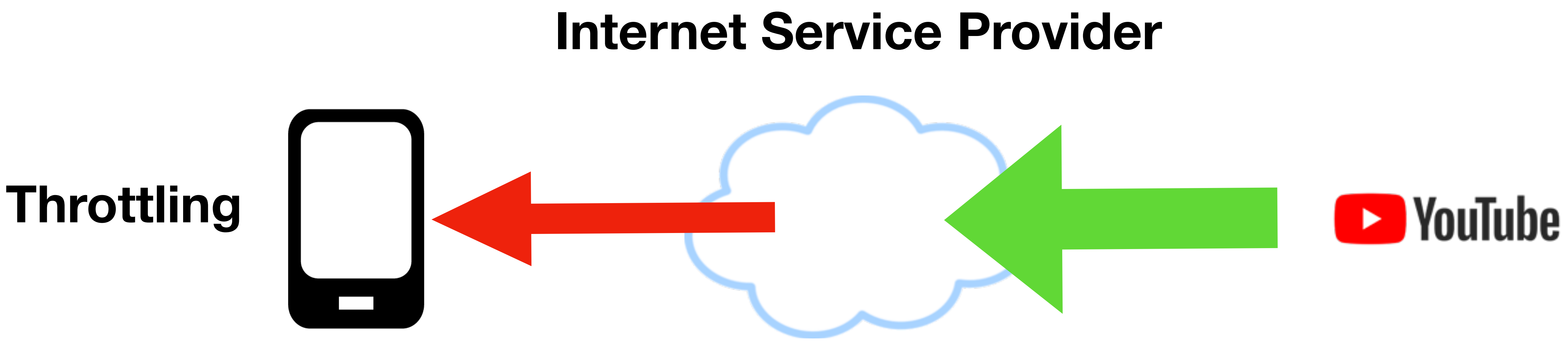
liberate, (n):

A library for exposing (traffic-classification) rules and avoiding them efficiently

Fangfan Li, Abbas Razaghpanah, Arash Molavi Kakhki,
Arian Akhavan Niaki, David Choffnes, Phillipa Gill, Alan Mislove

Traffic management

Traffic management

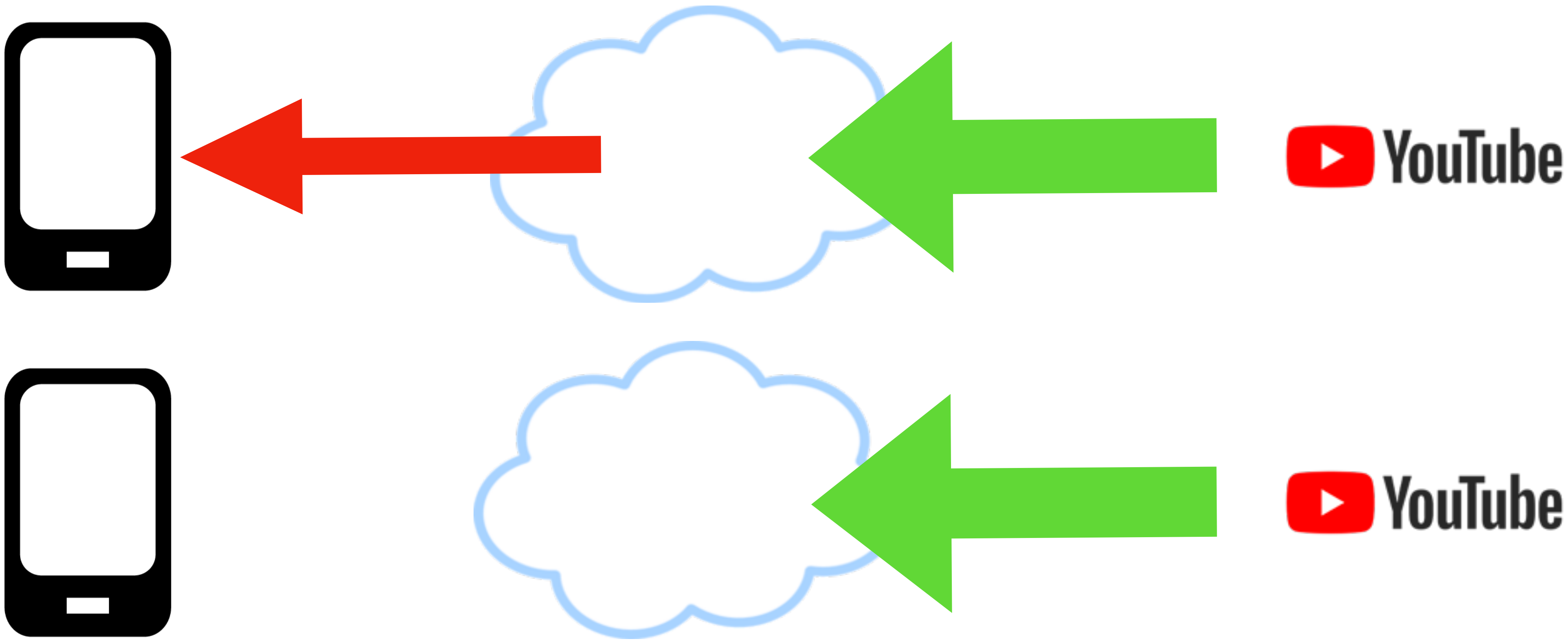


Internet Service Provider

Throttling

Traffic management

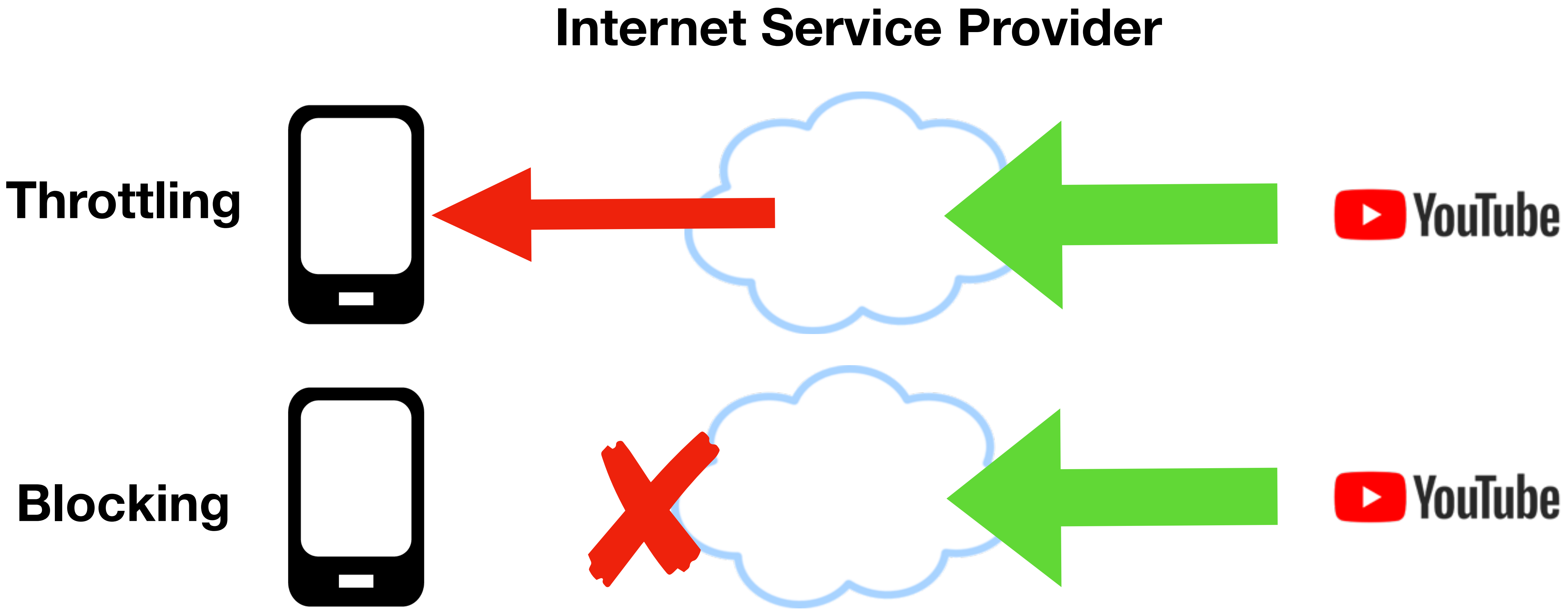
Internet Service Provider



Throttling

Blocking

Traffic management

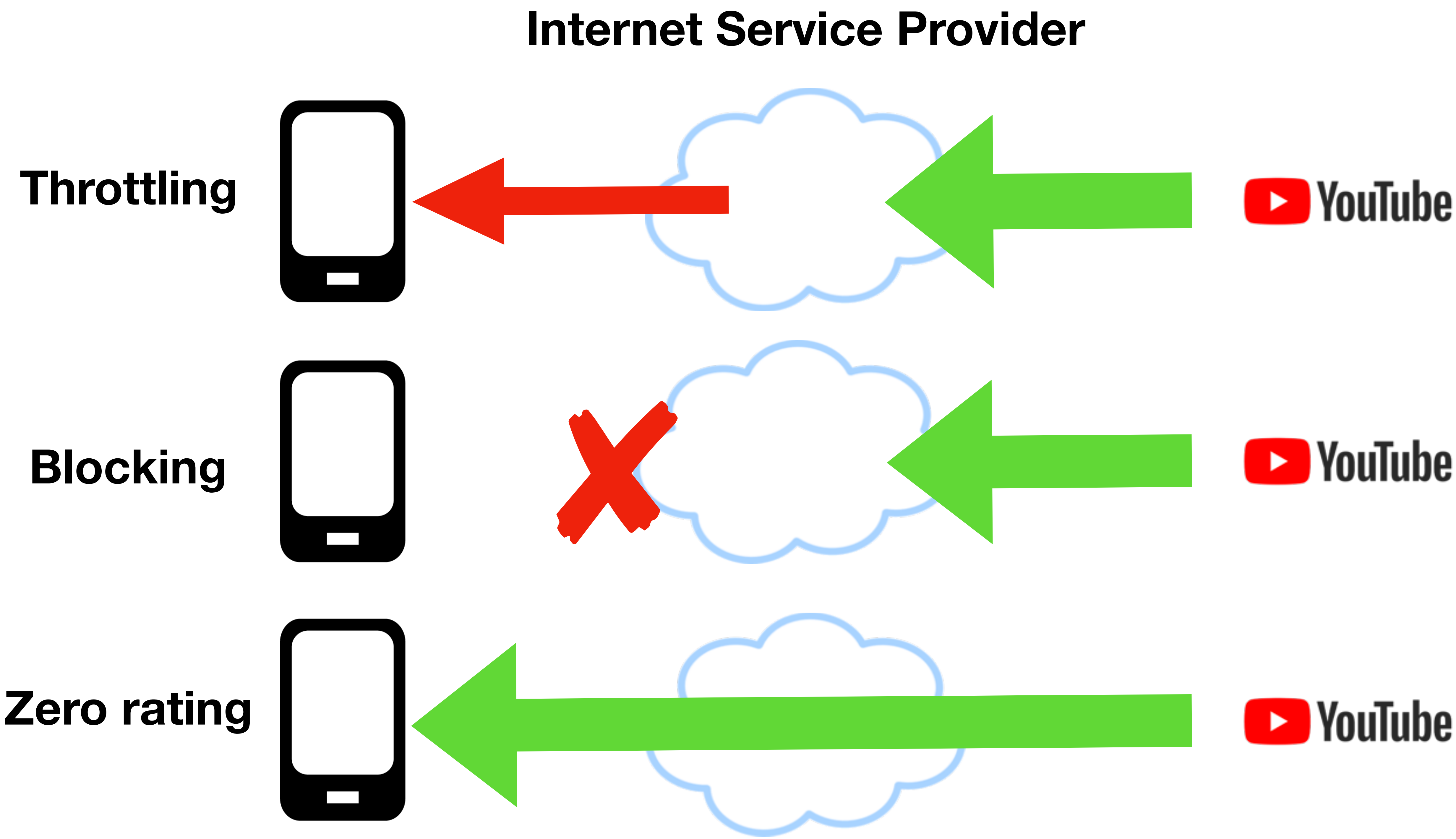


Internet Service Provider

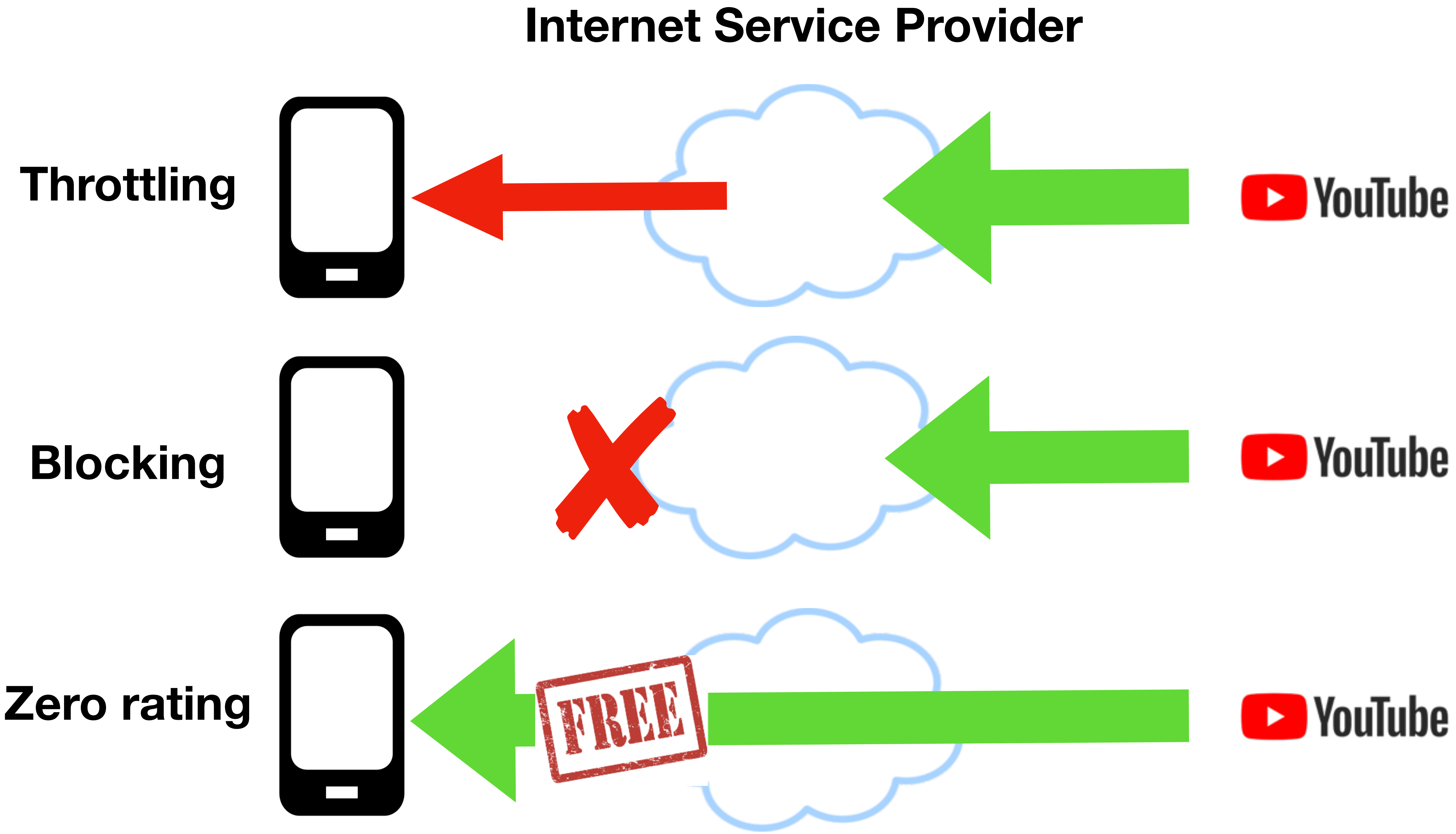
Throttling

Blocking

Traffic management



Traffic management



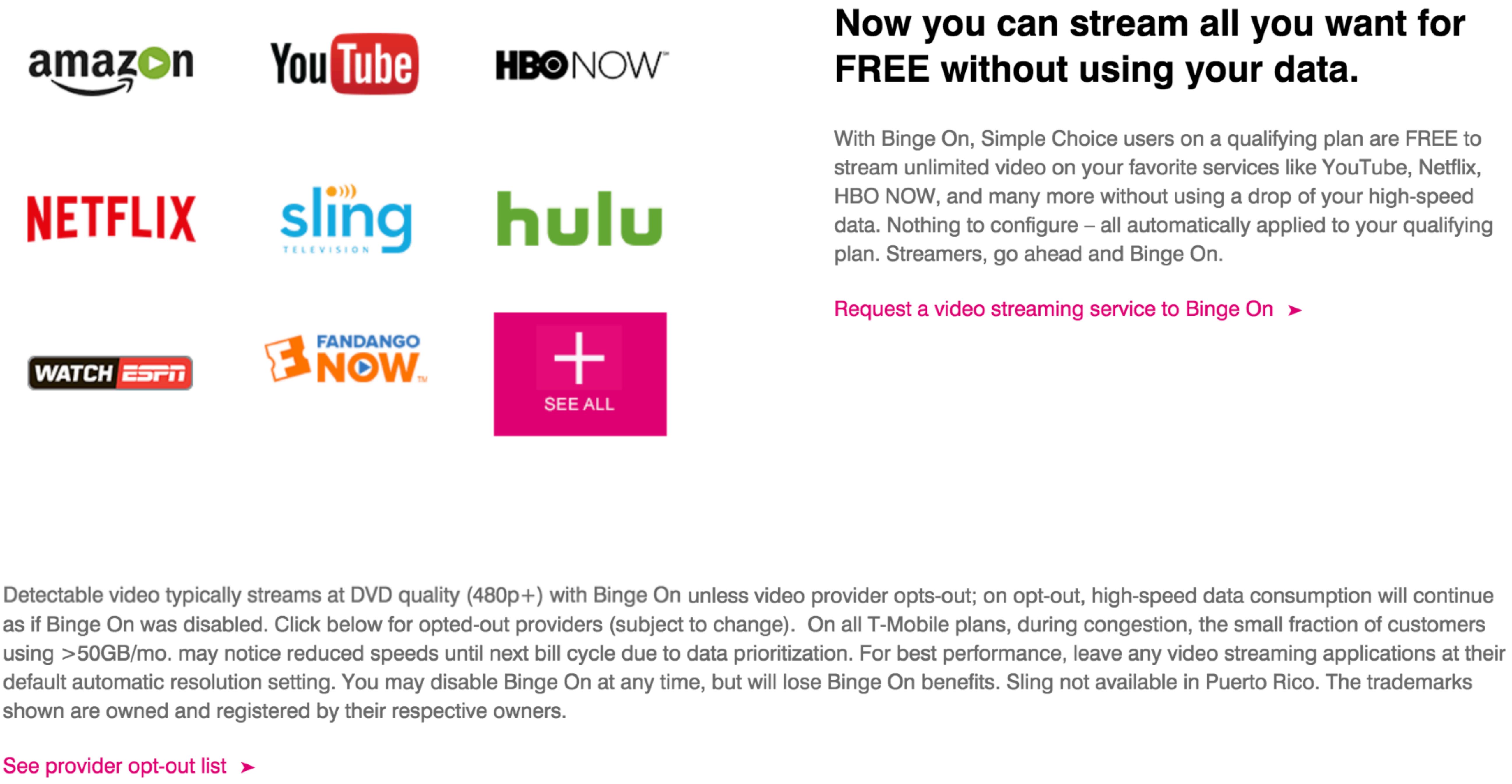
Internet Service Provider

Throttling

Blocking

Zero rating

Example policy



Now you can stream all you want for FREE without using your data.

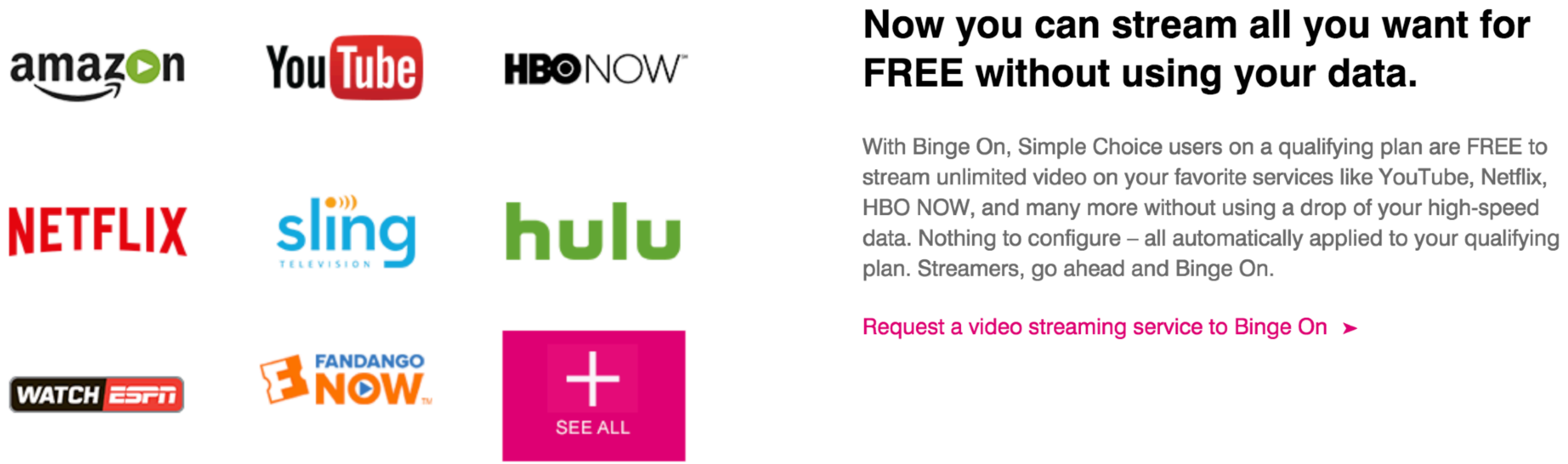
With Binge On, Simple Choice users on a qualifying plan are FREE to stream unlimited video on your favorite services like YouTube, Netflix, HBO NOW, and many more without using a drop of your high-speed data. Nothing to configure – all automatically applied to your qualifying plan. Streamers, go ahead and Binge On.

Request a video streaming service to Binge On ➤

Detectable video typically streams at DVD quality (480p+) with Binge On unless video provider opts-out; on opt-out, high-speed data consumption will continue as if Binge On was disabled. Click below for opted-out providers (subject to change). On all T-Mobile plans, during congestion, the small fraction of customers using >50GB/mo. may notice reduced speeds until next bill cycle due to data prioritization. For best performance, leave any video streaming applications at their default automatic resolution setting. You may disable Binge On at any time, but will lose Binge On benefits. Sling not available in Puerto Rico. The trademarks shown are owned and registered by their respective owners.

See provider opt-out list ➤

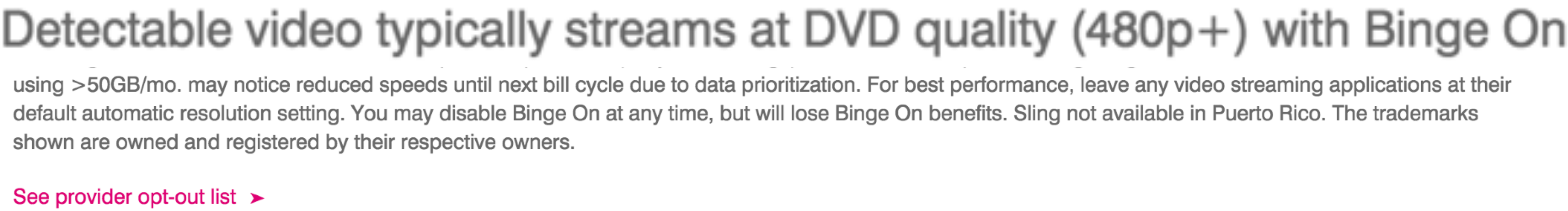
Example policy



Now you can stream all you want for FREE without using your data.

With Binge On, Simple Choice users on a qualifying plan are FREE to stream unlimited video on your favorite services like YouTube, Netflix, HBO NOW, and many more without using a drop of your high-speed data. Nothing to configure – all automatically applied to your qualifying plan. Streamers, go ahead and Binge On.

Request a video streaming service to Binge On ➤

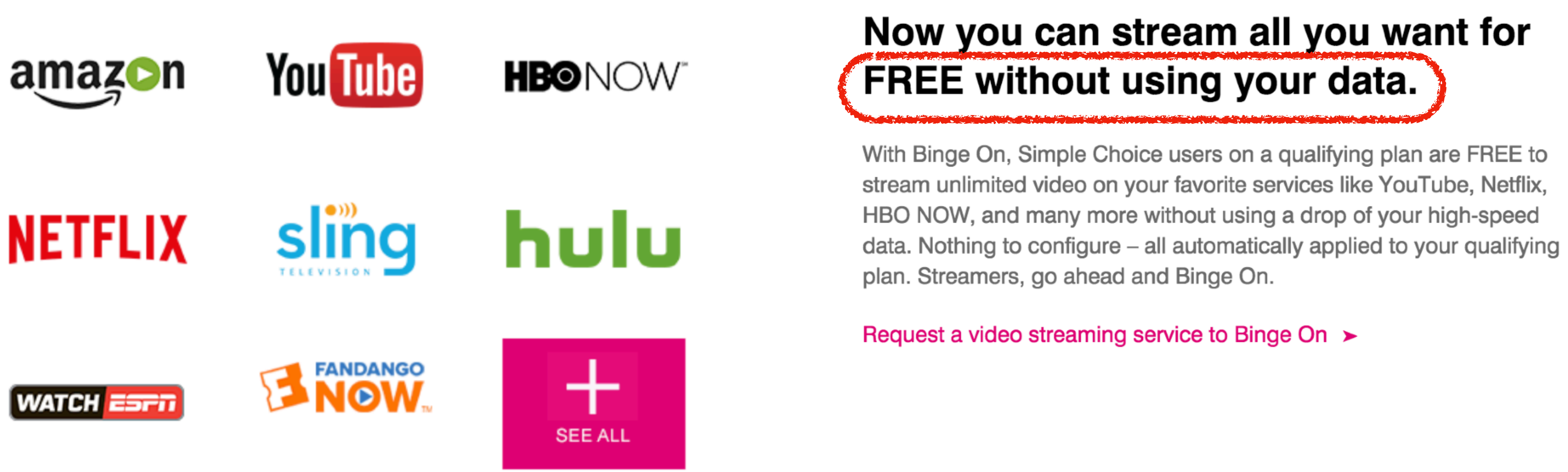


Detectable video typically streams at DVD quality (480p+) with Binge On

using >50GB/mo. may notice reduced speeds until next bill cycle due to data prioritization. For best performance, leave any video streaming applications at their default automatic resolution setting. You may disable Binge On at any time, but will lose Binge On benefits. Sling not available in Puerto Rico. The trademarks shown are owned and registered by their respective owners.

See provider opt-out list ➤

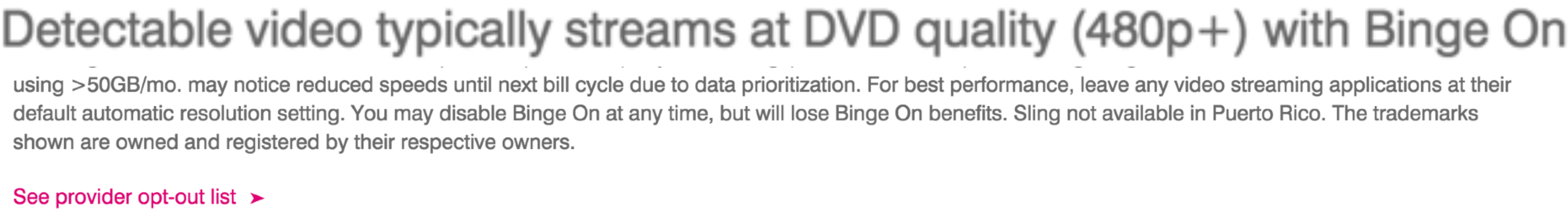
Example policy



Now you can stream all you want for FREE without using your data.

With Binge On, Simple Choice users on a qualifying plan are FREE to stream unlimited video on your favorite services like YouTube, Netflix, HBO NOW, and many more without using a drop of your high-speed data. Nothing to configure – all automatically applied to your qualifying plan. Streamers, go ahead and Binge On.

Request a video streaming service to Binge On ➤

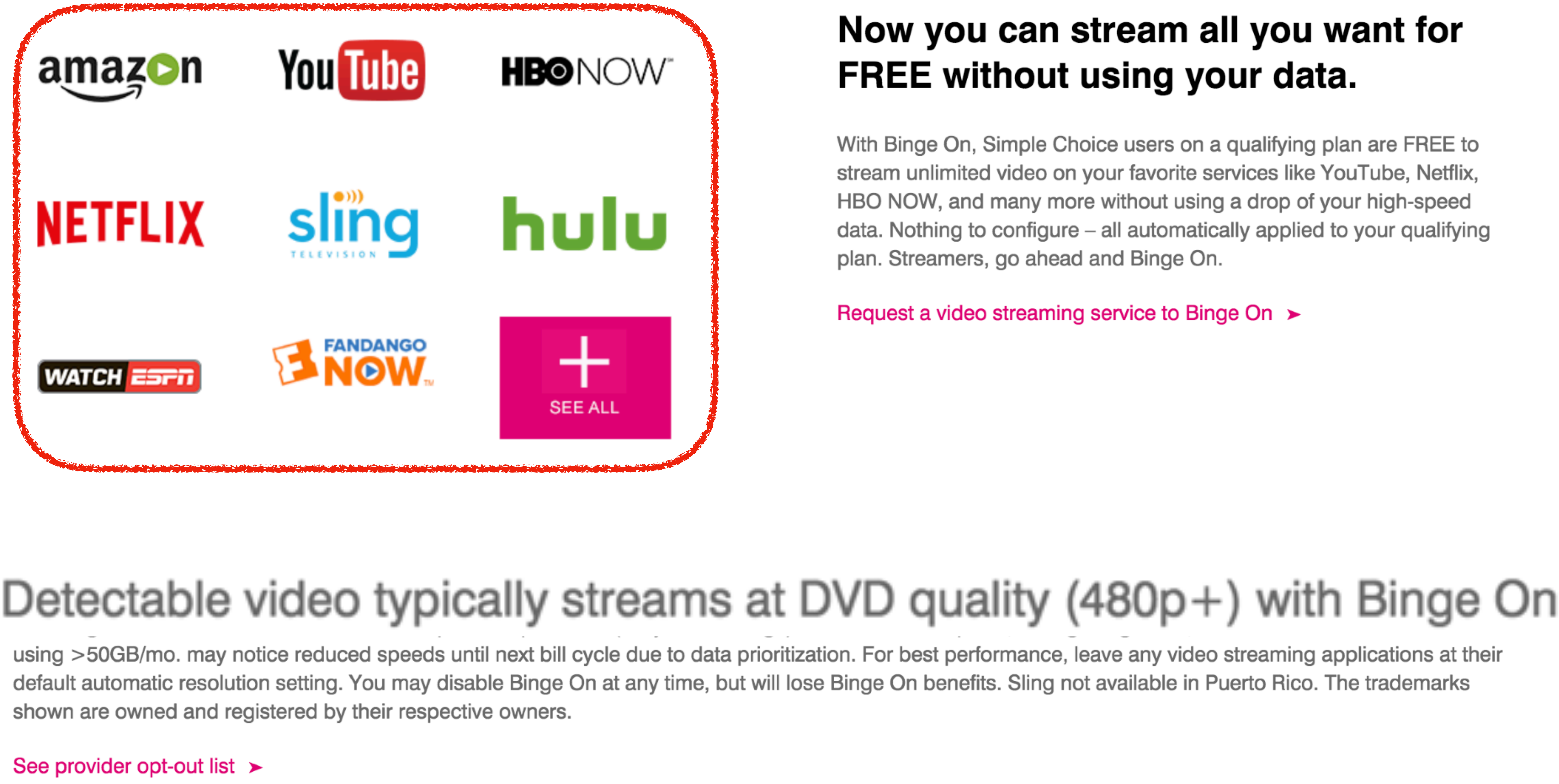


Detectable video typically streams at DVD quality (480p+) with Binge On

using >50GB/mo. may notice reduced speeds until next bill cycle due to data prioritization. For best performance, leave any video streaming applications at their default automatic resolution setting. You may disable Binge On at any time, but will lose Binge On benefits. Sling not available in Puerto Rico. The trademarks shown are owned and registered by their respective owners.

