Identifying Traffic Differentiation on Cellular Data Networks

Arash Molavi Kakhki[§], Abbas Razaghpanah[†], Rajesh Golani[†], David Choffnes[§], Phillipa Gill[†], Alan Mislove[§]

Northeastern University[§], Stony Brook University[†]

Northeastern University
College of Computer and Information Science
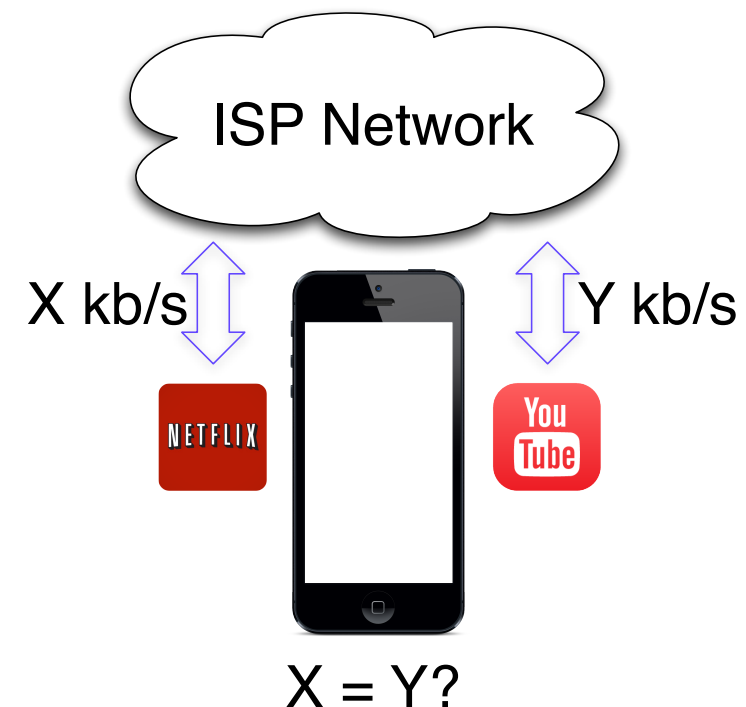
Stony Brook University

Introduction

Traffic differentiation

selectively changing the performance of network traffic

Reasons for differentiation:
- traffic engineering
- bandwidth management
- business reasons

ISP Network

X kb/s

Y kb/s

NETFLIX

You Tube

X = Y?

Do certain types of network traffic
receive better (or worse) performance?

Goals

Reliably detect differentiation in cellular networks

- On **any** app traffic

- *Without* requiring root privileges or OS modifications

- with few assumptions about traffic characteristics or packet shaper implementations

Our approach is the only known way to test differentiation from non-rooted mobile devices

Assumptions

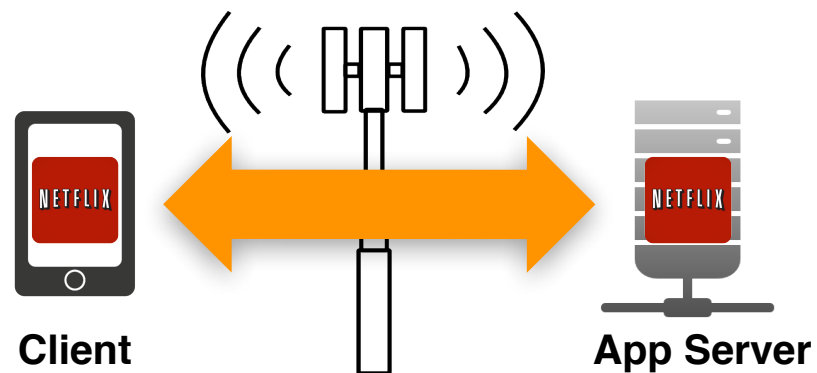What triggers differentiation?

We don't know.

