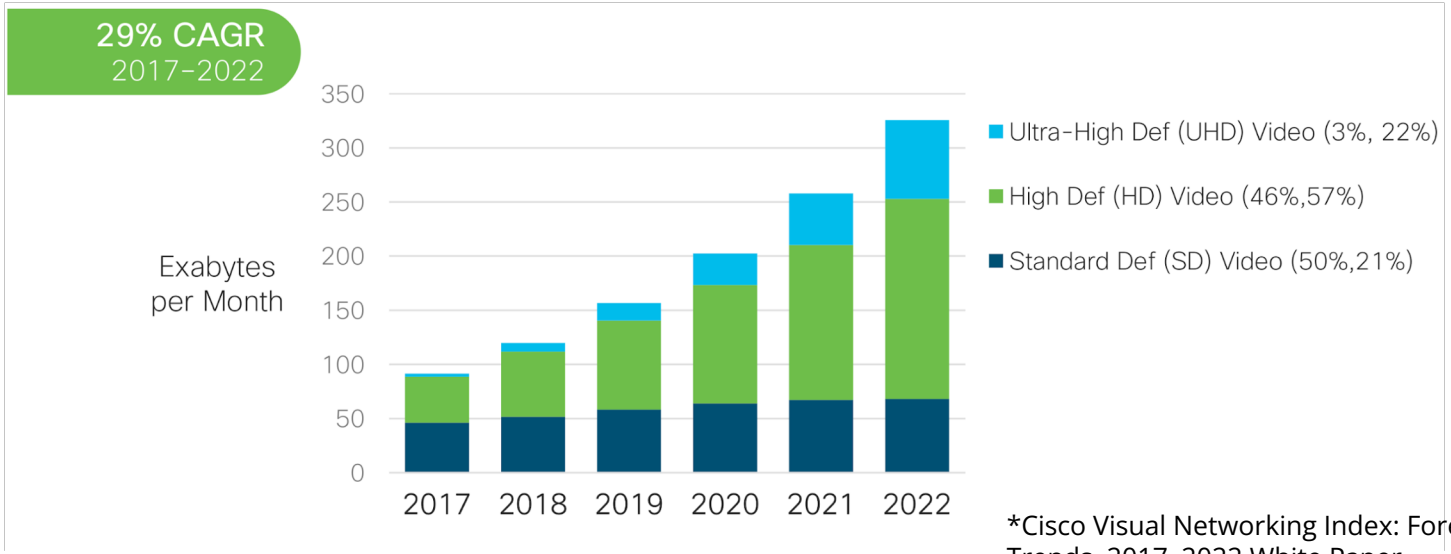
Resolution



Resolution switches

Why study video streaming?

Video accounts for most amount of traffic in the internet



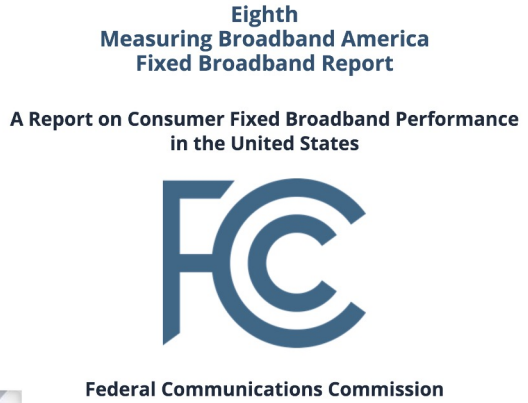
*Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper

- Increasing amount of video traffic in ISP networks
- Monitoring tools do not provide enough insights to reveal video streaming quality

Who cares about video streaming performance?

- Home users

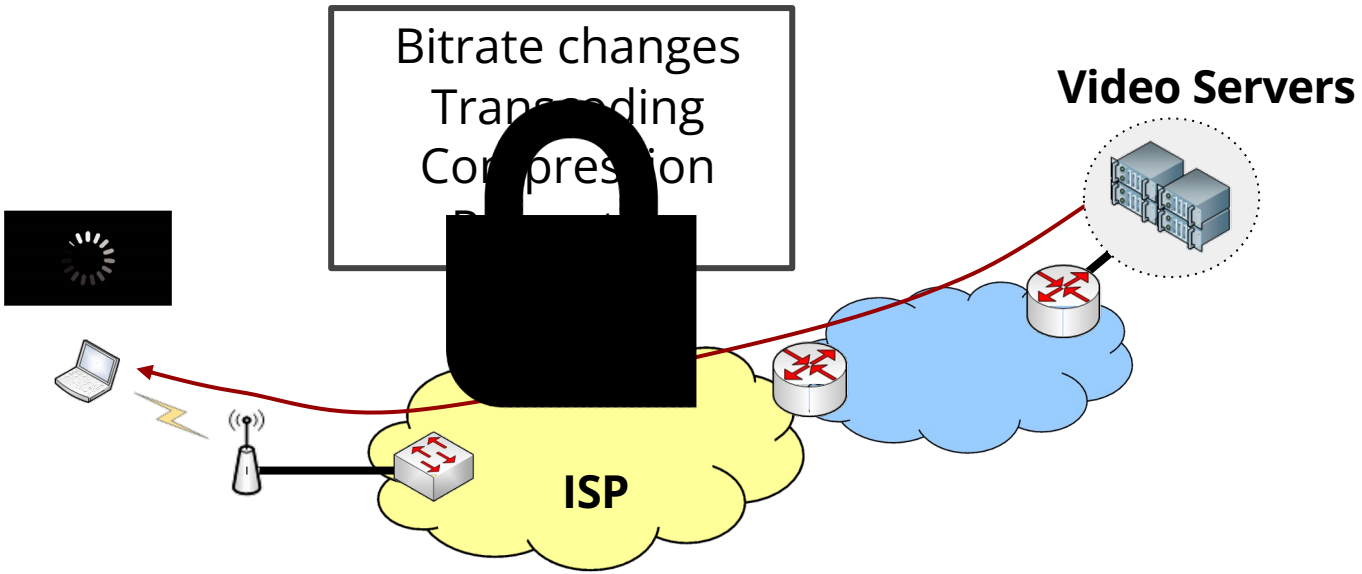
- Regulators (e.g., Federal Communications Commission)



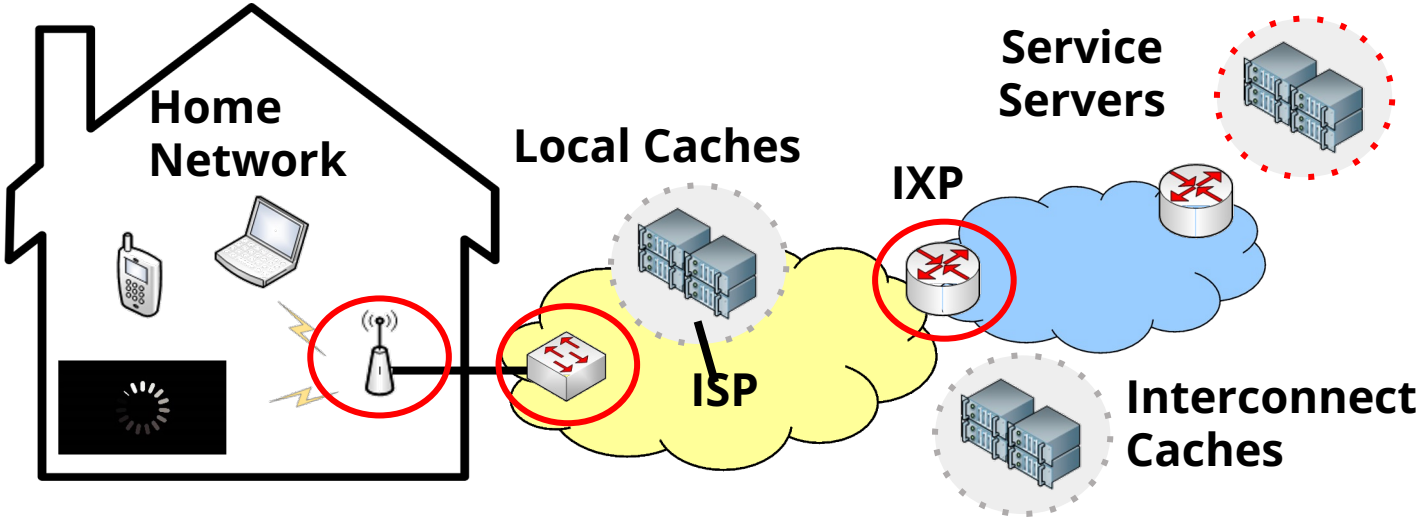
- Internet service providers (ISPs), content providers

Why it is hard to infer video quality?

ISPs aim to perform network optimizations

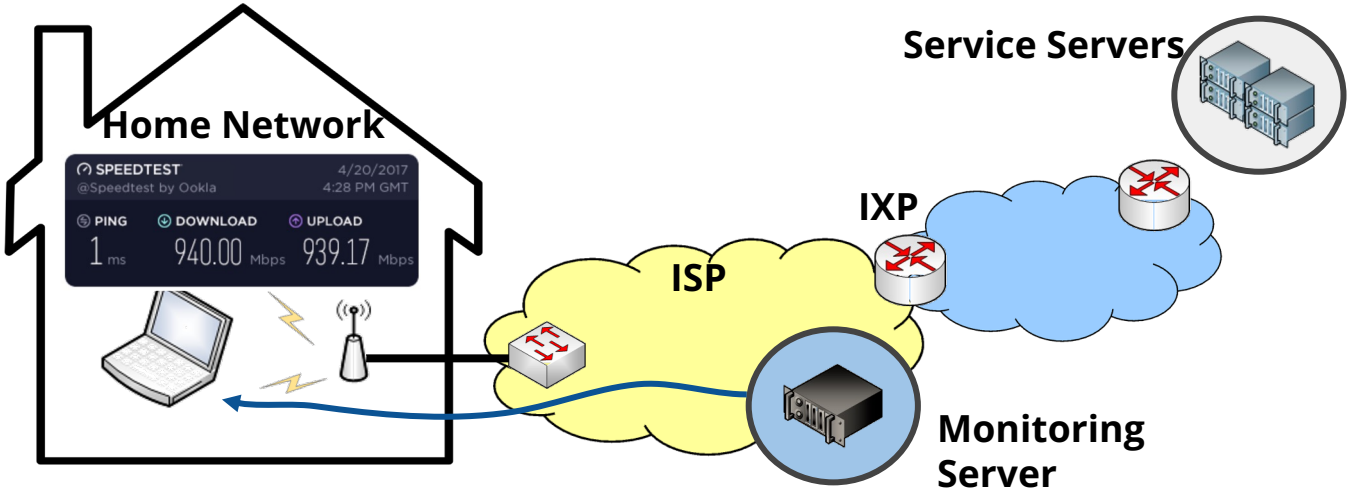


Video quality degradation root causes



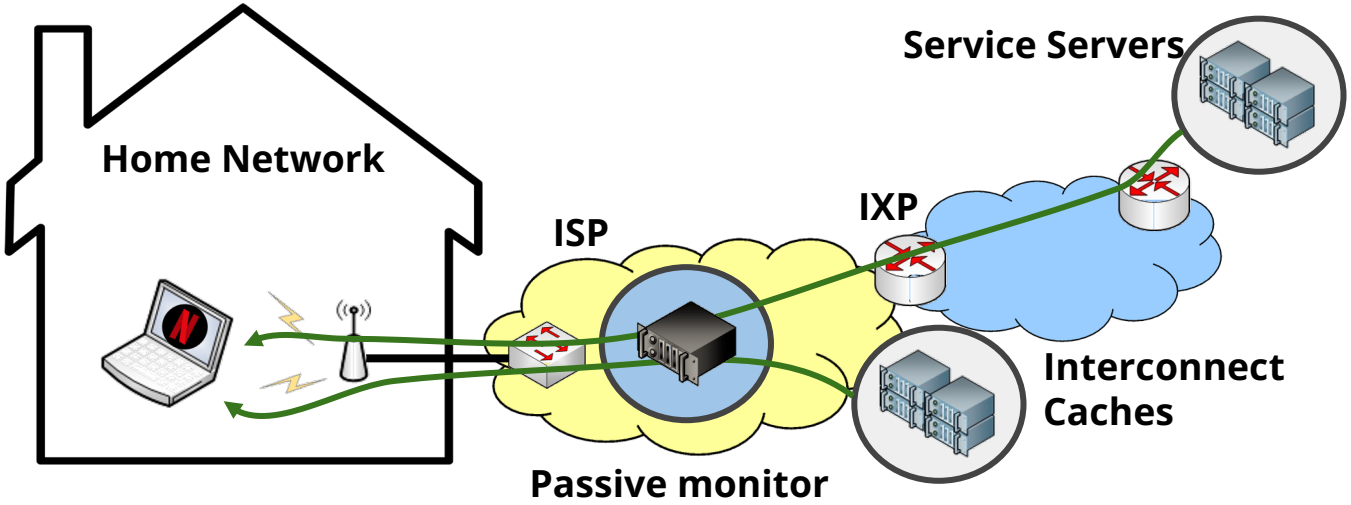
- Service traffic traverses a multitude of networks owned by different entities
  - Challenging to pinpoint and correlate the root causes of impairments from a single location

Estimating performance: active measurements



- Disruptive as speeds increase
- May reveal a different bottleneck other than the access link
- Typically measures a path that's different than the application traffic
- Application-specific measurements need to emulate clients, need cooperation from services