See provider opt-out list ➤

Example policy



Now you can stream all you want for FREE without using your data.

With Binge On, Simple Choice users on a qualifying plan are FREE to stream unlimited video on your favorite services like YouTube, Netflix, HBO NOW, and many more without using a drop of your high-speed data. Nothing to configure – all automatically applied to your qualifying plan. Streamers, go ahead and Binge On.

Request a video streaming service to Binge On ➤

Detectable video typically streams at DVD quality (480p+) with Binge On

using >50GB/mo. may notice reduced speeds until next bill cycle due to data prioritization. For best performance, leave any video streaming applications at their default automatic resolution setting. You may disable Binge On at any time, but will lose Binge On benefits. Sling not available in Puerto Rico. The trademarks shown are owned and registered by their respective owners.

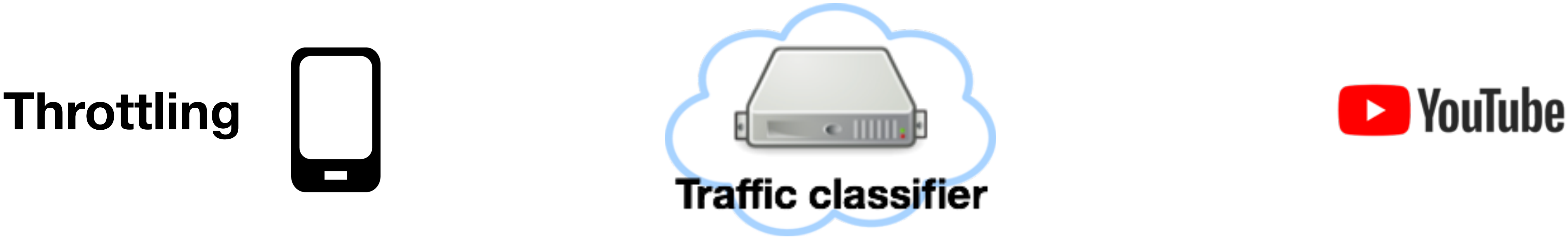
See provider opt-out list ➤

Lack of user control

Throttling

Lack of user control

- Policies are implemented by DPI (Deep Packet Inspection) devices [IMC 16]

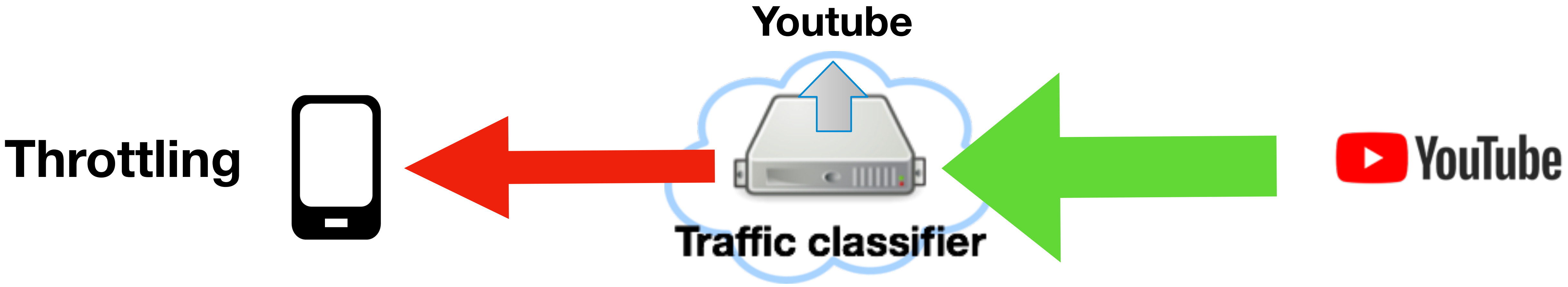


Throttling

Traffic classifier

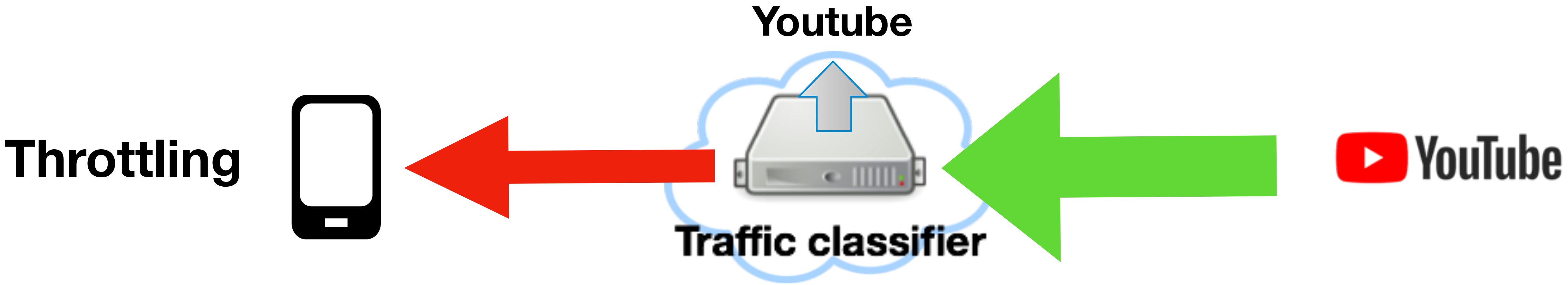
Lack of user control

- Policies are implemented by DPI (Deep Packet Inspection) devices [IMC 16]



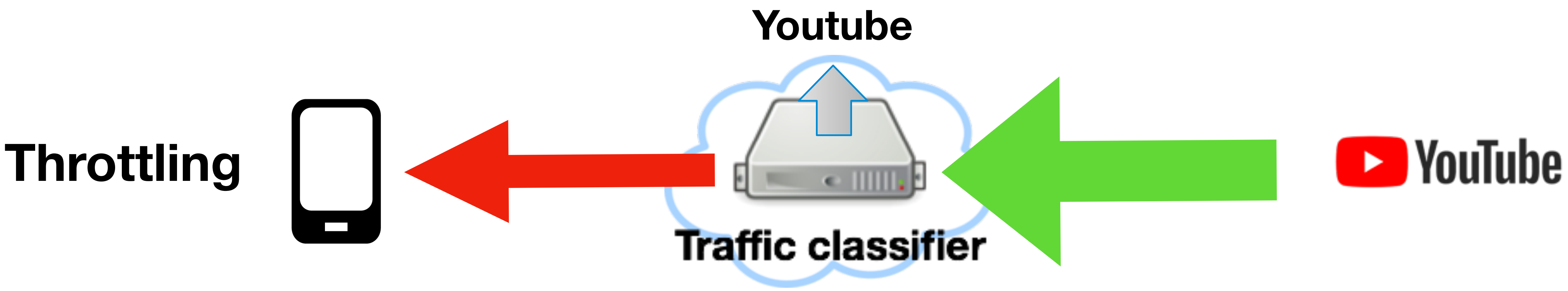
Lack of user control

- Policies are implemented by DPI (Deep Packet Inspection) devices [IMC 16]
- Differentiation policy can be harmful or unwanted to users/content providers



Lack of user control

- Policies are implemented by DPI (Deep Packet Inspection) devices [IMC 16]
- Differentiation policy can be harmful or unwanted to users/content providers
- Users/content providers have no control over these policies



Previous work

Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

- **Limitations:**

Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

- **Limitations:**
  - Brittle

Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

- **Limitations:**
  - Brittle
  - Development effort

Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

- **Limitations:**
  - Brittle
  - Development effort
  - Performance

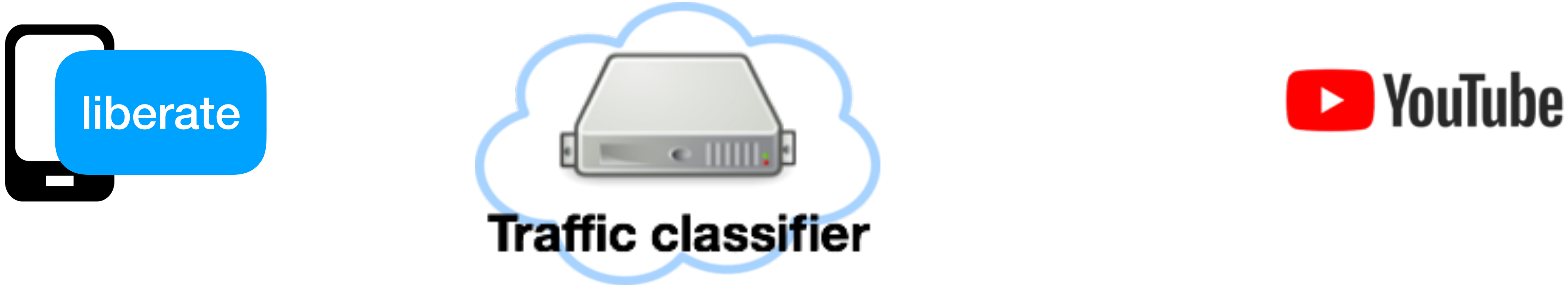
Previous work

- **Approaches**:
  - VPNs and proxies
  - Covert channels
  - Obfuscating traffic
  - Domain fronting

- **Limitations:**
  - Brittle
  - Development effort
  - Performance
  - Manual inspection

Goals of liberate

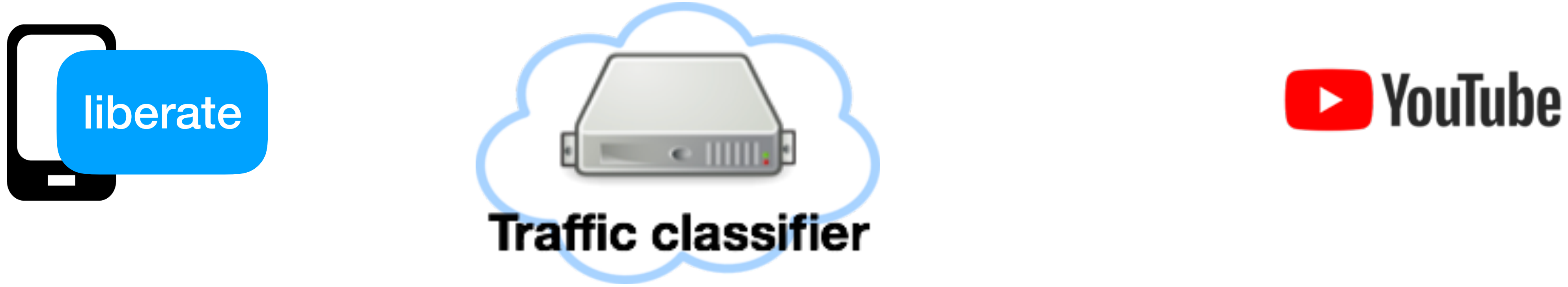
Evade throttling



Goals of liberate

- A technical solution for **detecting** and **evading** unwanted policies

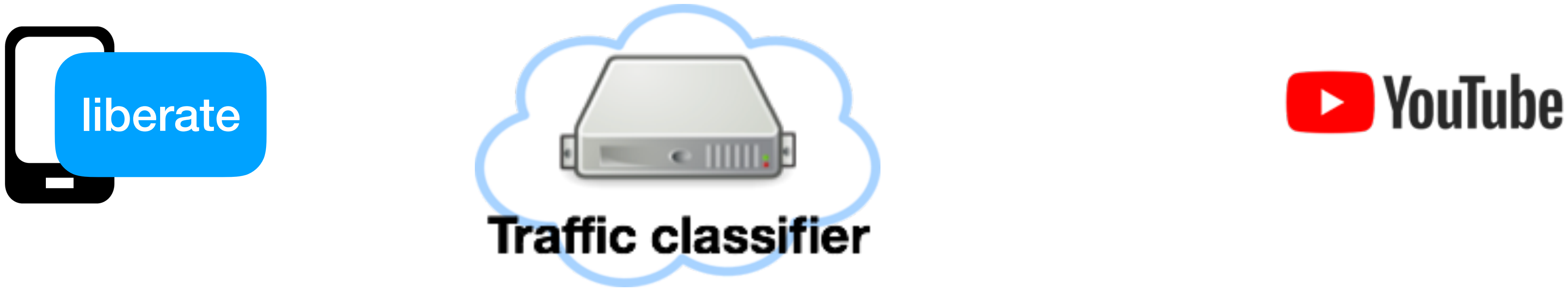
Evade throttling



Goals of liberate

- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade

Evade throttling



Goals of liberate

- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade
  - Automatically

Evade throttling

liberate
Traffic classifier
YouTube

Goals of liberate

- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade
  - Automatically
  - Adaptively

Evade throttling

liberate
Traffic classifier
YouTube

Goals of liberate

- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade
  - Automatically
  - Adaptively
  - Unilaterally

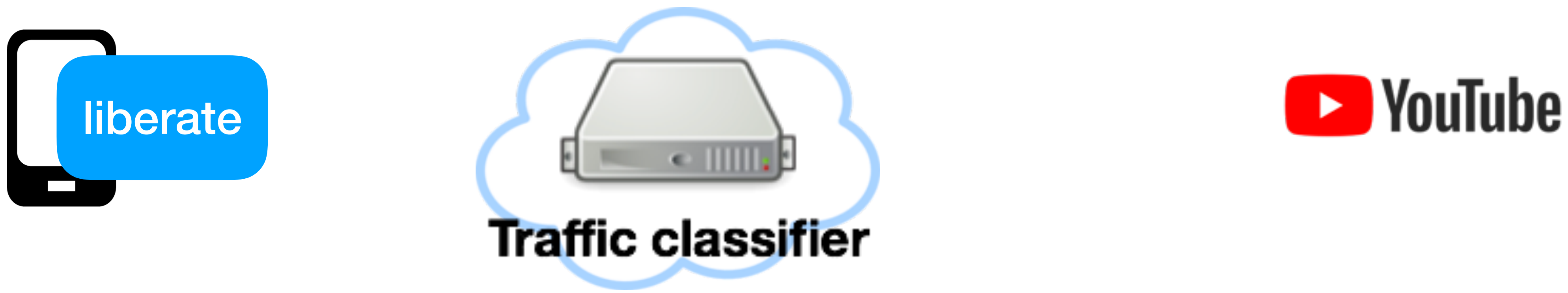
Evade throttling

liberate
Traffic classifier
YouTube

Goals of liberate

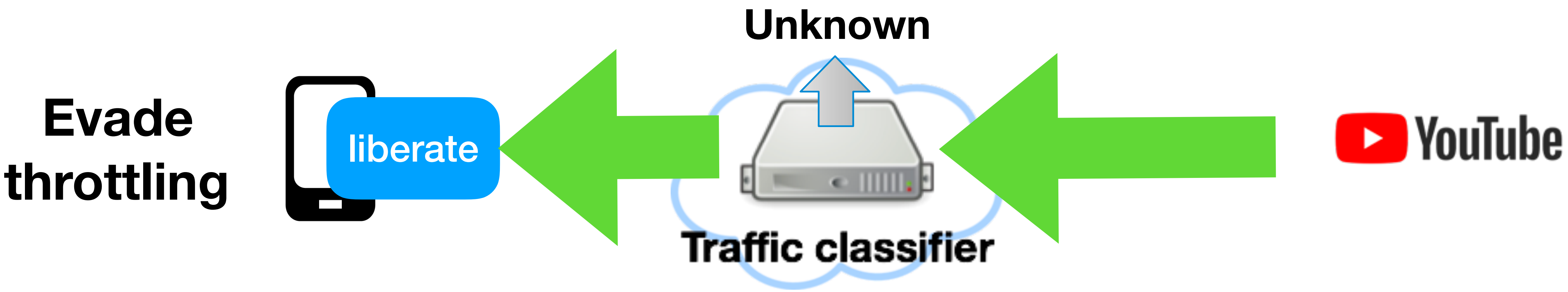
- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade
  - Automatically
  - Adaptively
  - Unilaterally
  - With low overhead

Evade throttling



Goals of liberate

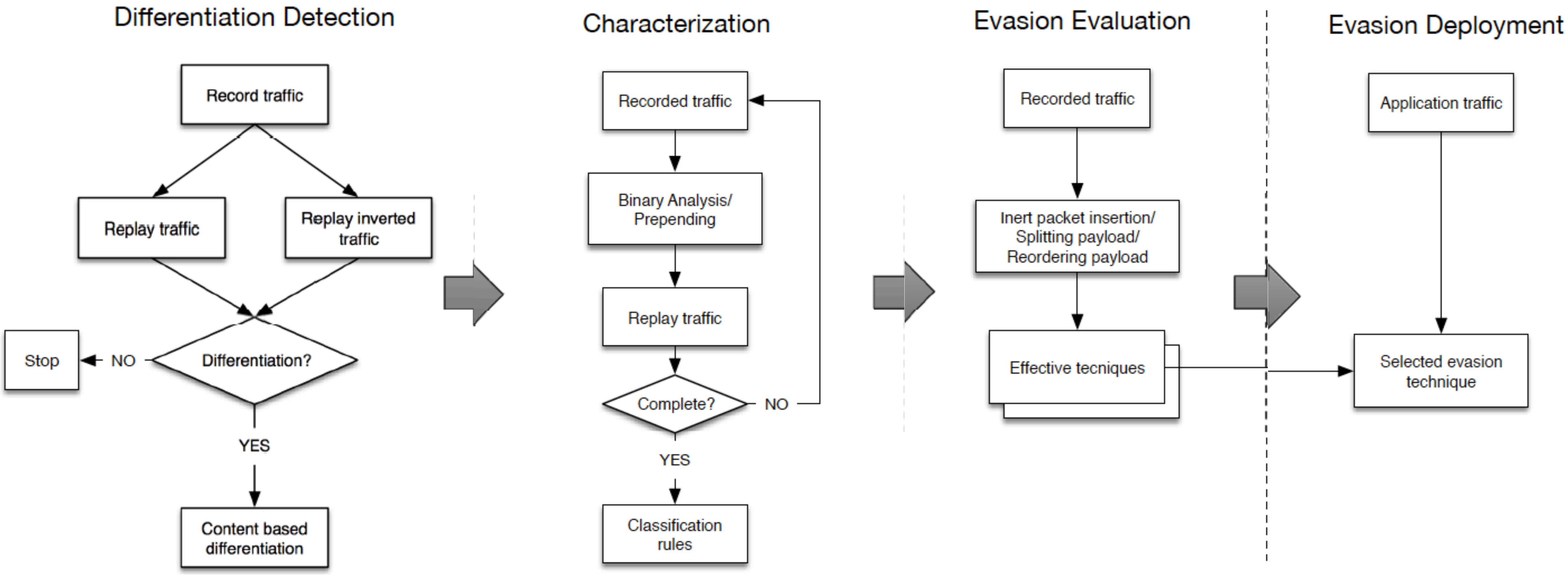
- A technical solution for **detecting** and **evading** unwanted policies
- Enables unmodified applications to evade
  - Automatically
  - Adaptively
  - Unilaterally
  - With low overhead



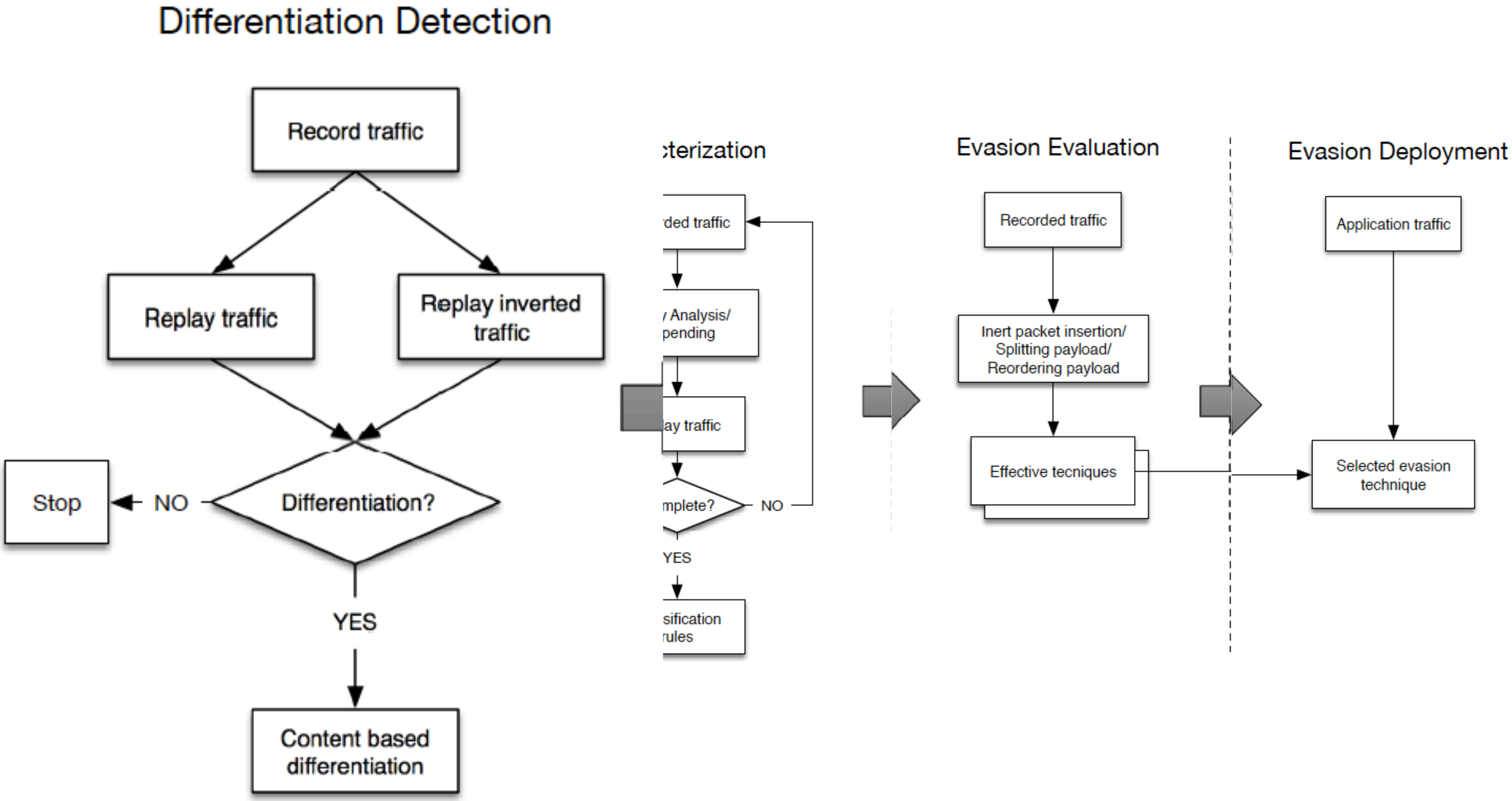
Outline

- Design and implementation
    - Traffic-classification rules detection
    - Evasion techniques
    - Implementation
- Evaluation
    - Effectiveness across multiple networks

