Exposing and Evading Middlebox Policies

DAVID CHOFFNES

Northeastern University
College of Computer and Information Science

Middleboxes are pervasive

In-network functionality can be really helpful

- Security (IPS)
- Performance (proxies)
- Fairness (traffic management)

Middleboxes are pervasive

In-network functionality can be really helpful

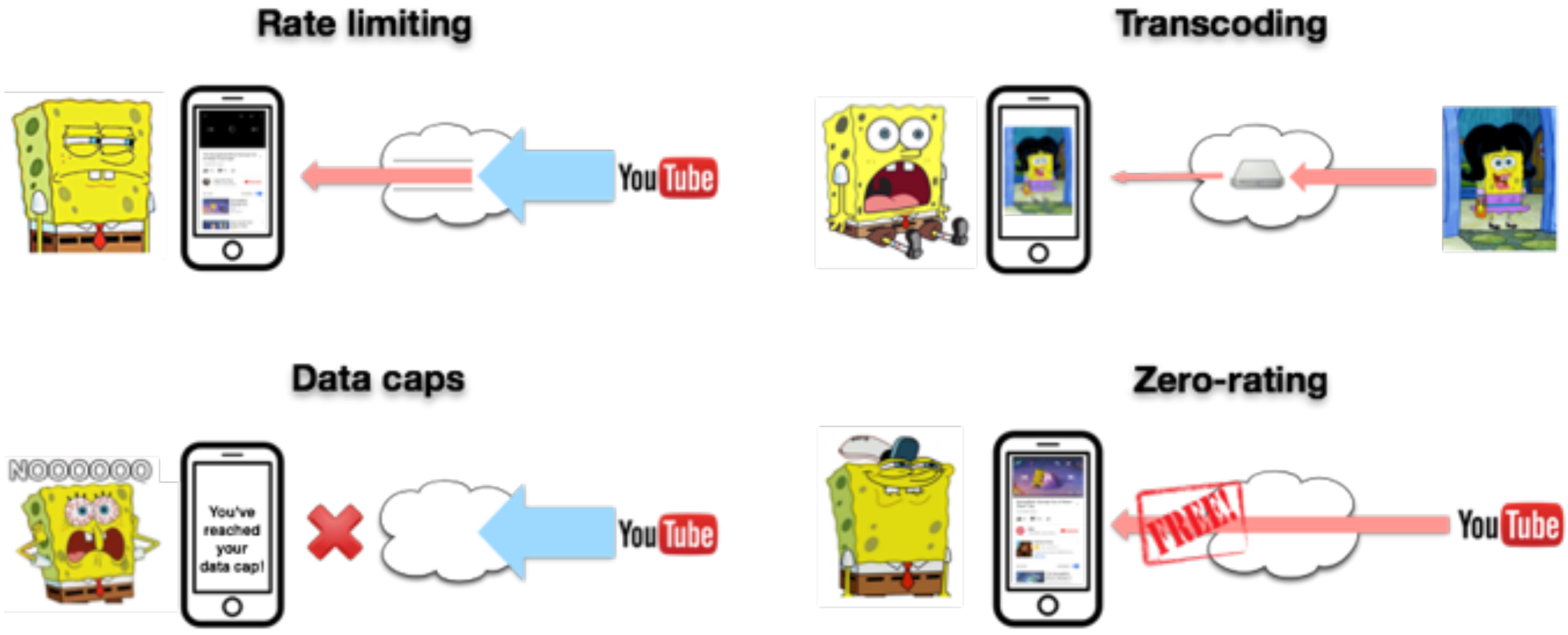
- Security (IPS)
- Performance (proxies)
- Fairness (traffic management)

Double-edged sword

- “Security” (censorship)
- “Performance” (transcoding to degraded quality)
- “Fairness” (throttling or boosting specific apps)

Context

Some device in the network (**middlebox**) uses **DPI** to **classify** traffic and apply **policies** accordingly



Key open questions

What is the nature of **deployed middlebox policies**?

How do middleboxes **enforce** policies?

What are **(un)intentional consequences**?

What can **users** do about this?

Challenges for middlebox research

Middleboxes are protected, undisclosed systems

- Expensive (5-6 figures)
- Hard to acquire
- Little-to-no documentation
- (Almost) never acknowledged

Challenges for middlebox research

Middleboxes are protected, undisclosed systems
- Expensive (5-6 figures)
- Hard to acquire
- Little-to-no documentation
- (Almost) never acknowledged

Understanding policies requires targeted traffic
- Need to identify potential targets
- Potentially requires lots of tests
- Not clear a priori what signals to use to detect classification

Our approach

Examine (in detail) a small testbed of DPI middleboxes

- Clear signals for classification
- Control over policies applied to classes

Our approach

Examine (in detail) a small testbed of DPI middleboxes

- Clear signals for classification
- Control over policies applied to classes

Extend to operationally deployed devices

Our approach

Examine (in detail) a small testbed of DPI middleboxes
- Clear signals for classification
- Control over policies applied to classes

Extend to operationally deployed devices

Use application-generated traffic to trigger policies
- Then explore what part of traffic triggered them
- Identify implications of inferred implementations

Our approach

Examine (in detail) a small testbed of DPI middleboxes

- Clear signals for classification
- Control over policies applied to classes

Extend to operationally deployed devices

Use application-generated traffic to trigger policies

- Then explore what part of traffic triggered them
- Identify implications of inferred implementations

Systematically violate assumptions in classifiers

What are middleboxes doing?