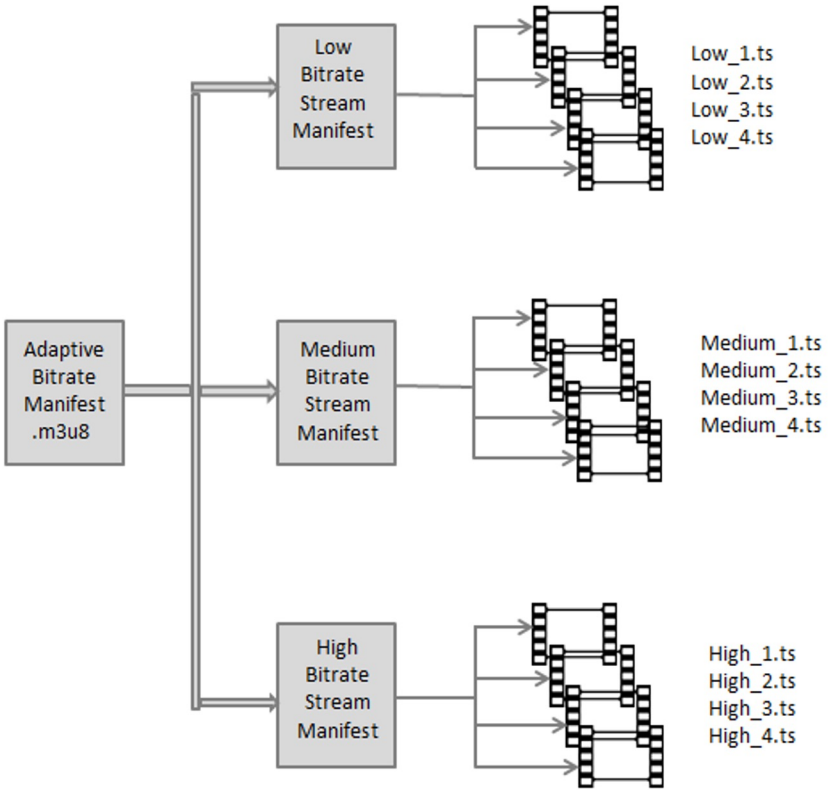
Estimating performance: passive measurements



- Infer application quality passively from network traffic
- Challenges:
  - Identify application sessions within network traffic
  - Large diversity of applications and services
  - Increasing line rates

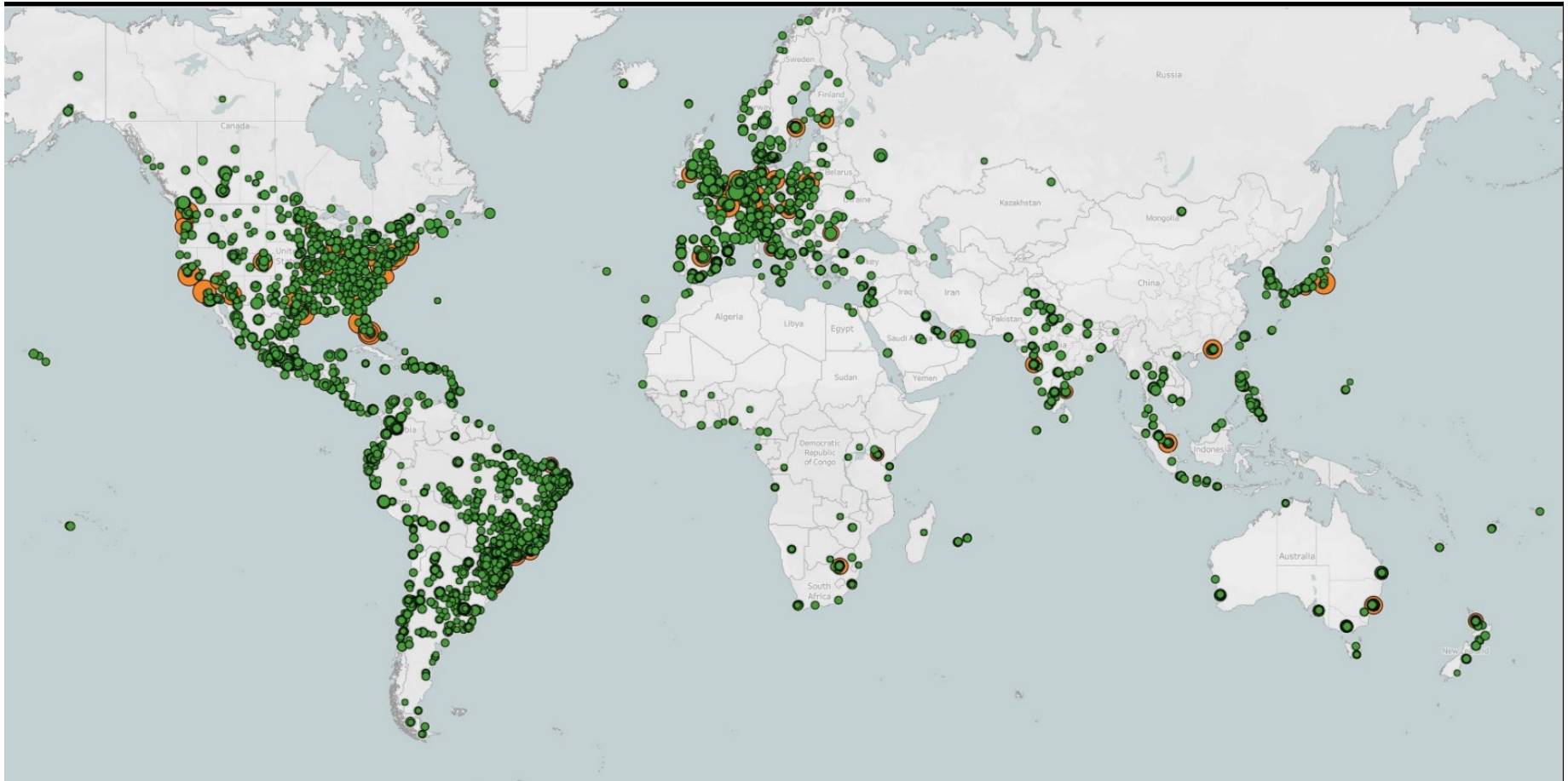
Dynamic Adapting Streaming over HTTP (DASH)

DASH video streaming

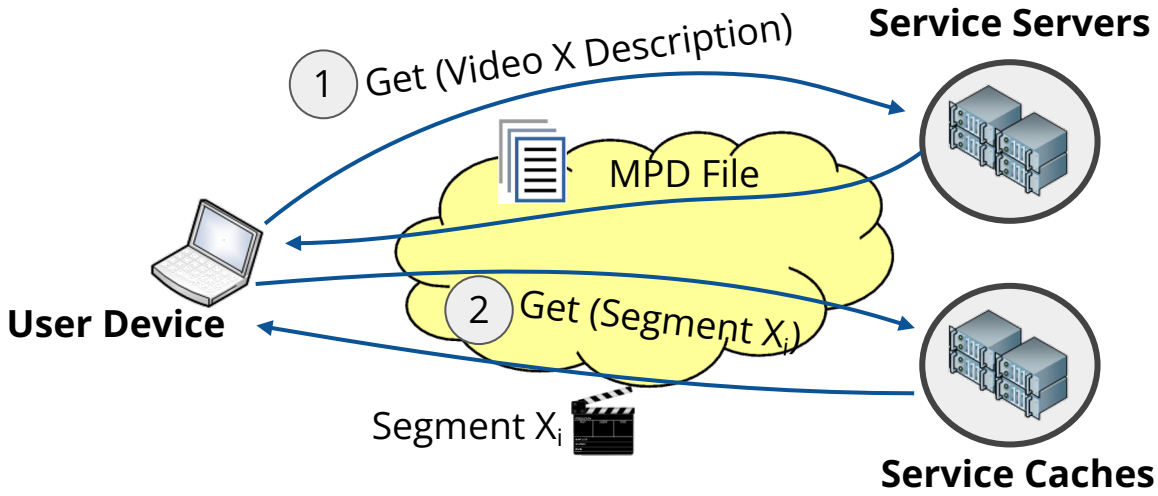


- Dynamic Adaptive Streaming over HTTP (DASH)
- Encode the video into multiple qualities / resolutions
- Split each quality level into segments of short duration
  - 2 / 4 seconds

Netflix content server deployment map

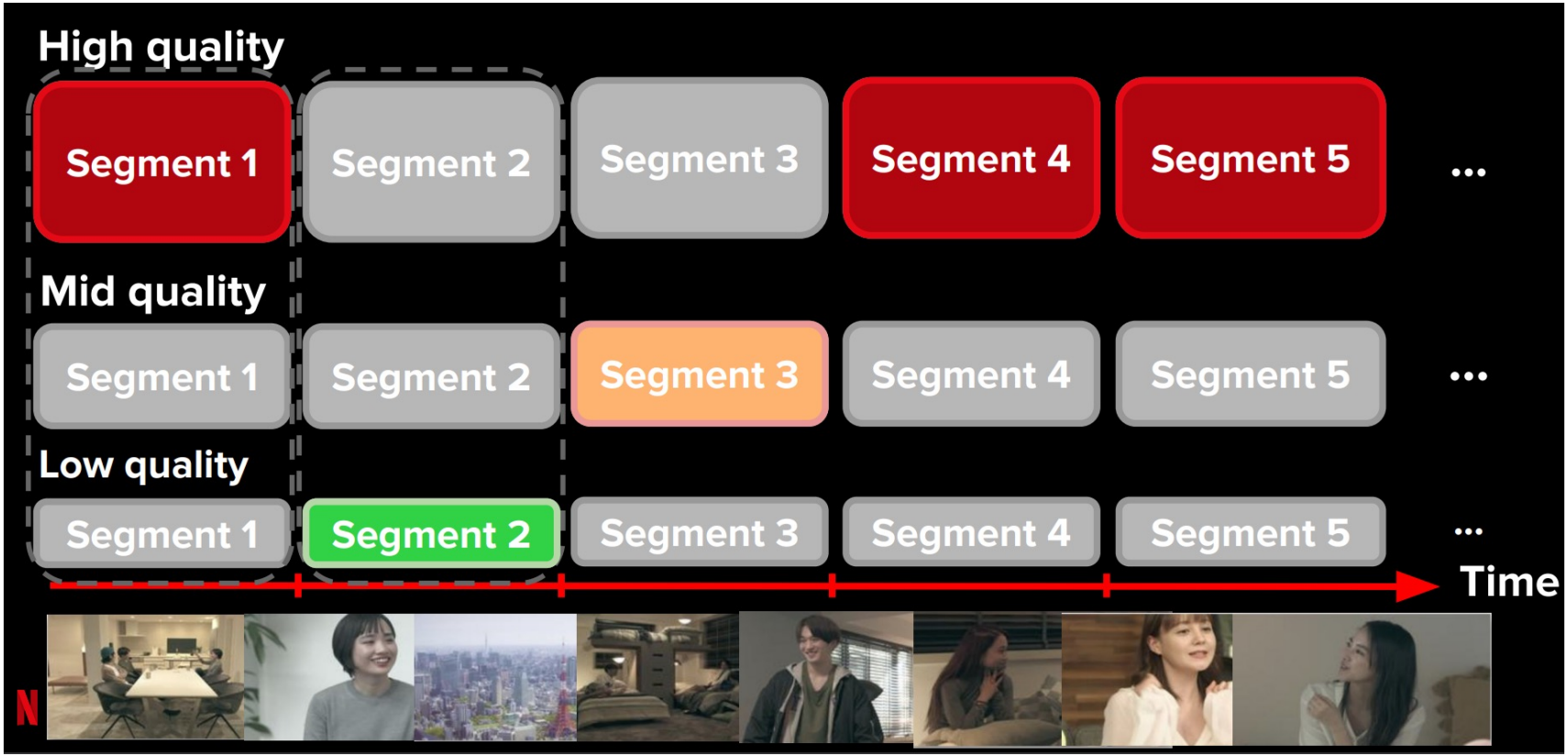


DASH video streaming



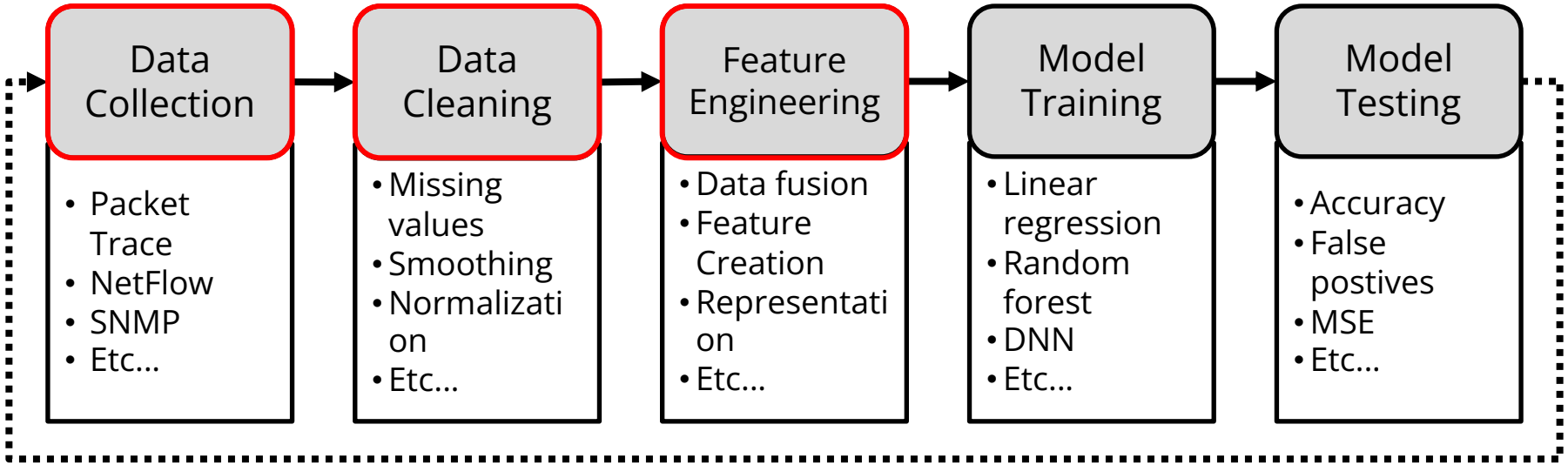
- Use a complex architecture of servers and caches
- Sequentially fetch video segments from a location of choice
- Use encrypted protocols (HTTPS)