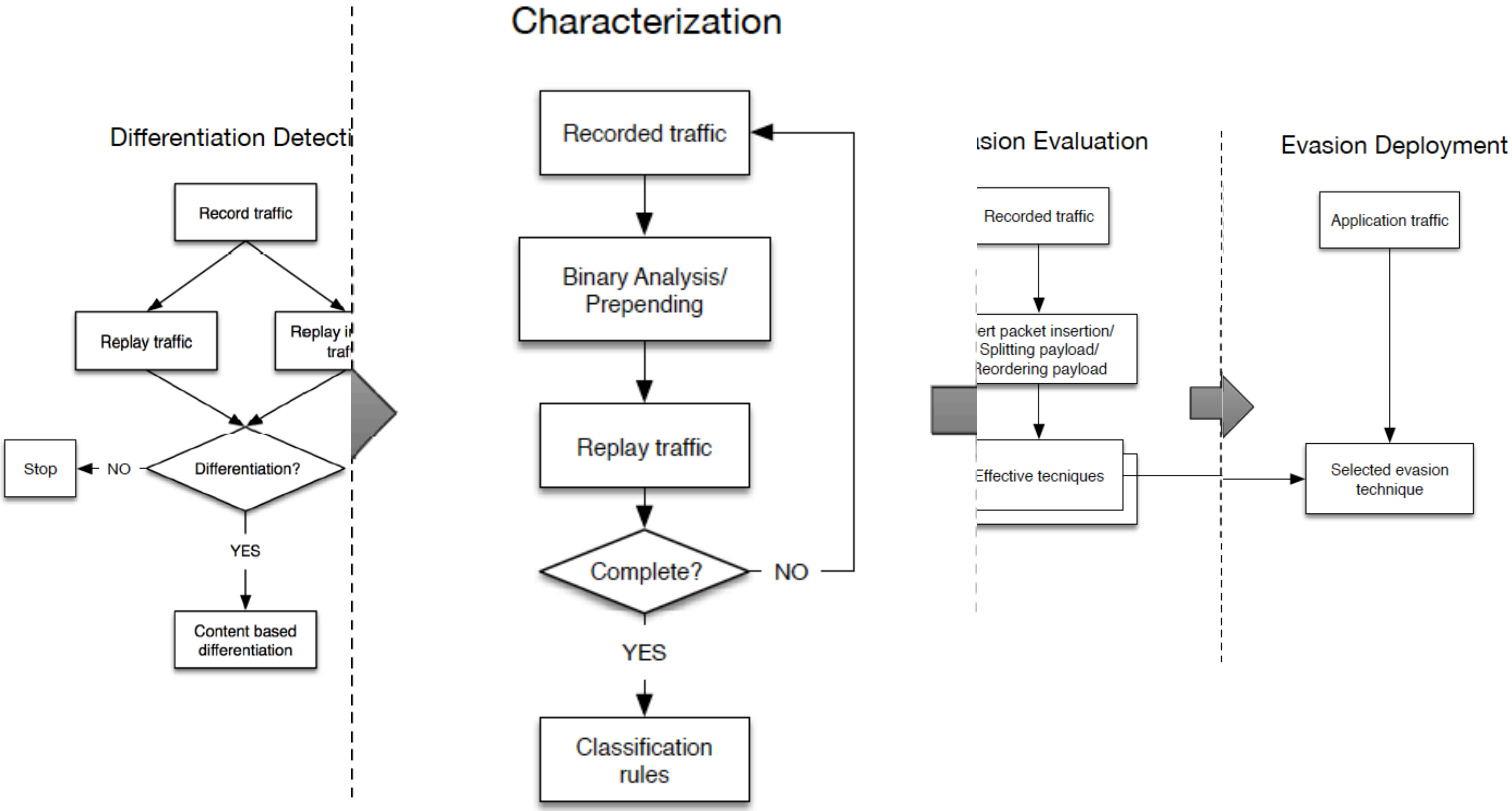
Overview of liberate



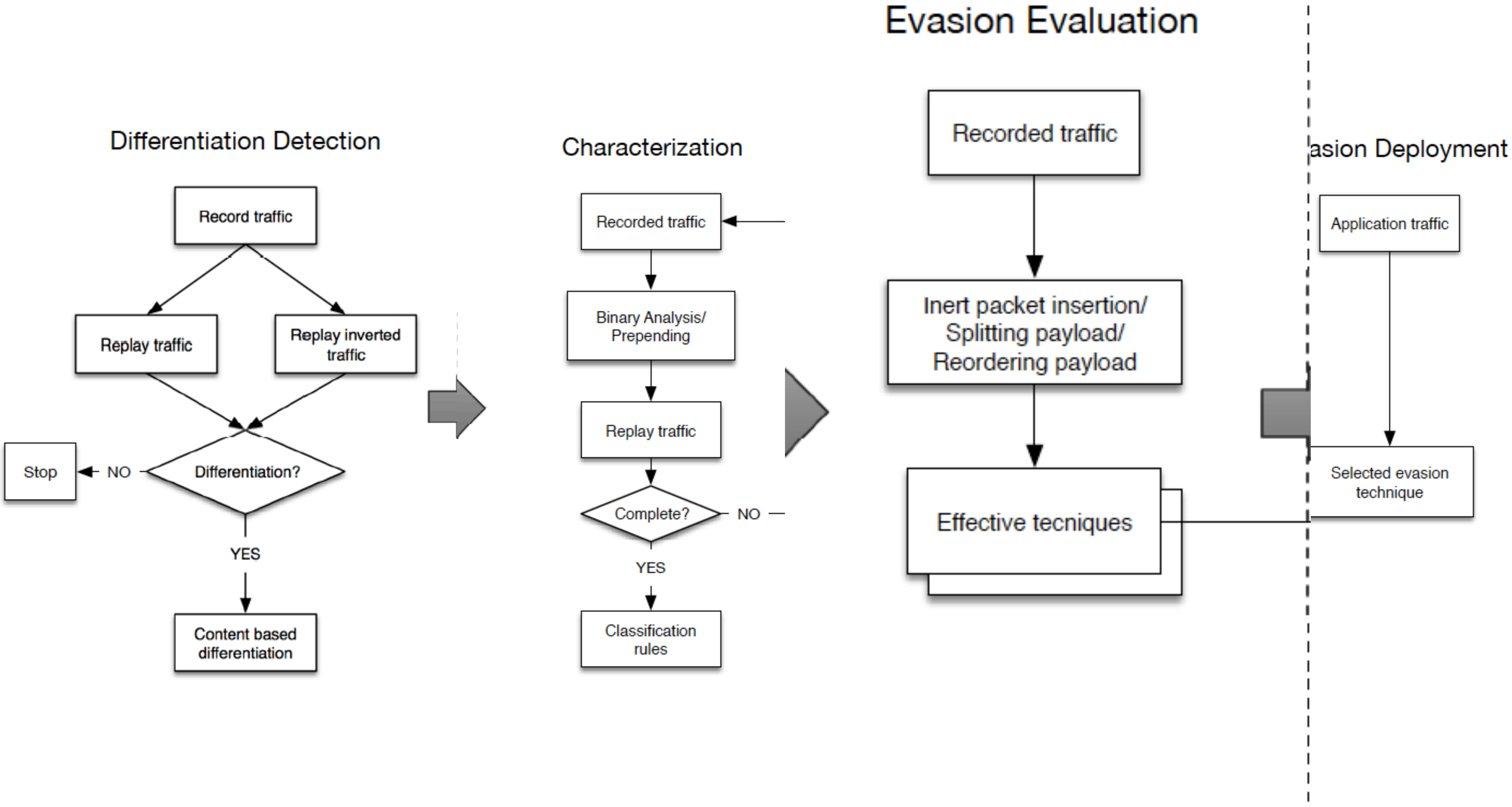
Overview of liberate



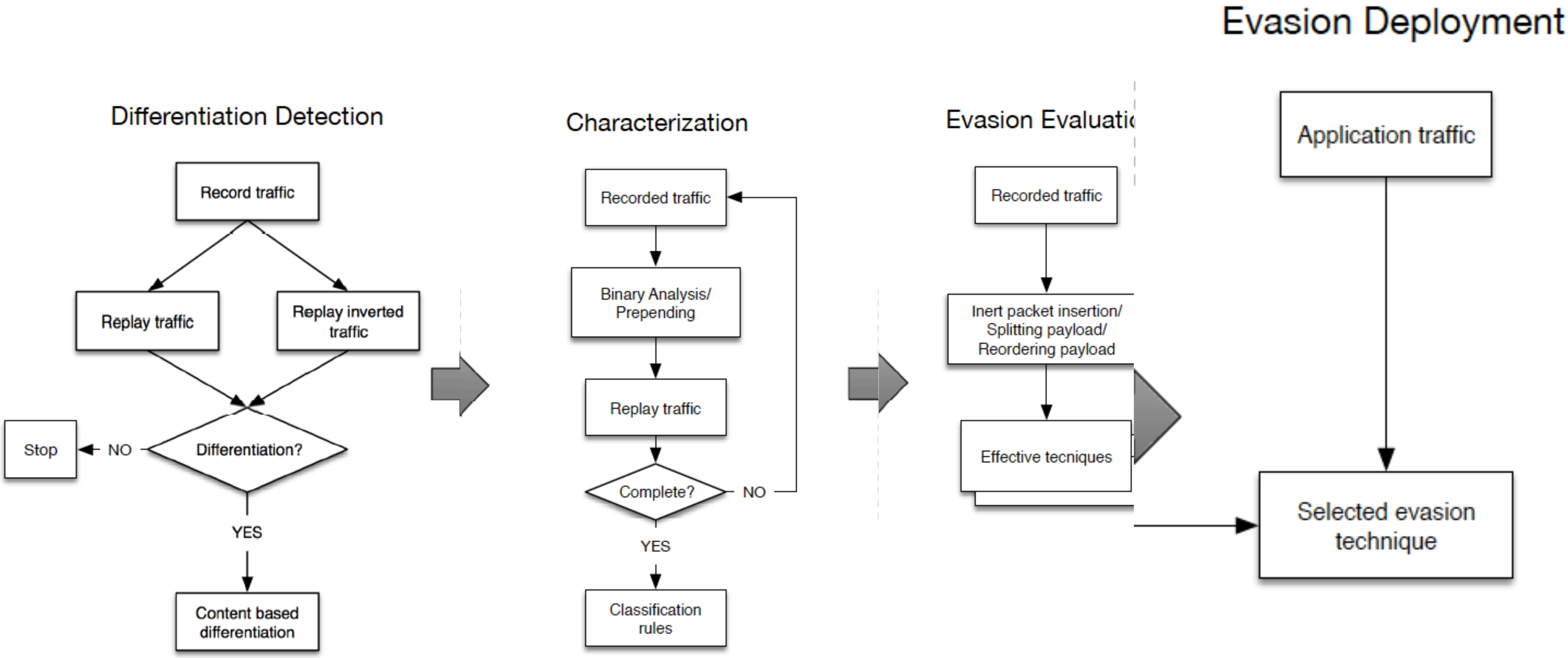
Overview of liberate



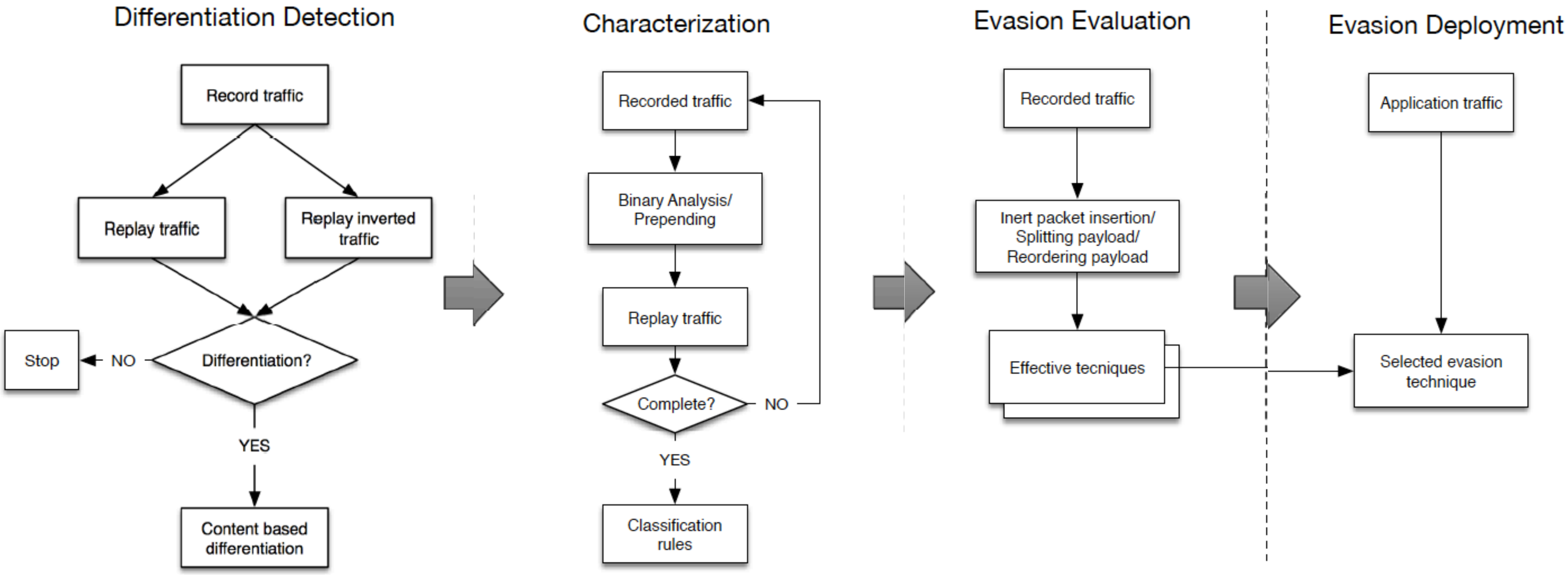
Overview of liberate



Overview of liberate



Overview of liberate



Outline

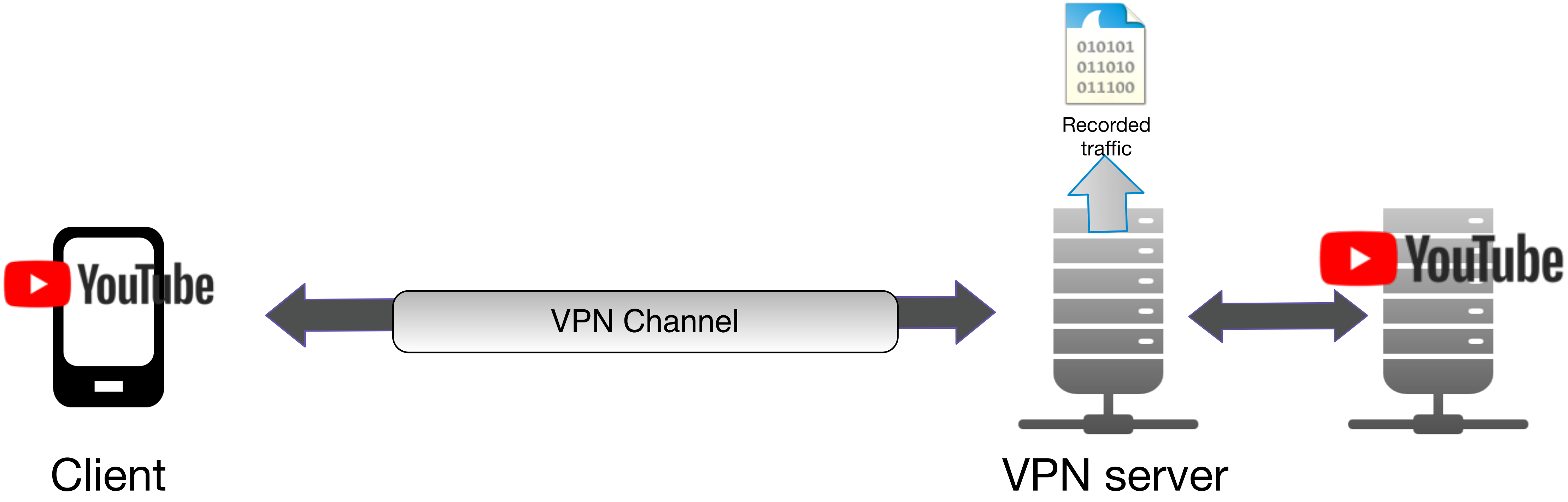
- Design and implementation
  - Traffic-classification rules detection
  - Evasion techniques
  - Implementation
- Evaluation
  - Effectiveness across multiple networks

Design

Traffic-classification rules detection

Design

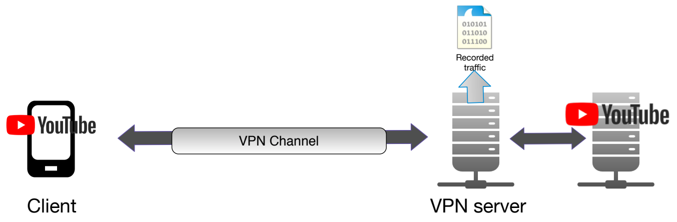
Traffic-classification rules detection

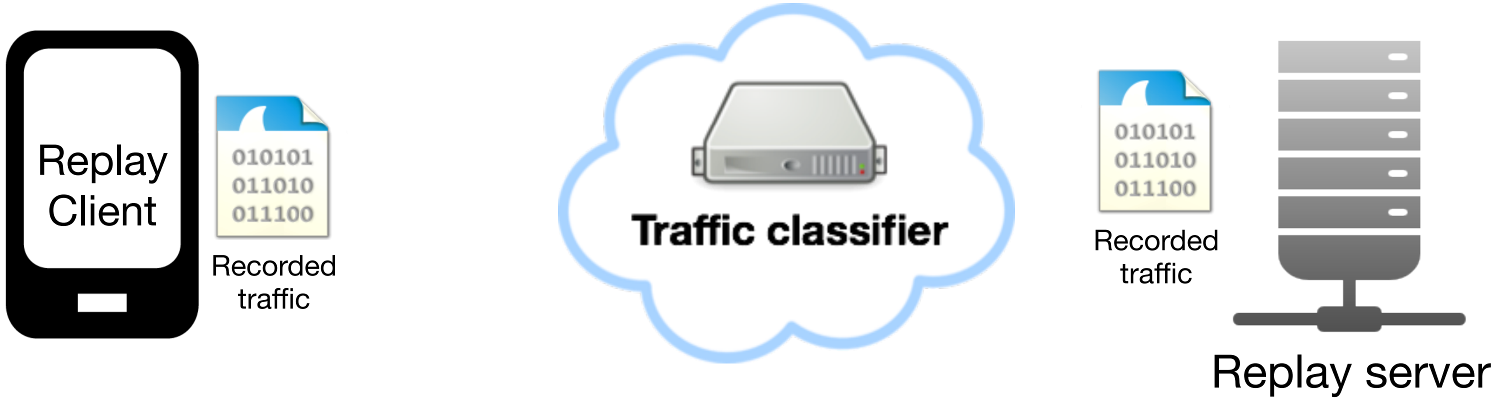


- How to detect differentiation?
  - Record and Replay [IMC 15]

Design

Traffic-classification rules detection

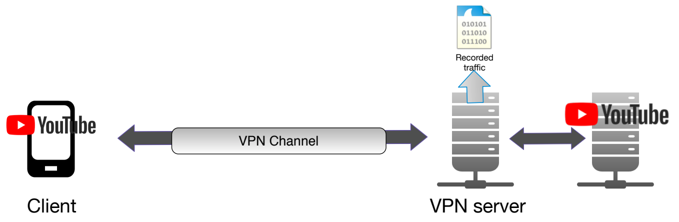


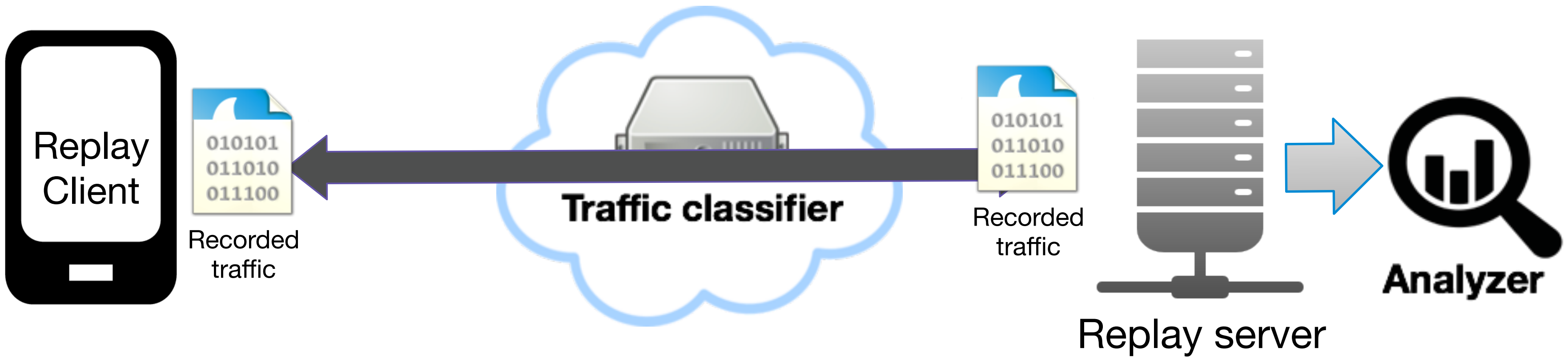


- How to detect differentiation?
    - Record and Replay [IMC 15]

Design

Traffic-classification rules detection

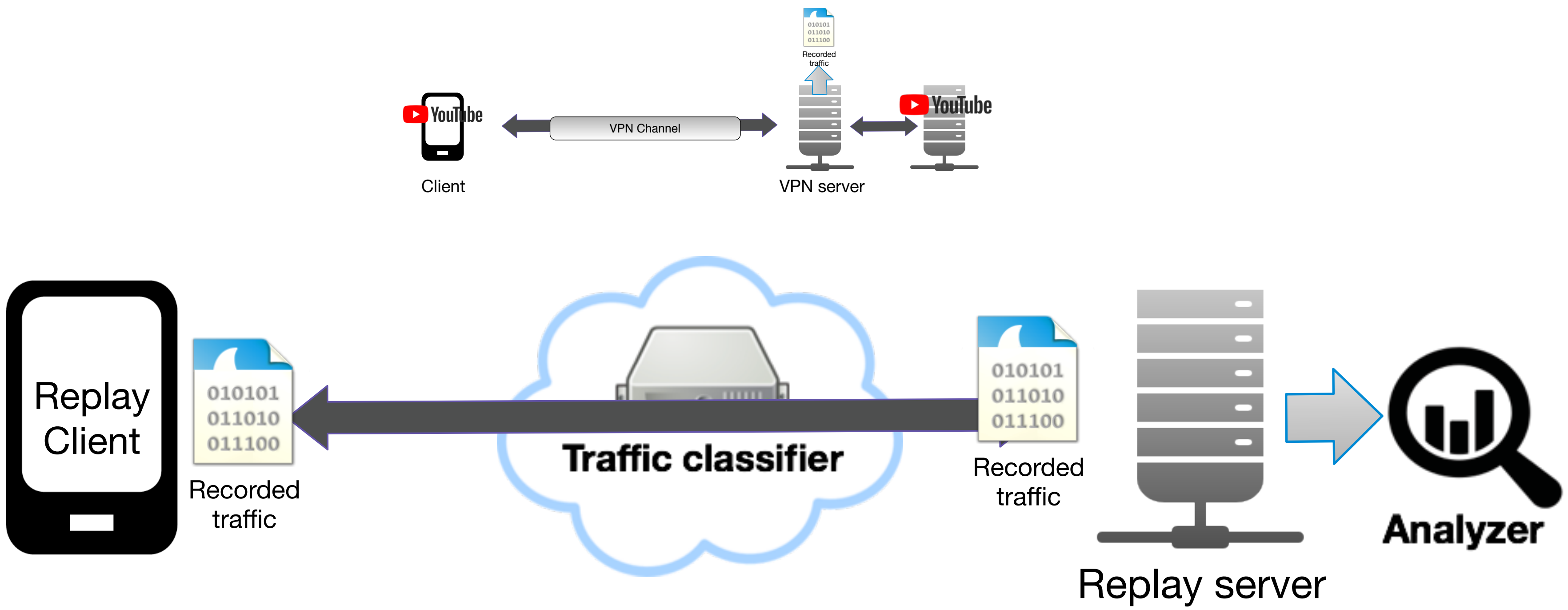




- How to detect differentiation?
    - Record and Replay [IMC 15]

Design

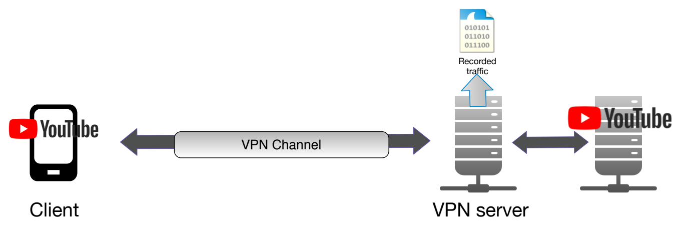
Traffic-classification rules detection

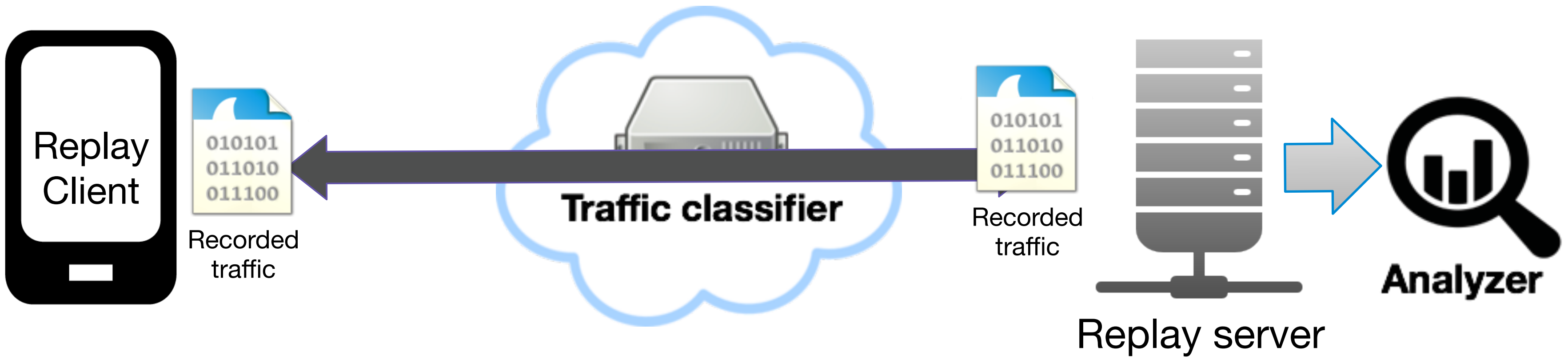


- How to detect differentiation?
  - Record and Replay [IMC 15]
- How to evade differentiation efficiently?

Design

Traffic-classification rules detection

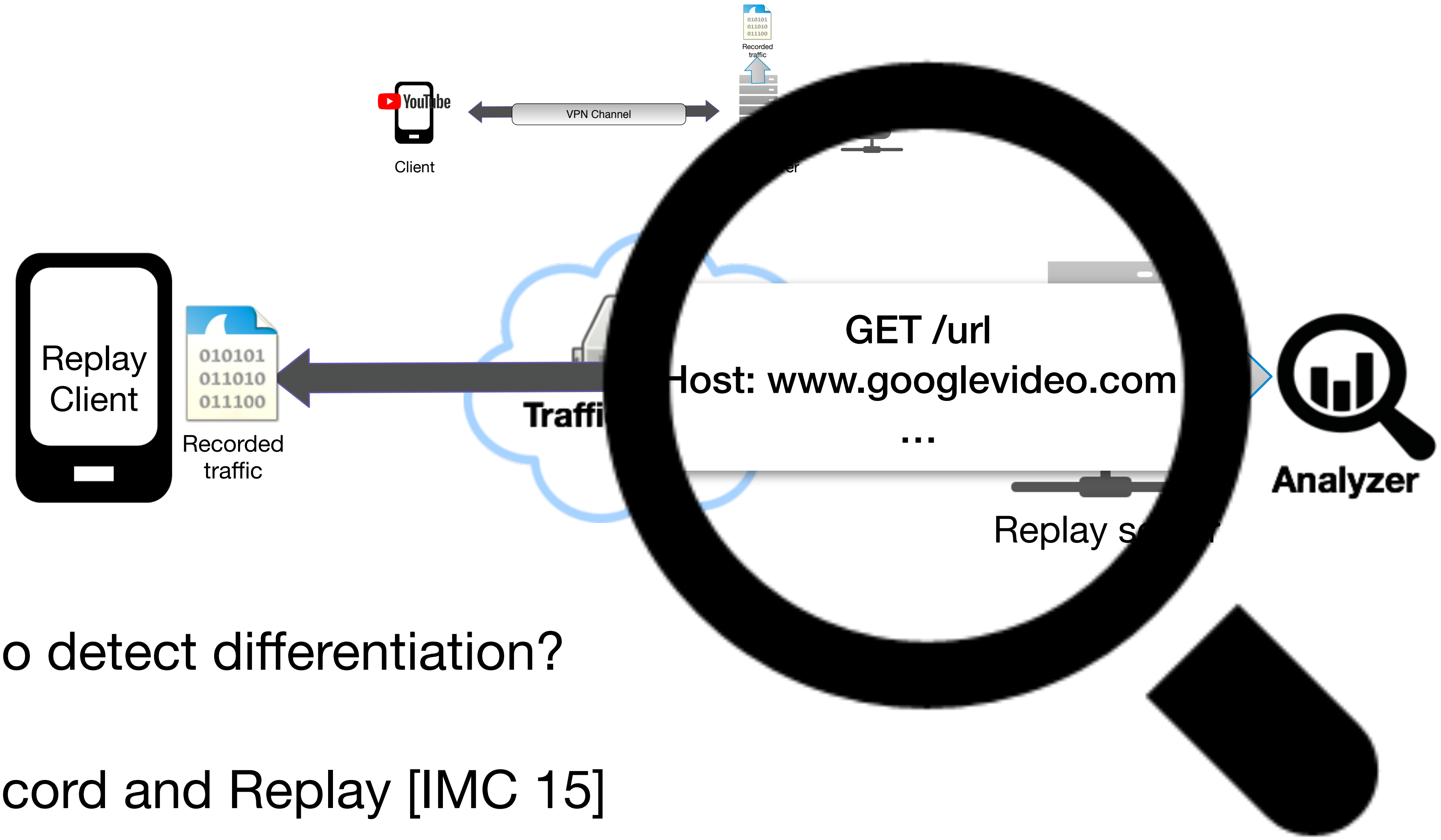




- How to detect differentiation?
  - Record and Replay [IMC 15]
- How to evade differentiation efficiently?
  - Understand classification rules [IMC 16]

Design

Traffic-classification rules detection

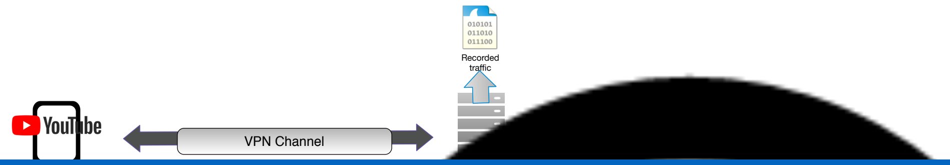


GET /url
Host: www.googlevideo.com
…

- How to detect differentiation?
  - Record and Replay [IMC 15]
- How to evade differentiation efficiently?
  - Understand classification rules [IMC 16]

Design

Traffic-classification rules detection



Header	Example matching content
URI	site.js{…}-**nbcsports**-com
Host	Host: www.**spotify.**com
User-Agent	User-Agent: **Pandora** 5.0{…}
Content-Type	Content-Type: **video**
SNI	**googlevideo.com**

- Understand classification rules [IMC 16]

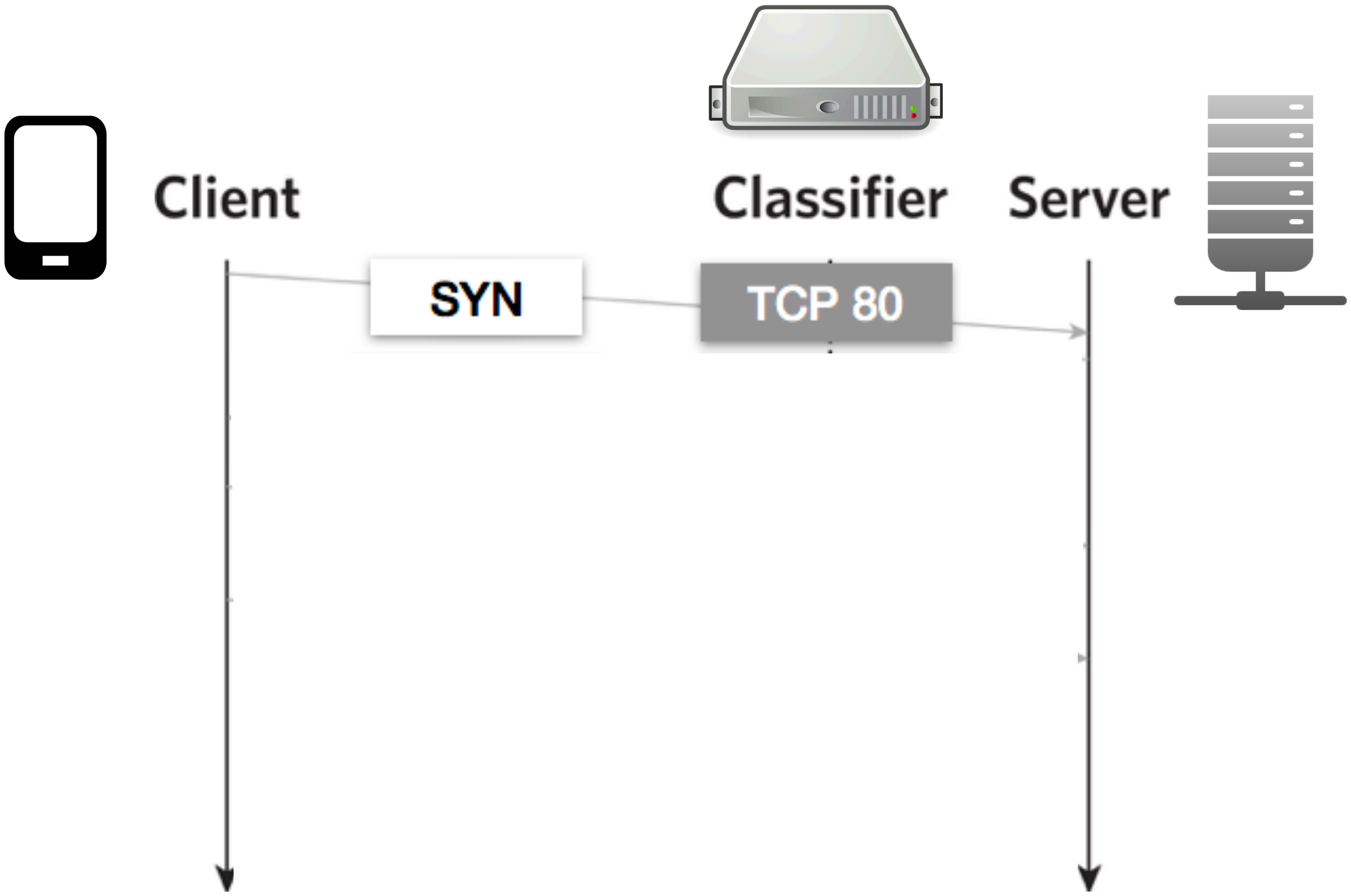
Outline

- Design and implementation
  - Traffic-classification rules detection
  - Evasion techniques
  - Implementation
- Evaluation
  - Effectiveness across multiple networks

Design

Example classification

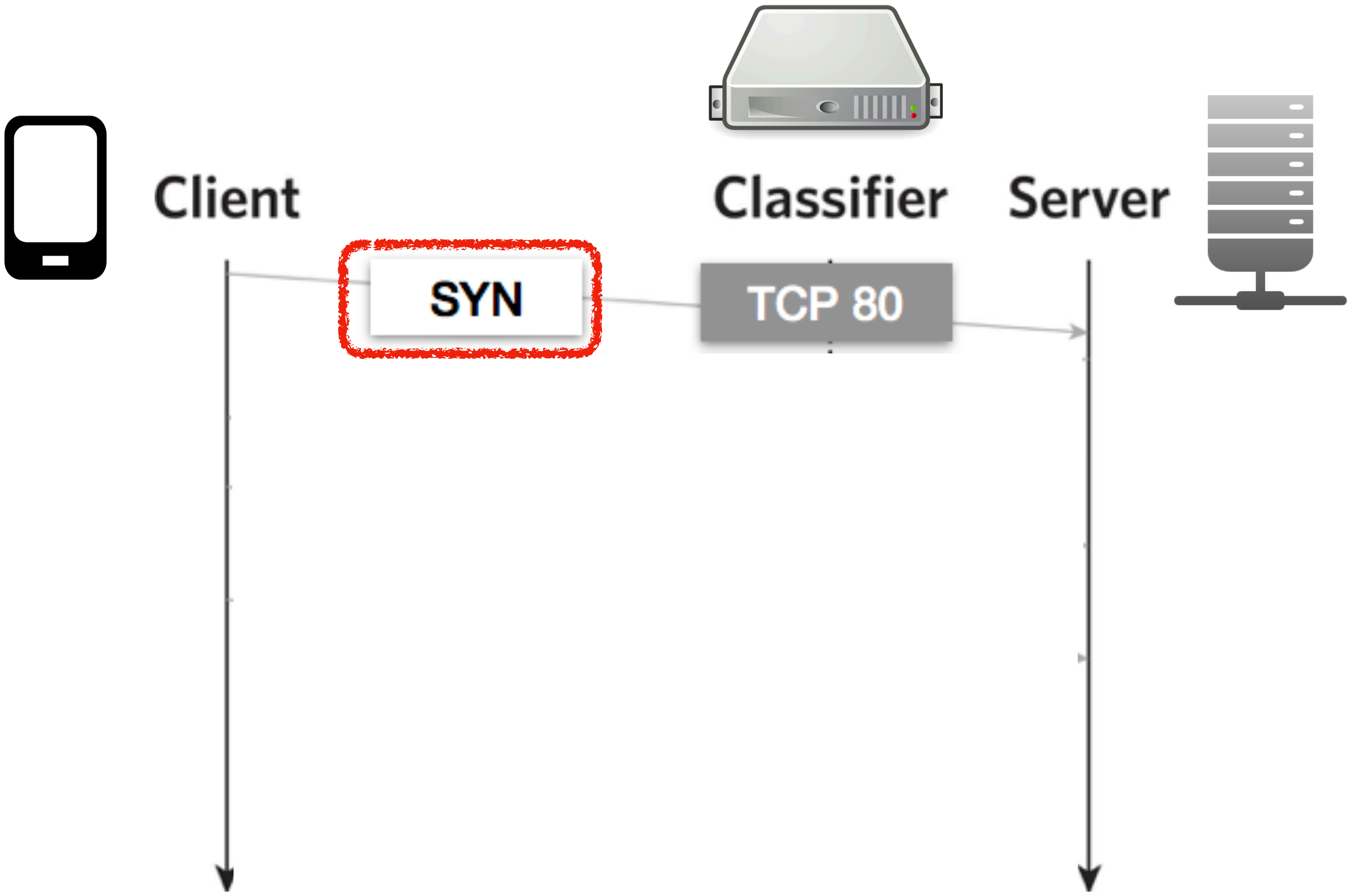
How does classifier classify application B?