ISP	YouTube	Netflix	Spotify
Verizon			
Tmobile			
ATT			
Sprint			
Boost			
BlackWireless			
H2O			
SimpleMobile			
NET10			

What are middleboxes doing (2015)?

ISP	YouTube	Netflix	Spotify
Verizon	m	m	m
Tmobile	-	-	-
ATT	m	m	m
Sprint	m	m	m
Boost	m	m	m
BlackWireless	60%	-	-
H2O	37%	45%	65%
SimpleMobile	36%	-	-
NET10	p	p	p

m: content modified on the fly

p: translucent proxies change connection behavior

What are middleboxes doing (2015)?

ISP	YouTube	Netflix	Spotify
Verizon	m	m	m
Tmobile	-	-	-
ATT	m	m	m
Sprint	m	m	m
Boost			
BlackWireless			
H2O			
SimpleMobile			
NET10	p	p	p

m: content modified on the fly

p: translucent proxies change connection behavior

Stopped after Open Internet Order…

We will keep monitoring…

What are middleboxes doing (2015)?

ISP	YouTube	Netflix	Spotify
Verizon	m	m	m
Tmobile	-	-	-
ATT	m	m	m
Sprint	m	m	m
Boost			
BlackWireless			
H2O			
SimpleMobile			
NET10	p	p	p

m: content modified on the fly

p: translucent proxies change connection behavior

Stopped after Open Internet Order…

We will keep monitoring…

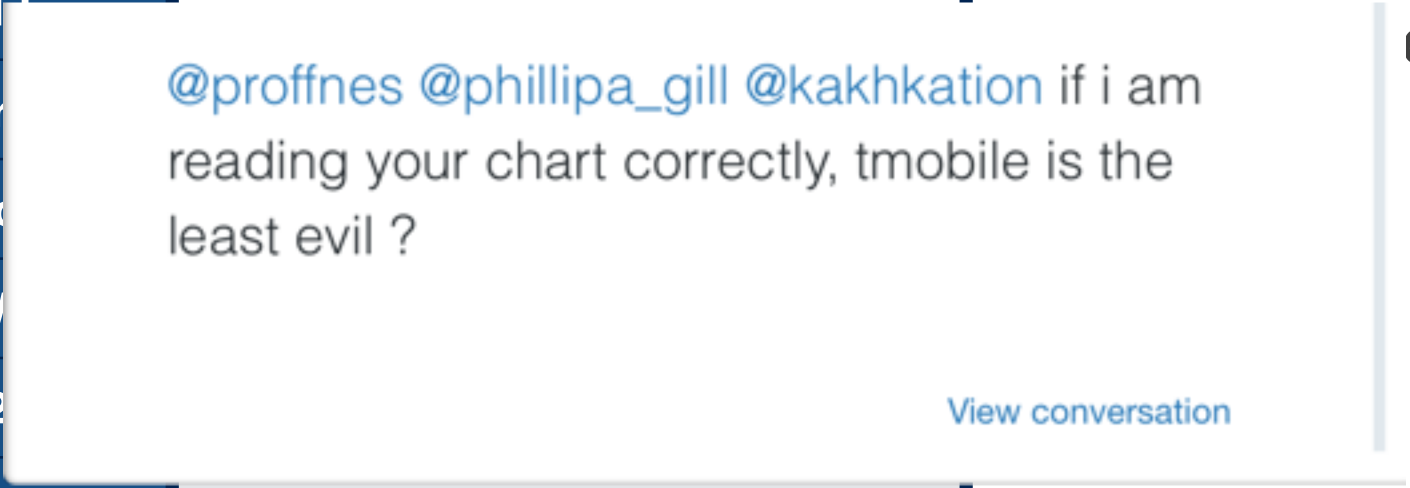
What are middleboxes doing (2015)?

ISP	YouTube	Netflix	Spotify
Verizon	m	m	m
Tmobile	-	-	-
ATT			
Spr			
Bo			
BlackW			
H2			
SimpleMobile	36%	-	-
NET10	p	p	p

m: content modified on the fly

p: translucent proxies change

ehavior



@proffnes @phillipa_gill @kakhkation if i am reading your chart correctly, tmobile is the least evil ?

How do they classify traffic?

DPI: It’s dumber than you think

What ***isn’t*** it looking at?

- IP addresses
- Traffic timings
- …

How do they classify traffic?