Adapt video quality based on changing conditions



A quick point on machine learning pipelines

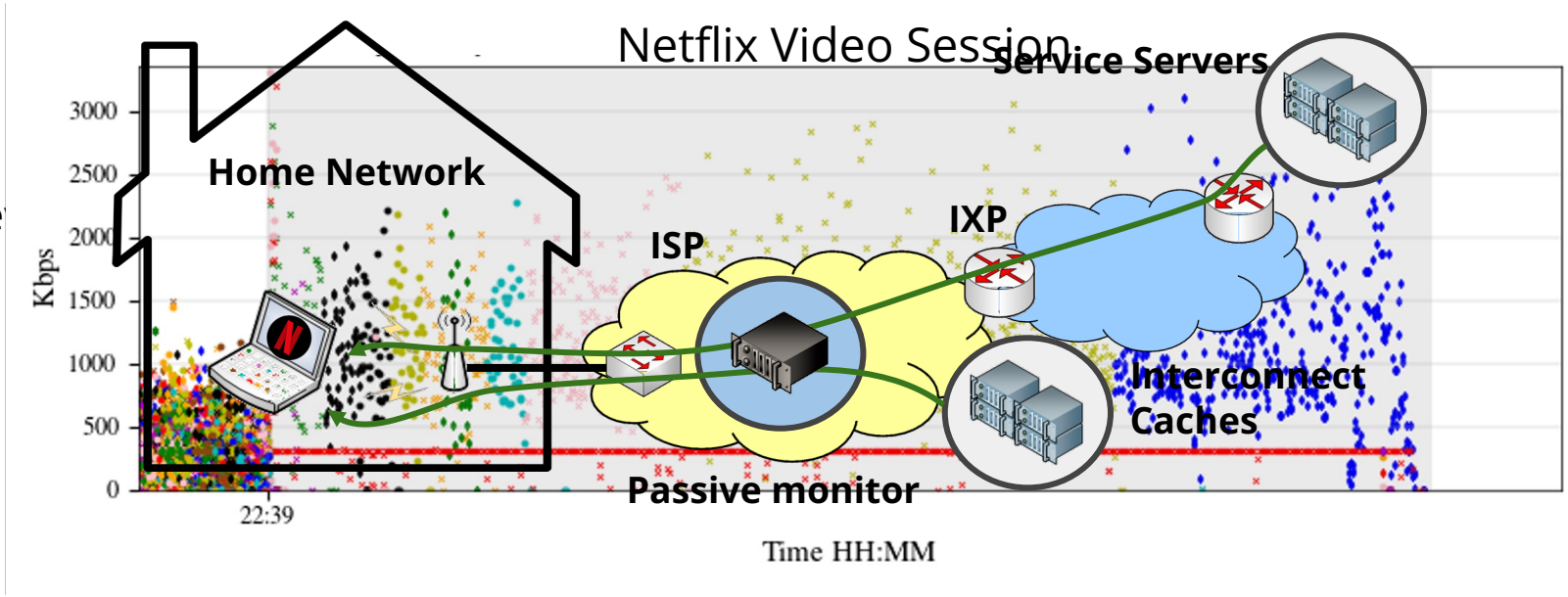
Building a machine learning pipeline for network traffic (supervised learning)



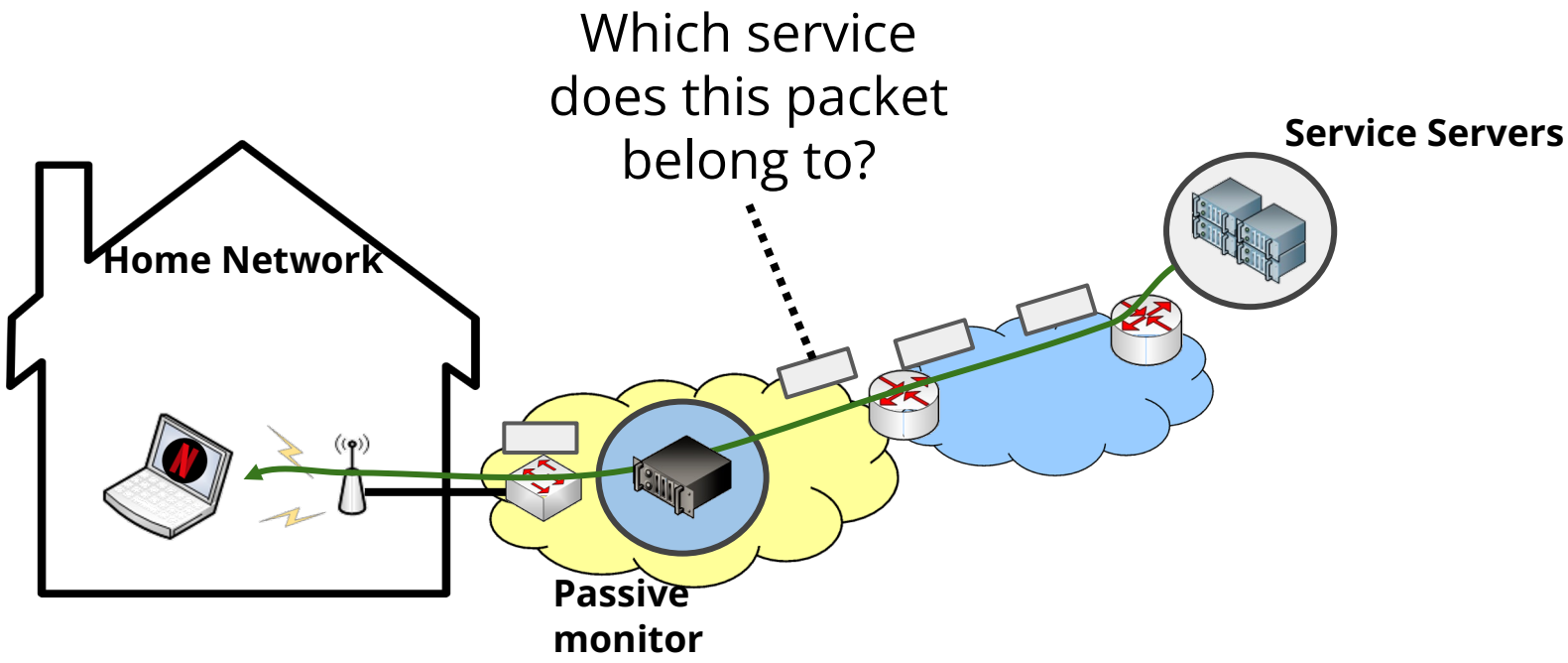
Passively extracting features from traffic

What ISPs see with encryption

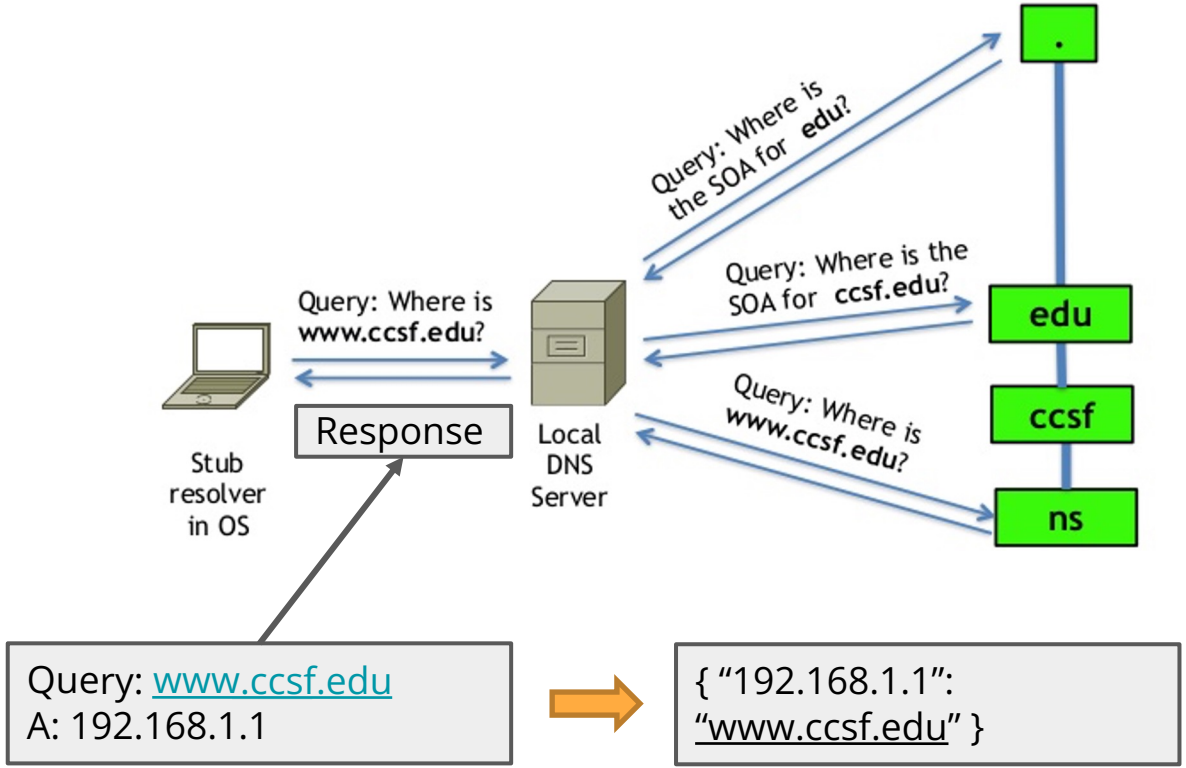
MPD
Retrie



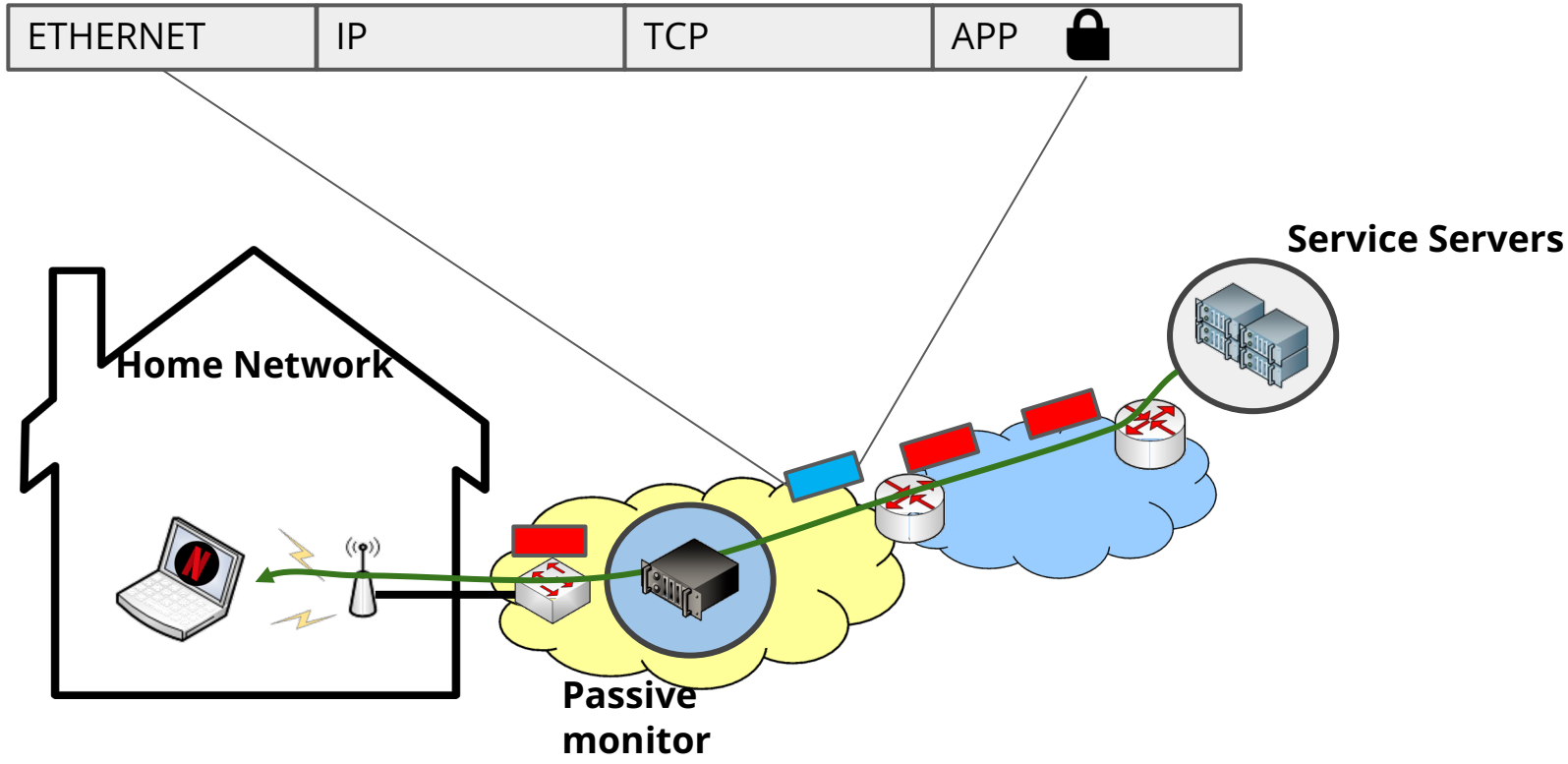
Extracting features: step 1



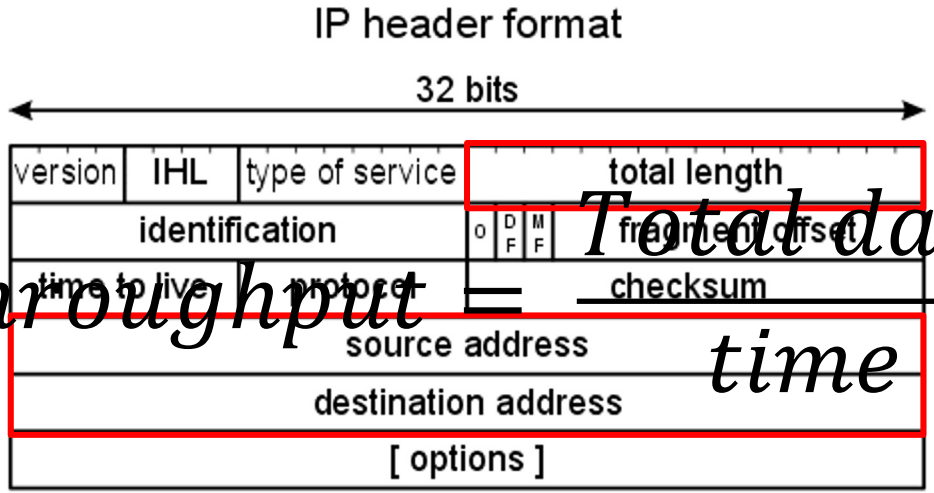
Identifying services through DNS traffic