We assume that they might trigger on:
- **IP addresses**
- **ports**
- total **number of connections**
- **payload signatures**
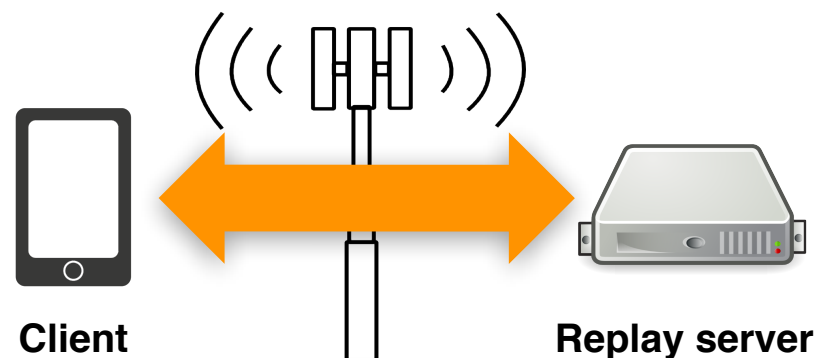- total **bandwidth**
- **time of day**

This is consistent with online manuals for DPI boxes
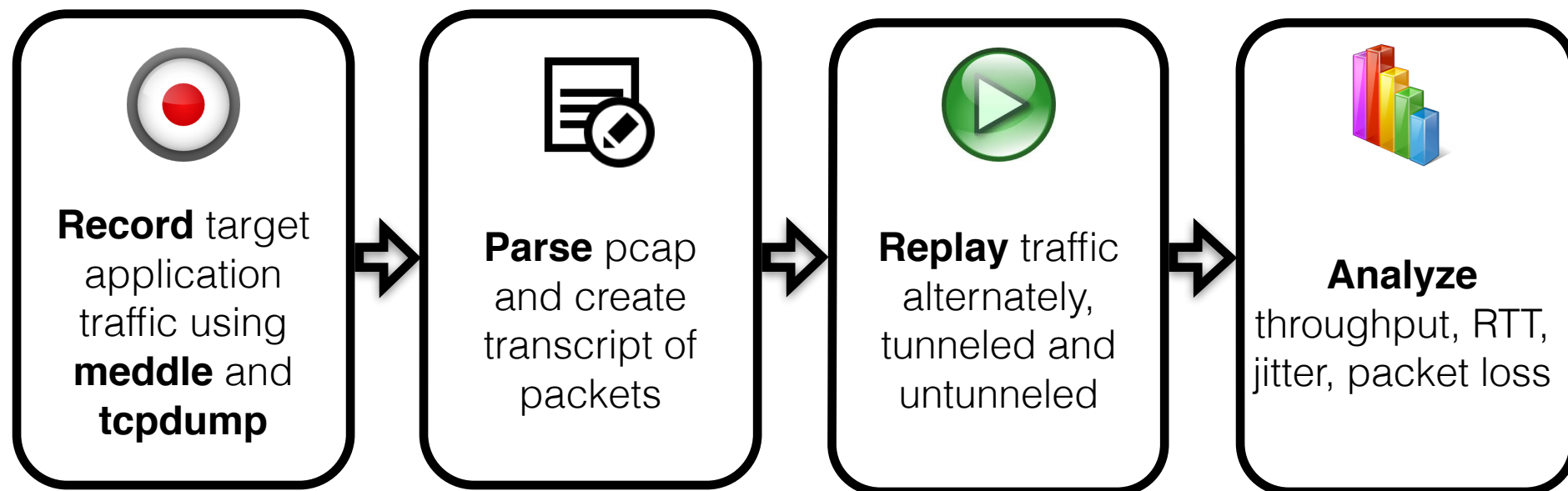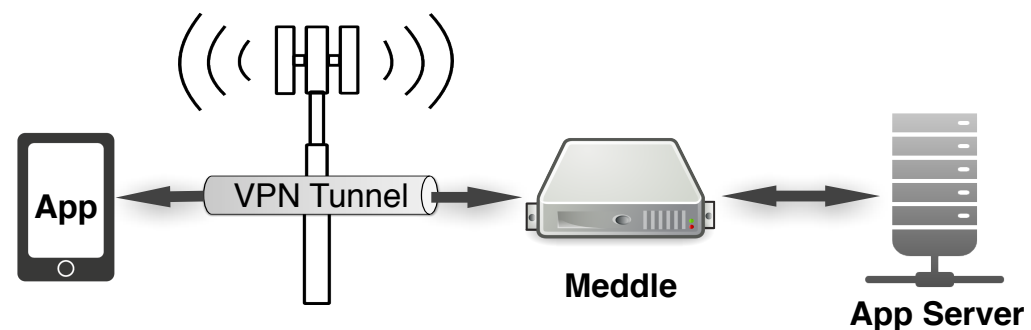
Record & Replay