DPI: It’s dumber than you think

What ***isn’t*** it looking at?

- IP addresses
- Traffic timings
- …

What ***is*** it looking for?

- Specific keywords (or bytes)
- With limited understanding of deployed protocols

How do they classify traffic?

Header	Example Value	Example Application
URI	site.js{…}-**nbcsports**-com	NBC Sports
Host	Host: www**.spotify.**com	Spotify
User-Agent	User-Agent: **Pandora** 5.0{…}	Pandora
Content-Type	Content-Type: video/**quicktime**	QuickTime

What are unintentional consequences?

Header	Example Value
User-Agent	User-Agent: GalaxyWarsMultiplayer

What are unintentional consequences?

Header	Example Value	Example Application
User-Agent	User-Agent: GalaxyWarsMultiplayer	iPlayer

What are unintentional consequences?

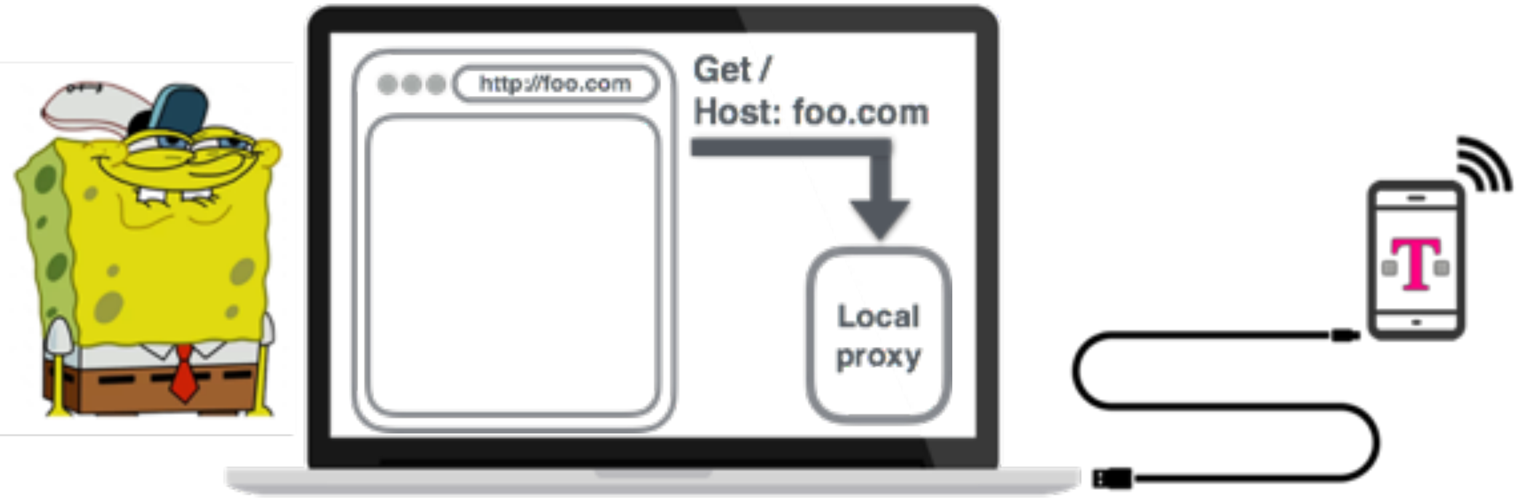
Header	Example Value	Example Application
User-Agent	User-Agent: GalaxyWarsMultiplayer	iPlayer

What are unintentional consequences?

Free riding on T-Mobile

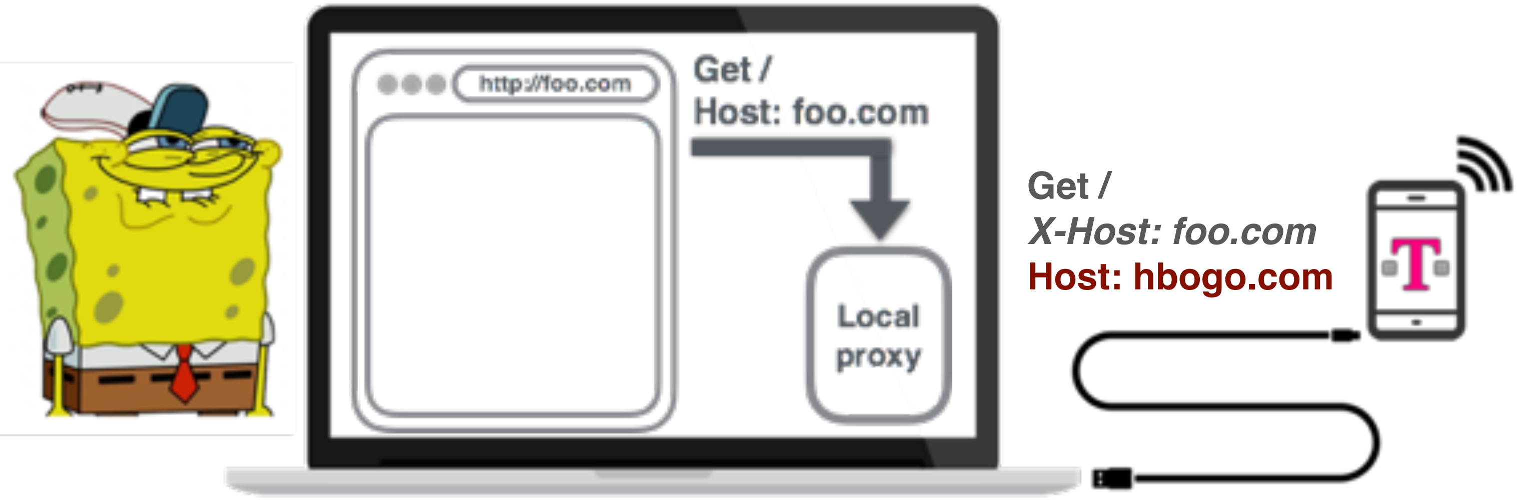
What are unintentional consequences?