Design

Example classification

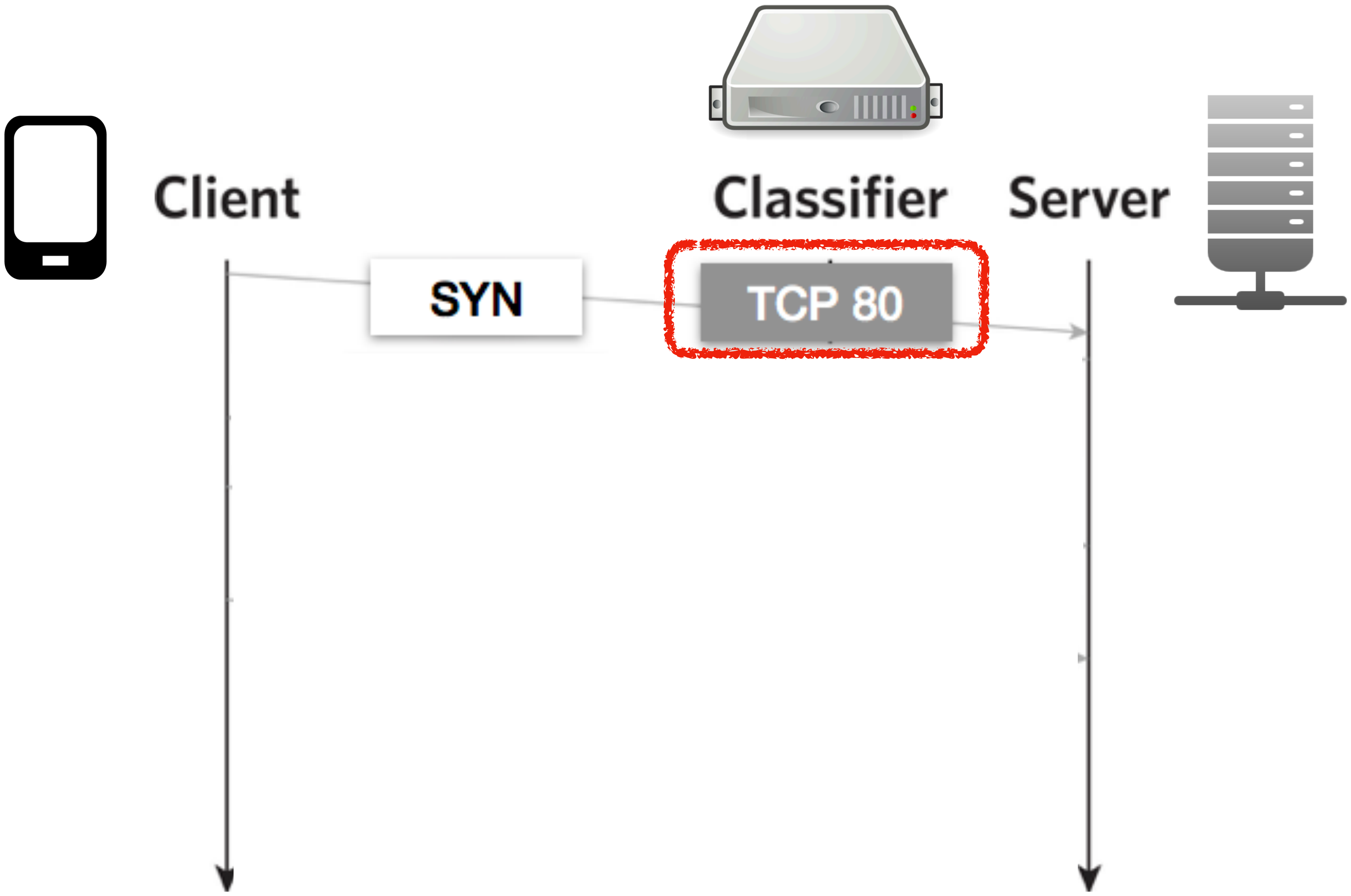
How does classifier classify application B?



Design

Example classification

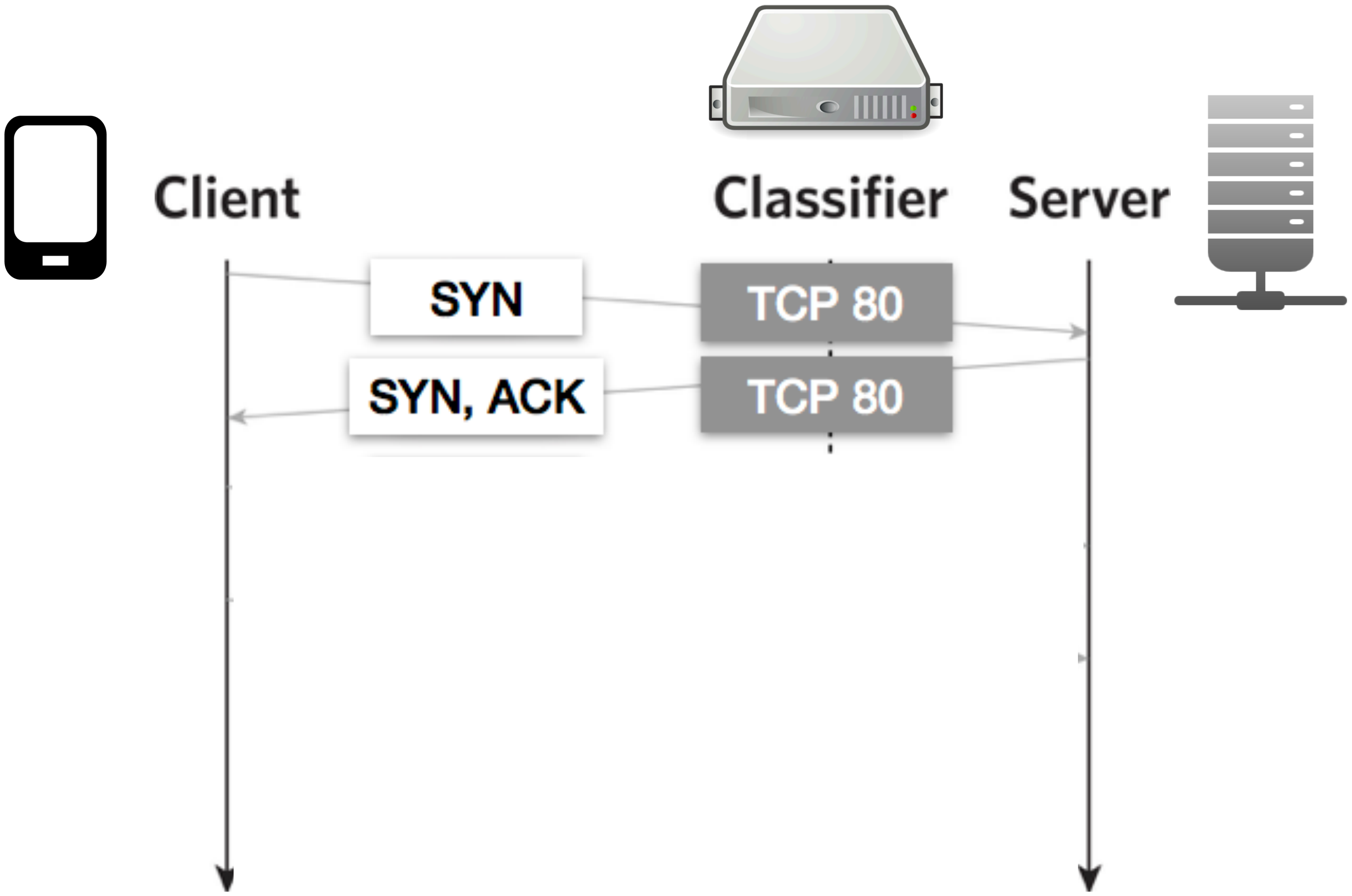
How does classifier classify application B?



Design

Example classification

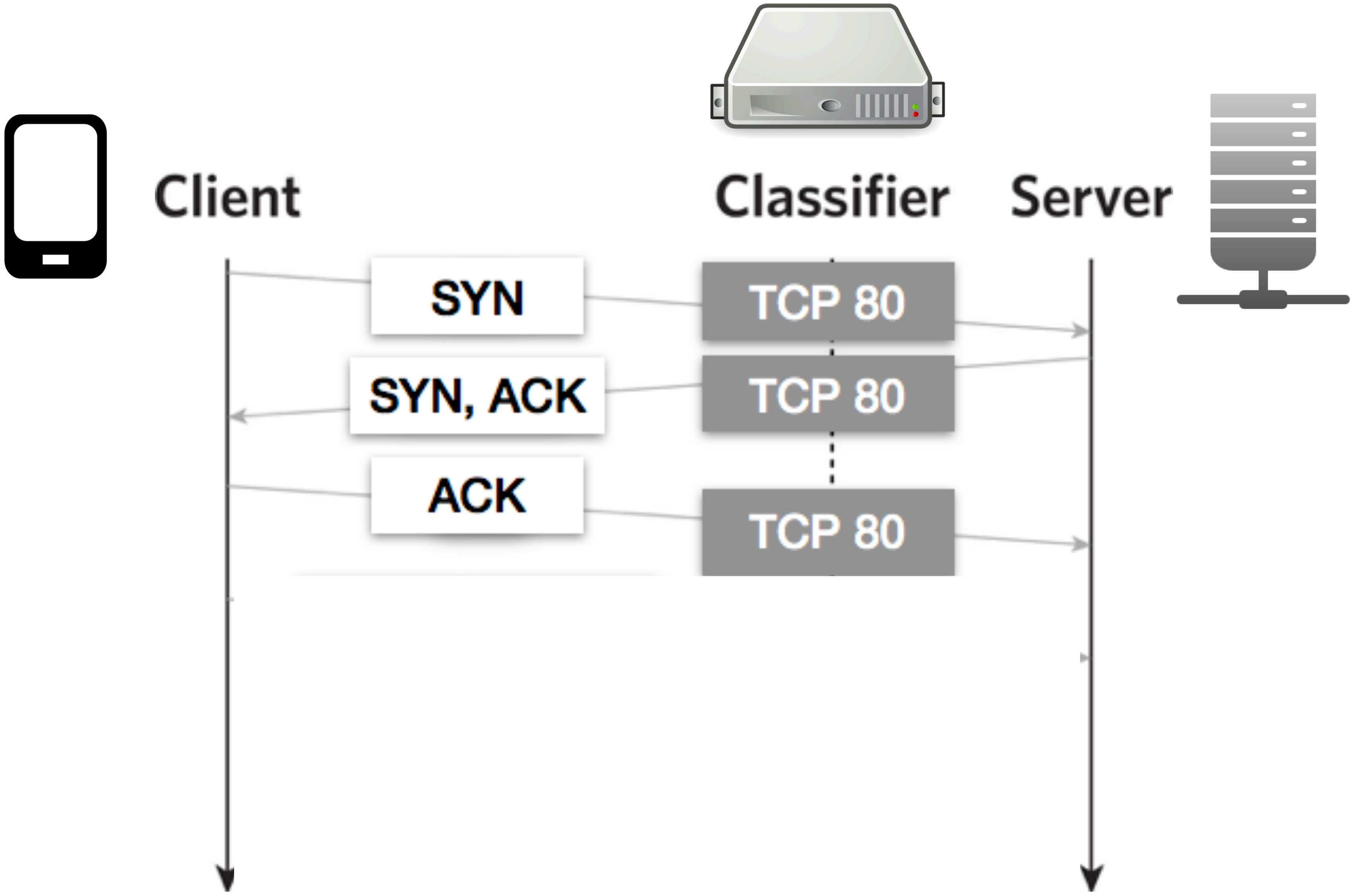
How does classifier classify application B?



Design

Example classification

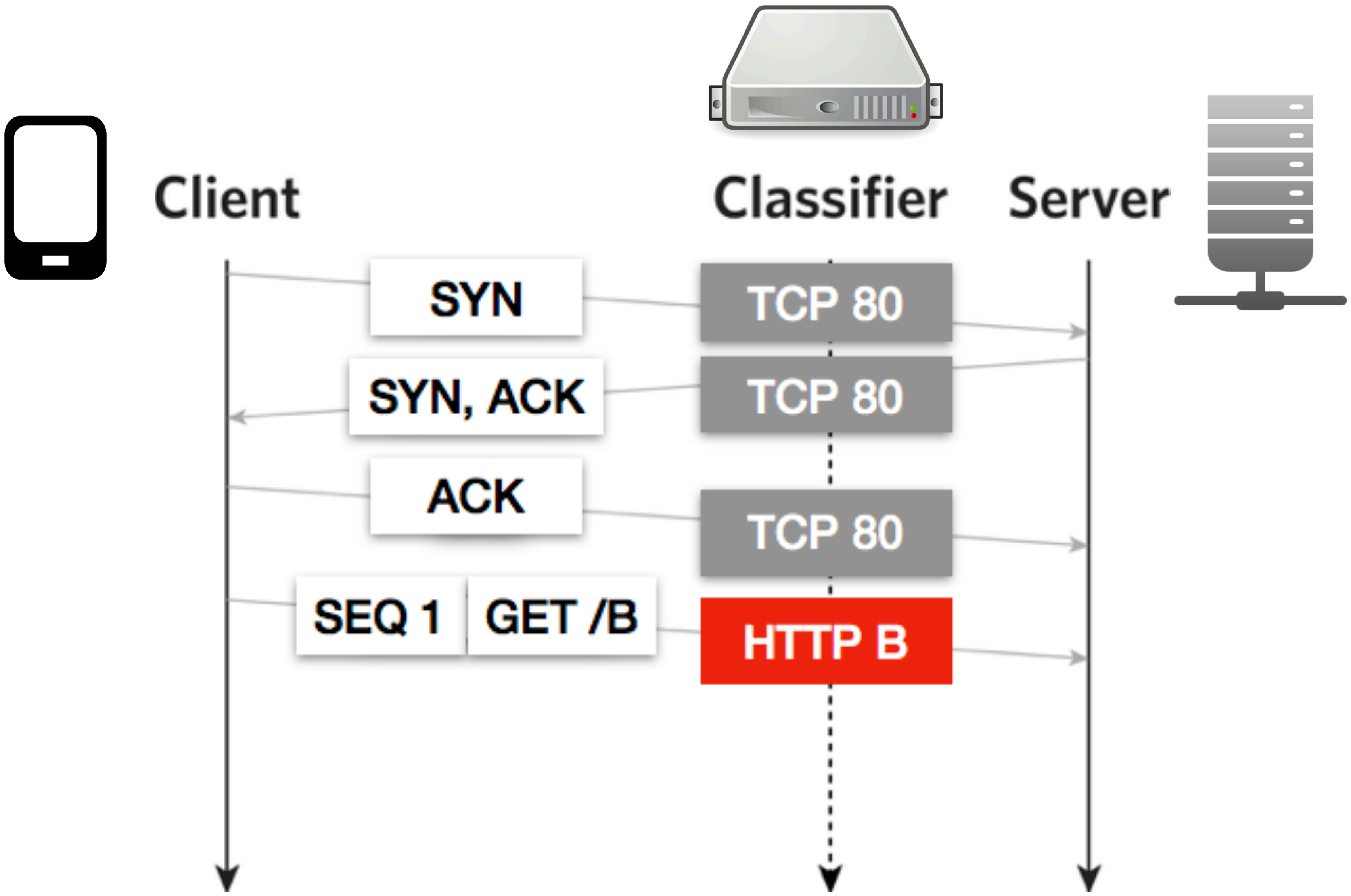
How does classifier classify application B?



Design

Example classification

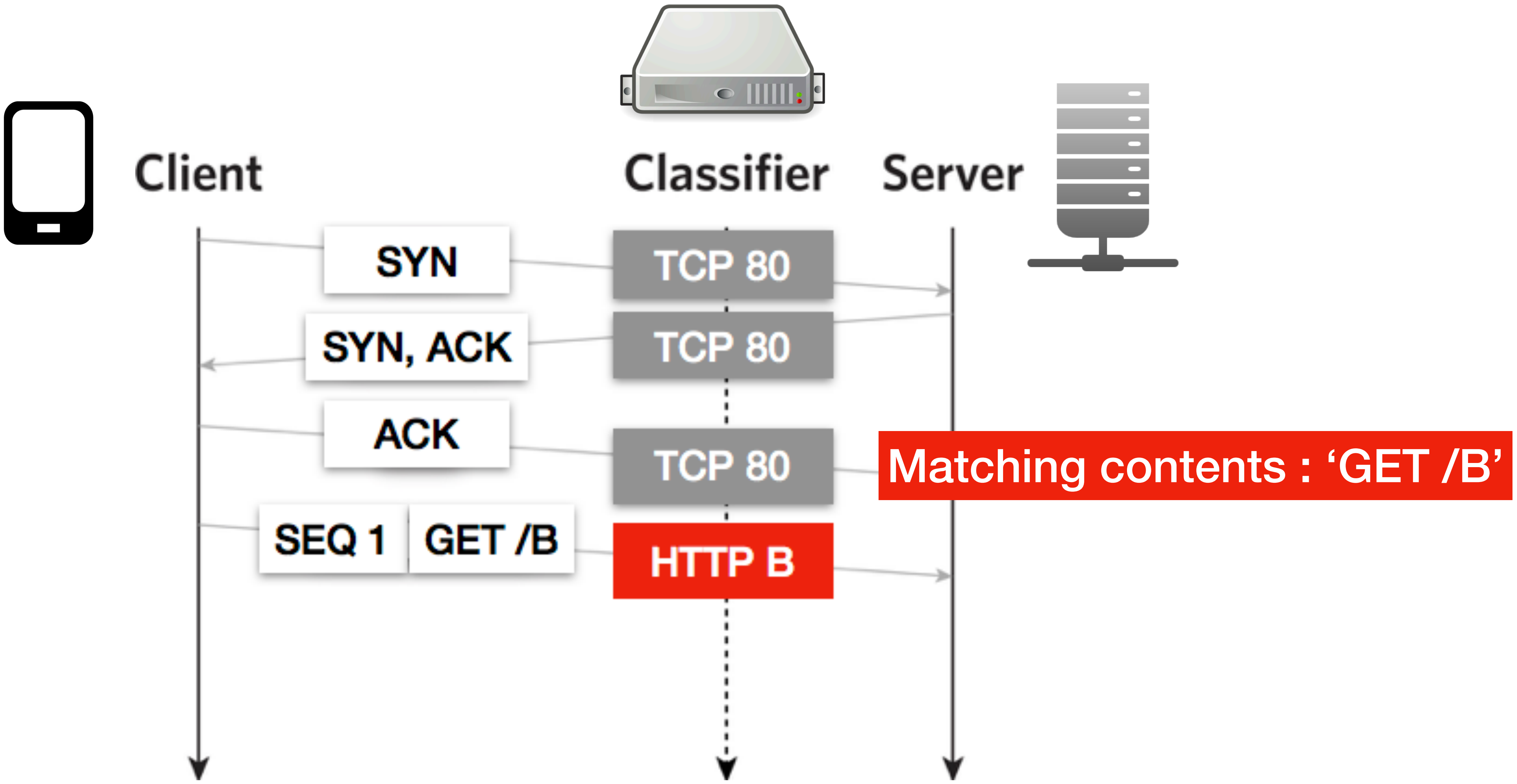
How does classifier classify application B?



Design

Example classification

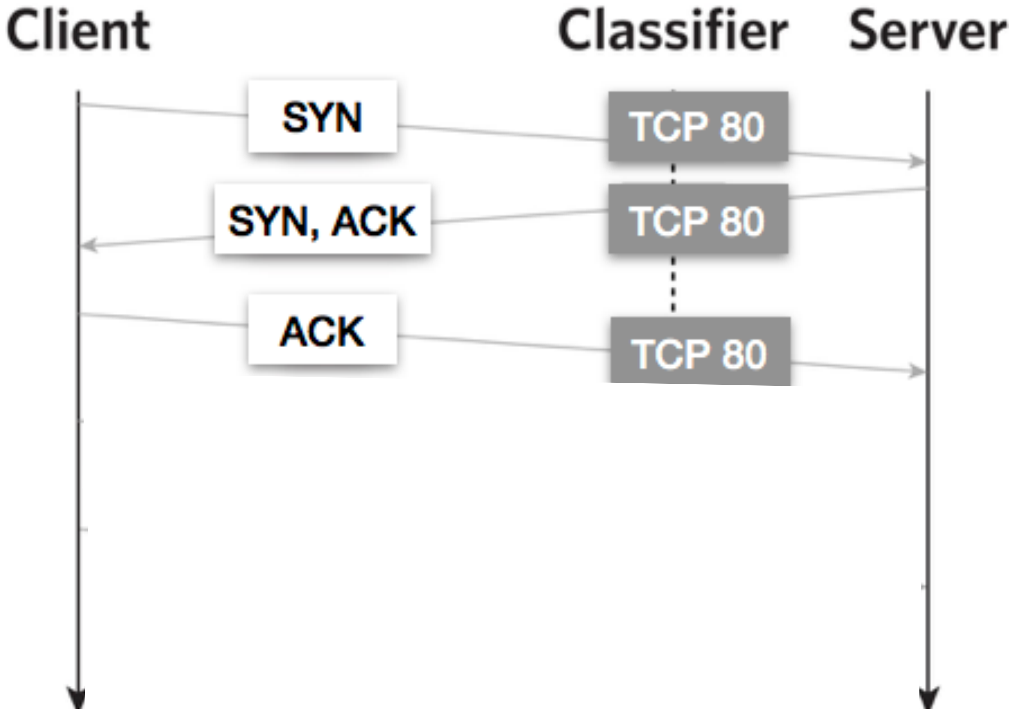
How does classifier classify application B?



Matching contents : ‘GET /B’

Design

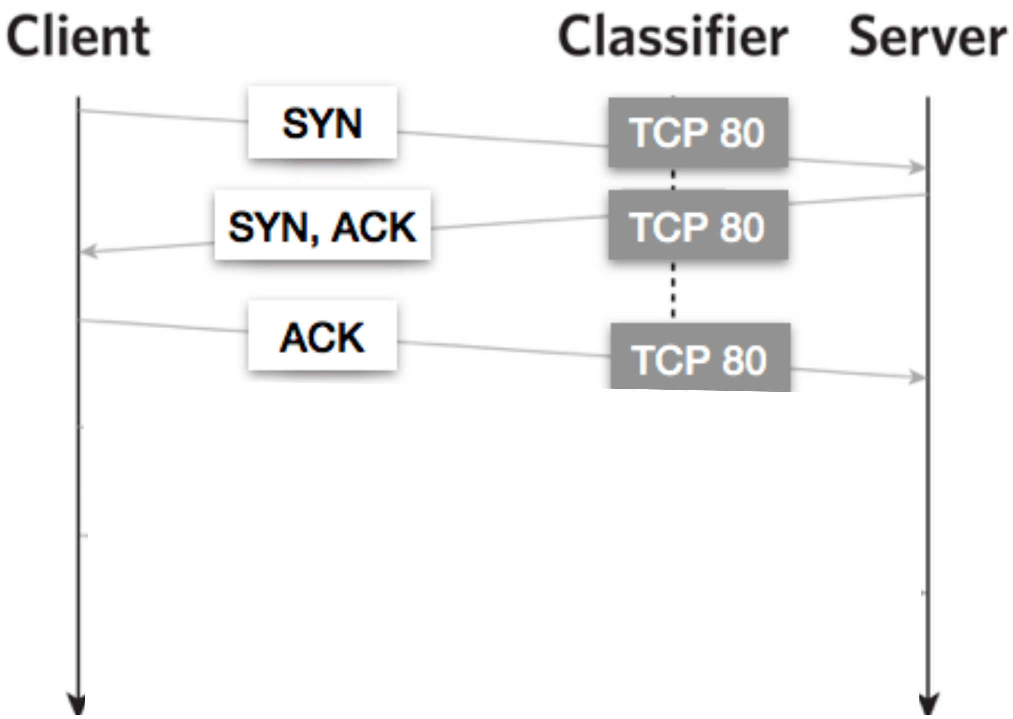
Evasion techniques



Design

Evasion techniques

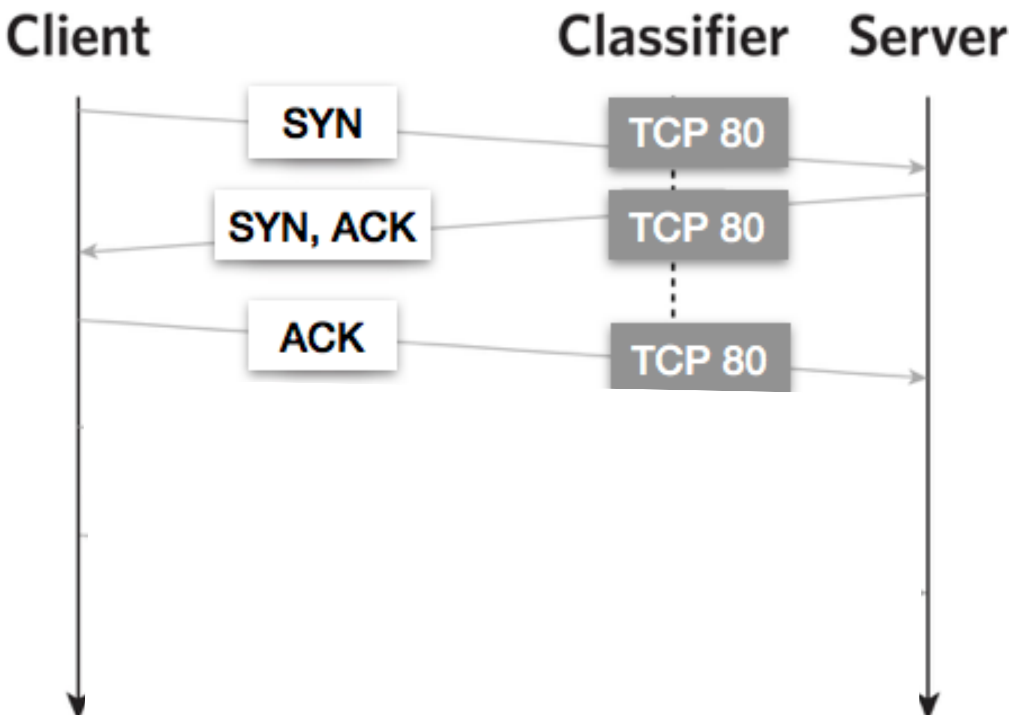
- Observation:
  - ‘Match and forget’ behavior



Design

Evasion techniques

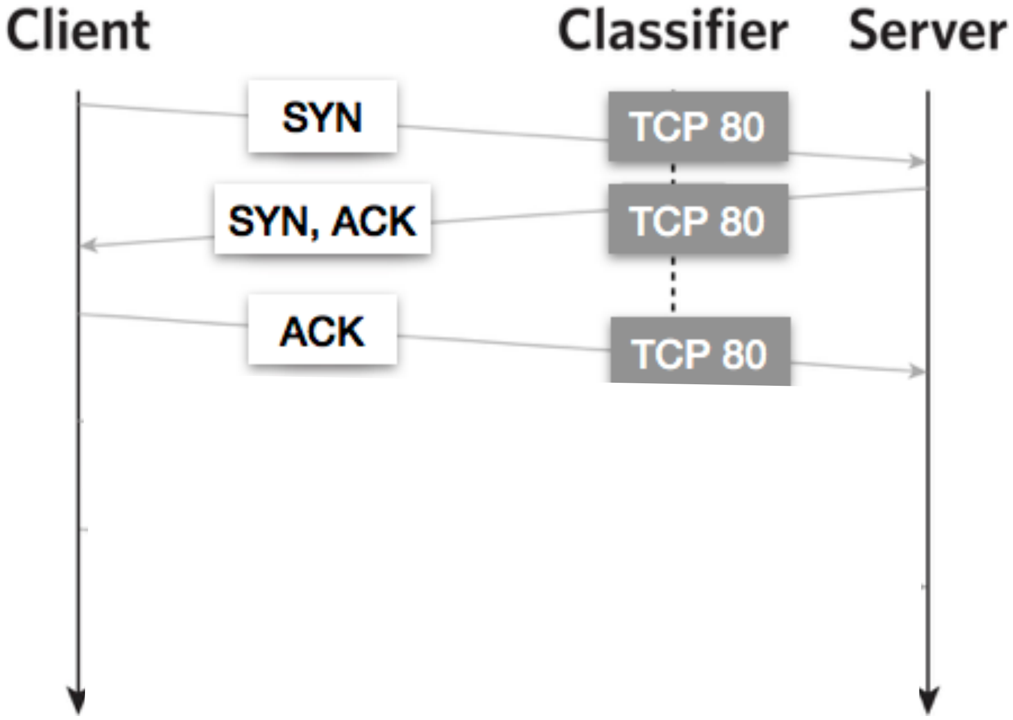
- Observation:
  - ‘Match and forget’ behavior
  - Incomplete views of the connection



Design

Evasion techniques

- Observation:
  - ‘Match and forget’ behavior
  - Incomplete views of the connection
- **Inert packet insertion*** : Traffic processed only by a classifier but not endpoint



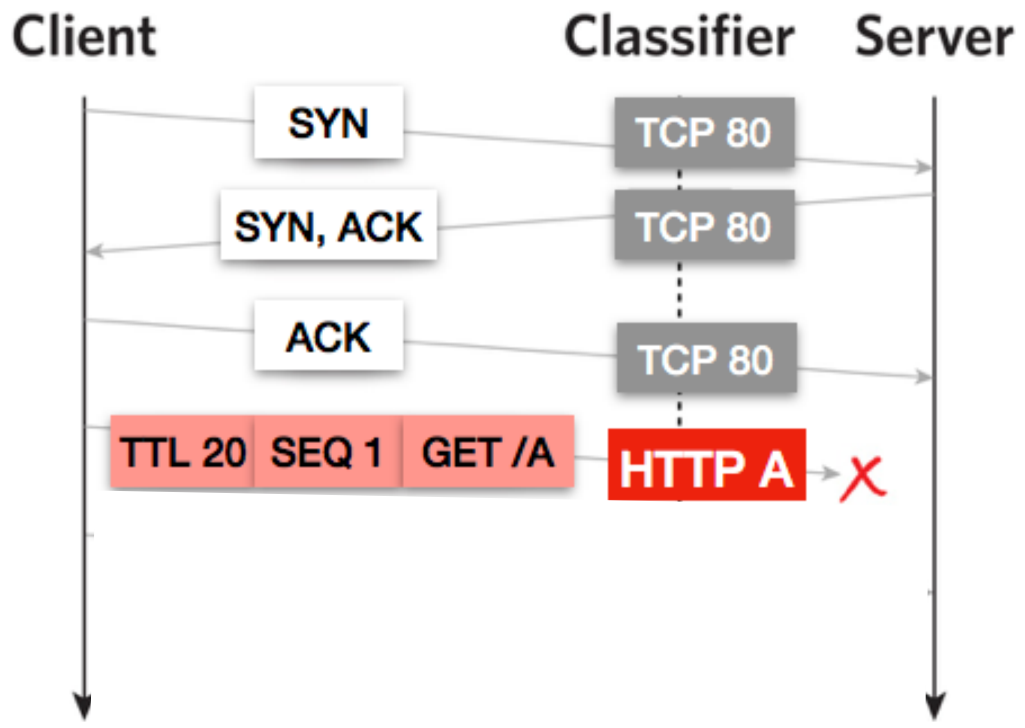
Using a small TTL value

* Christian Kreibich et al. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics.