Extracting features: step 2



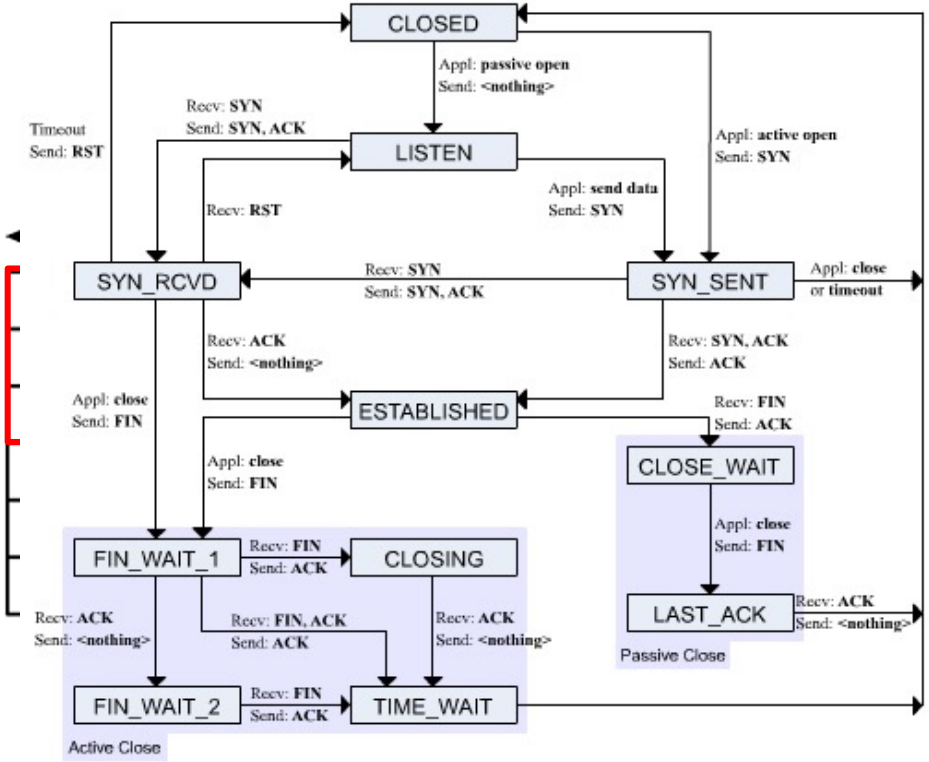
Network features



$$throughput = \frac{Total\ data}{time}$$

Transport features

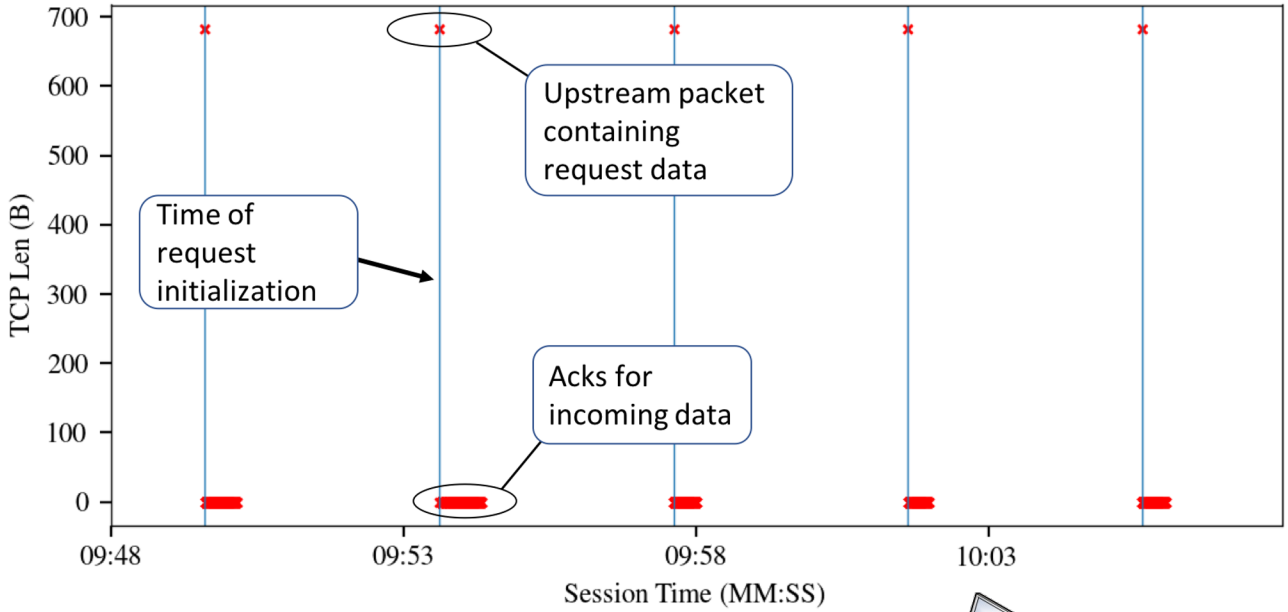
ETHERNET	IP	TCP	APP



Application features

- Application layer is encrypted
- How to extract information?

Segment estimation using traffic patterns



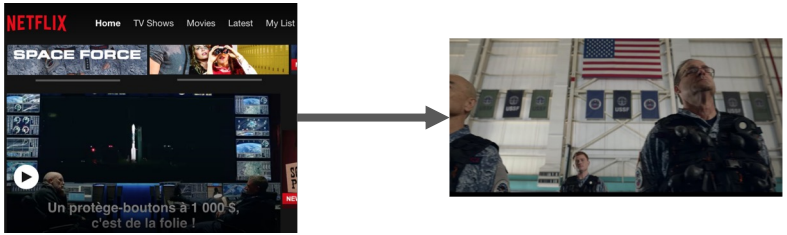
- HTTP traffic for video requests is sequential
- Upstream traffic reveals requests boundaries

Input features from encrypted video traffic

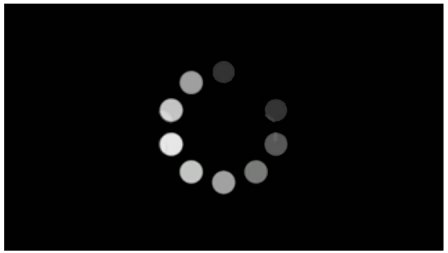
Network layer	Transport layer	Application layer
throughput up/down	#flags up/down	seg. sizes (all, last-10, cumulative)
throughput down diff	rcv window size up/down	seg. request interarrivals
pkt count up/down	idle time up/down	seg. completions interarrivals
byte count up/down	goodput up/down	#pending requests
pkt interarrivals up/down	bytes per pkt up/down	#downloaded seg.
#parallel flows	round trip time	#requested seg.
	bytes in flight up/down	
	#retransmissions up/down	
	#out of order pks up/down	

Creating a labeled dataset of video sessions

Video quality metrics

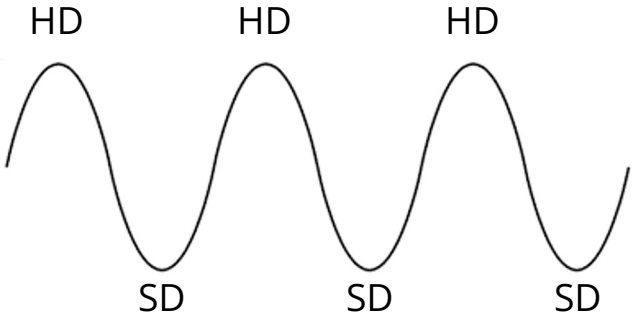


Startup delay



Rebuffering

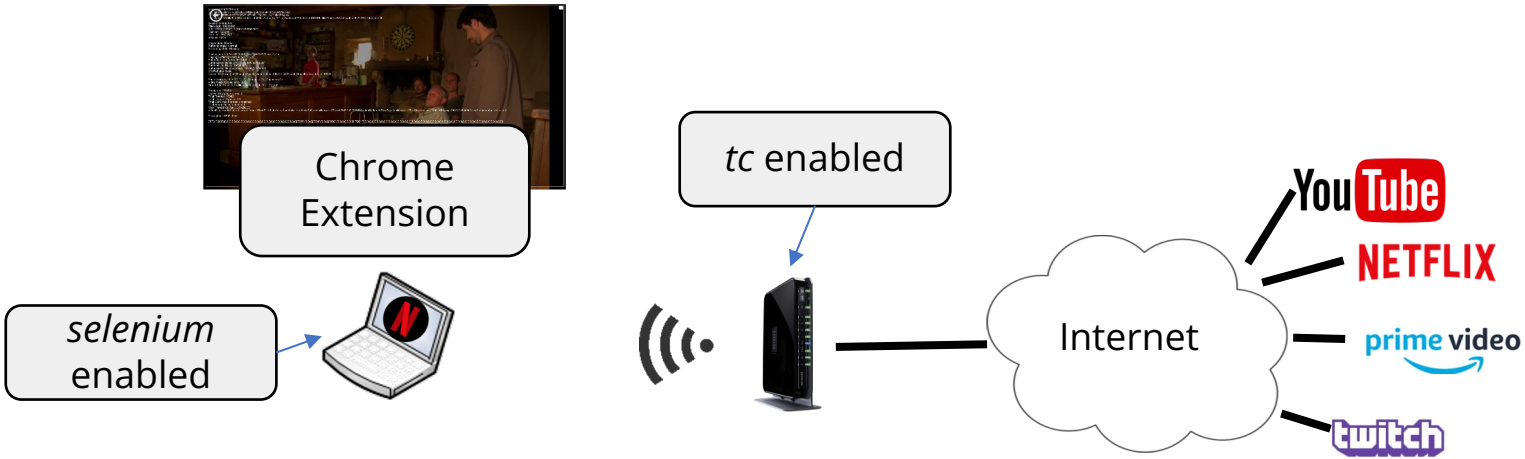
Resolution



Resolution switches

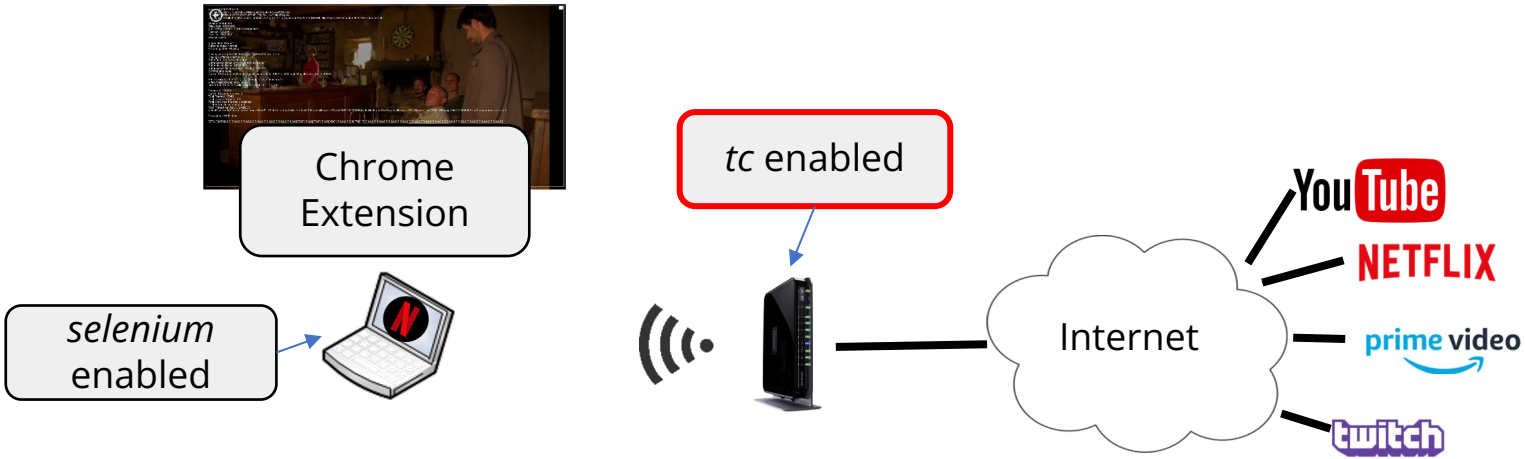
Creating a labeled dataset

- Chrome extension to collect labeled video sessions for four services
- Network traffic through pcap traffic traces
- Diverse emulated network conditions
- >13k video sessions collected
  - Dataset and extension are available for download