Free riding on T-Mobile



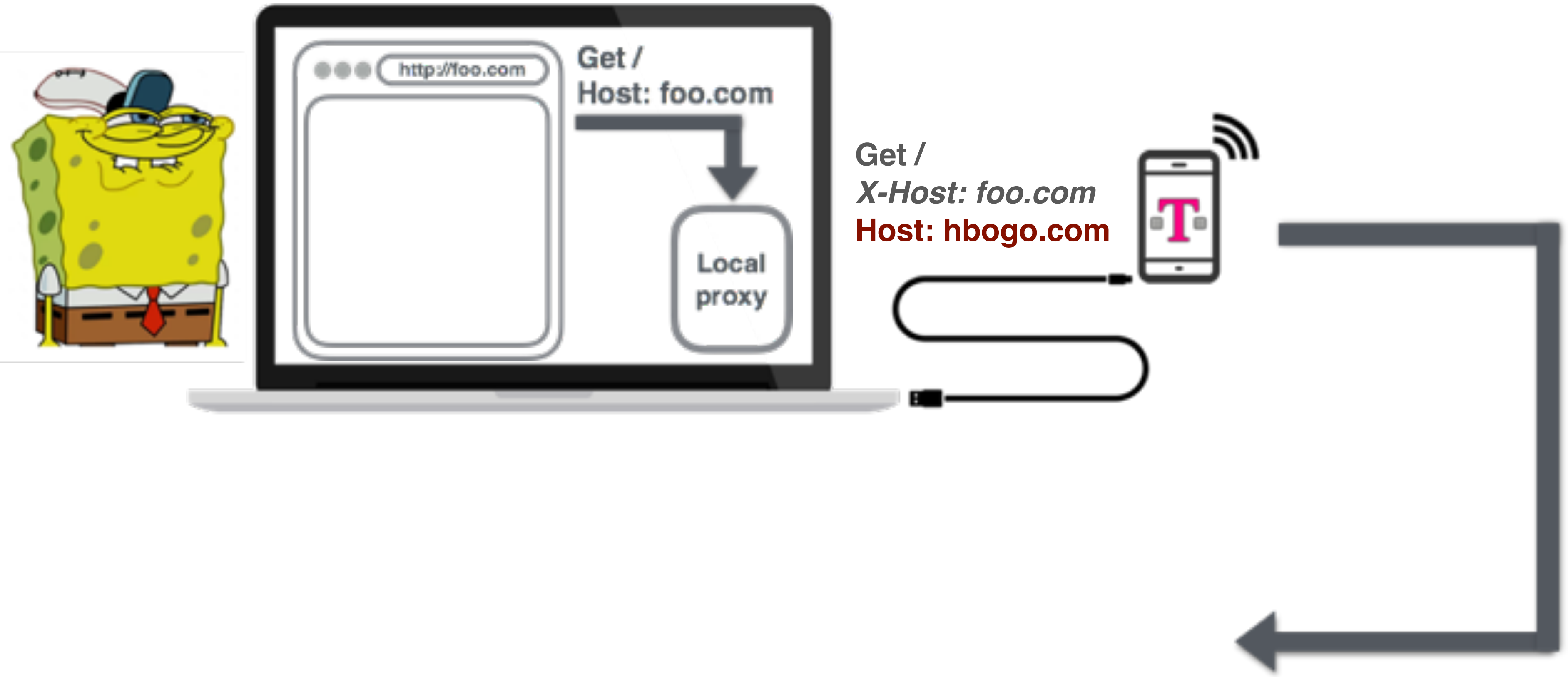
What are unintentional consequences?