Design

Evasion techniques

- Observation:
    - ‘Match and forget’ behavior
    - Incomplete views of the connection
- **Inert packet insertion*** : Traffic processed only by a classifier but not endpoint



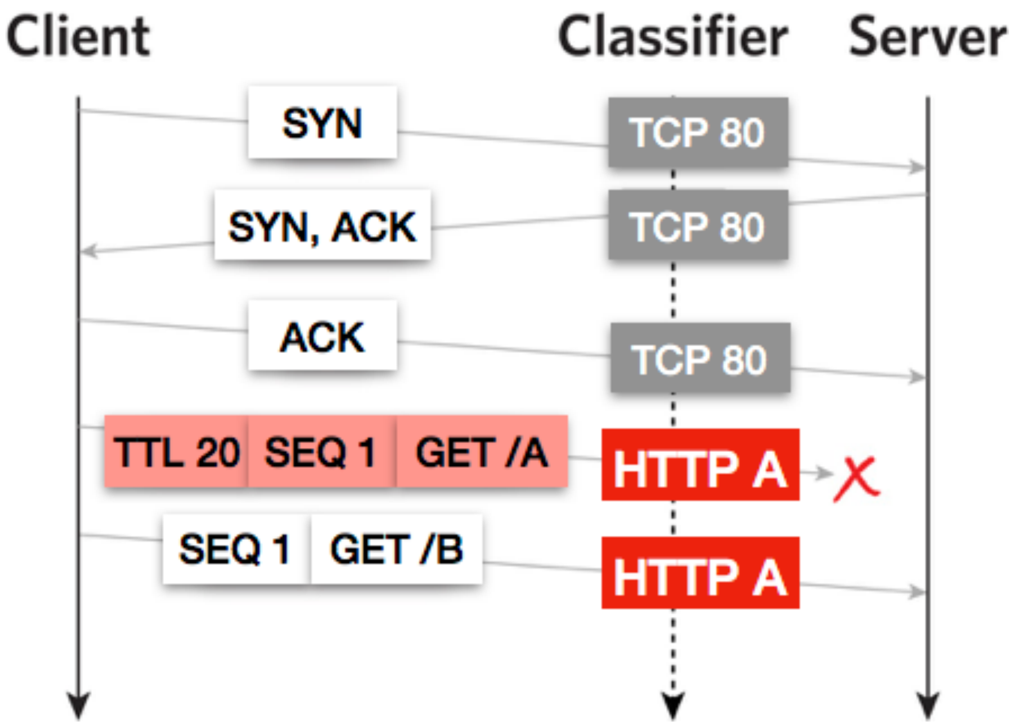
Using a small TTL value

* Christian Kreibich et al. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics.

Design

Evasion techniques

- Observation:
  - ‘Match and forget’ behavior
  - Incomplete views of the connection
- **Inert packet insertion*** : Traffic processed only by a classifier but not endpoint



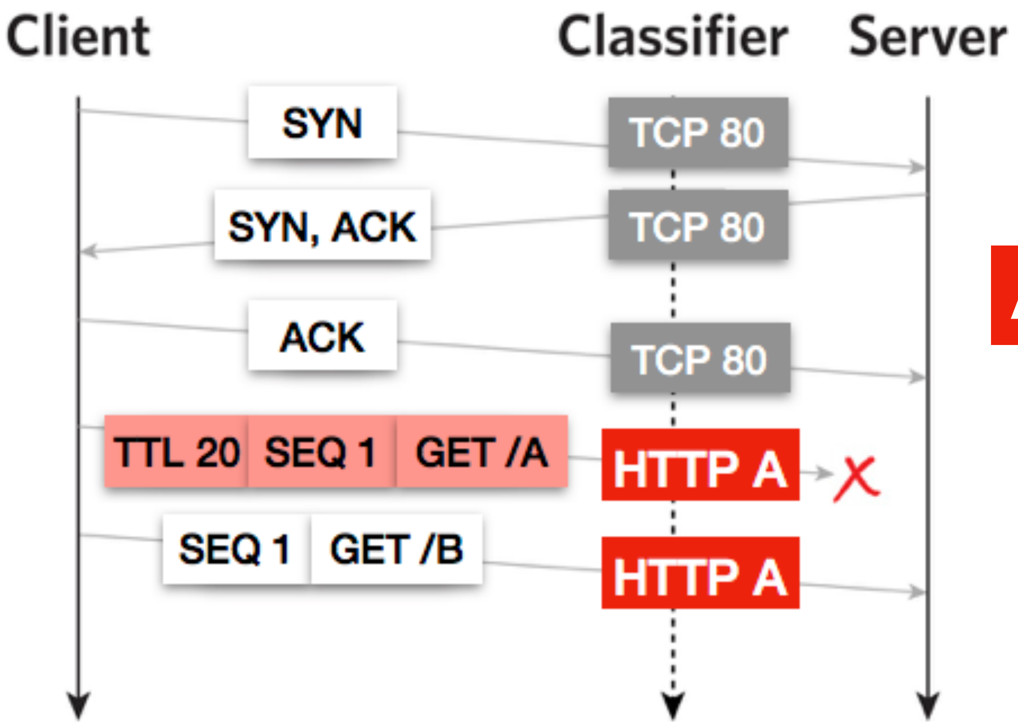
Using a small TTL value

* Christian Kreibich et al. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics.

Design

Evasion techniques

- Observation:
  - ‘Match and forget’ behavior
  - Incomplete views of the connection
- **Inert packet insertion*** : Traffic processed only by a classifier but not endpoint



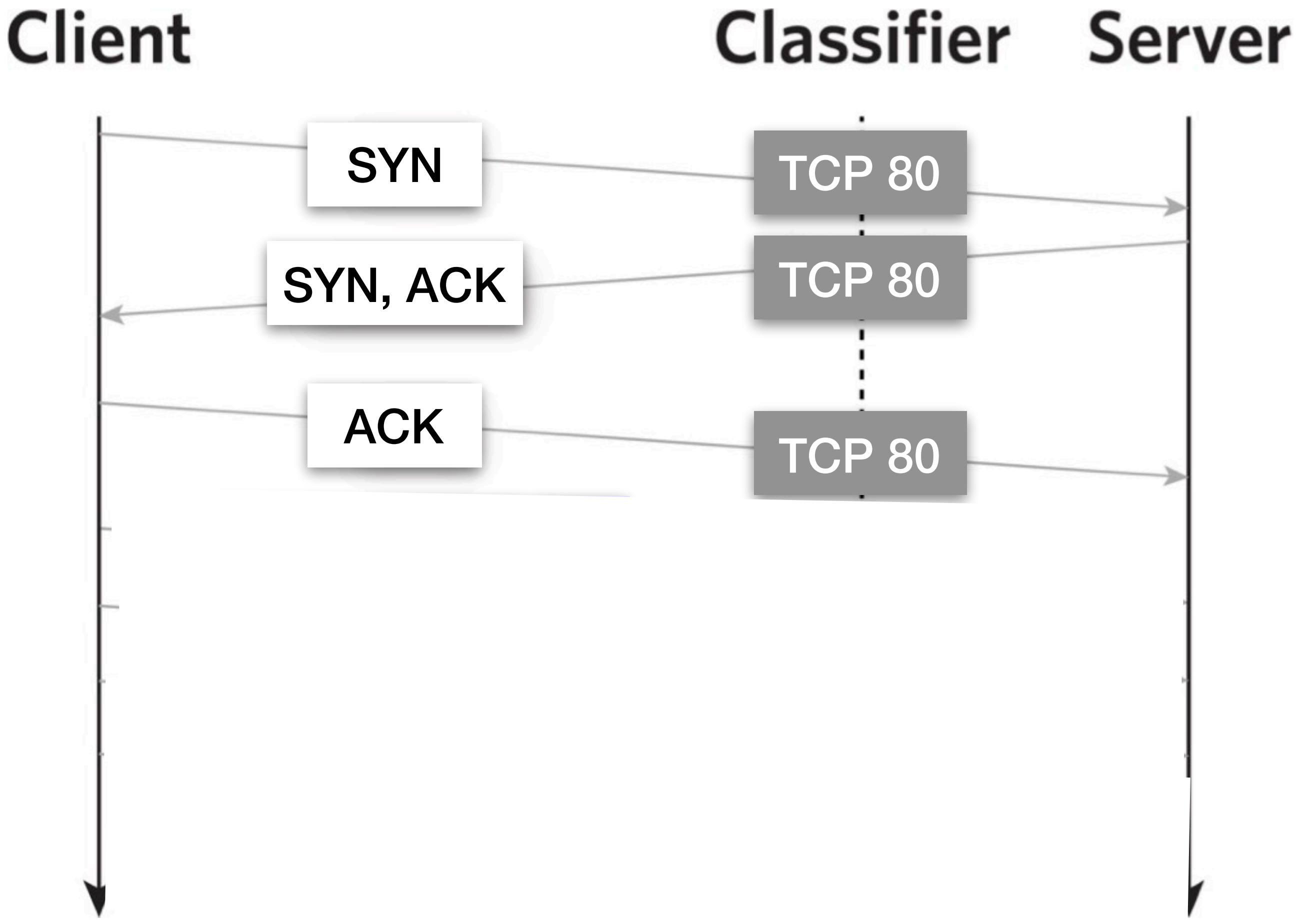
App B is classified as App A

Using a small TTL value

* Christian Kreibich et al. 2001. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics.

Design

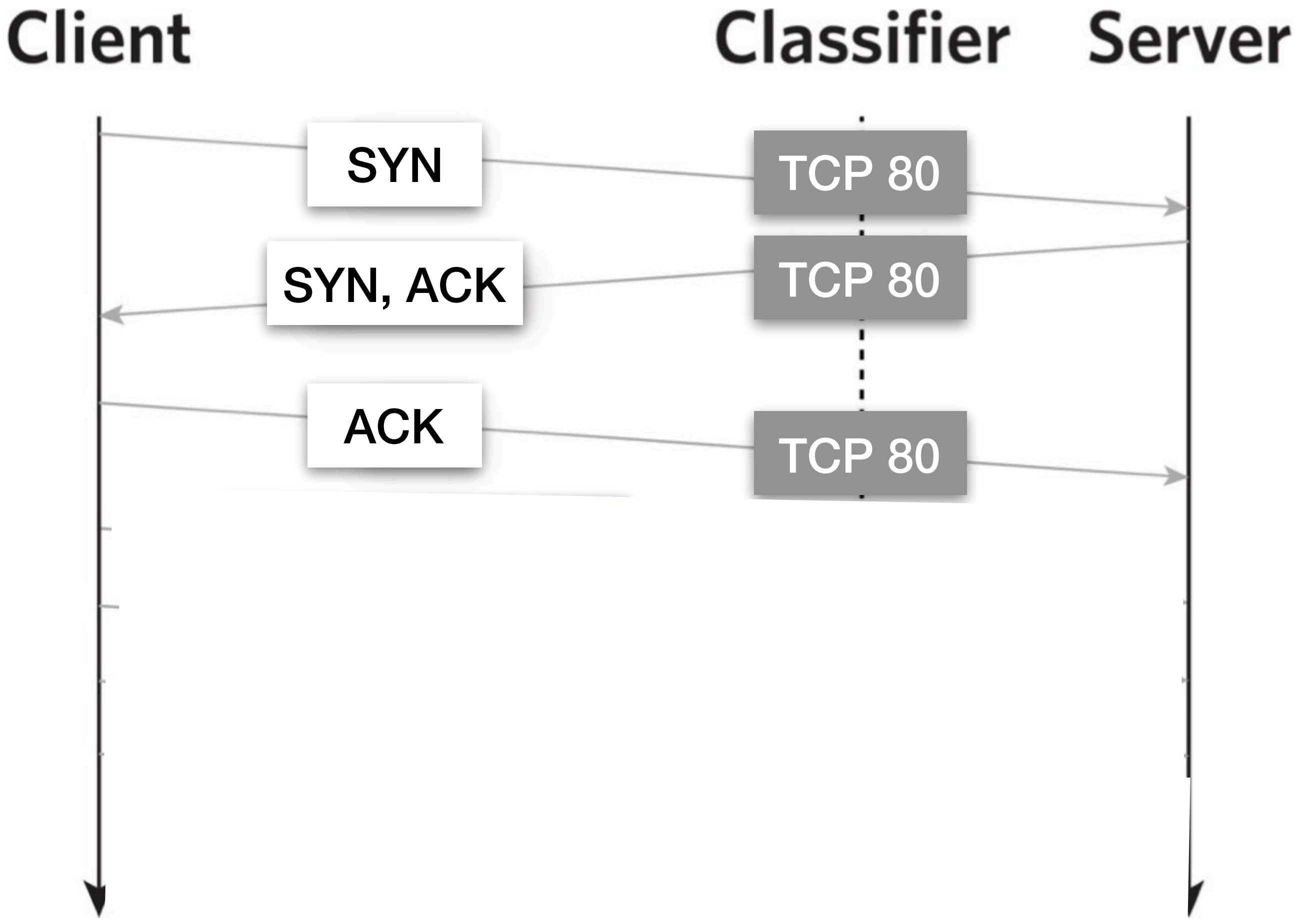
Evasion techniques



Design

Evasion techniques

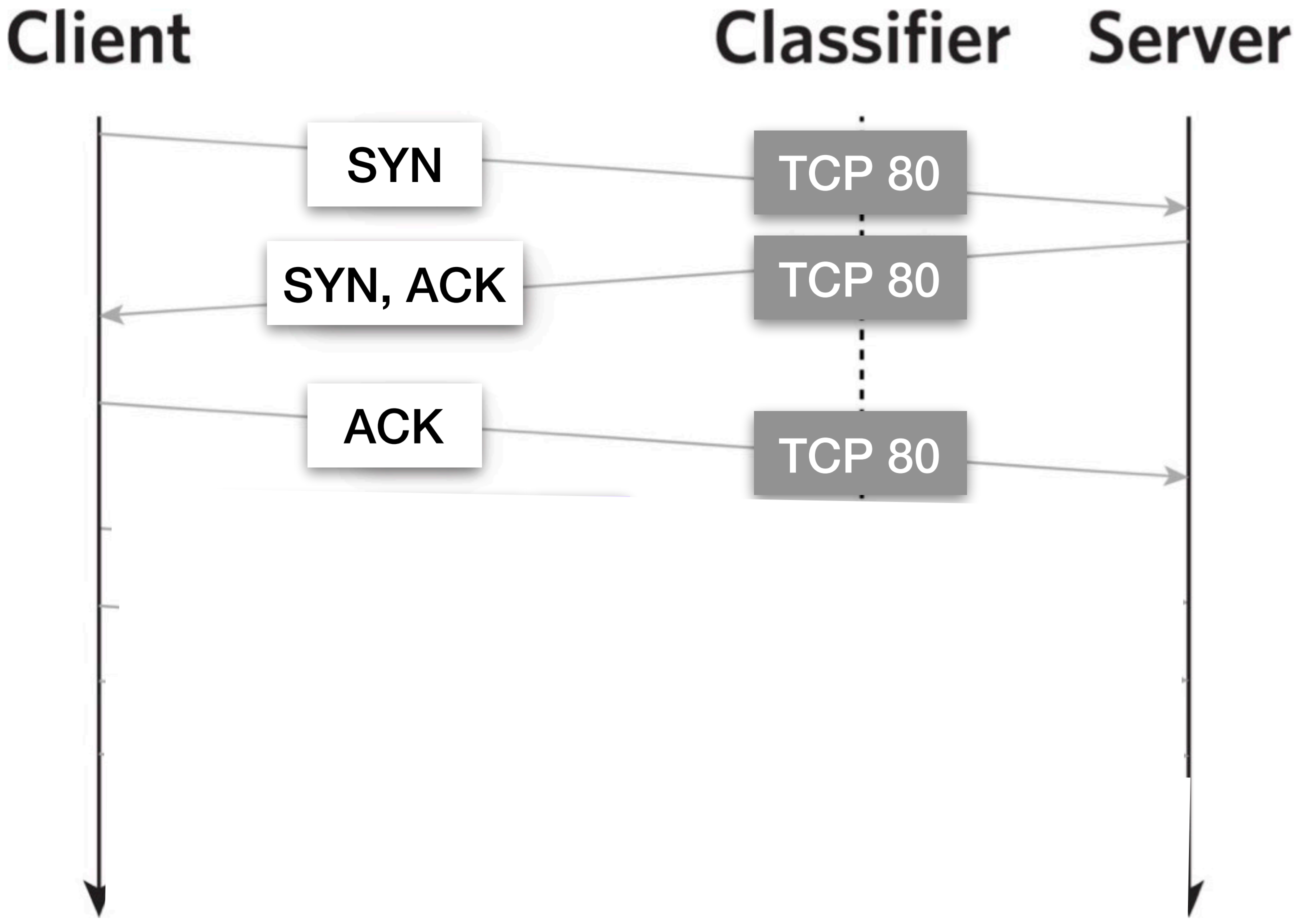
- Observation:
  - Each packet is searched independently for matching contents



Design

Evasion techniques

- Observation:
  - Each packet is searched independently for matching contents
- **Splitting/Reordering**: splitting the matching contents across multiple packets

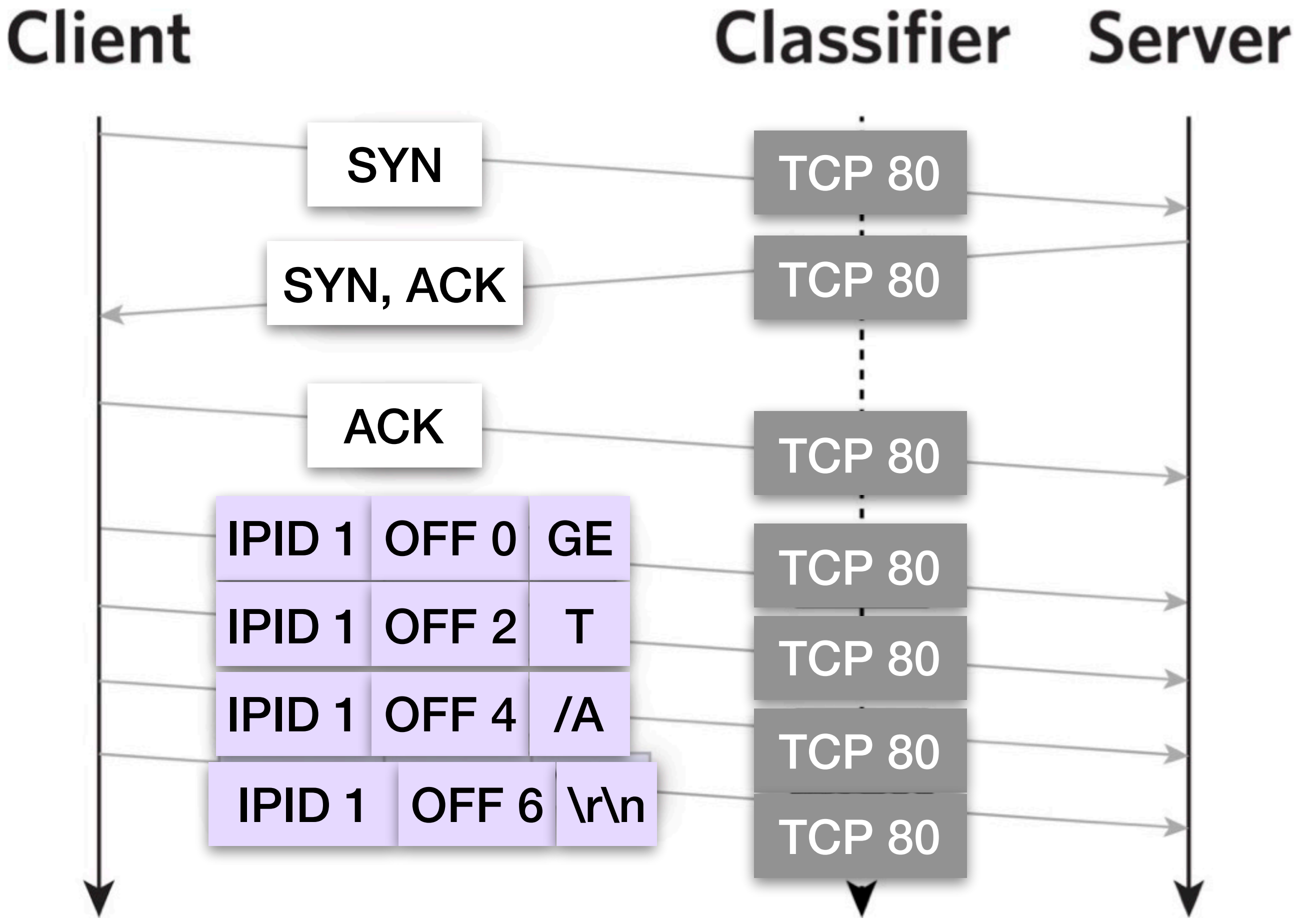


Fragmenting the IP packet

Design

Evasion techniques

- Observation:
    - Each packet is searched independently for matching contents
- **Splitting/Reordering**: splitting the matching contents across multiple packets

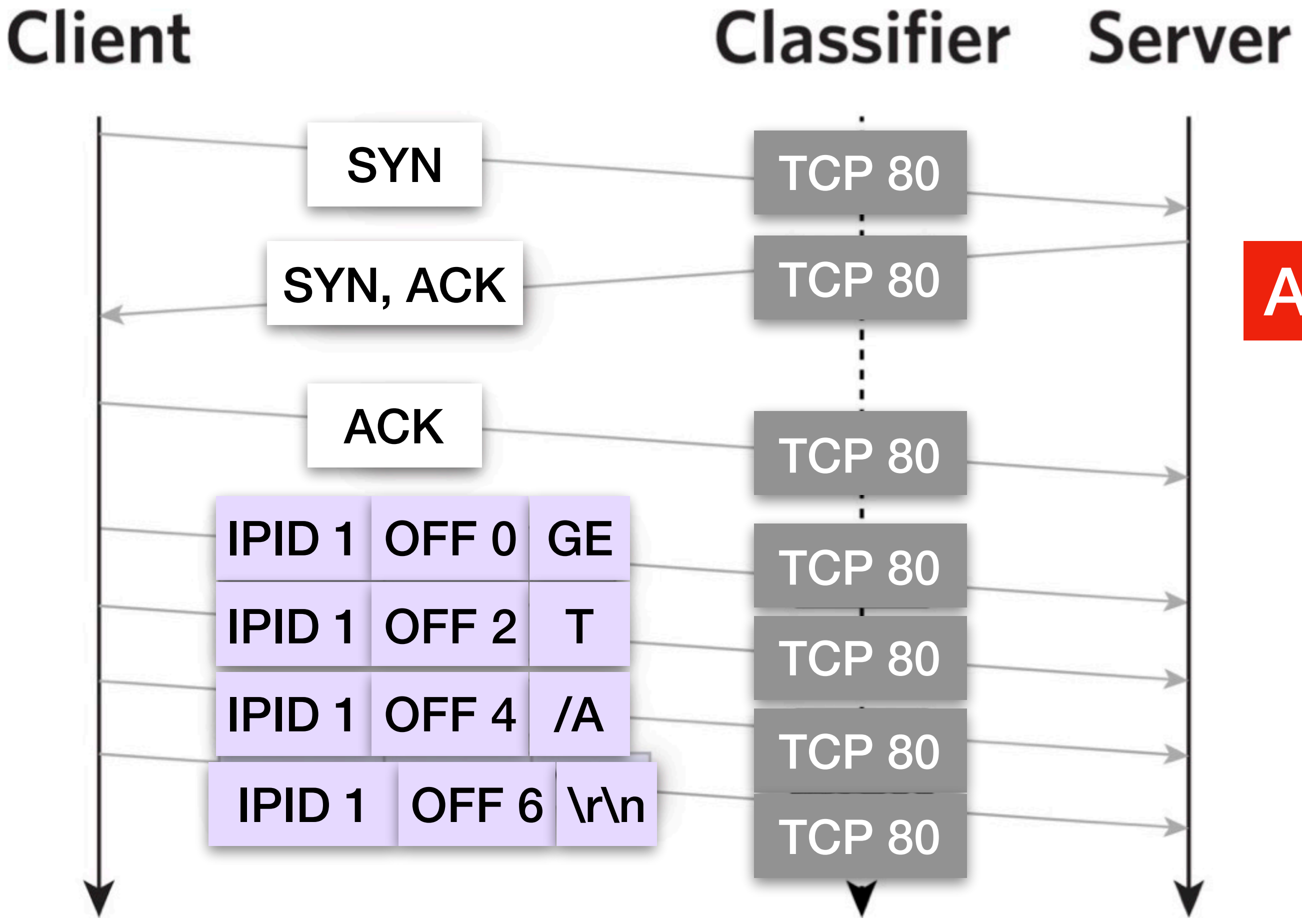


Fragmenting the IP packet

Design

Evasion techniques

- Observation:
  - Each packet is searched independently for matching contents
- **Splitting/Reordering**: splitting the matching contents across multiple packets

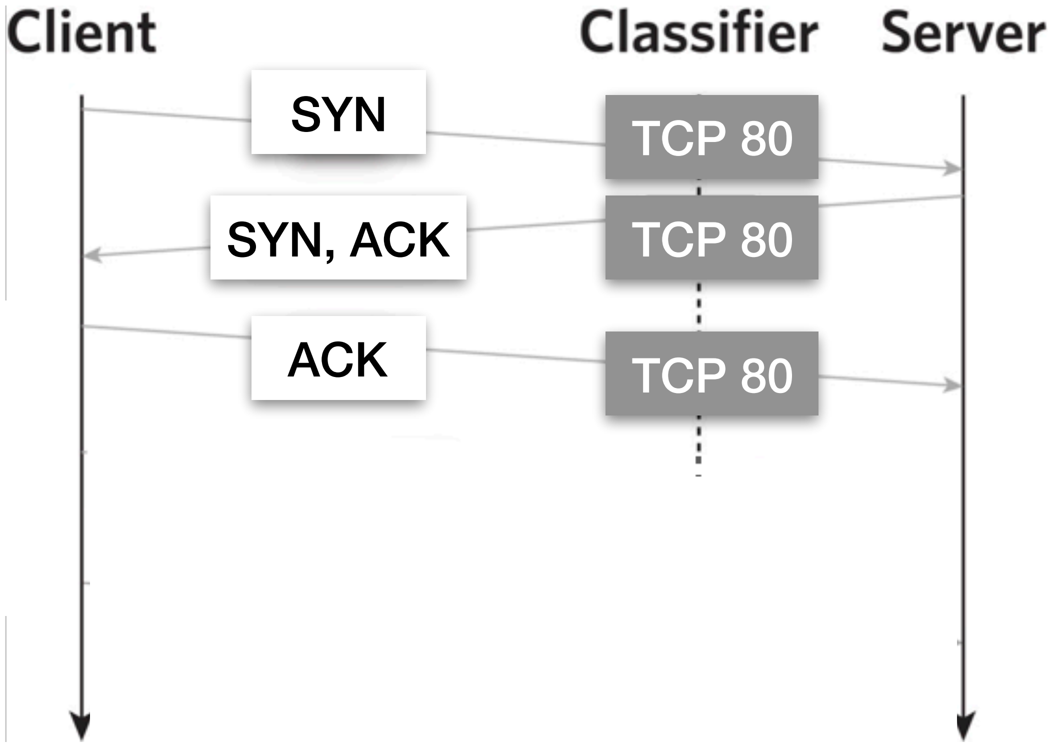


App A is unclassified

Fragmenting the IP packet

Design

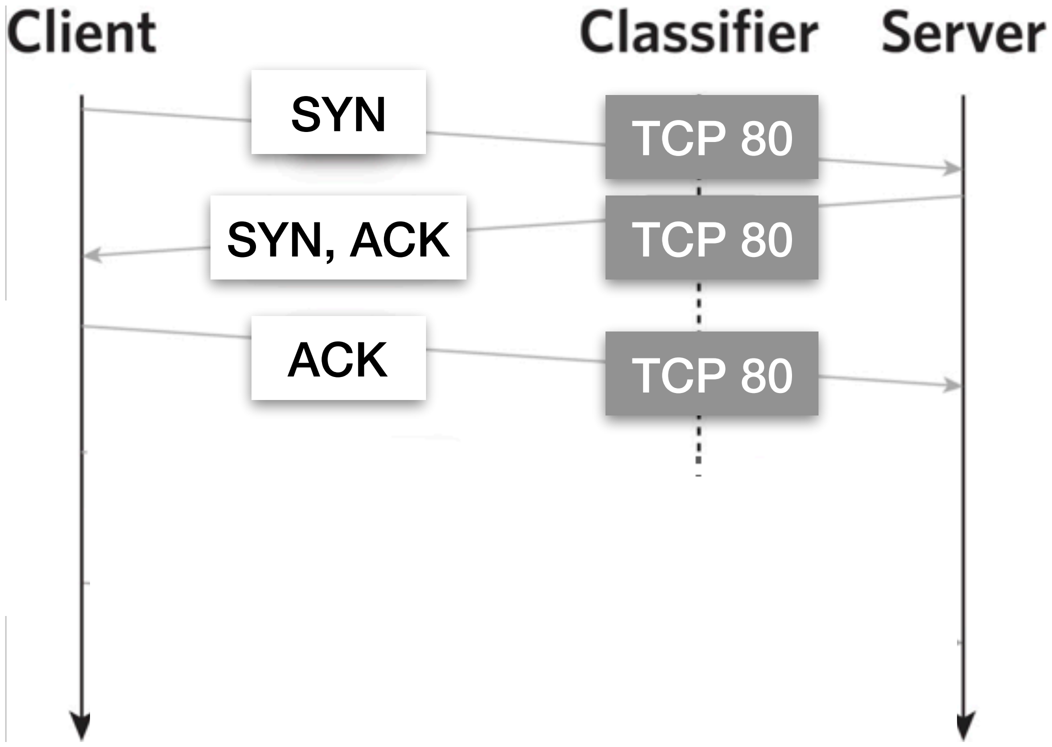
Evasion techniques



Design

Evasion techniques

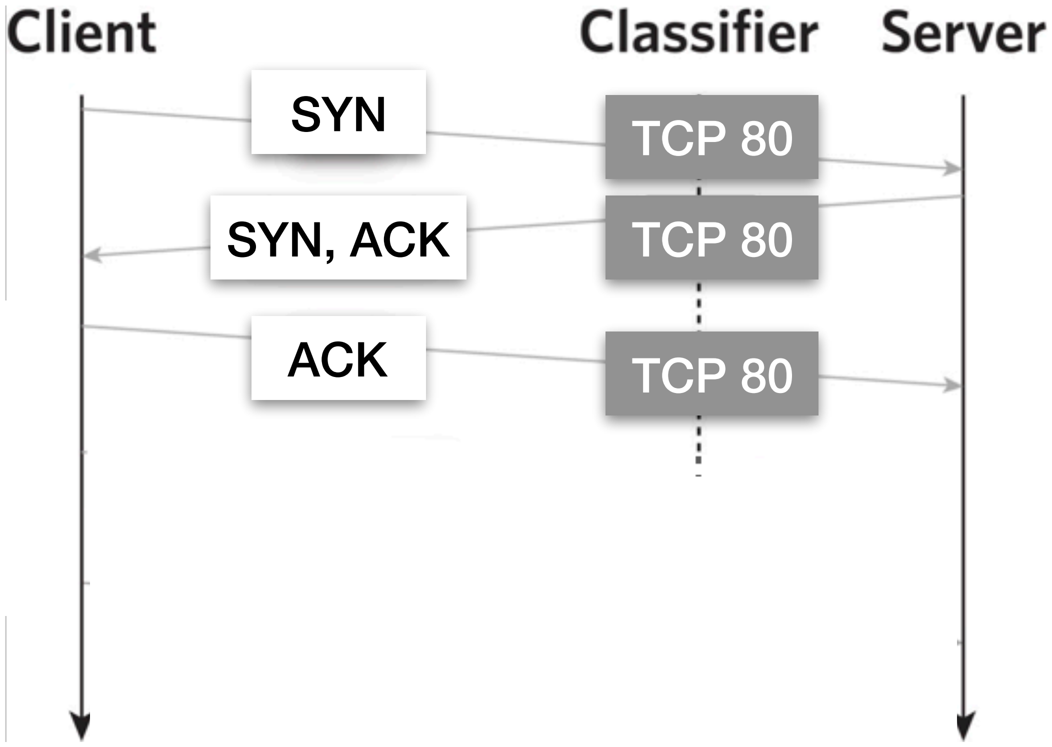
- Observation:
  - Classifiers do no retain classification results indefinitely