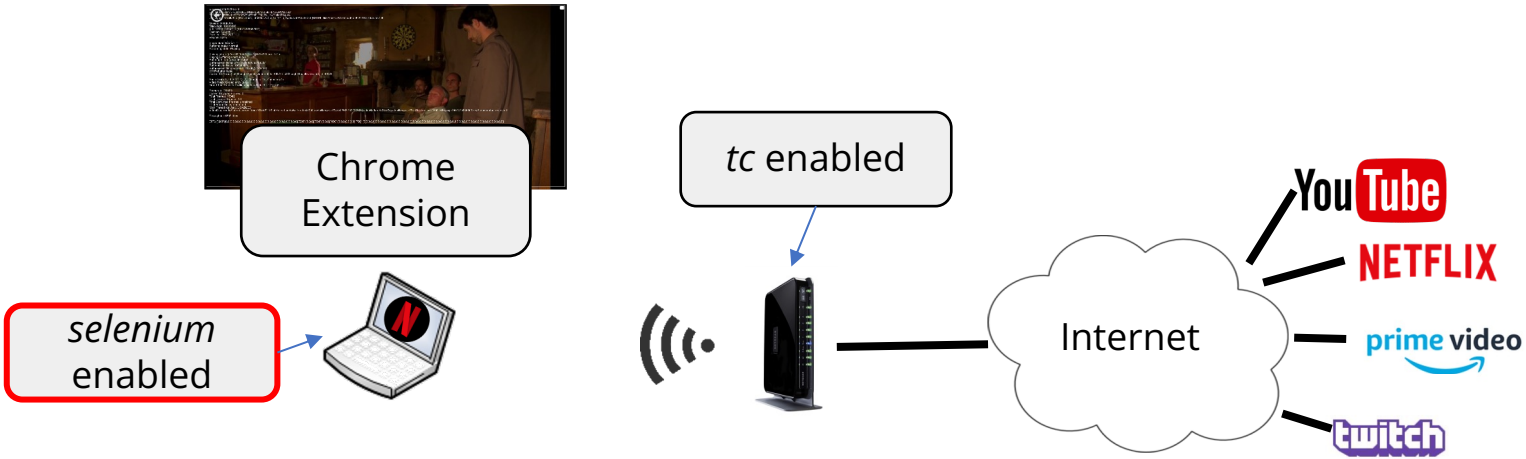
Emulating network conditions

- *tc* (https://man7.org/linux/man-pages/man8/tc.8.html) is a linux kernel tool to manipulate traffic control settings
  - Throughput
  - Latency
  - Packet loss



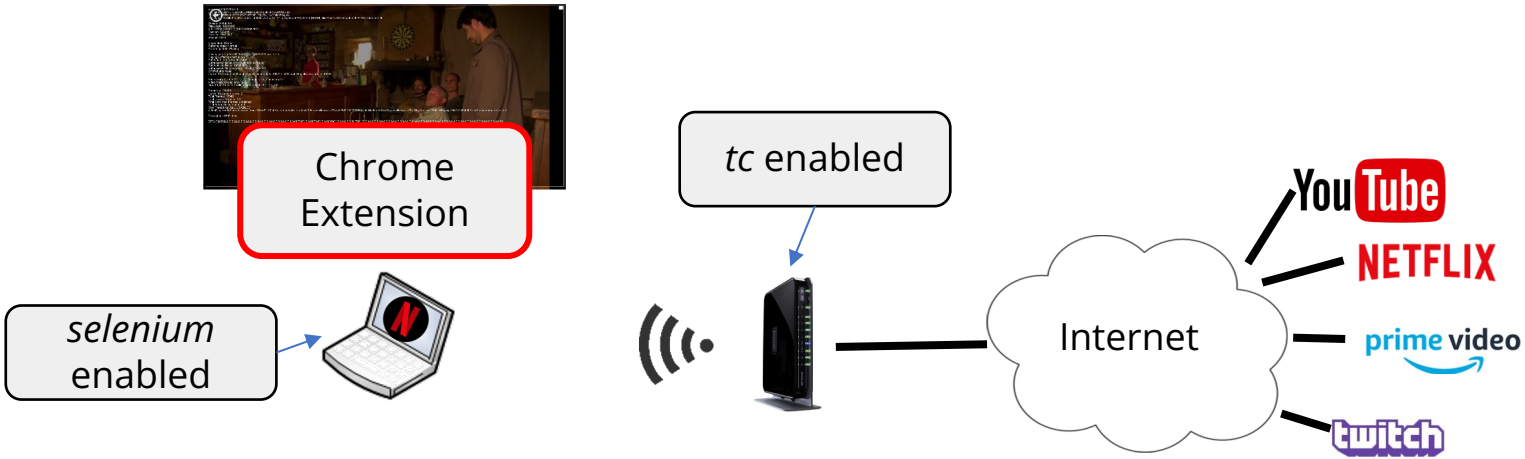
Automated collection

- *Selenium WebDriver* (https://selenium-python.readthedocs.io/index.html) provides a simple API to control the web browser from Python
  - Open/close web pages
  - Click on links



Collecting ground truth

- Chrome extension to automatically detect when a video is playing and extract video player information
  - YouTube / HTML5 APIs
  - Netflix on-screen logs
  - https://github.com/inria-muse/video_collection



Create video inference models

What questions to answer when creating a model

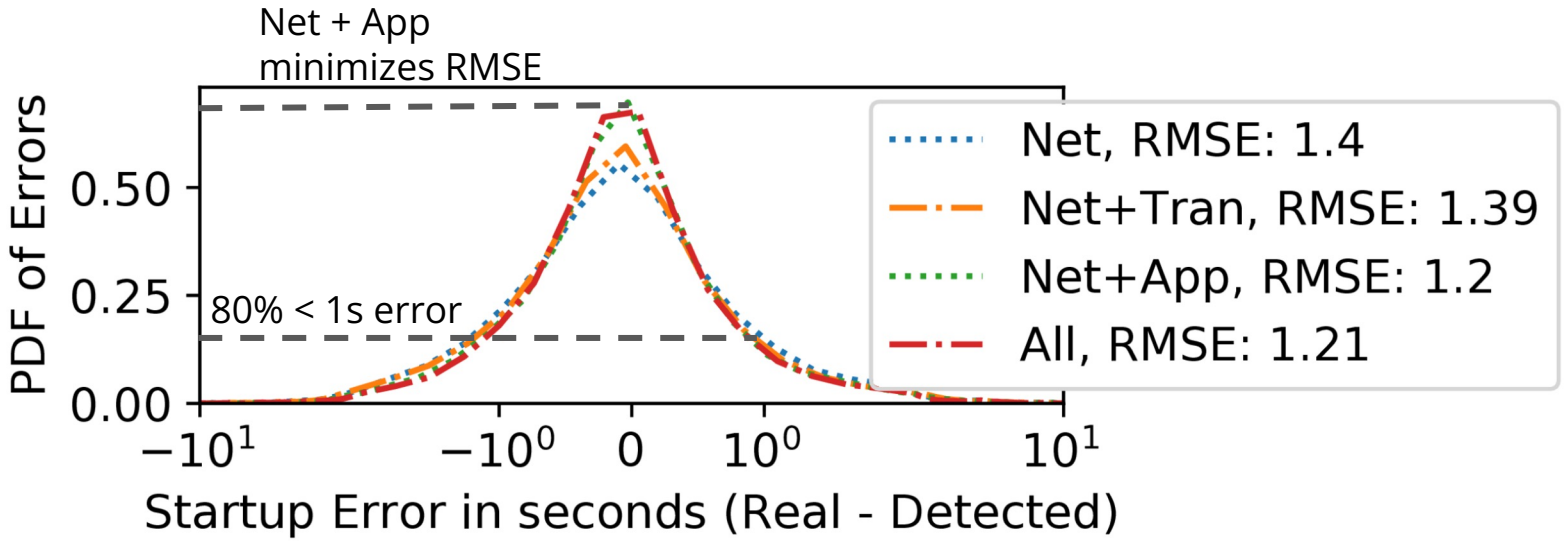
- Time granularity for inference?
- Which features to use?
- Create one model per service or a single one for all services?
- Which model to use?

Model validation

- Infer quality metrics at periodic time intervals
  - 10 seconds
- Different sets of input features
  - Net, Tran, App, Net+Tran, Net+App, All
- Different sets of services
  - Specific: train one service - test same service
  - Combined: train on all services - test on one of the services
  - Excluded: train on three out of four services - test on the remaining one
- Different models
  - Selected random forest (for both regression and classification)

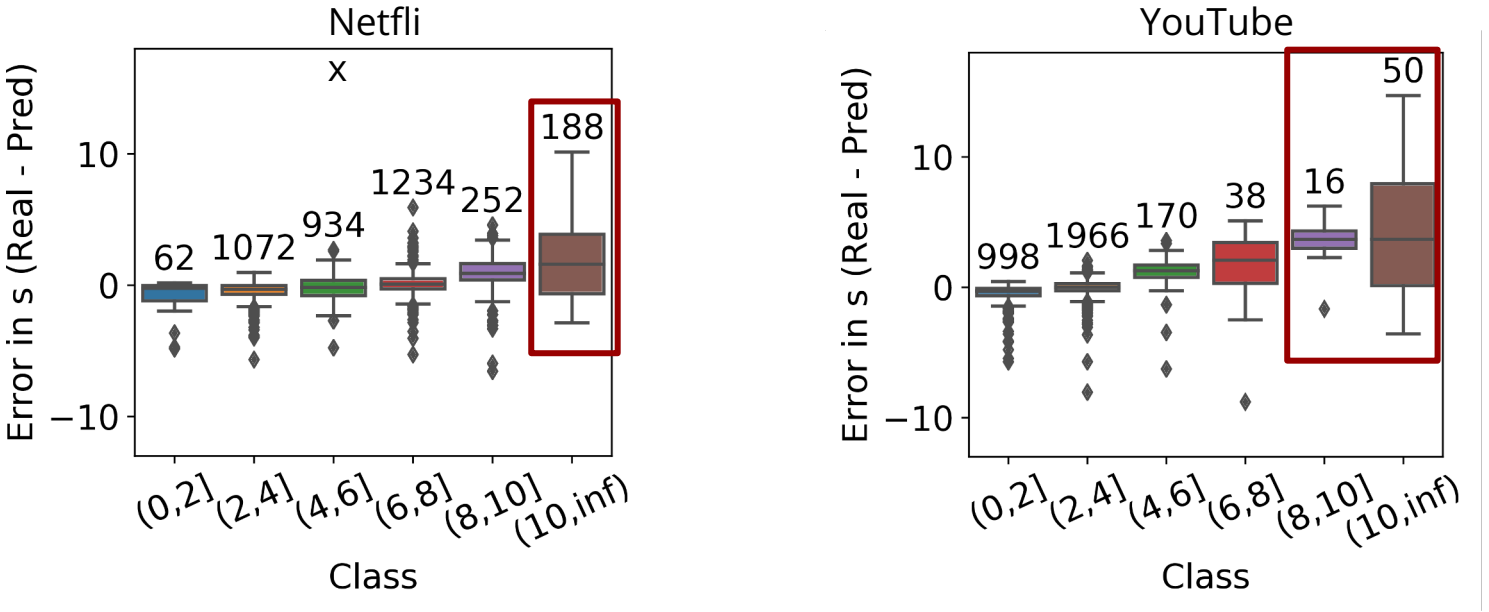
Results

Startup Delay - Feature Selection



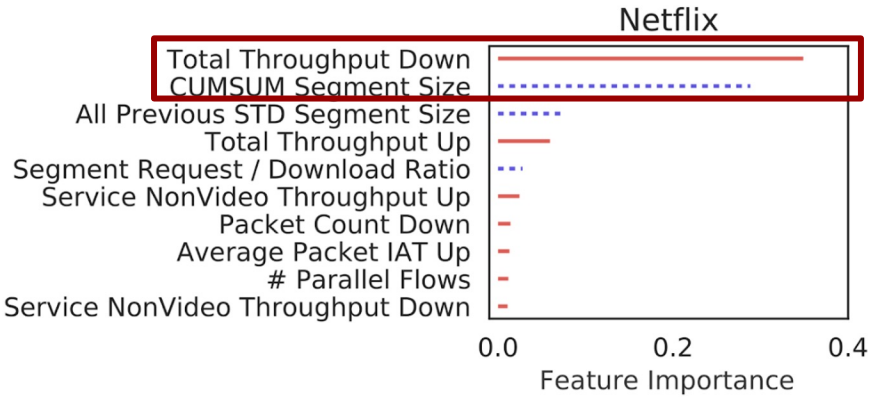
- Regression infers the exact time required to start a session
- Magnitude of the inference errors for our model prediction