Free riding on T-Mobile



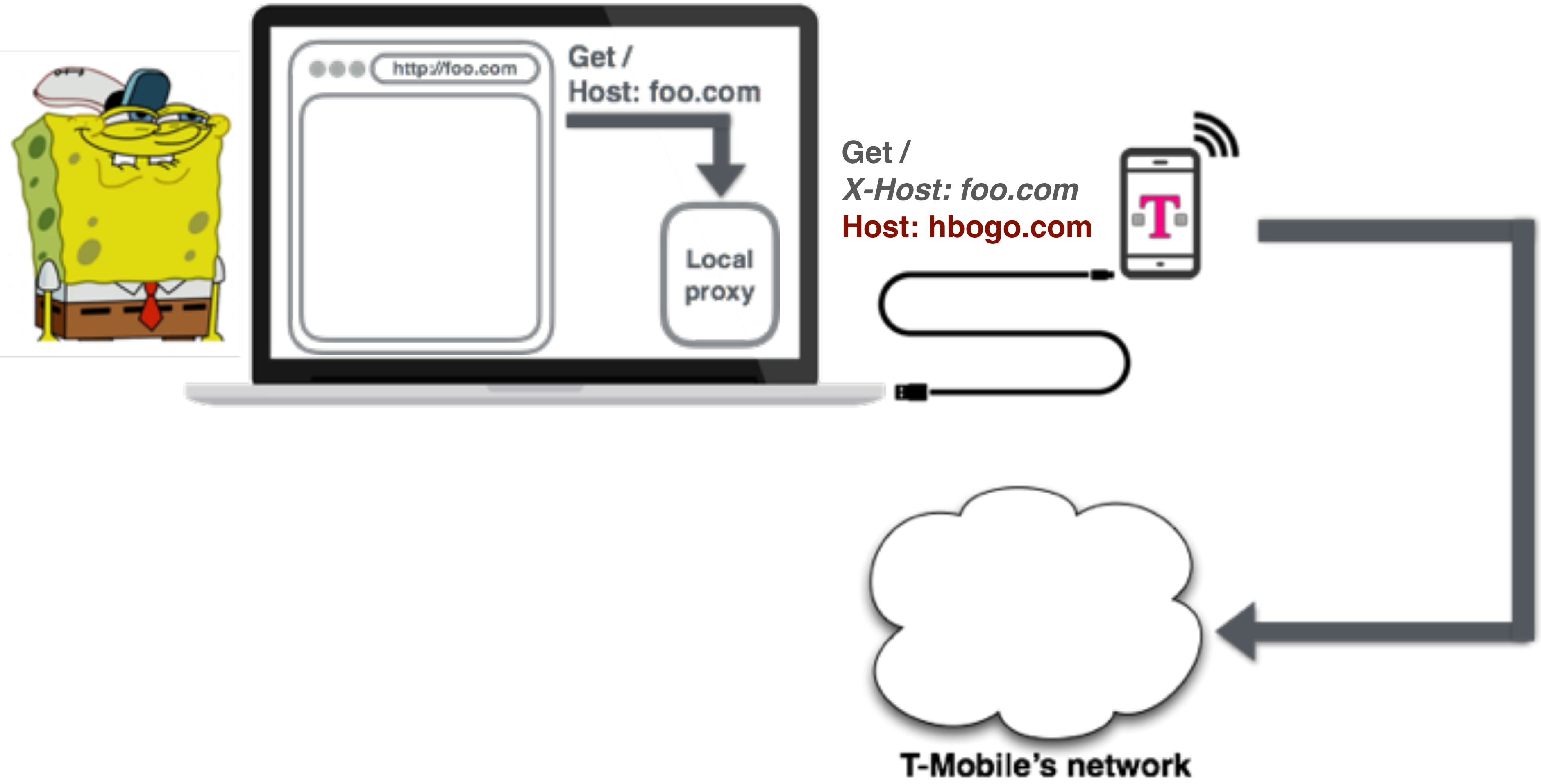
What are unintentional consequences?