Design

Evasion techniques

- Observation:
  - Classifiers do no retain classification results indefinitely
- **Flushing**: causing the classifier to remove the classification state for the flow

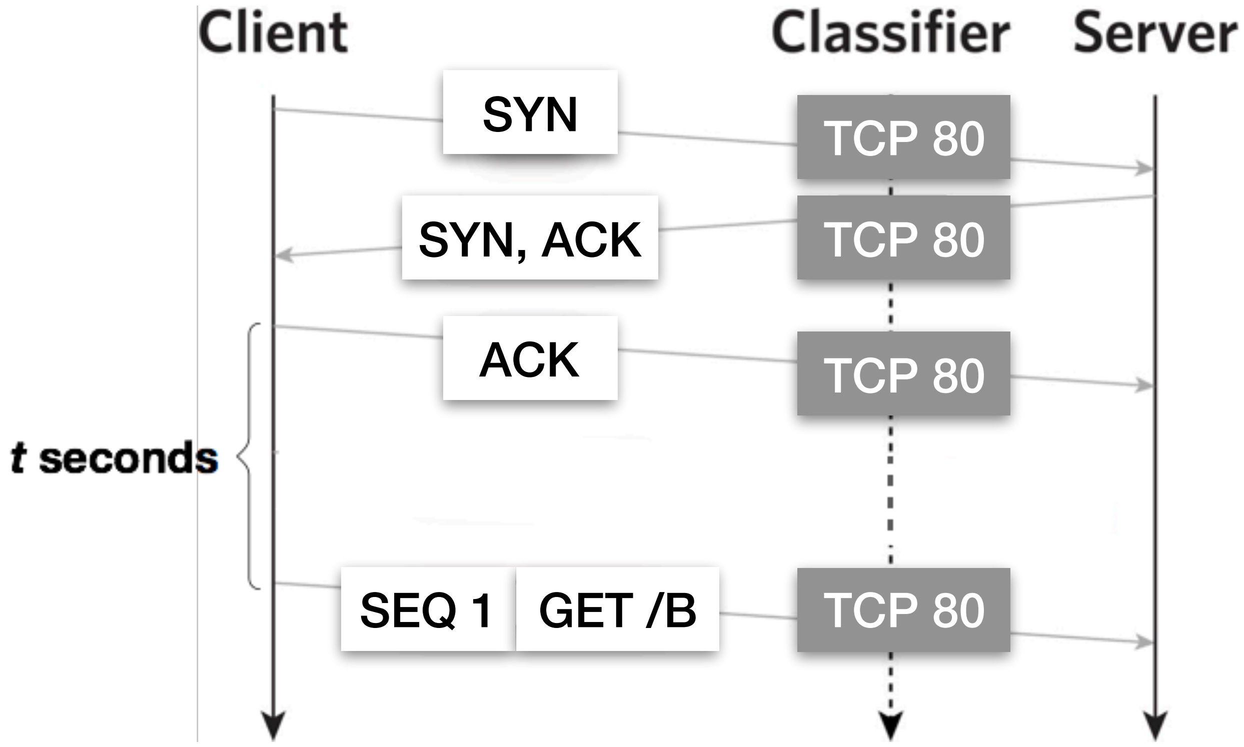


Inserting large delays

Design

Evasion techniques

- Observation:
    - Classifiers do no retain classification results indefinitely
- **Flushing**: causing the classifier to remove the classification state for the flow

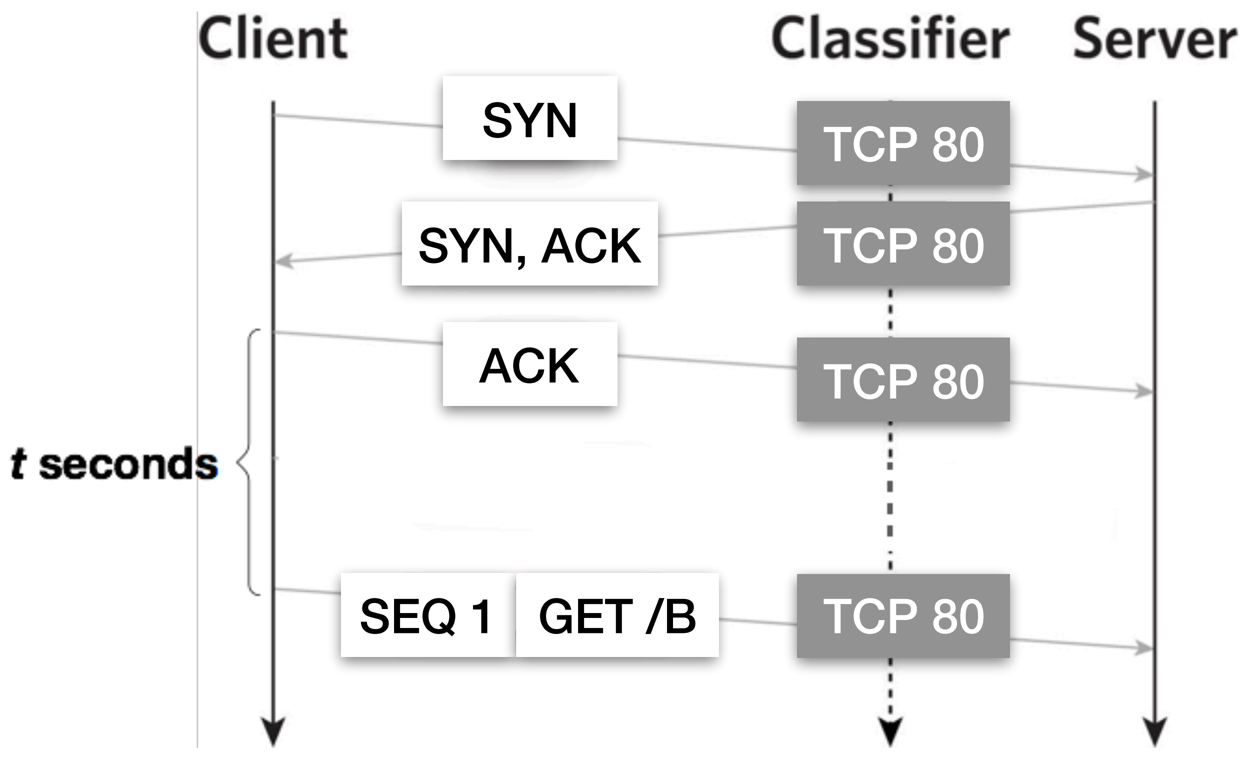


Inserting large delays

Design

Evasion techniques

- Observation:
  - Classifiers do no retain classification results indefinitely
- **Flushing**: causing the classifier to remove the classification state for the flow



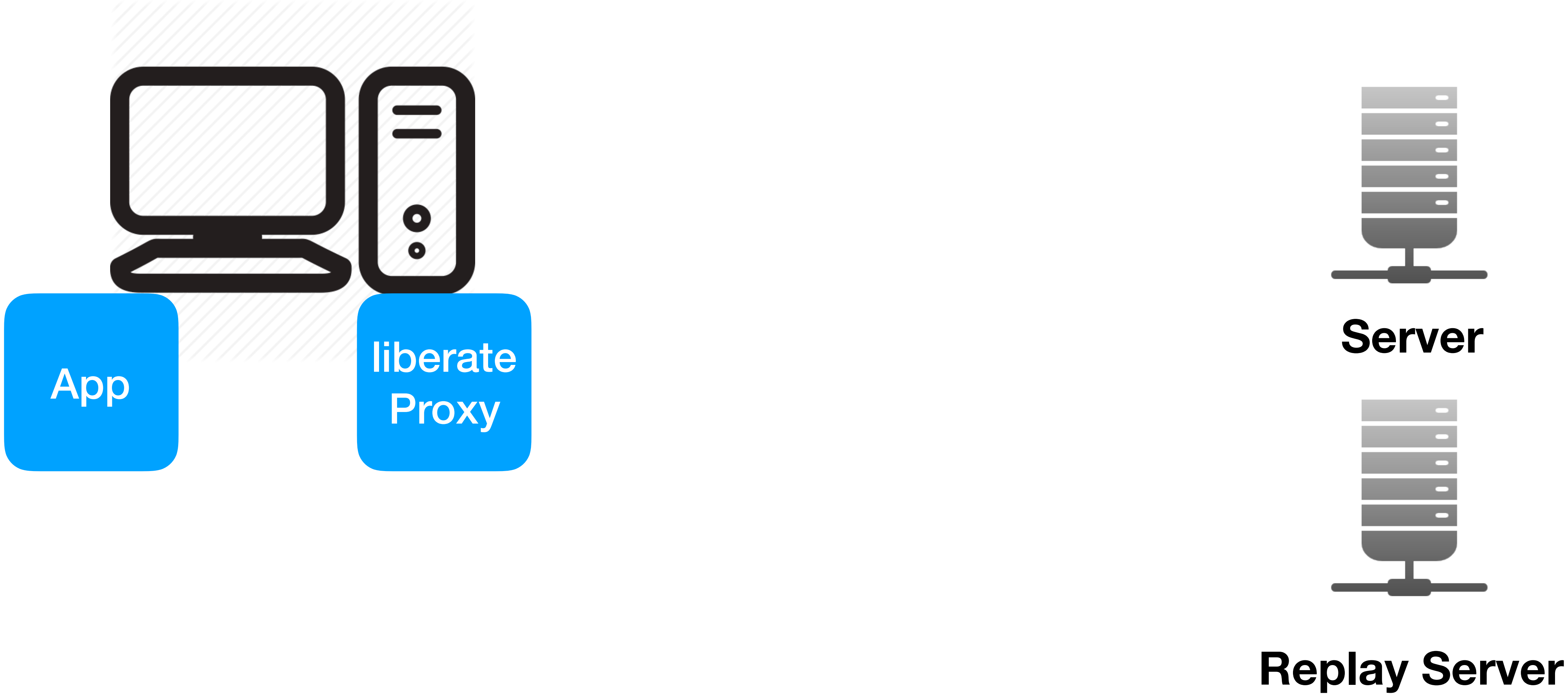
App B is unclassified

Inserting large delays

Outline

- Design and implementation
  - Traffic-classification rules detection
  - Evasion techniques
  - Implementation
- Evaluation
  - Effectiveness across multiple networks

Implementation



App

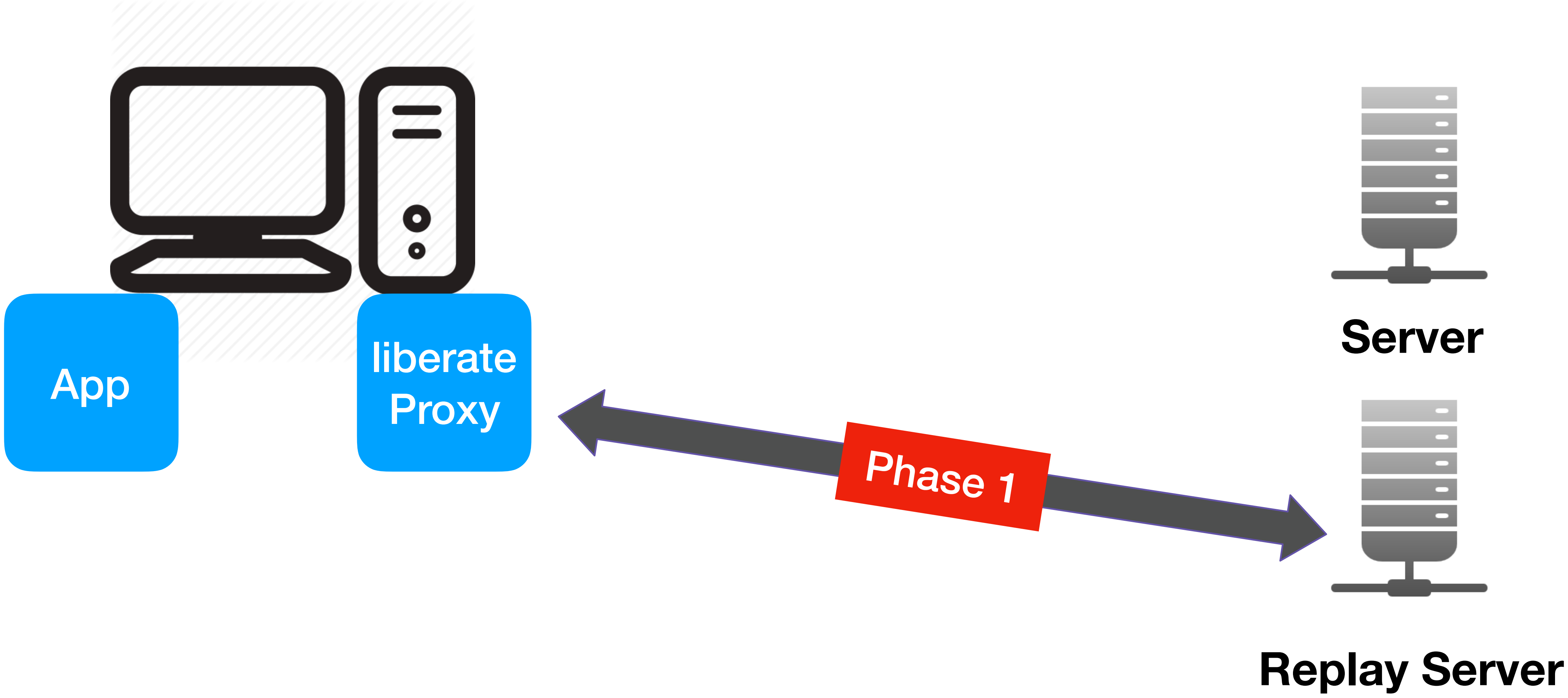
liberate Proxy

Server

Replay Server

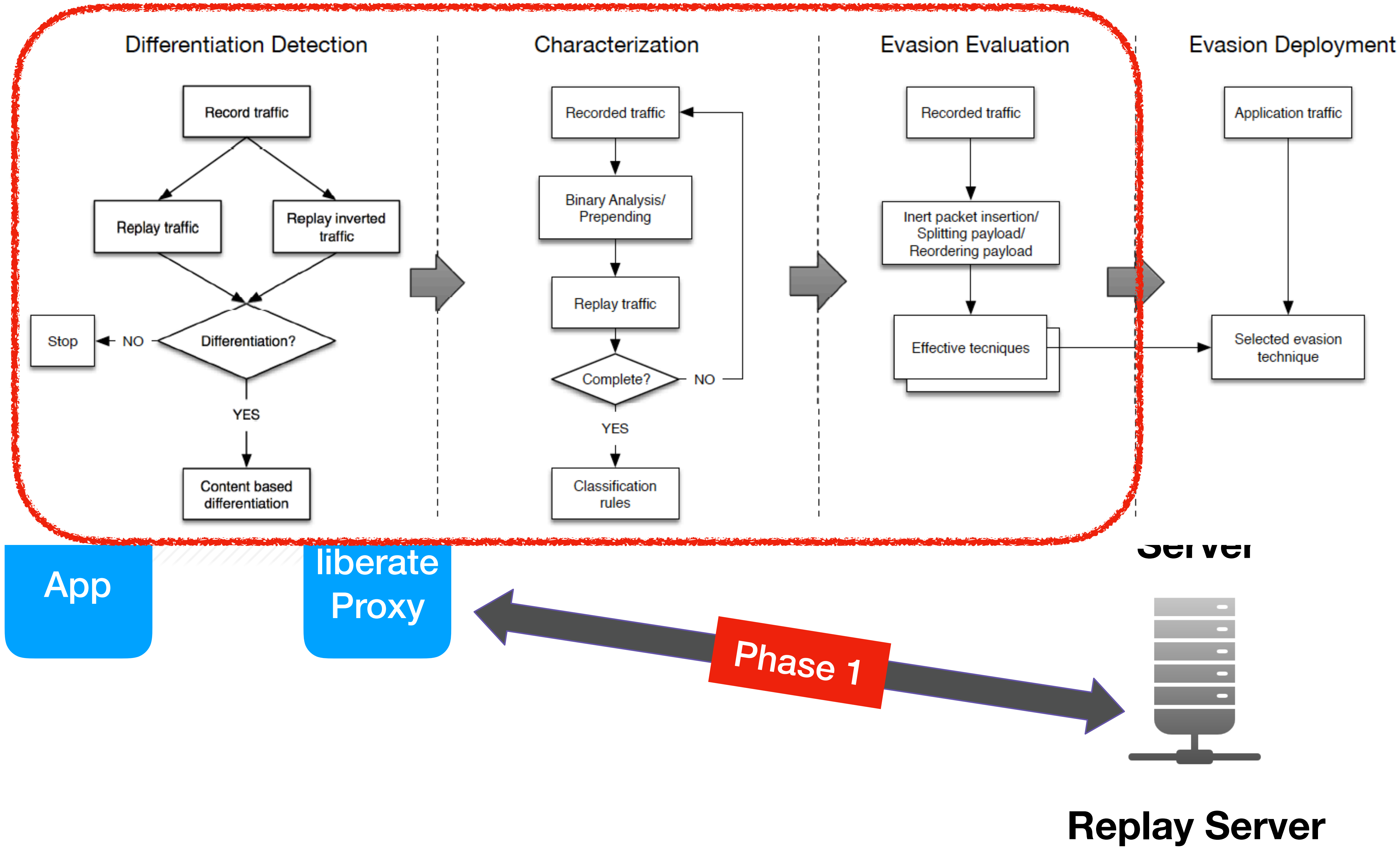
Implementation

- Phase 1: liberate does the analysis using a replay server



Implementation

Phase 1



Differentiation Detection

Record traffic

Replay traffic

Replay inverted traffic

Differentiation?

NO

Stop

YES

Content based differentiation

Characterization

Recorded traffic

Binary Analysis/ Prepending

Replay traffic

Complete?

NO

YES

Classification rules

Evasion Evaluation

Recorded traffic

Inert packet insertion/ Splitting payload/ Reordering payload

Effective tecniques

Evasion Deployment

Application traffic

Selected evasion technique

App

liberate Proxy

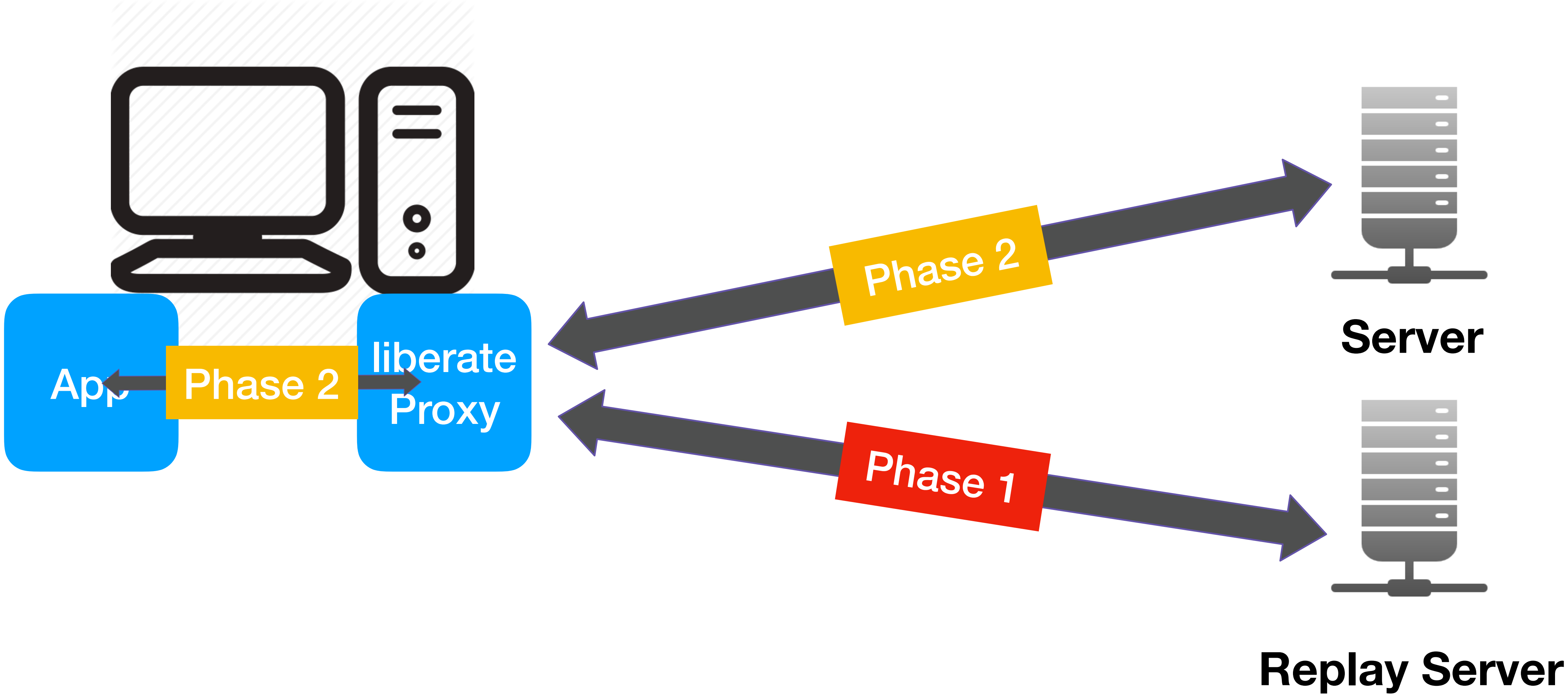
Phase 1

Server

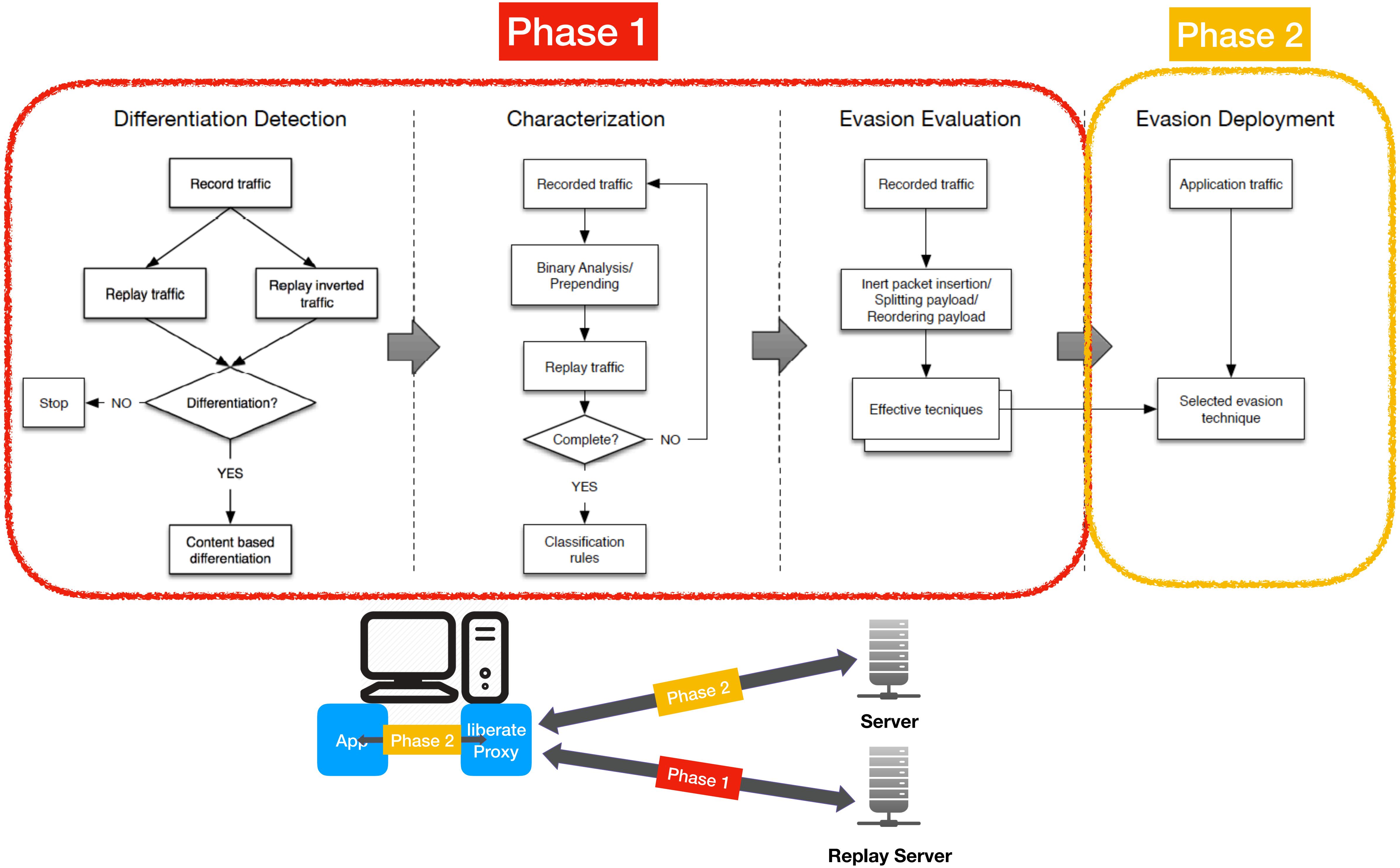
Replay Server

Implementation

- Phase 1: liberate does the analysis using a replay server
- Phase 2: liberate applies evasion technique to traffic in-flight



Implementation

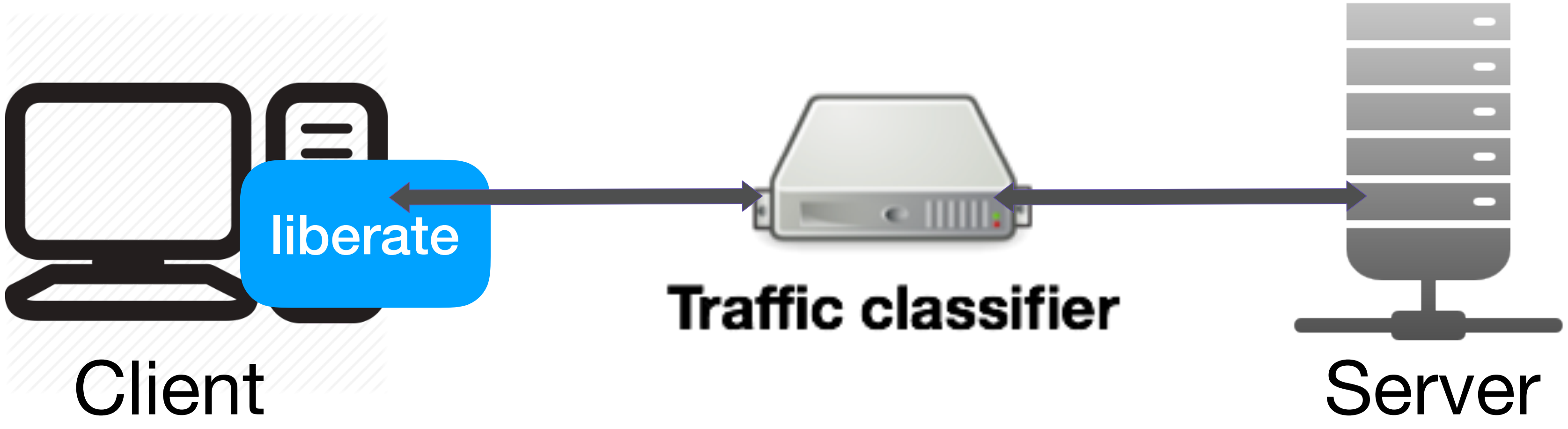


Outline

- Design and implementation
  - Traffic-classification rules detection
  - Evasion techniques
  - Implementation
- Evaluation
  - Effectiveness across multiple networks

Evaluation

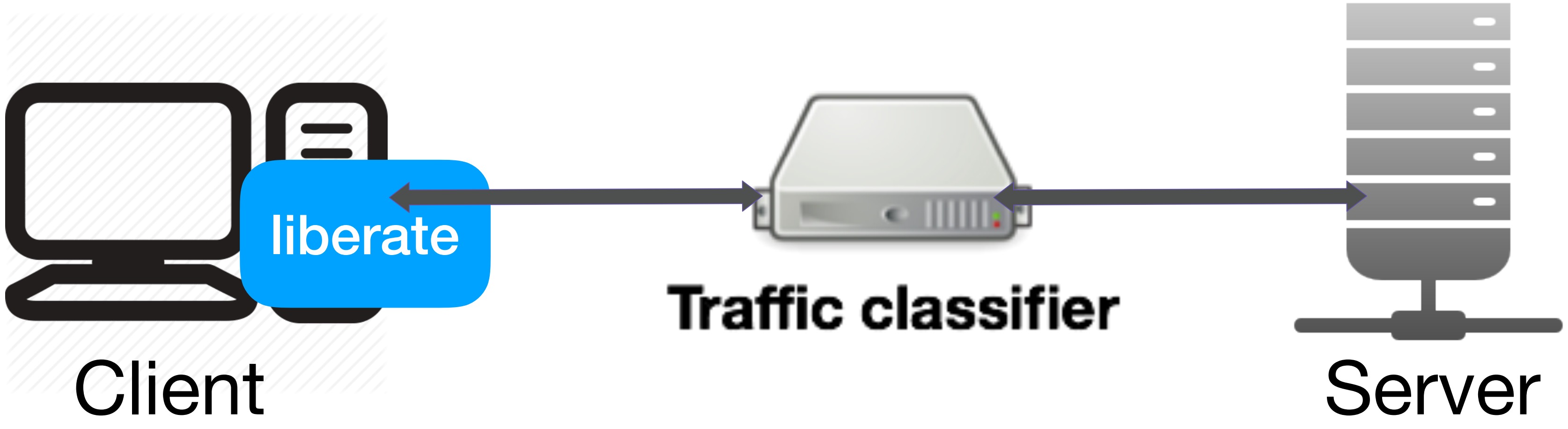
Testbed and in the wild



Evaluation

Testbed and in the wild

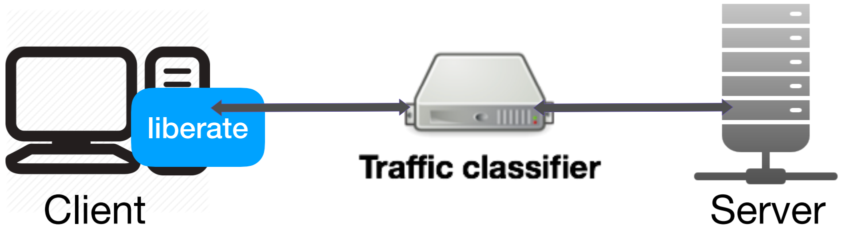
- Testbed evaluation



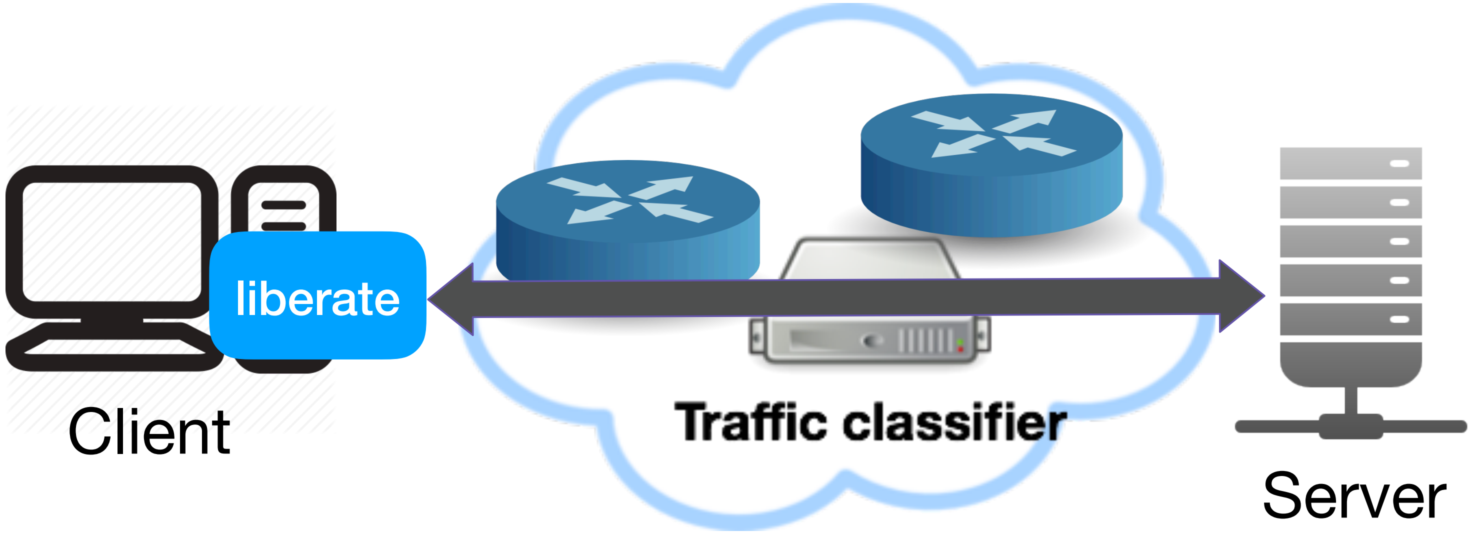
Evaluation

Testbed and in the wild

- Testbed evaluation



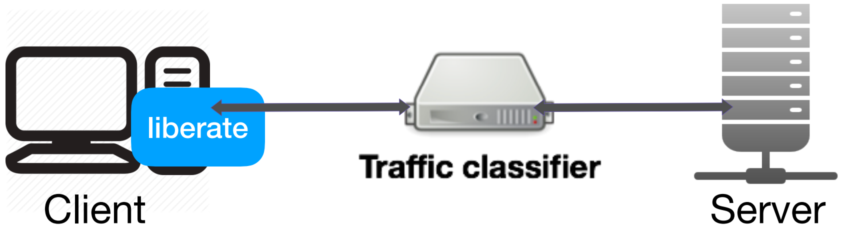
- Evaluation “in the wild”