1- Record:

Client   **App Server**

2- Replay:

Client   **Replay server**

Methodology

Record target application traffic using **meddle** and **tcpdump** → **Parse** pcap and create transcript of packets → **Replay** traffic alternately, tunneled and untunneled → **Analyze** throughput, RTT, jitter, packet loss
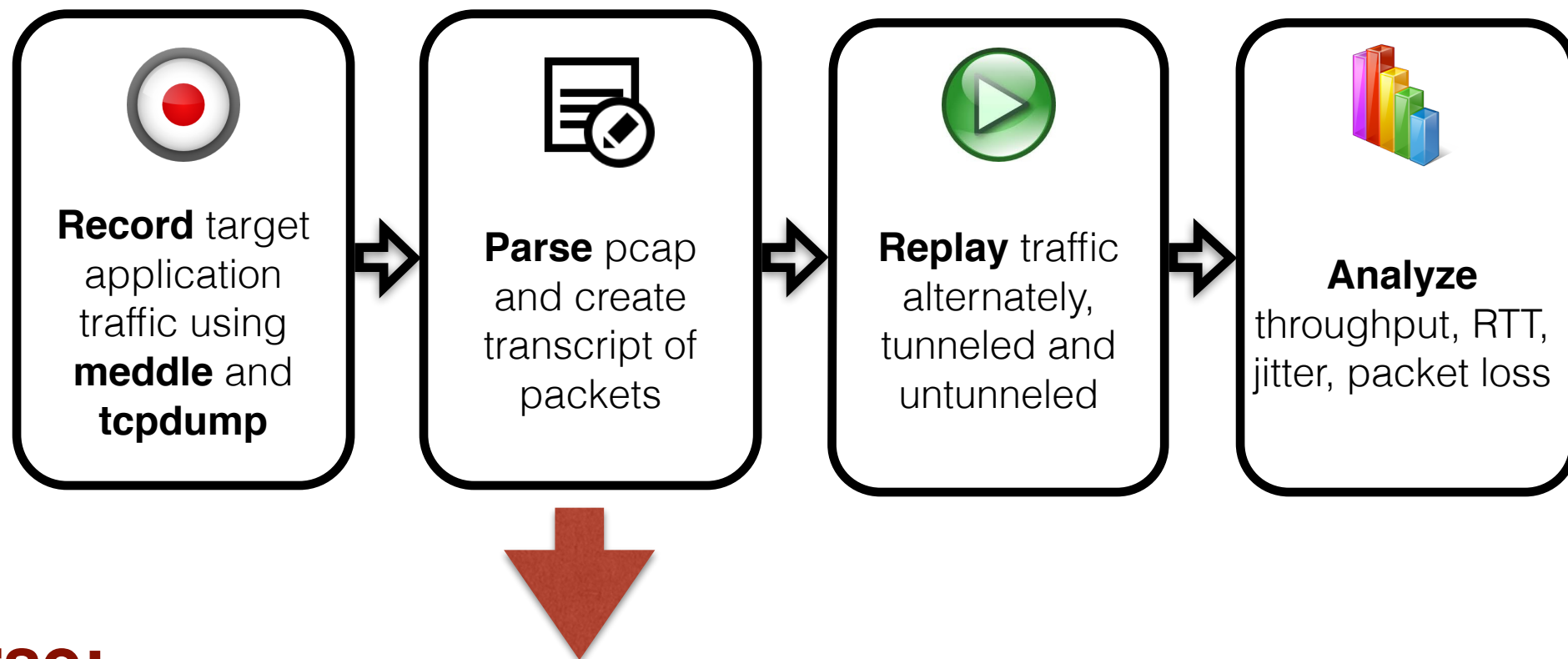
Record:

- Avoid running TCPDUMP on users' devices.
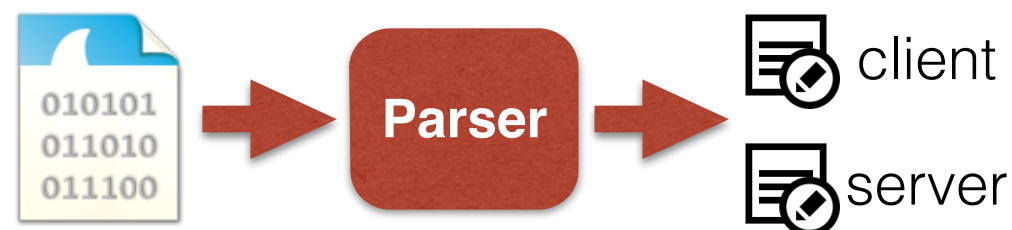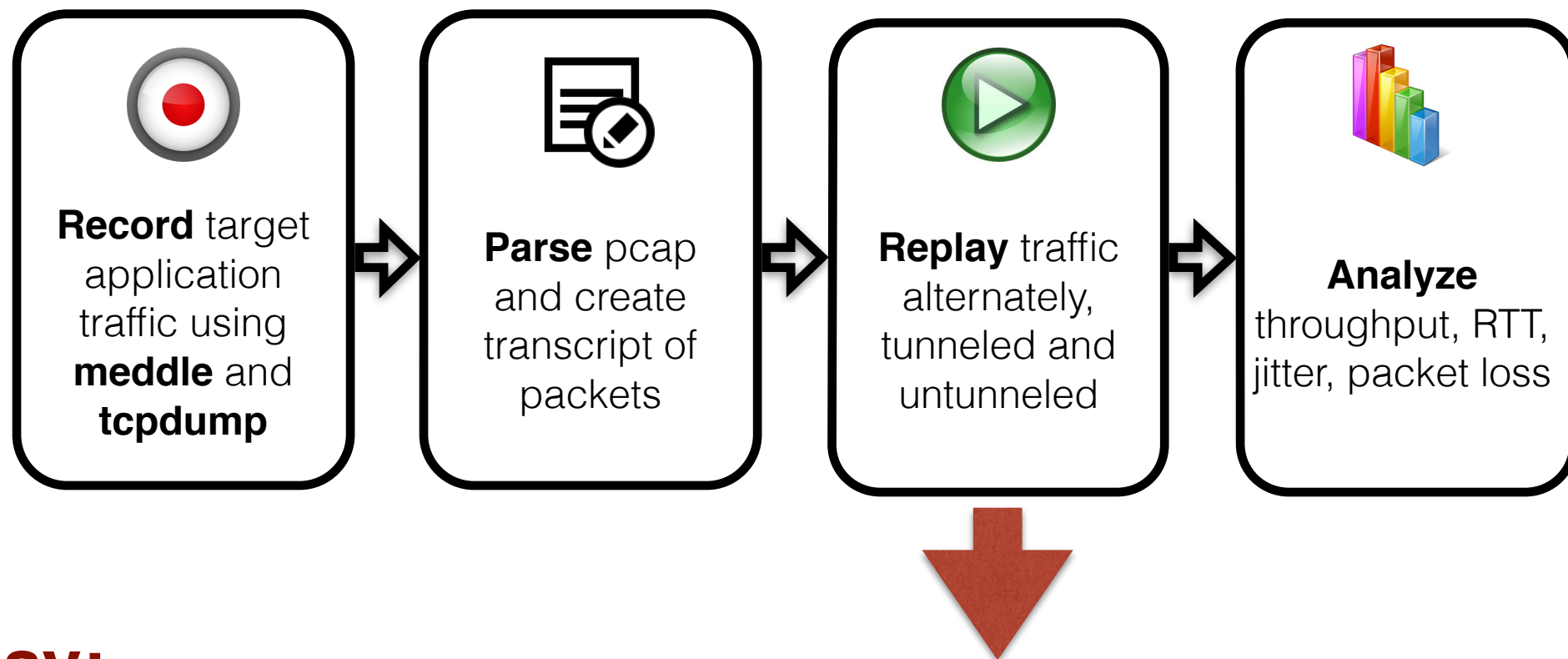- Utilize Meddle (a VPN proxy over IPSec) to record network traffic.