Free riding on T-Mobile



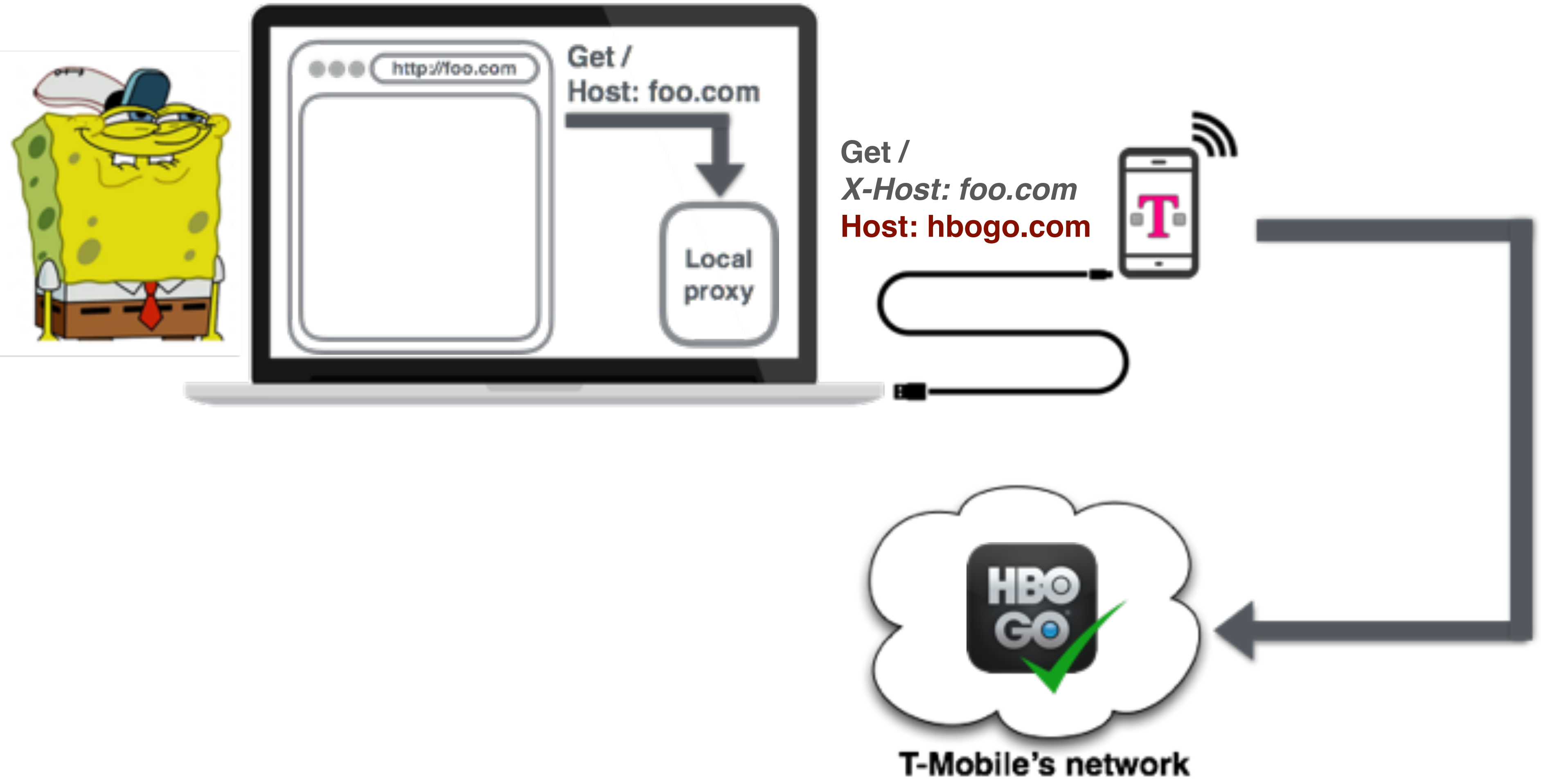
What are unintentional consequences?

Free riding on T-Mobile



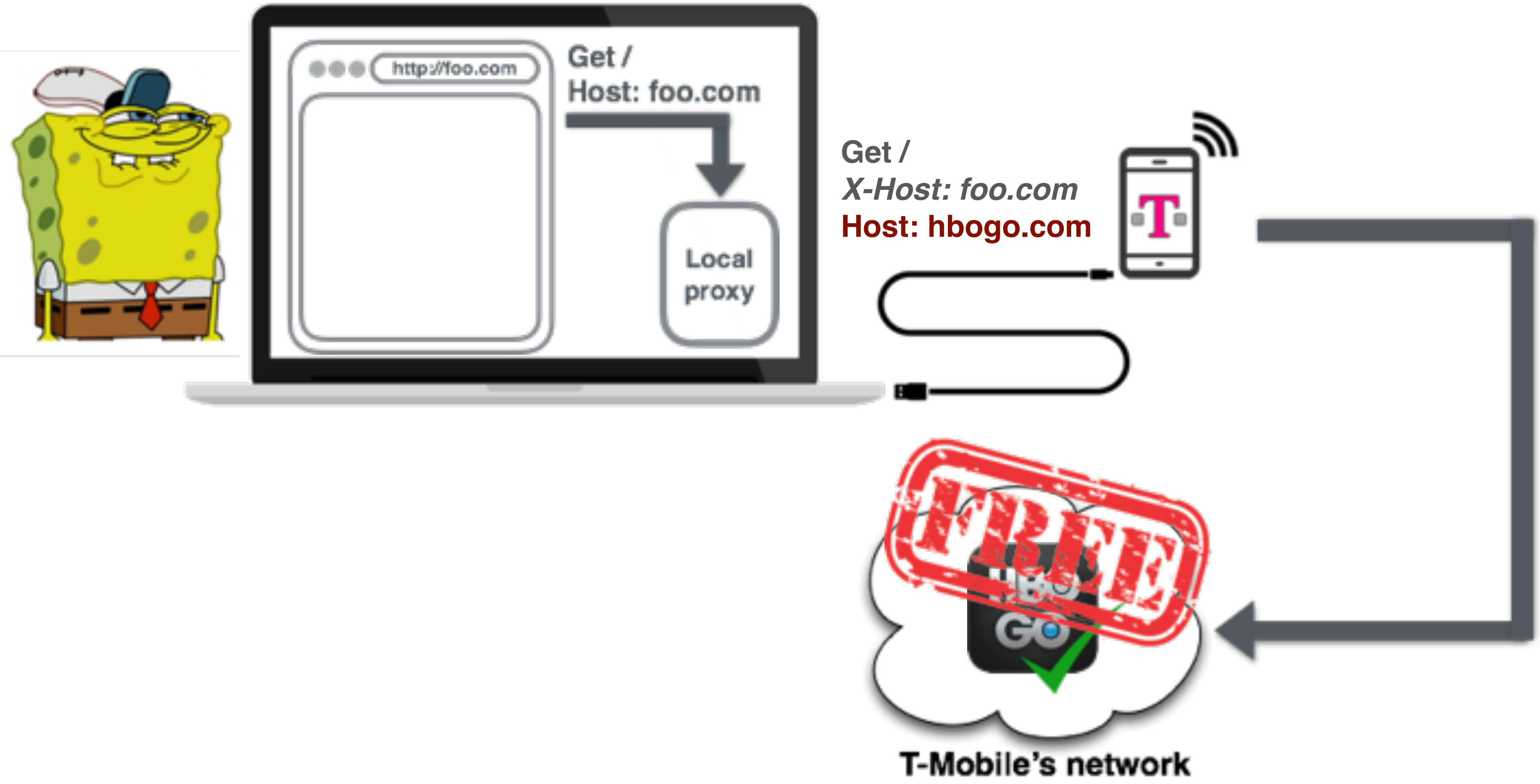
What are unintentional consequences?