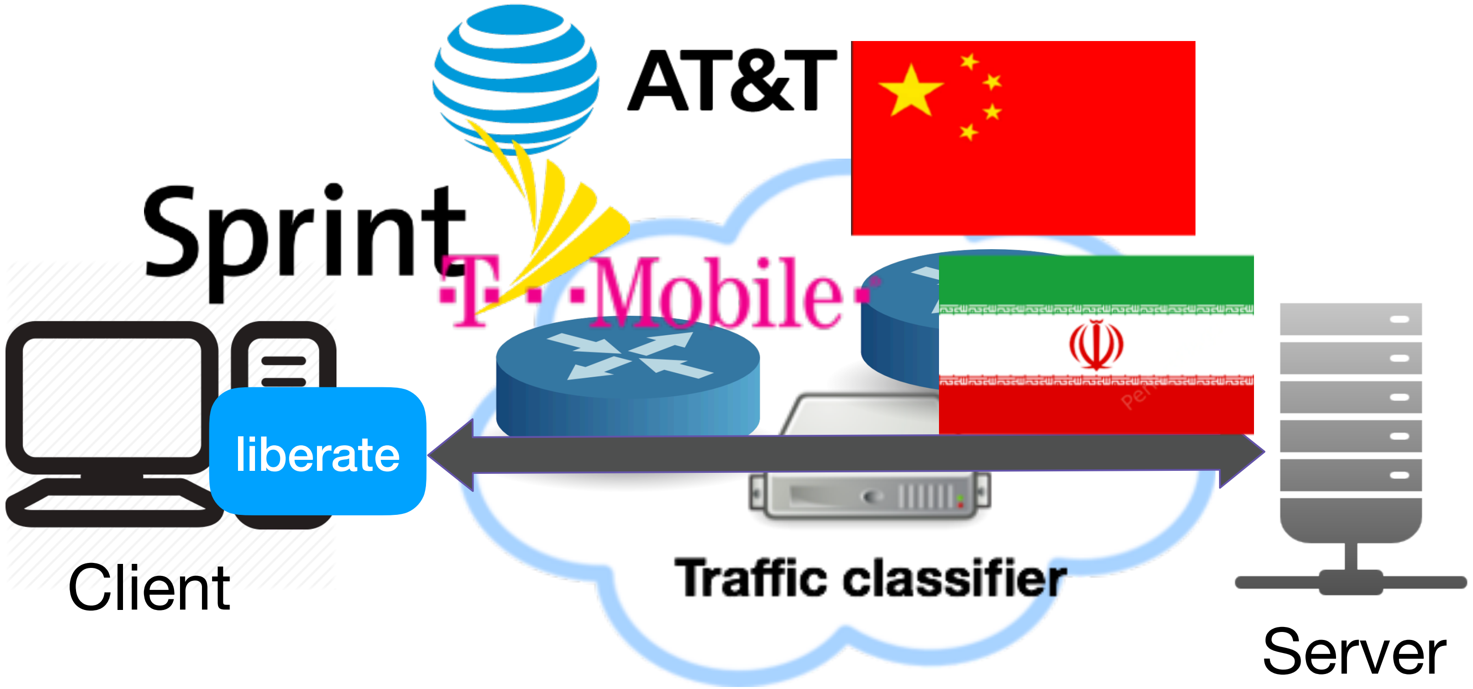
Evaluation

Testbed and in the wild

- Testbed evaluation



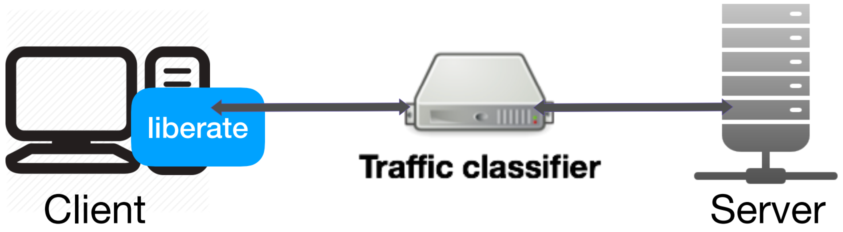
- Evaluation “in the wild”



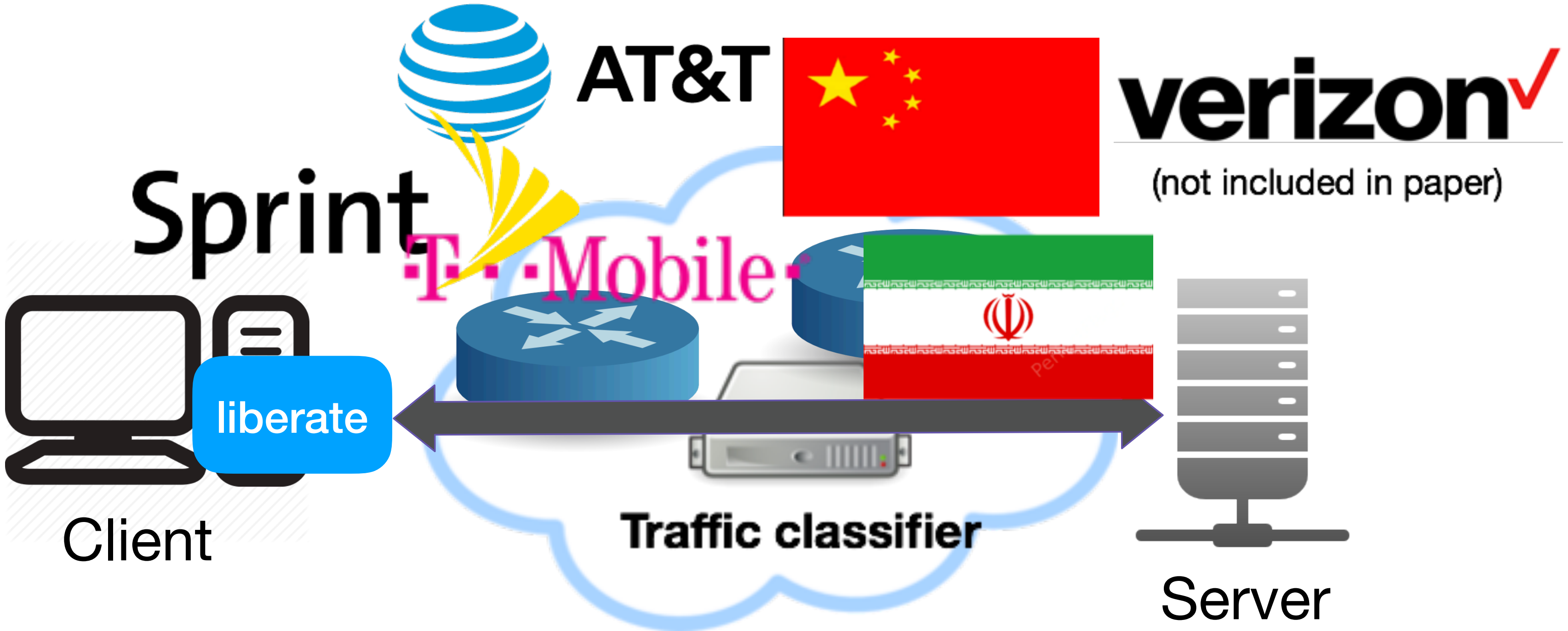
Evaluation

Testbed and in the wild

- Testbed evaluation

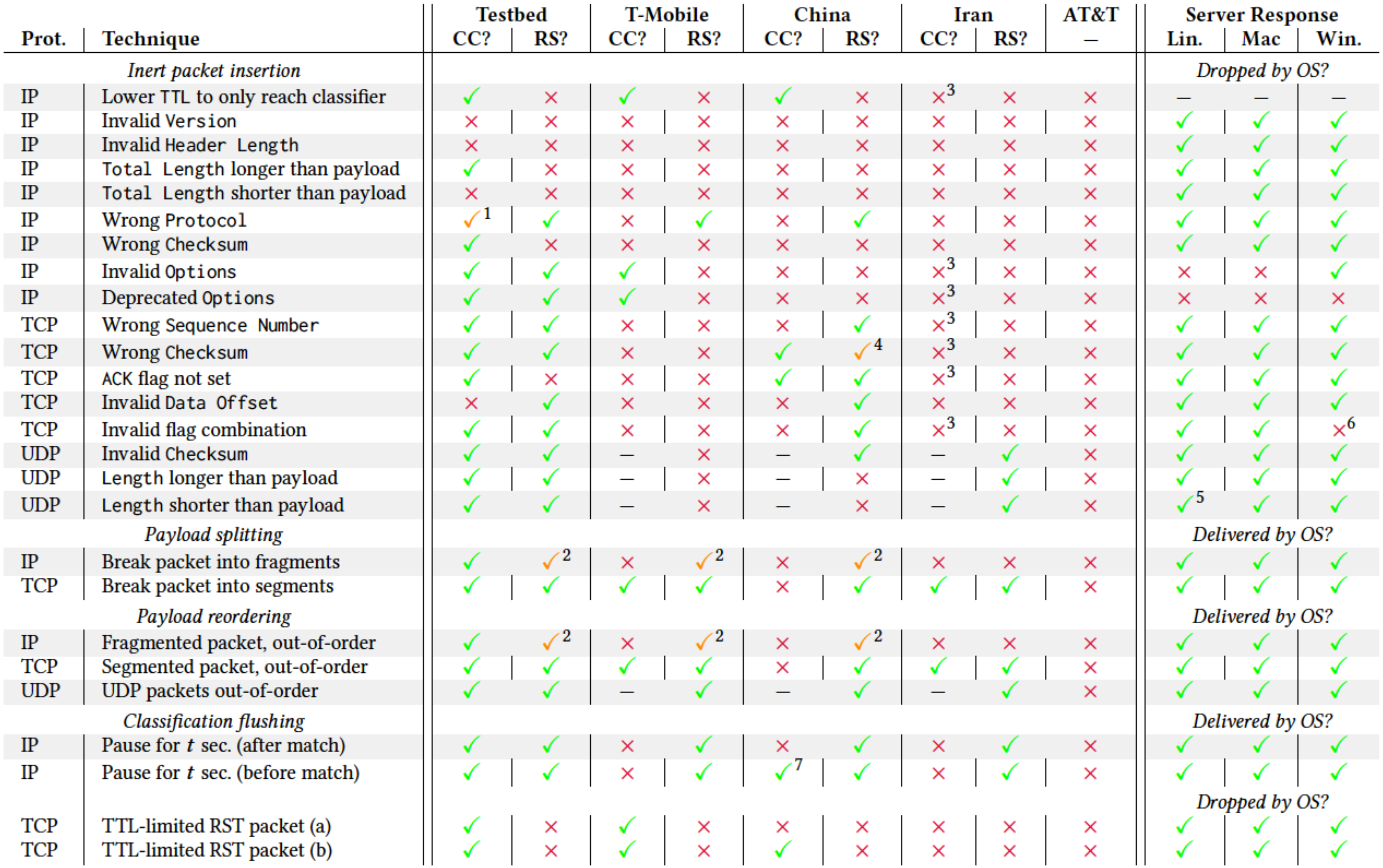


- Evaluation “in the wild”



Evaluation

Results



Prot.	Technique	Testbed CC?	Testbed RS?	T-Mobile CC?	T-Mobile RS?	China CC?	China RS?	Iran CC?	Iran RS?	AT&T —	Server Response Lin.	Server Response Mac	Server Response Win.
	Inert packet insertion										*Dropped by OS?*		
IP	Lower TTL to only reach classifier	✓	✗	✓	✗	✓	✗	✗[3]	✗	✗	—	—	—
IP	Invalid Version	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
IP	Invalid Header Length	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
IP	Total Length longer than payload	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
IP	Total Length shorter than payload	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
IP	Wrong Protocol	✓[1]	✓	✗	✓	✗	✓	✗	✗	✗	✓	✓	✓
IP	Wrong Checksum	✓	✗	✗	✗	✗	✗	✗	✗	✗	✓	✓	✓
IP	Invalid Options	✓	✓	✓	✗	✗	✗	✗[3]	✗	✗	✗	✗	✓
IP	Deprecated Options	✓	✓	✓	✗	✗	✗	✗[3]	✗	✗	✗	✗	✗
TCP	Wrong Sequence Number	✓	✓	✗	✗	✗	✓	✗[3]	✗	✗	✓	✓	✓
TCP	Wrong Checksum	✓	✓	✗	✗	✓	✓[4]	✗[3]	✗	✗	✓	✓	✓
TCP	ACK flag not set	✓	✗	✗	✗	✓	✓	✗[3]	✗	✗	✓	✓	✓
TCP	Invalid Data Offset	✗	✓	✗	✗	✗	✓	✗	✗	✗	✓	✓	✓
TCP	Invalid flag combination	✓	✓	✗	✗	✗	✓	✗[3]	✗	✗	✓	✓	✗[6]
UDP	Invalid Checksum	✓	✓	—	✗	—	✓	—	✓	✗	✓	✓	✓
UDP	Length longer than payload	✓	✓	—	✗	—	✗	—	✓	✗	✓	✓	✓
UDP	Length shorter than payload	✓	✓	—	✗	—	✗	—	✓	✗	✓[5]	✓	✓
	Payload splitting										*Delivered by OS?*		
IP	Break packet into fragments	✓	✓[2]	✗	✓[2]	✗	✓[2]	✗	✗	✗	✓	✓	✓
TCP	Break packet into segments	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓
	Payload reordering										*Delivered by OS?*		
IP	Fragmented packet, out-of-order	✓	✓[2]	✗	✓[2]	✗	✓[2]	✗	✗	✗	✓	✓	✓
TCP	Segmented packet, out-of-order	✓	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓	✓
UDP	UDP packets out-of-order	✓	✓	—	✓	—	✓	—	✓	✗	✓	✓	✓
	Classification flushing										*Delivered by OS?*		
IP	Pause for t sec. (after match)	✓	✓	✗	✓	✗	✓	✗	✓	✗	✓	✓	✓
IP	Pause for t sec. (before match)	✓	✓	✗	✓	✓[7]	✓	✗	✓	✗	✓	✓	✓
											Dropped by OS?		
TCP	TTL-limited RST packet (a)	✓	✗	✓	✗	✗	✗	✗	✗	✗	✓	✓	✓
TCP	TTL-limited RST packet (b)	✓	✗	✓	✗	✓	✗	✗	✗	✗	✓	✓	✓

Evaluation

Example result table

Technique		Test case 1	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✗	
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✗	

Evaluation

Example result table

Technique		Test case 1	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✗	
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✗	

Evaluation

Example result table

Technique		Test case 1	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✗	
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✗	

Evaluation

Example result table

Technique		Test case 1	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✗	
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✗	

Evaluation

Testbed results

Technique		Testbed	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✓	Break packet into two IP fragments
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✓	TTL-limited RST packet before classification

Evaluation

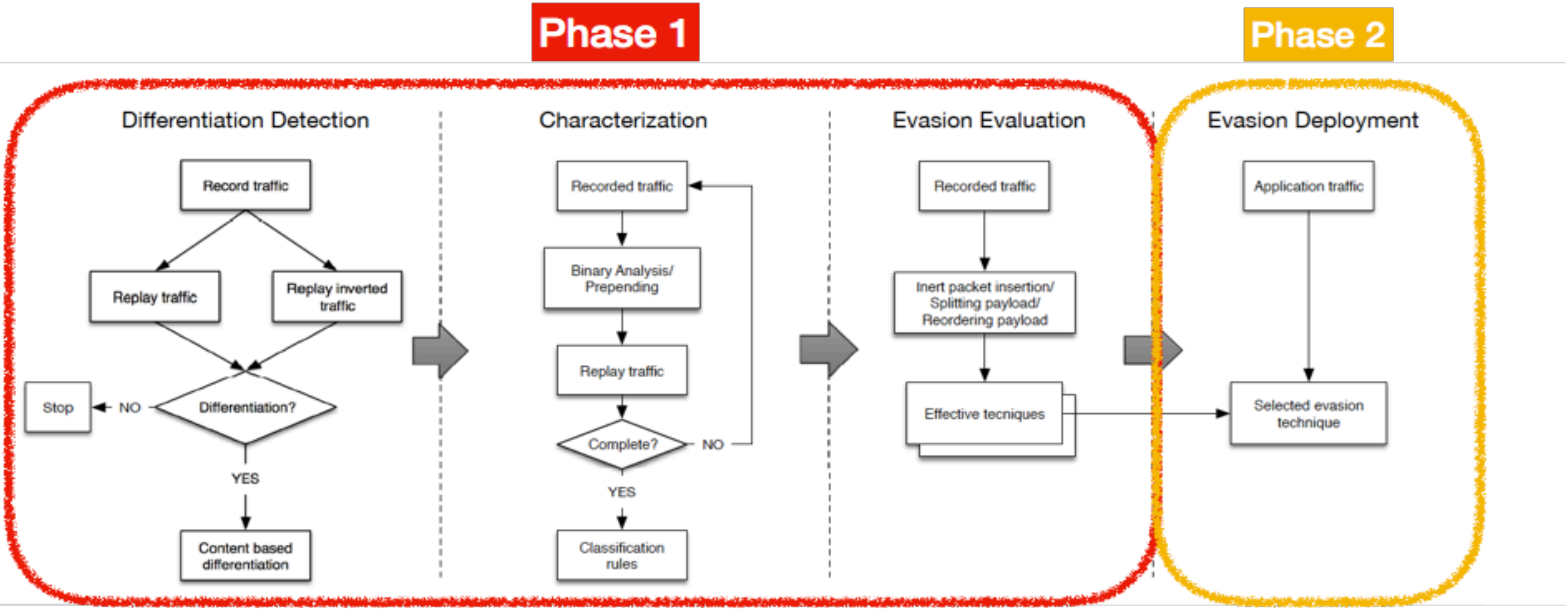
Testbed results

Technique		Testbed	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✓	Break packet into two IP fragments
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✓	TTL-limited RST packet before classification

- Efficiency:
  - **One-time overhead** (phase 1) : 13 minutes

Evaluation

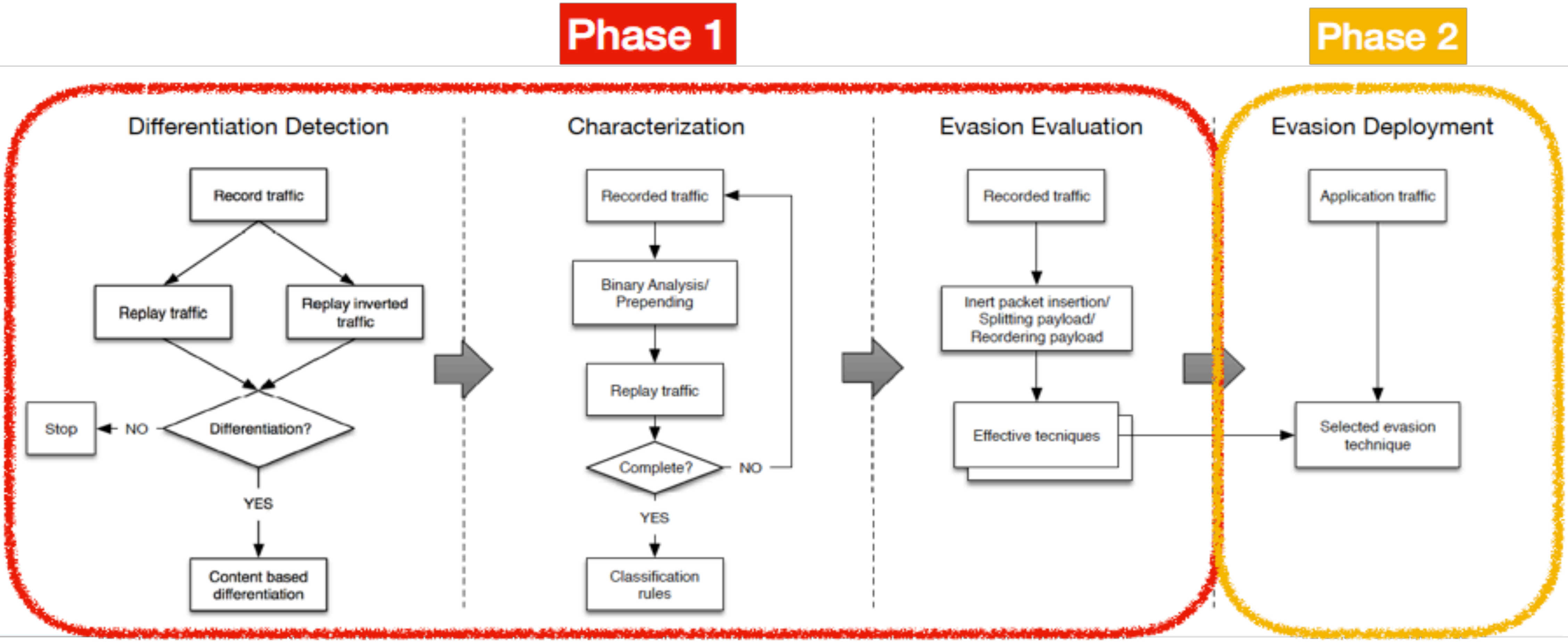
Testbed results



- Efficiency:
  - **One-time overhead** (phase 1) : 13 minutes

Evaluation

Testbed results



- Efficiency:
  - **One-time overhead** (phase 1) : 13 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow

Evaluation

Testbed results

Technique		Testbed	Example technique
Inert packet insertion	IP	✓	Lower TTL to only reach classifier
	TCP	✓	Wrong sequence number
	UDP	✓	Wrong checksum
Payload Splitting		✓	Break packet into two IP fragments
Payload Reordering		✓	Reverse the transmission of first two fragments
Classification flushing		✓	TTL-limited RST packet before classification

- Efficiency:
  - **One-time overhead** (phase 1) : 13 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - All types of techniques were effective in testbed

Evaluation

T mobile ‘Binge On’

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

Evaluation

T mobile 'Binge On'

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

- Classified video (HTTP/S) was throttled to 1.5 Mbps and zero-rated
- Efficiency:
  - **One-time overhead** (phase 1) : 30 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow

Evaluation

T mobile ‘Binge On’

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

- Classified video (HTTP/S) was throttled to 1.5 Mbps and zero-rated
- Efficiency:
  - **One-time overhead** (phase 1) : 30 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - UDP traffic (e.g., Youtube video in QUIC) was not classified

Evaluation

T mobile ‘Binge On’

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

- Classified video (HTTP/S) was throttled to 1.5 Mbps and zero-rated
- Efficiency:
  - **One-time overhead** (phase 1) : 30 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - UDP traffic (e.g., Youtube video in QUIC) was not classified
  - Breaking packet into **5** TCP segments evaded classification

Evaluation

T mobile ‘Binge On’

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

- Classified video (HTTP/S) was throttled to 1.5 Mbps and zero-rated
- Efficiency:
  - **One-time overhead** (phase 1) : 30 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - UDP traffic (e.g., Youtube video in QUIC) was not classified
  - Breaking packet into **5** TCP segments evaded classification
  - Reversing the order of initial packets was effective

Evaluation

T mobile ‘Binge On’

Technique		Testbed	T mobile	Example technique
Inert packet insertion	IP	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	
	UDP	✓	—	
Payload Splitting		✓	✓	Break packet into five TCP segments
Payload Reordering		✓	✓	Reverse the transmission of first two segments
Classification flushing		✓	✓	TTL-limited RST packet before classification

- Classified video (HTTP/S) was throttled to 1.5 Mbps and zero-rated
- Efficiency:
  - **One-time overhead** (phase 1) : 30 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - UDP traffic (e.g., Youtube video in QUIC) was not classified
  - Breaking packet into **5** TCP segments evaded classification
  - Reversing the order of initial packets was effective

Evaluation

The Great Firewall of China

Technique		Testbed	T mobile	GFC	Example technique
Inert packet insertion	IP	✓	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	✓	Wrong Checksum
	UDP	✓	—	—	
Payload Splitting		✓	✓	✗	
Payload Reordering		✓	✓	✗	
Classification flushing		✓	✓	✓	Pause for ***t*** seconds before classification

Evaluation

The Great Firewall of China

Technique		Testbed	T mobile	GFC	Example technique
Inert packet insertion	IP	✓	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	✓	Wrong Checksum
	UDP	✓	—	—	
Payload Splitting		✓	✓	✗	
Payload Reordering		✓	✓	✗	
Classification flushing		✓	✓	✓	Pause for ***t*** seconds before classification

- Classified HTTP content was blocked by 3-5 RST packets
- Efficiency:
  - **One-time overhead** (phase 1) : 20 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow

Evaluation

The Great Firewall of China

Technique		Testbed	T mobile	GFC	Example technique
Inert packet insertion	IP	✓	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	✓	Wrong Checksum
	UDP	✓	—	—	
Payload Splitting		✓	✓	✗	
Payload Reordering		✓	✓	✗	
Classification flushing		✓	✓	✓	Pause for *t* seconds before classification

- Classified HTTP content was blocked by 3-5 RST packets
- Efficiency:
  - **One-time overhead** (phase 1) : 20 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - Both IP/ TCP inert insertion succeeded

Evaluation

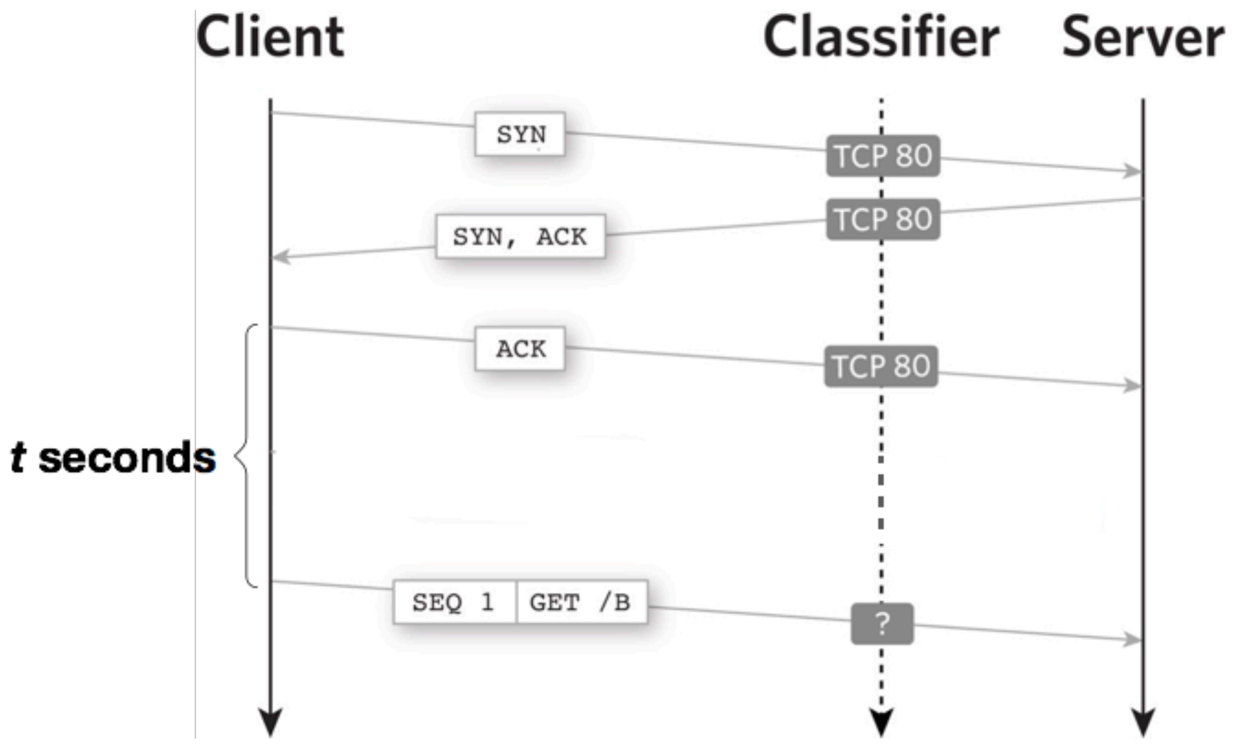
The Great Firewall of China

Technique		Testbed	T mobile	GFC	Example technique
Inert packet insertion	IP	✓	✓	✓	Lower TTL to only reach classifier
	TCP	✓	✗	✓	Wrong Checksum
	UDP	✓	—	—	
Payload Splitting		✓	✓	✗	
Payload Reordering		✓	✓	✗	
Classification flushing		✓	✓	✓	Pause for ***t*** seconds before classification

- Classified HTTP content was blocked by 3-5 RST packets
- Efficiency:
  - **One-time overhead** (phase 1) : 20 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - Both IP/ TCP inert insertion succeeded
  - Flushing classification by pausing succeeded