App ←→ VPN Tunnel ←→ Meddle ←→ App Server

Methodology

Record target application traffic using **meddle** and **tcpdump**

Parse pcap and create transcript of packets

Replay traffic alternately, tunneled and untunneled

Analyze throughput, RTT, jitter, packet loss

Parse:

- Create two objects, one for the client side and one for server side.
- Handle unrelated/noise traffic.

client

server

Methodology

Record target application traffic using **meddle** and **tcpdump** → **Parse** pcap and create transcript of packets → **Replay** traffic alternately, tunneled and untunneled → **Analyze** throughput, RTT, jitter, packet loss
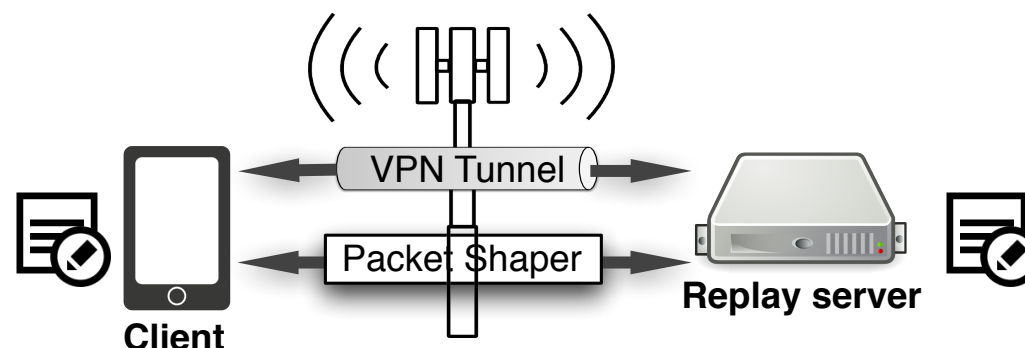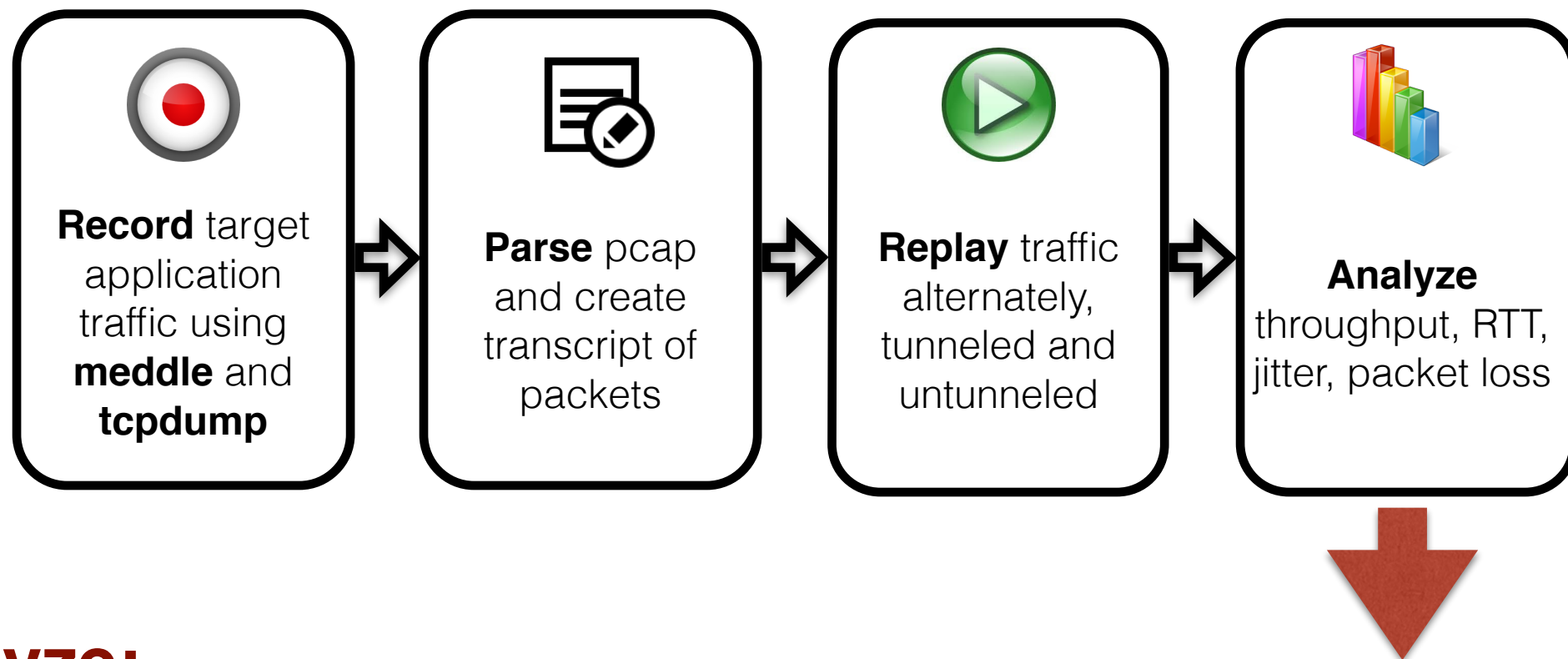
Replay:

- **replay** the salient features of application traffic such that it will be subject to differentiation from middleboxes

- **alternate** tunneling and plaintext to control visibility for packet shapers
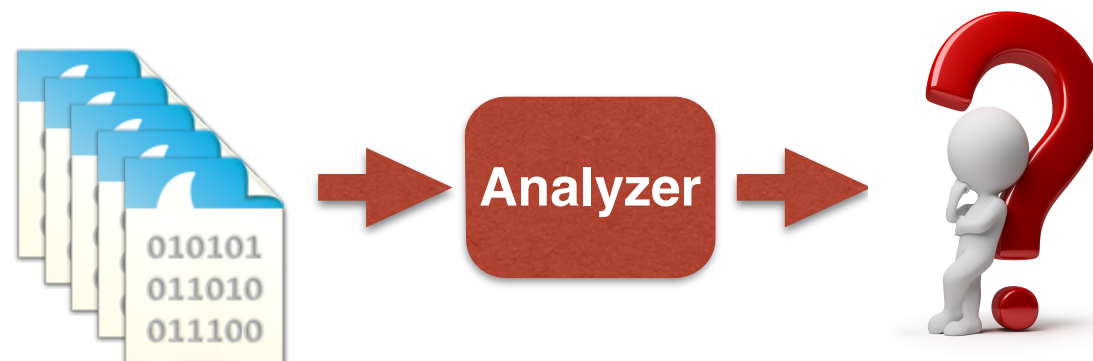