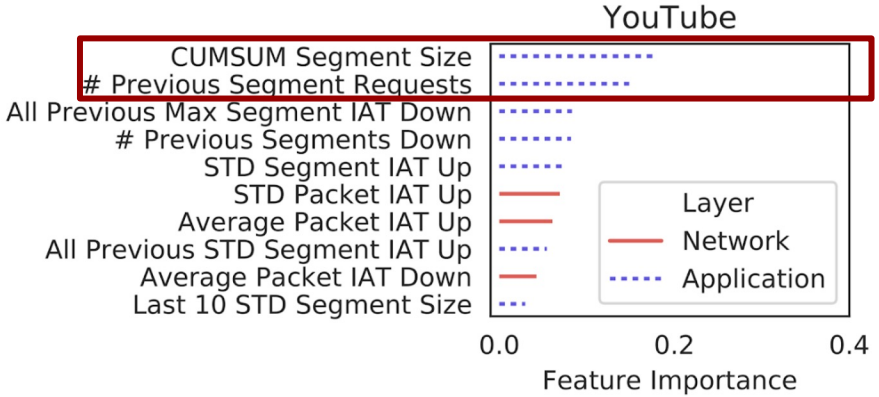
Startup Delay - Service Performance



- Relative errors split into two-second bins
- Goal: understand impact in real world usage

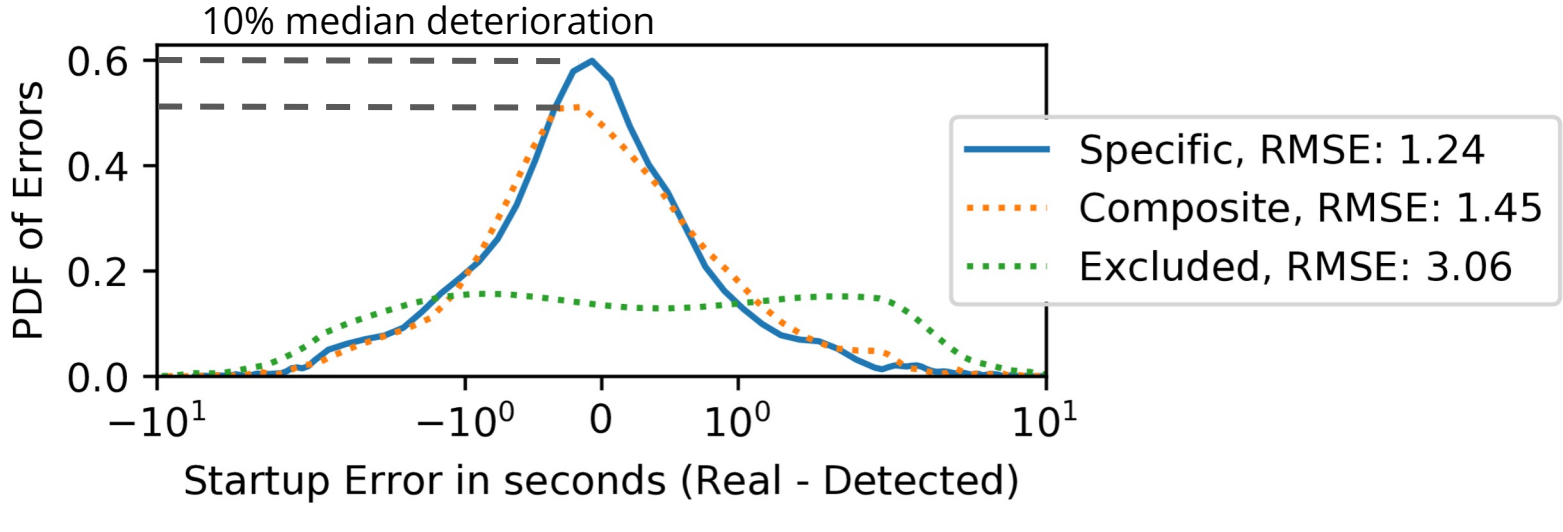
Startup Delay - Feature Importance





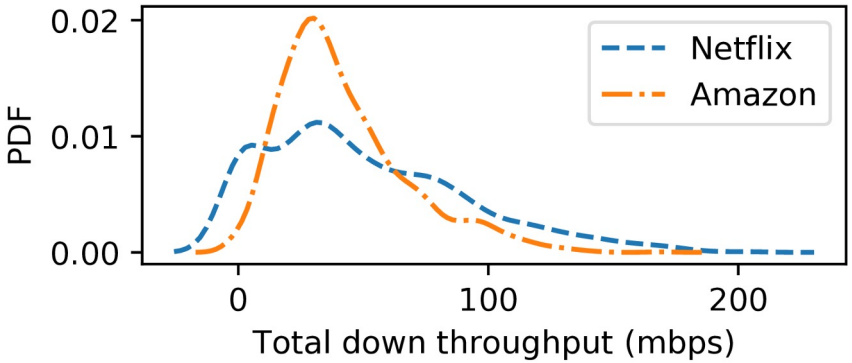
- GINI index
- Small subset of features have largest impact
  - High correlation to amount of downloaded data
- Opportunity for pruning / minimizing state

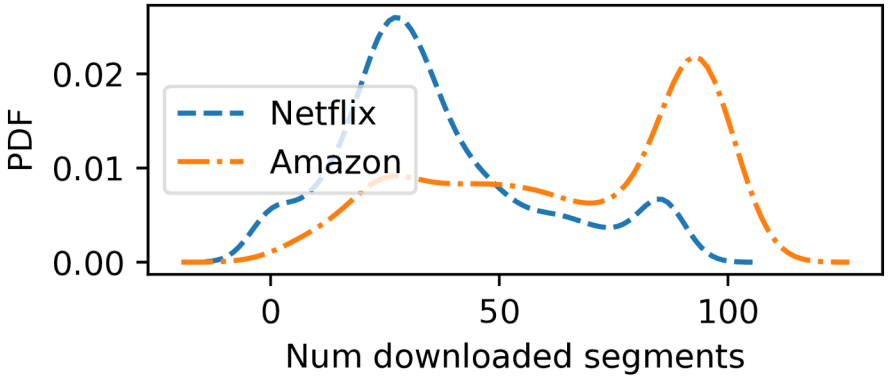
Startup Delay - Generalization



- Evaluate models generality
- Train the model with data from multiple services and predict quality of any video service

Startup Delay - Generalization



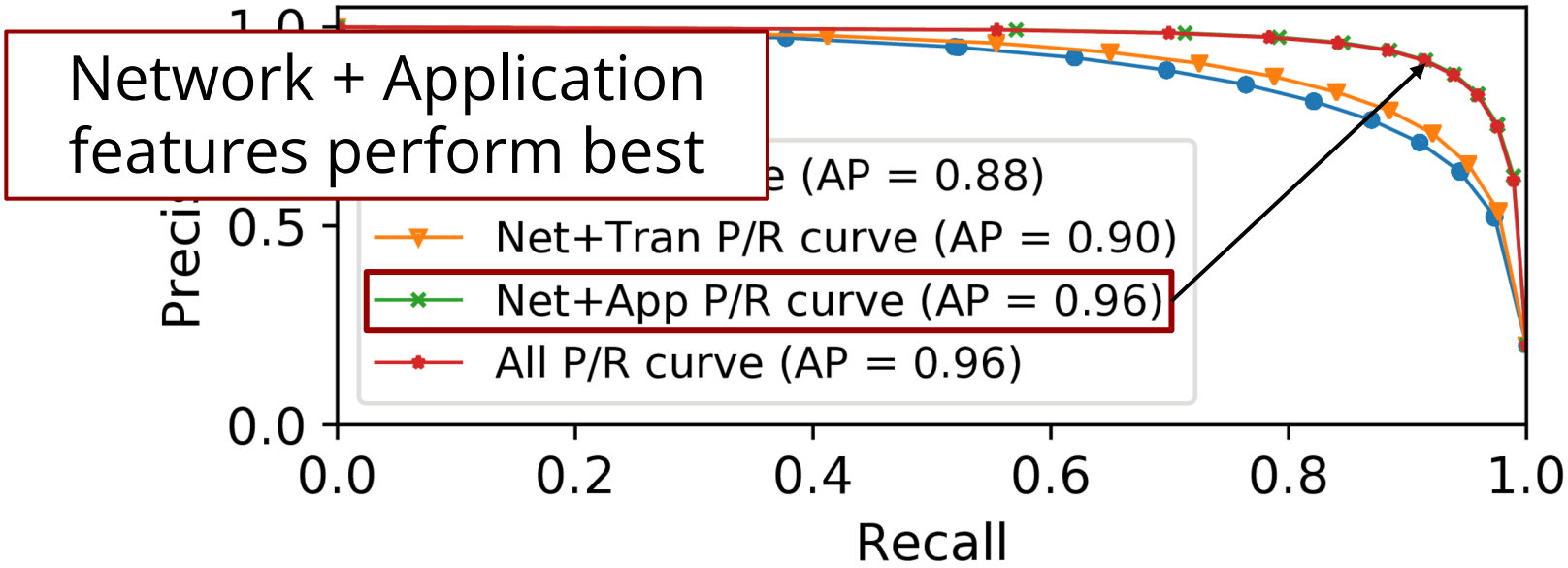


- Analysis of the differences and similarities of input features
- Netflix's behavior is often similar to that of Amazon
  - But only on a subset of features

Startup Delay - Summary

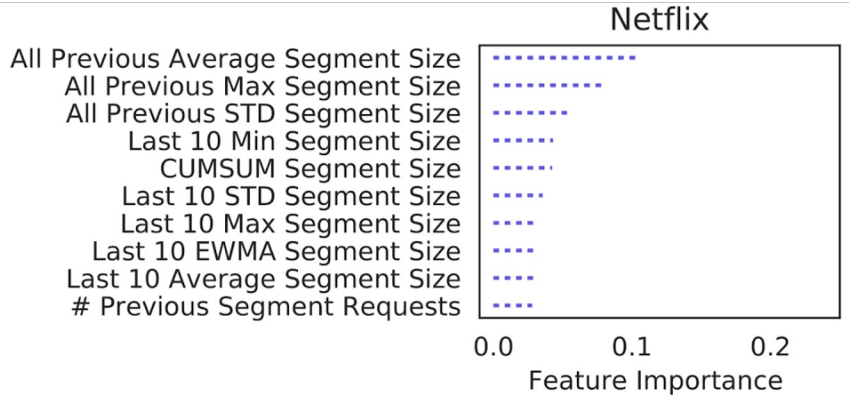
1. Good inference performance
    - Best with Net+App
2. Small subset of features have largest impact
    - Dominant features relate to the amount of downloaded data
3. Generalizes well if the service is present in the training dataset
    - Very poor performance if only using other services
    - Important features can have very different distributions

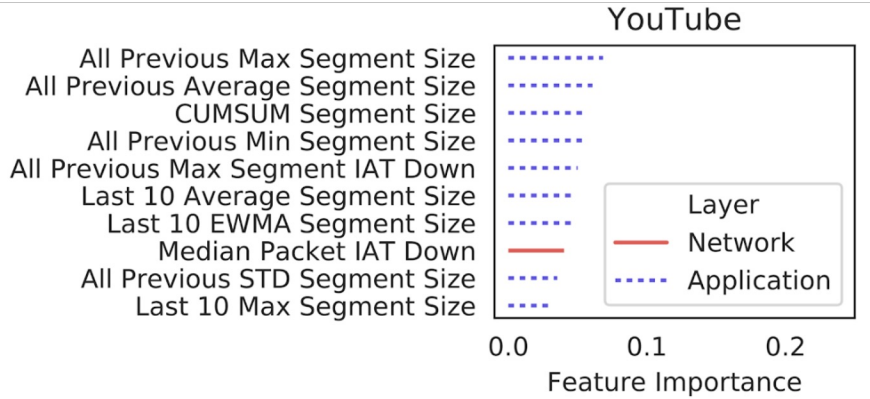
Resolution - Feature Selection



- Classification infers the current resolution
  - 240, 360, 480, 720, 1080
- Similar results for ROC plot

Startup Delay - Feature Importance





- Importance is more distributed across different features
- High correlation to the segments of video downloaded over time

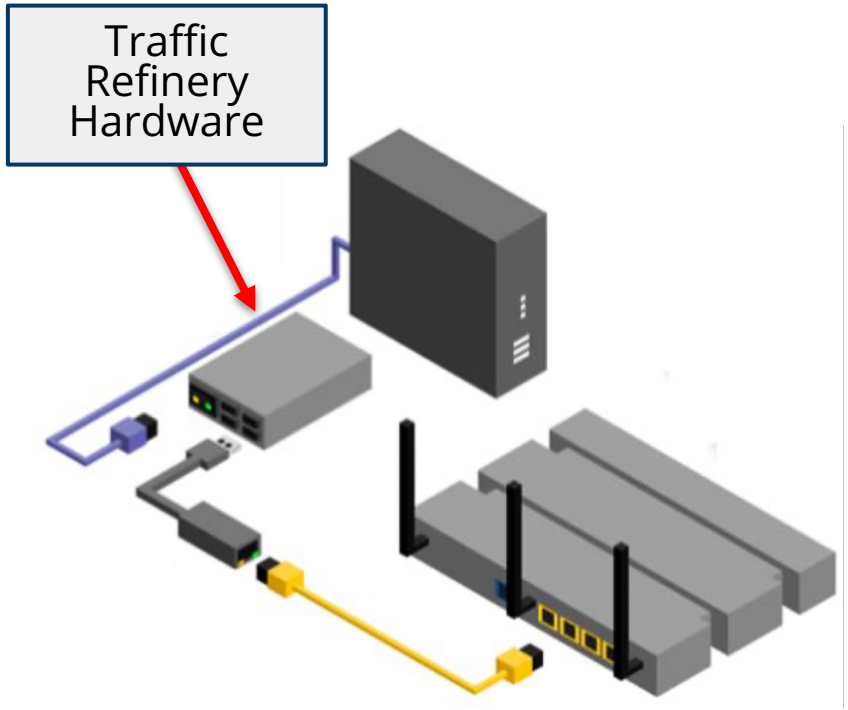
Resolution - Summary

1. Good inference performance
    - Best with Net+App
2. Dominant features relate to the application layer (segment downloads)
    - More distributed impact across different features
3. Generalizes well if the service is present in the training dataset
    - Very poor performance if only using other services
    - The majority of features have very different distributions

From lab models to the real world

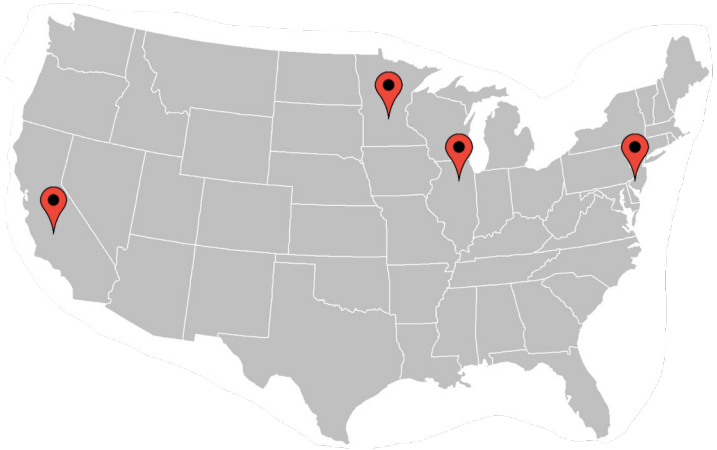
Traffic Refinery

- **Goal**: estimate application performance using passive measurements without breaking encryption
- Prototype placed in-line between
  - Cable modem and wireless router
  - View of devices behind NAT
- Implemented in Go for low-cost devices
  - (Raspberry Pi, Odroid) on home networks
- Potential for deployment in lots of different places in the network.