Free riding on T-Mobile



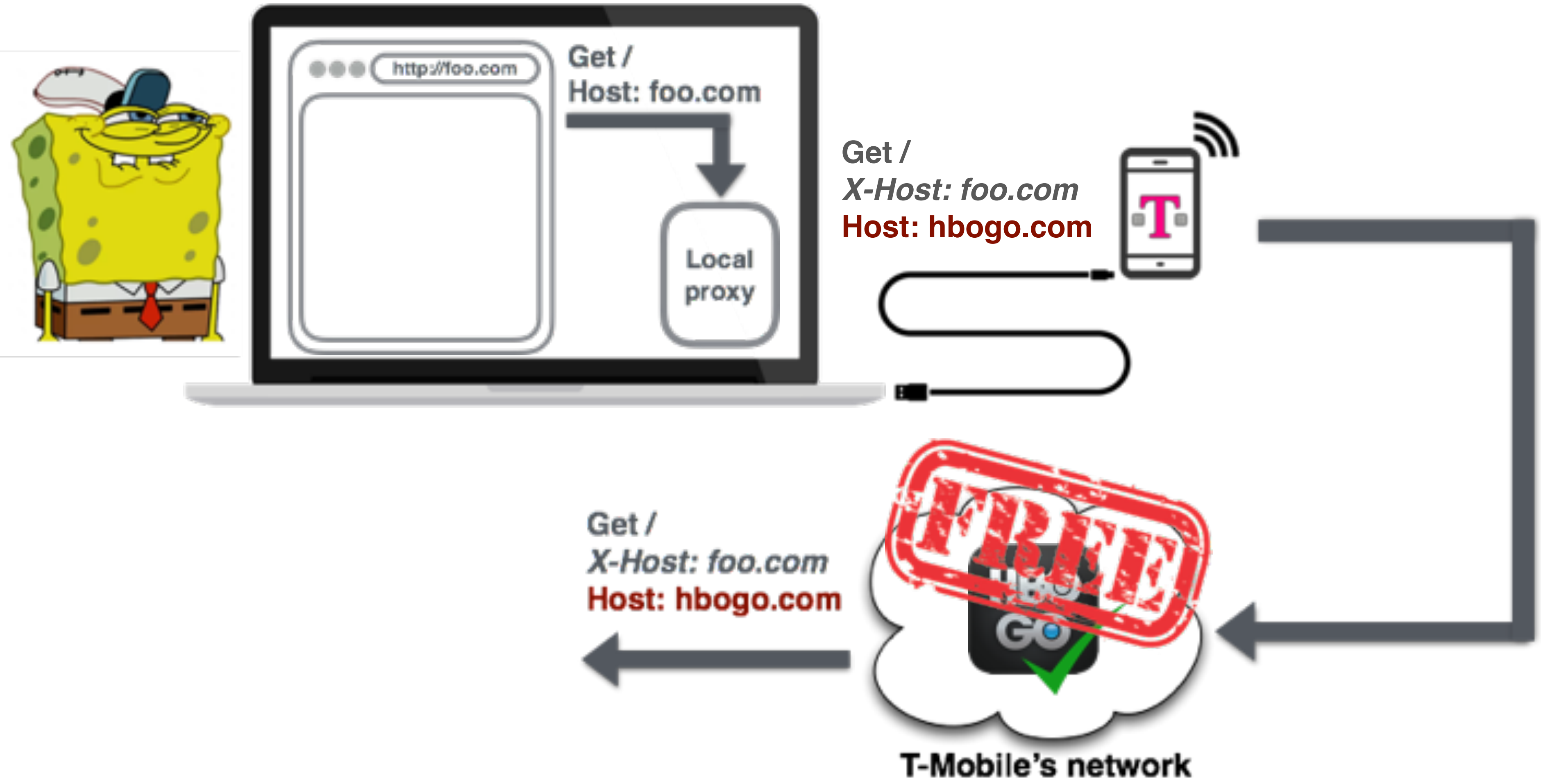
What are unintentional consequences?