VPN Tunnel

Packet Shaper

Client

Replay server

Methodology

Record target application traffic using **meddle** and **tcpdump** → **Parse** pcap and create transcript of packets → **Replay** traffic alternately, tunneled and untunneled → **Analyze** throughput, RTT, jitter, packet loss
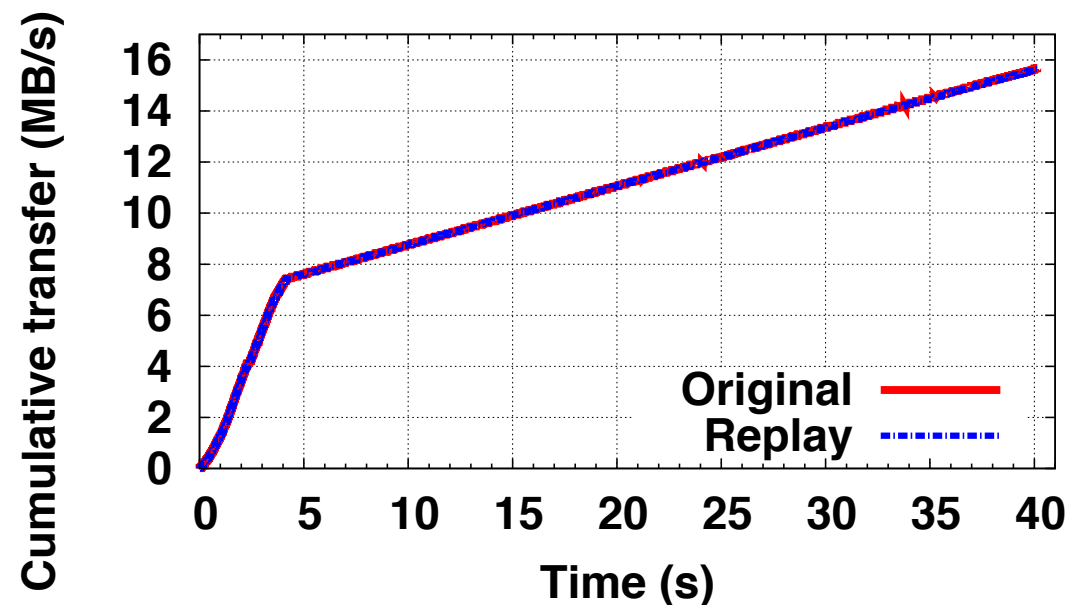
Analyze:

- **quantify** differentiation in term of throughput, round trip time, jitter, loss, …
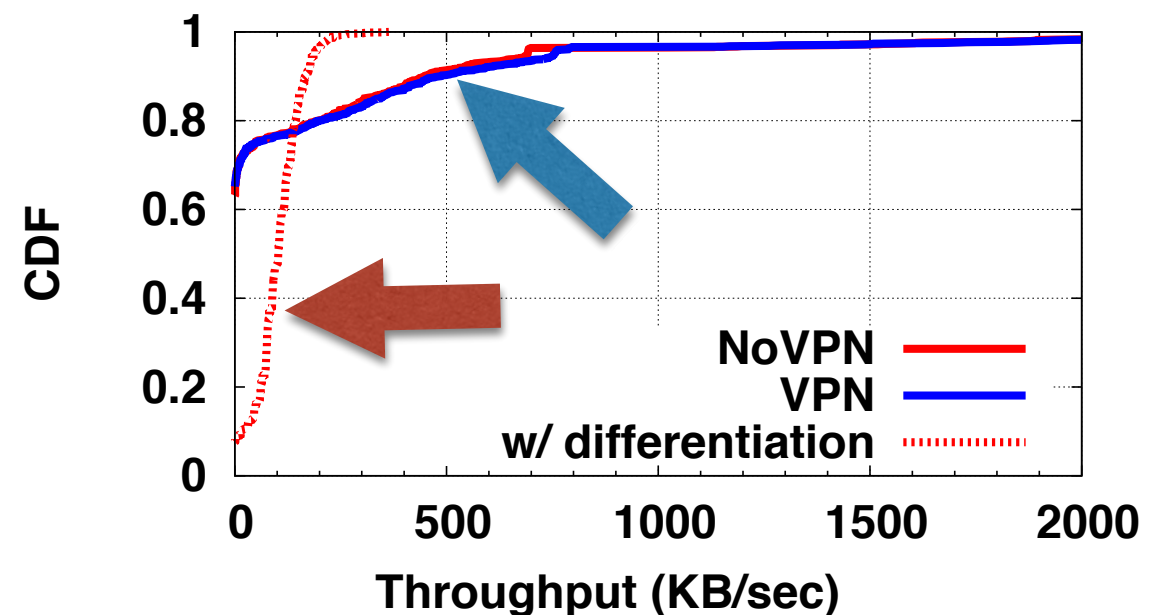
Proof of concept

Replay

YouTube traffic recorded on Verizon

Replay produces traffic **nearly identical** to the original traffic.