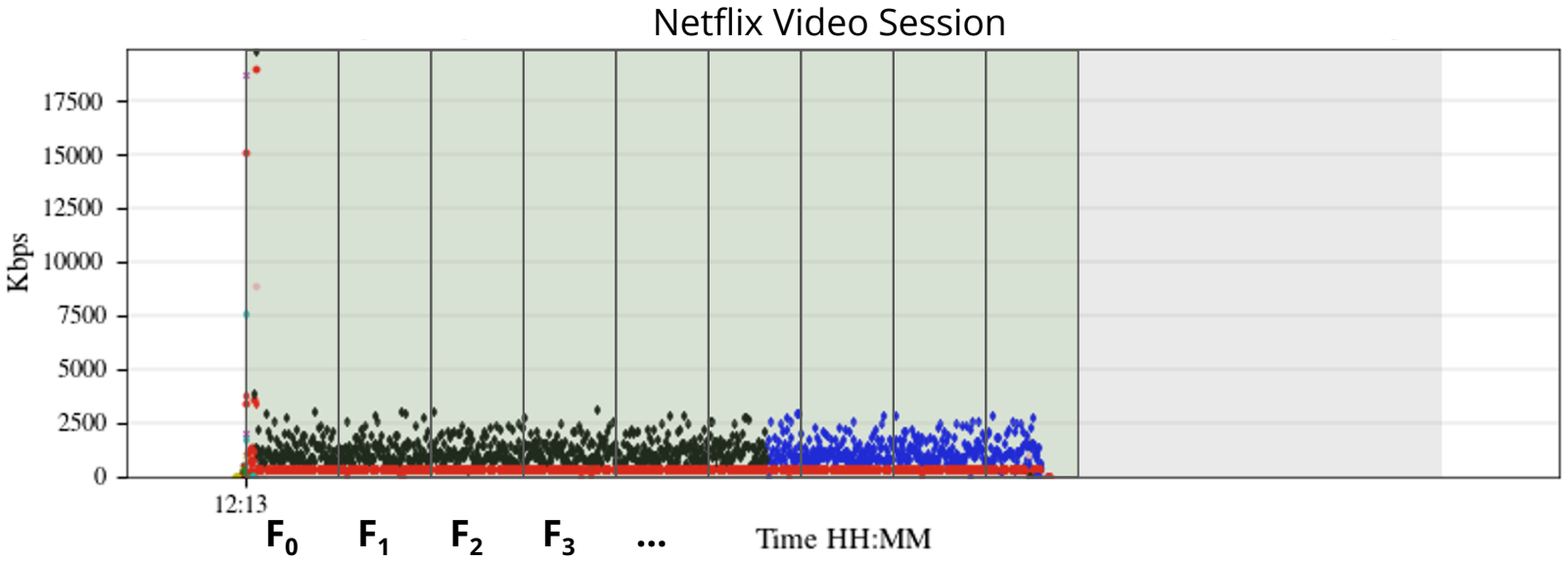


Deployment - Characterization

- Heterogeneous collection of homes
  - ~60 in the US, ~10 in Paris
- ~210k video sessions collected in homes over 14 months
  - Additional 2k sessions collected with ground truth (extension)
- Video metrics:
  - Startup delay, resolution

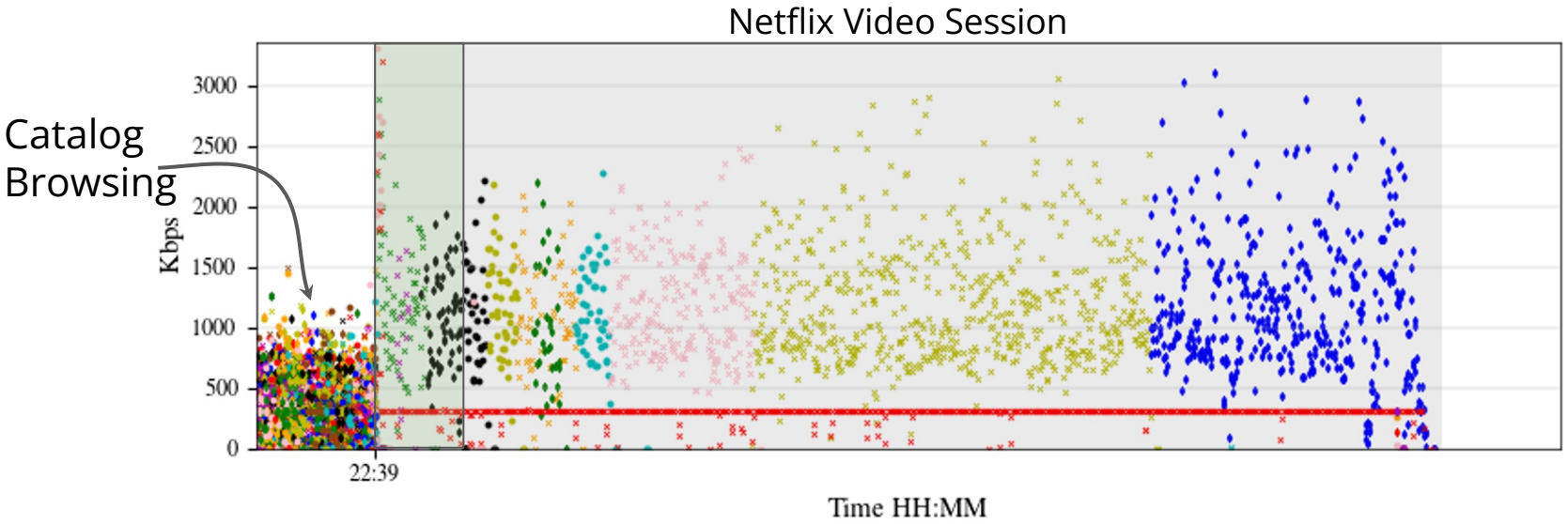


Deployment - Practical Challenges



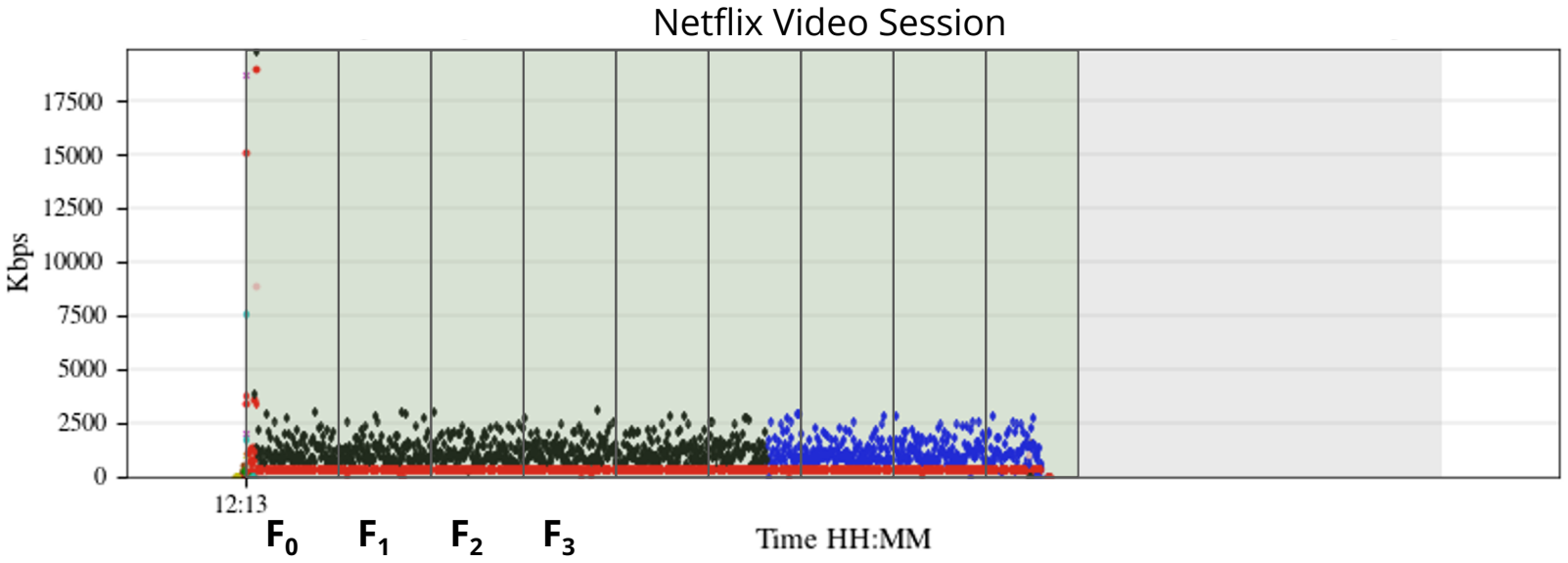
- Real world video session measurements do not reflect lab ones

Practical Challenges - Session Detection

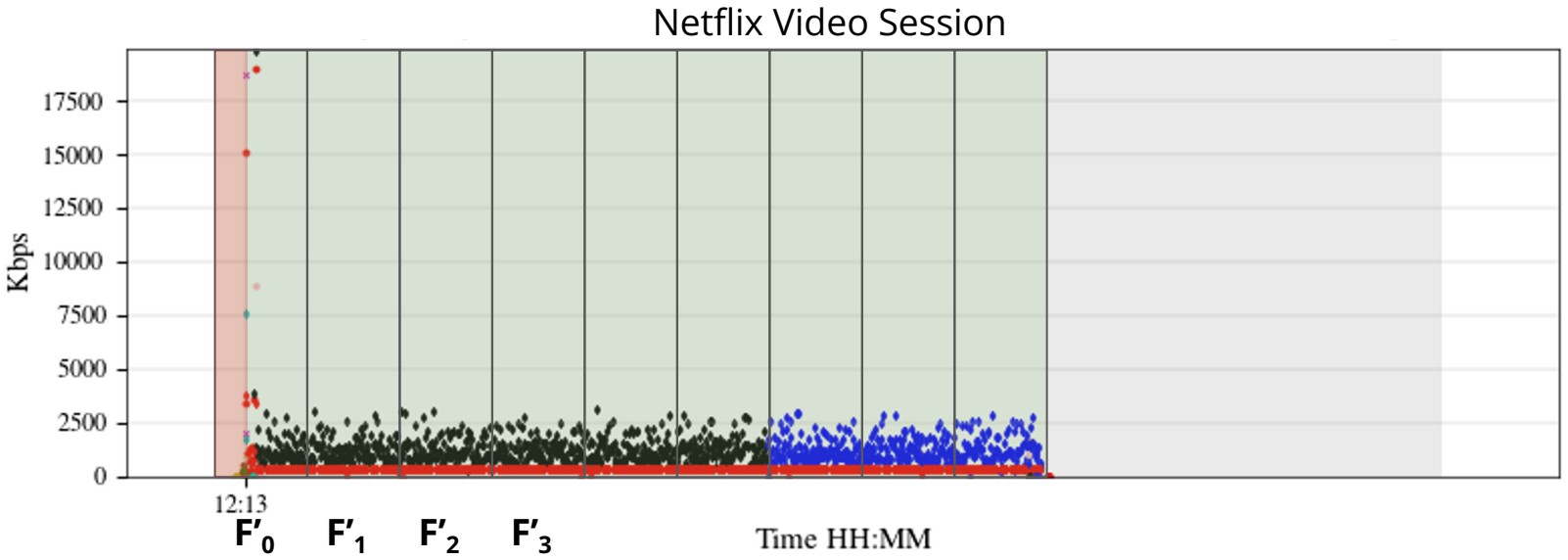


- Detect sessions boundaries
- Use traffic patterns to determine start and end of sessions
  - E.g. initial traffic spike