Evaluation

The Great Firewall of China

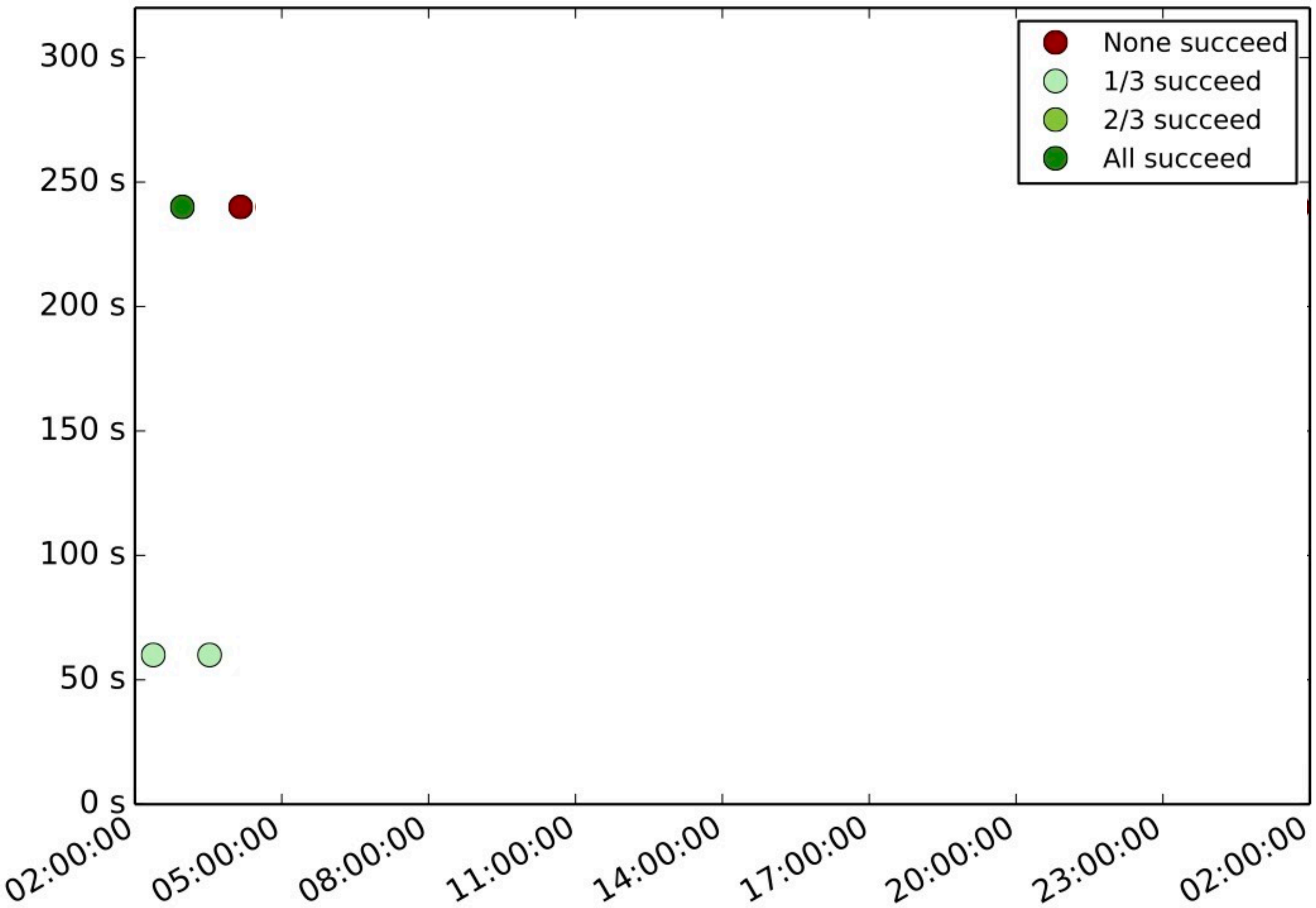


- Classified HTTP content was blocked by 3-5 RST packets
- Efficiency:
  - **One-time overhead** (phase 1) : 20 minutes
  - Run-time overhead (phase 2) : tens of bytes per flow
- Effectiveness:
  - Both IP/ TCP inert insertion succeeded
  - Flushing classification by pausing succeeded

Evaluation

The Great Firewall of China

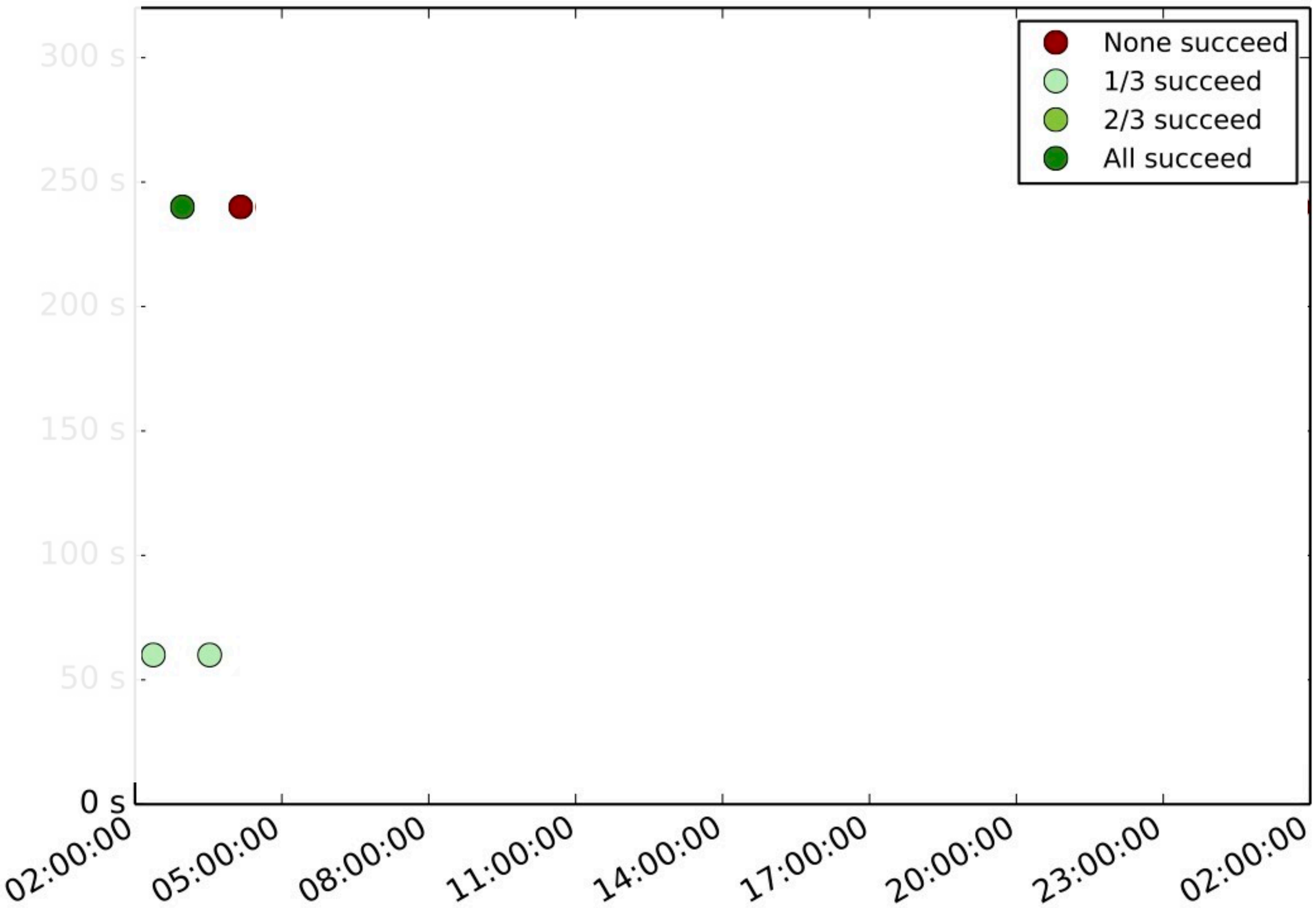
Time-of-day effects when flushing classification



Evaluation

The Great Firewall of China

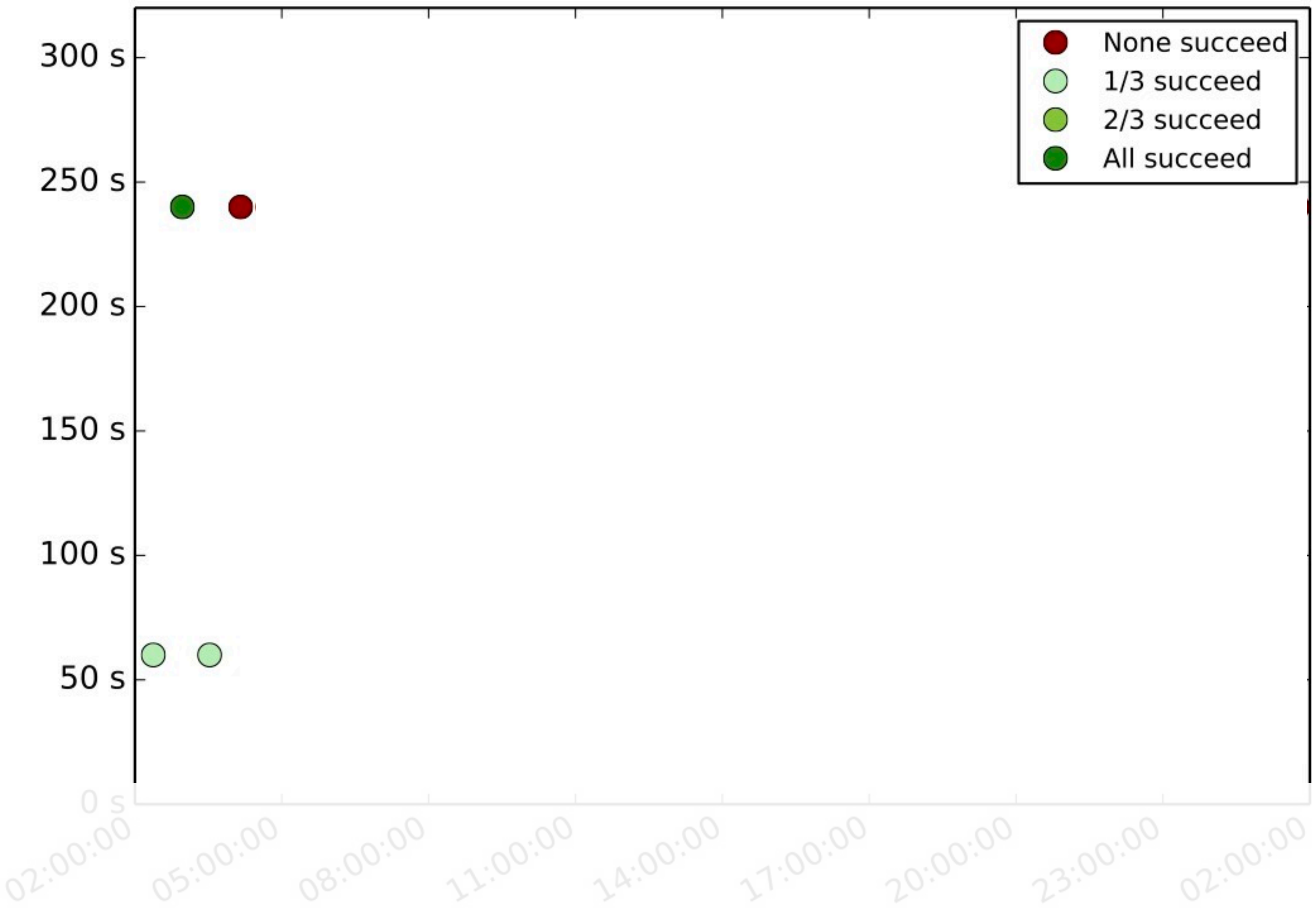
Time-of-day effects when flushing classification



Evaluation

The Great Firewall of China

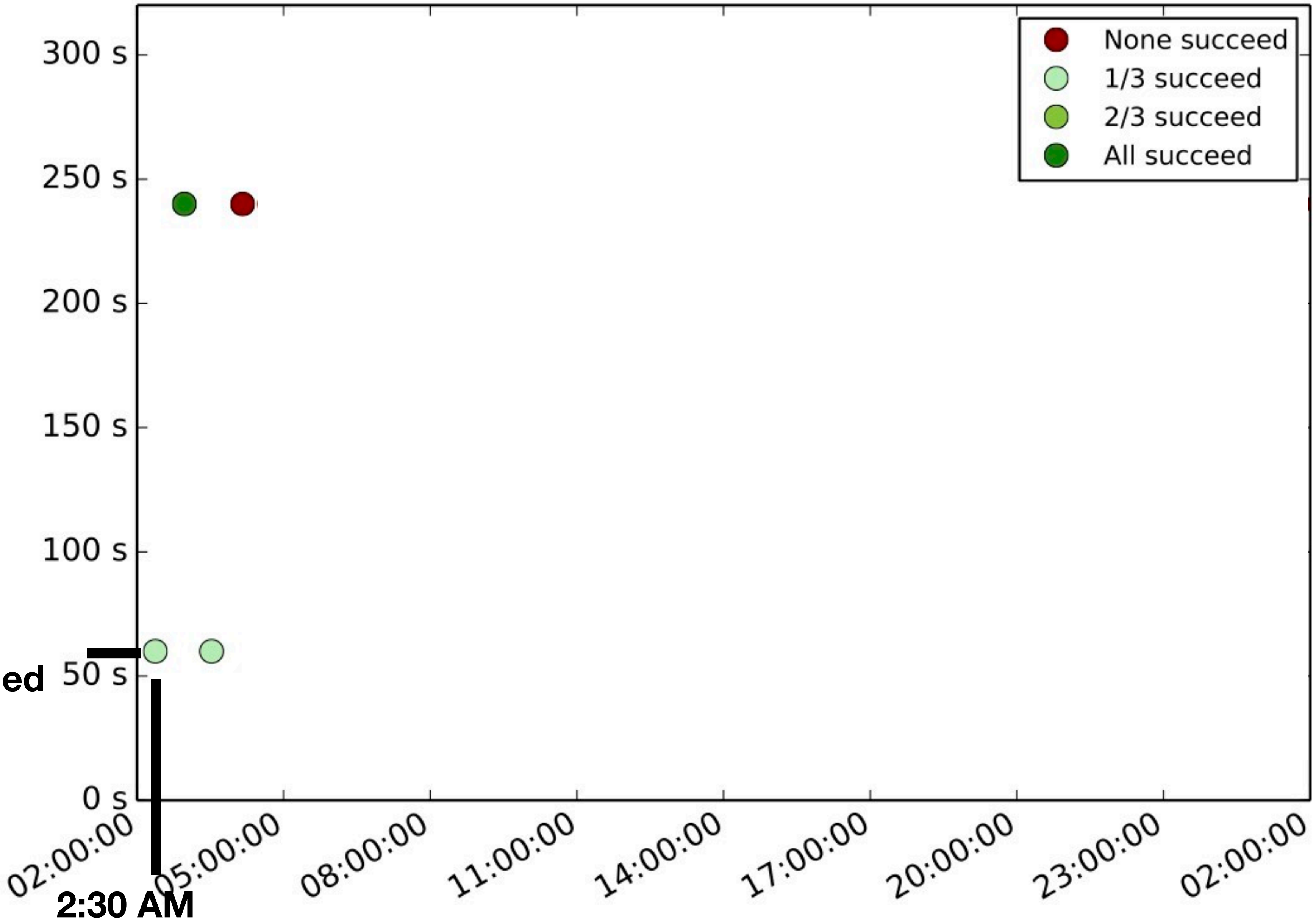
Time-of-day effects when flushing classification



Evaluation

The Great Firewall of China

Time-of-day effects when flushing classification



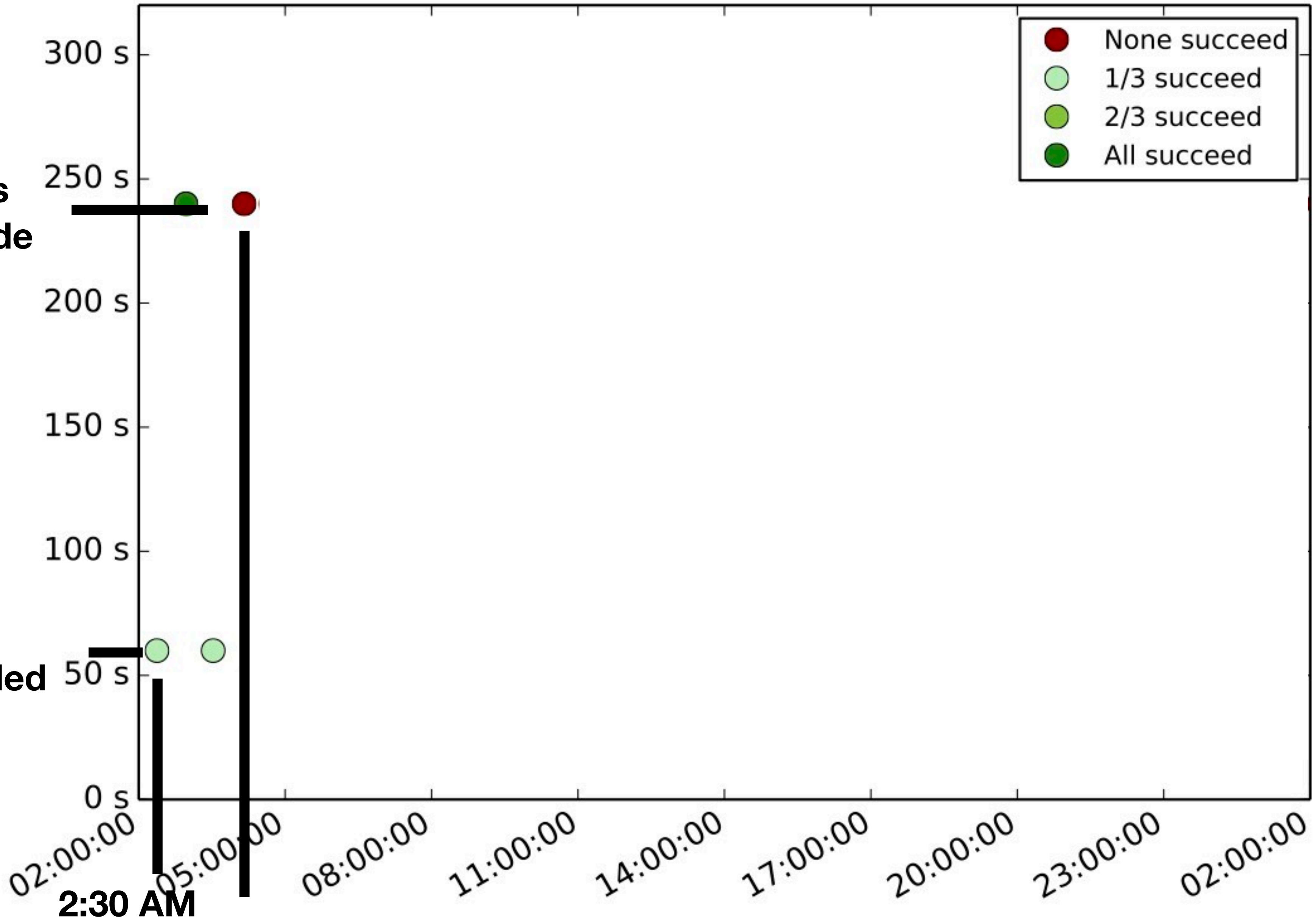
60 seconds successfully evaded

2:30 AM

Evaluation

The Great Firewall of China

Time-of-day effects when flushing classification



240 seconds failed to evade

60 seconds successfully evaded

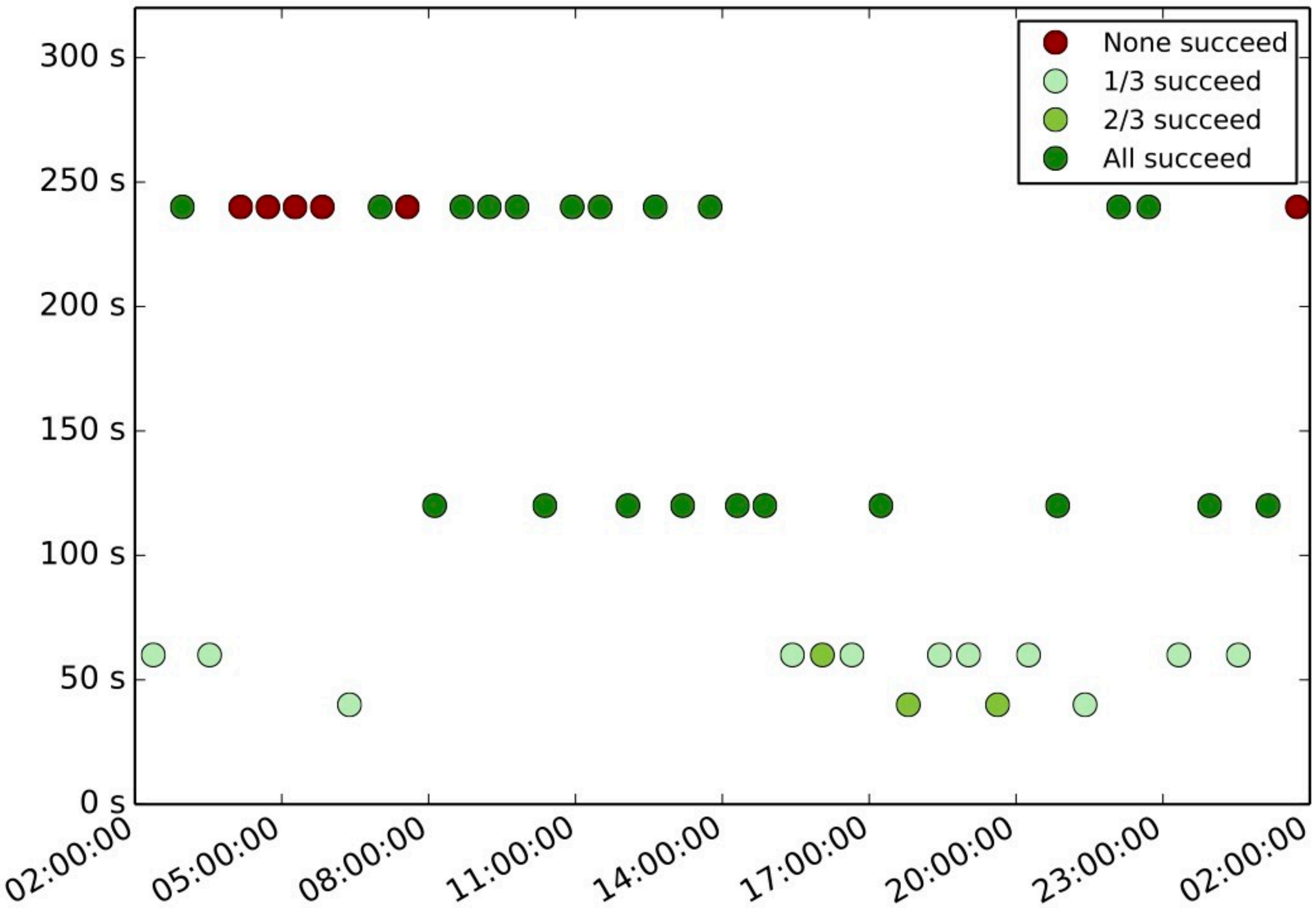
2:30 AM

4:00 AM

Evaluation

The Great Firewall of China

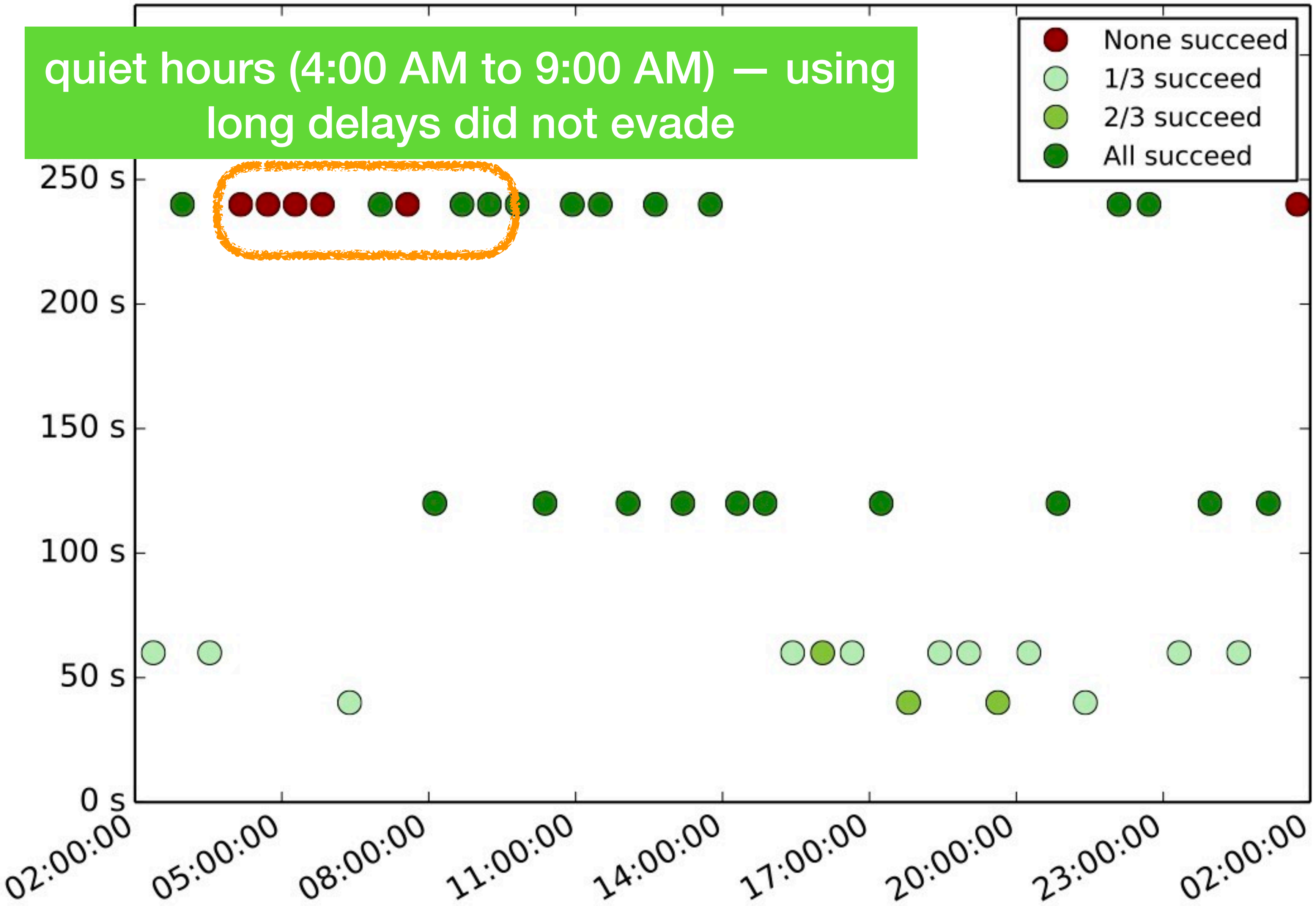
Time-of-day effects when flushing classification



Evaluation

The Great Firewall of China

Time-of-day effects when flushing classification

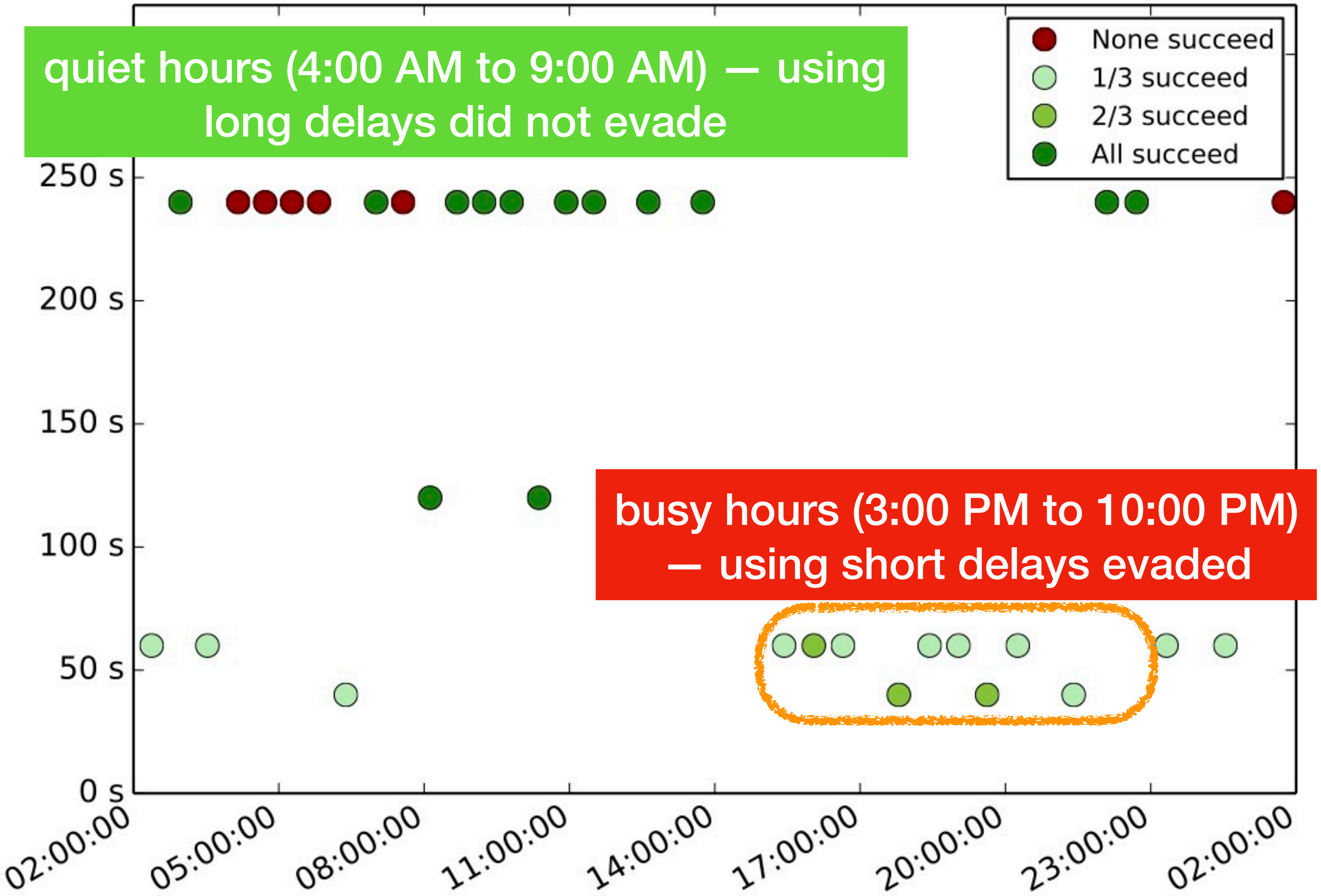


quiet hours (4:00 AM to 9:00 AM) — using long delays did not evade

Evaluation

The Great Firewall of China

Time-of-day effects when flushing classification



quiet hours (4:00 AM to 9:00 AM) — using long delays did not evade

busy hours (3:00 PM to 10:00 PM) — using short delays evaded

Conclusion

- A tool that **automatically** and **efficiently** evades differentiation
  - A **taxonomy of evasion techniques**
  - An **empirical measurement** of traffic classifiers
  - liberate **evaded** classifiers with low run-time overhead
- **Public, open-source** tools and datasets
- **Future work**: more resilient evasion techniques

Thanks

For more details about liberate, code, and data :
http://dd.meddle.mobi/liberate

NSF

Google