Analysis

3% loss and 10ms delay

Detect differentiation in a controlled environment

Future work

- Create a **mobile app** to enable wider adoption of the test (in progress). Run on networks we know there is differentiation to **validate** our techniques.

- Allow users to **record and replay custom traces** (longer term).

- Create a *Differentiation Watch* website and blog to report the results for different networks and blog about newly detected differentiation.

- **Source-spoofing** to detect destination IP-based differentiation.
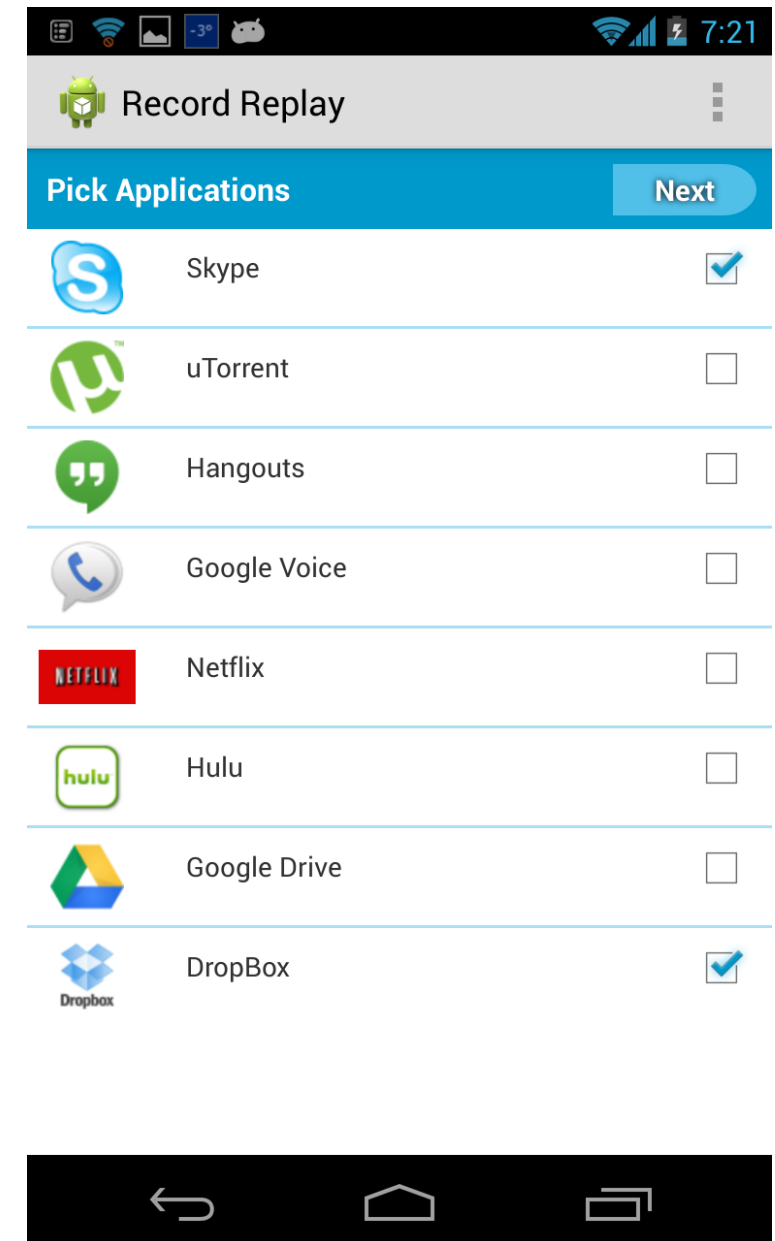
Related work

Previous work explored this problem for limited protocols and in limited environments.

	Switzerland	Glasnost	Us
Applications Tested	P2P	P2P and video	Any application
Features Tested	Packet manipulation	Performance	Both
Desktop App	Yes	Browser plugin	Yes
Smartphone App	No	No	Yes

Other closely related work

NetDiff, NetPolice, NANO

Thank you