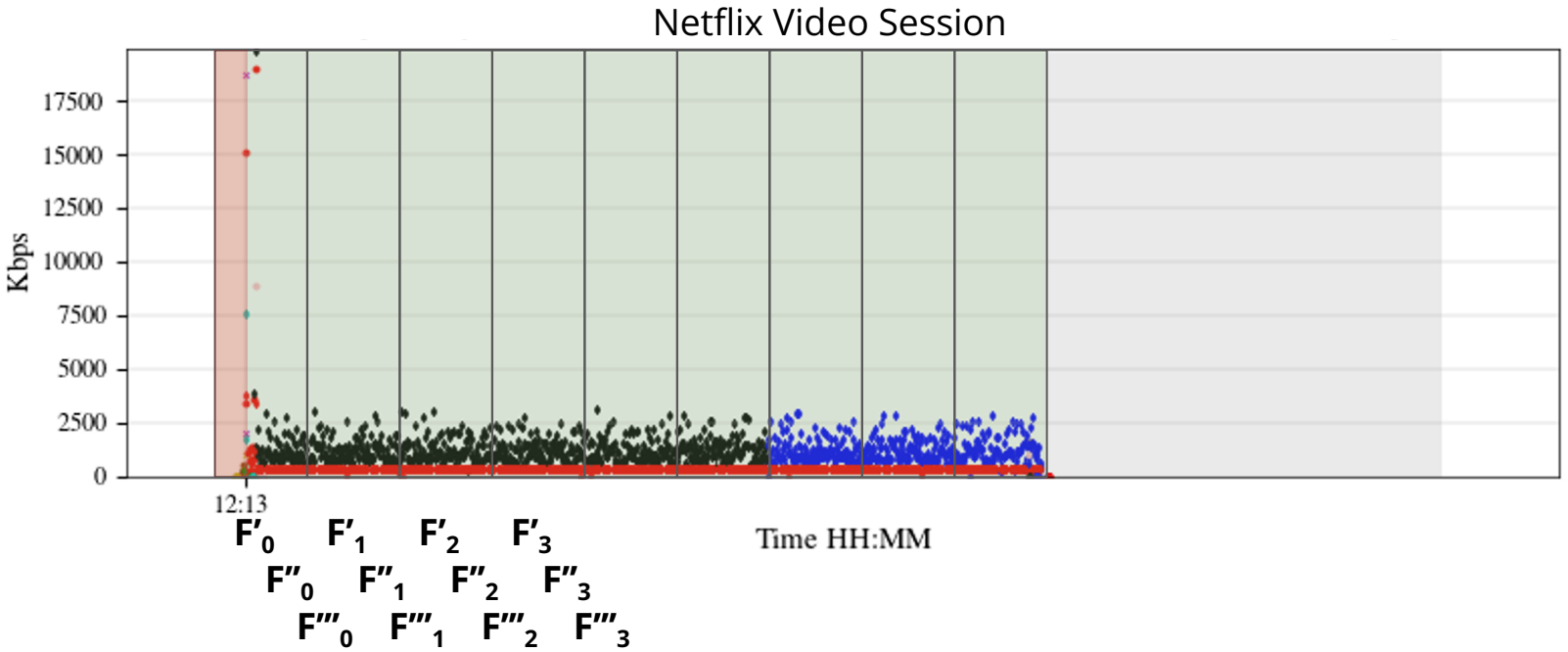
Practical Challenges - Data Granularity



Practical Challenges - Data Granularity



Practical Challenges - Data Granularity



- Train models using also noisy data

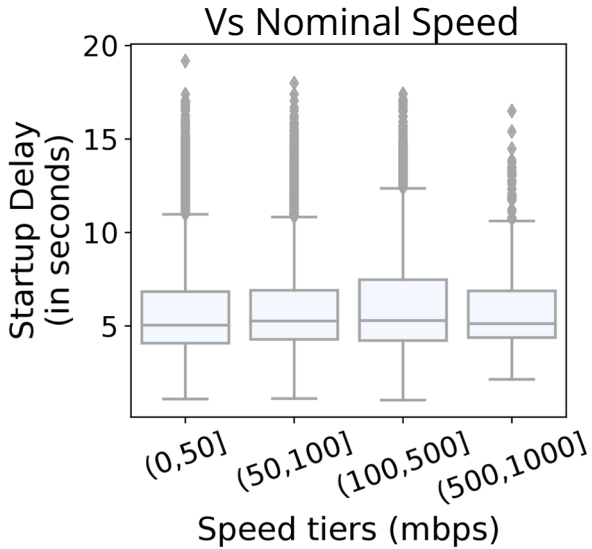
Communicating results to the wider audience

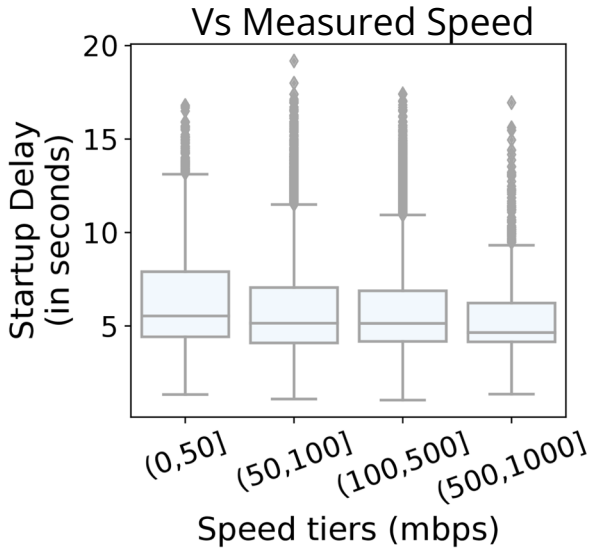
Deployment - Inference Results

- Apply the Net+App inference model to the collected sessions
  - Between January 23, 2018, and March 12, 2019

Speed (mbps)	Homes	Devices	# Video sessions			
			Netflix	YouTube	Amazon	Twitch
(0, 50]	20	329	21,774	65,488	2,612	2,584
(50, 100]	23	288	11,788	50,178	3,273	3,345
(100, 500]	19	277	13,201	38,691	2,030	197
(500, 1000]	4	38	523	442	86	1

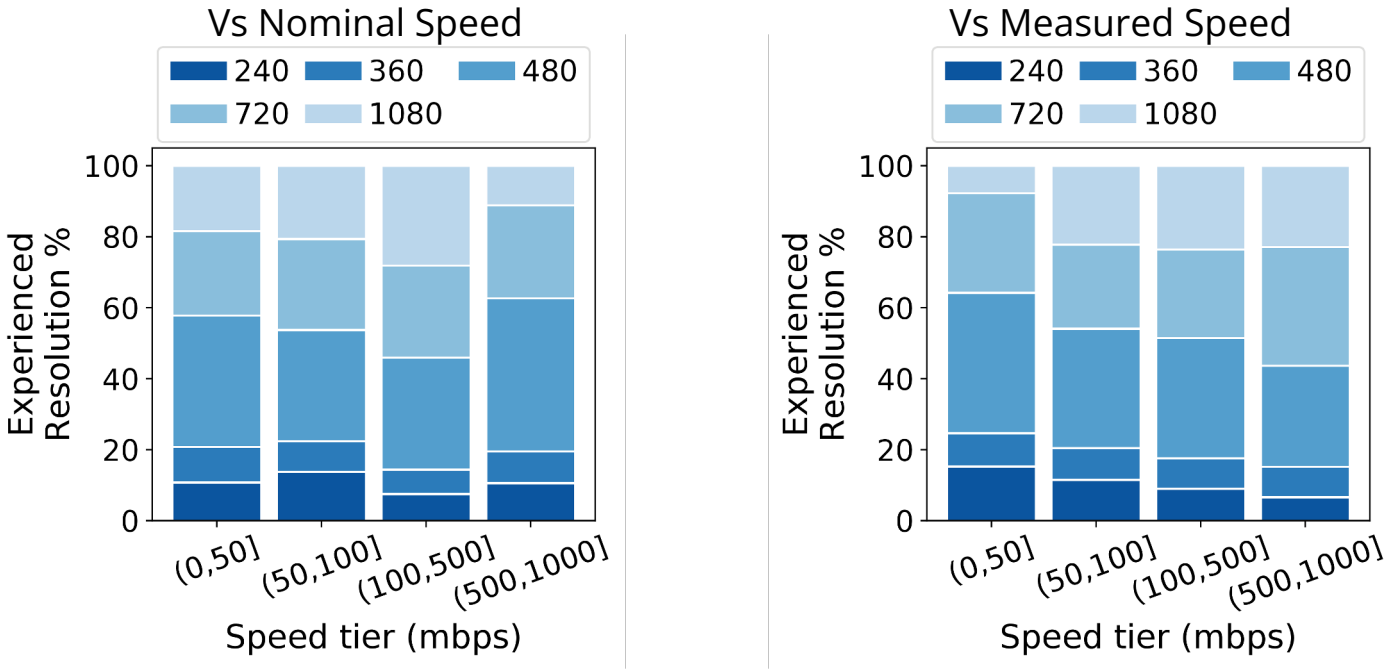
Inference Results - Startup Delay





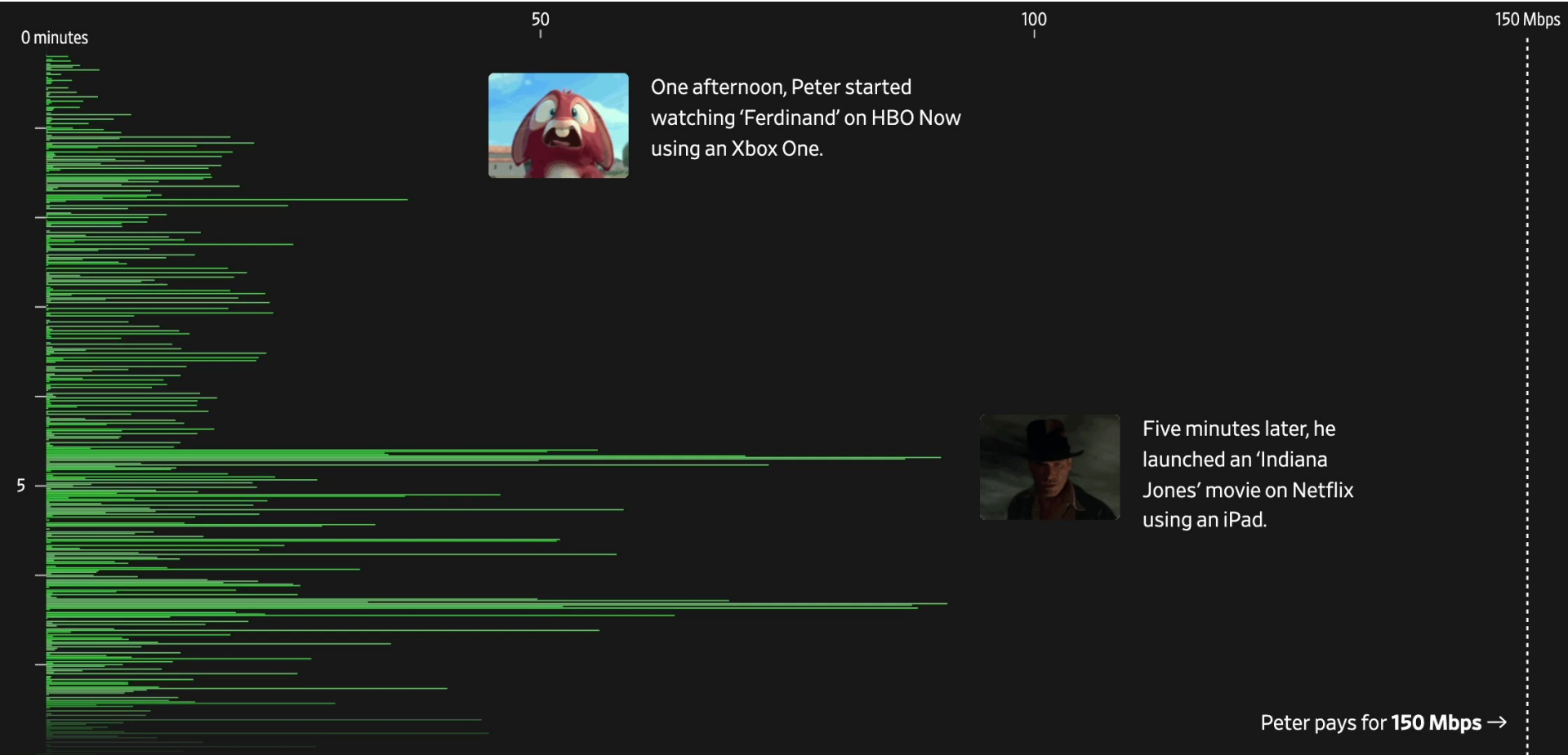
- How does access link capacity relate to startup delay?
- Median startup delays for each service tend to be similar across the subscription tiers

Inference Results - Resolution



- How does access link capacity relate to resolution?
  - Higher resolutions with higher capacities

How does that translate to a normal user?



Thank you!

francesco.bronzino@ens-lyon.fr

https://www.wsj.com/graphics/faster-internet-not-worth-it/
“Inferring streaming video quality from encrypted traffic: Practical models and deployment experience”, ACM SIGMETRICS 2020
https://github.com/inria-muse/video_collection
https://github.com/traffic-refinery/traffic-refinery
https://www.youtube.com/watch?v=kCshXyCmUho

Hands-on exercises

Exercise 1 : Extracting features from network traffic

https://colab.research.google.com/drive/1SXfKNGJfLLb9J7WPf-IvKXXLrqT0khKb?usp=sharing

GOAL: learn the basics of how to extract information from a pcap trace

1. Read a pcap file
2. Identify service types using DNS
3. Calculate network counters
4. Infer video segment downloads

Exercise 2 : Creating video models

https://colab.research.google.com/drive/1mjnjWv6xRIyNKQeePE5pbB4Xr6Be45p6?usp=sharing

GOAL: learn the basics of how to create a video resolution inference model using network traffic statistics

1. Clean the dataset
2. Simple resolution inference
3. Feature importance
4. The impact of different layers on the inference accuracy