Free riding on T-Mobile



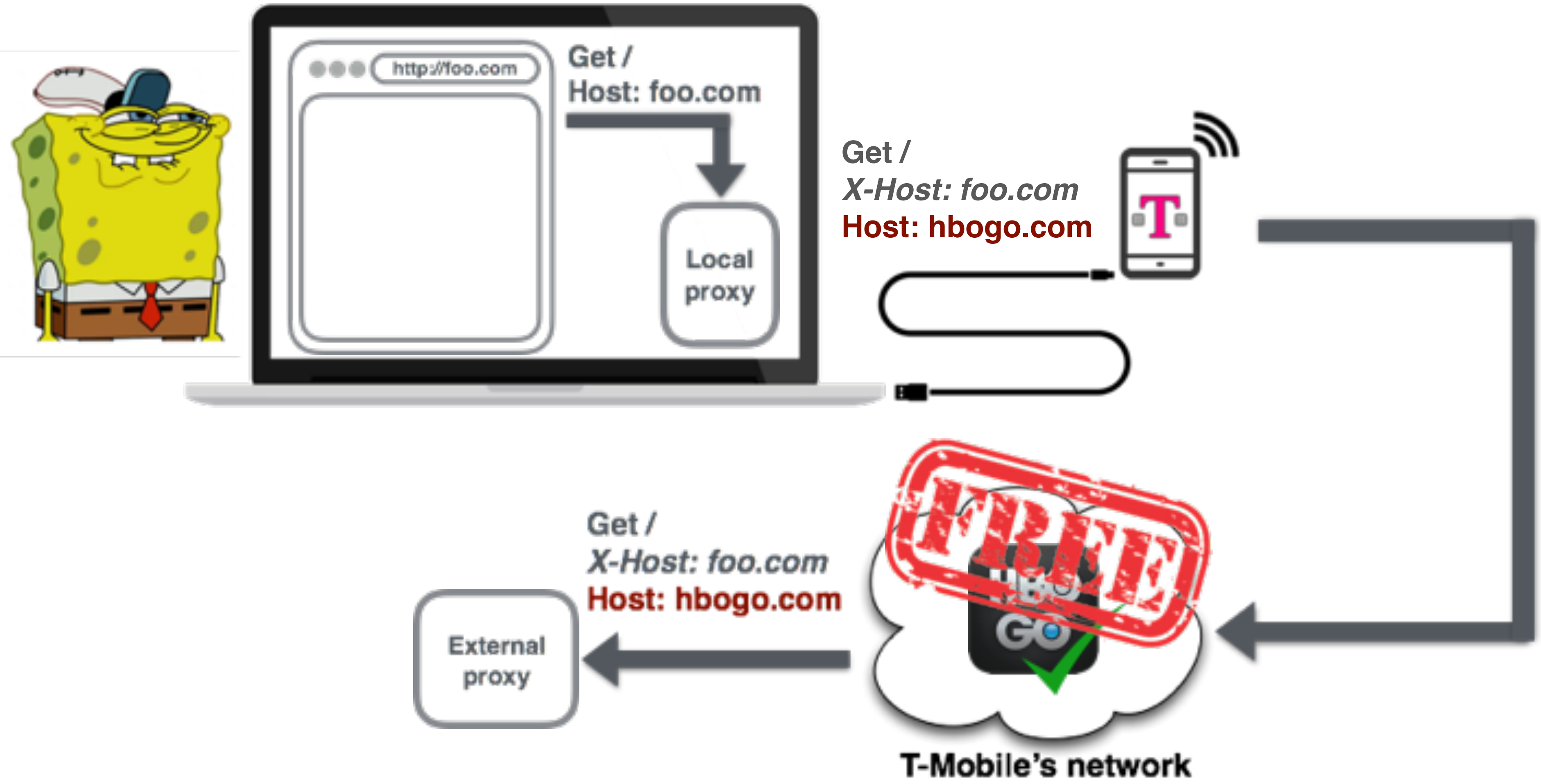
What are unintentional consequences?

Free riding on T-Mobile



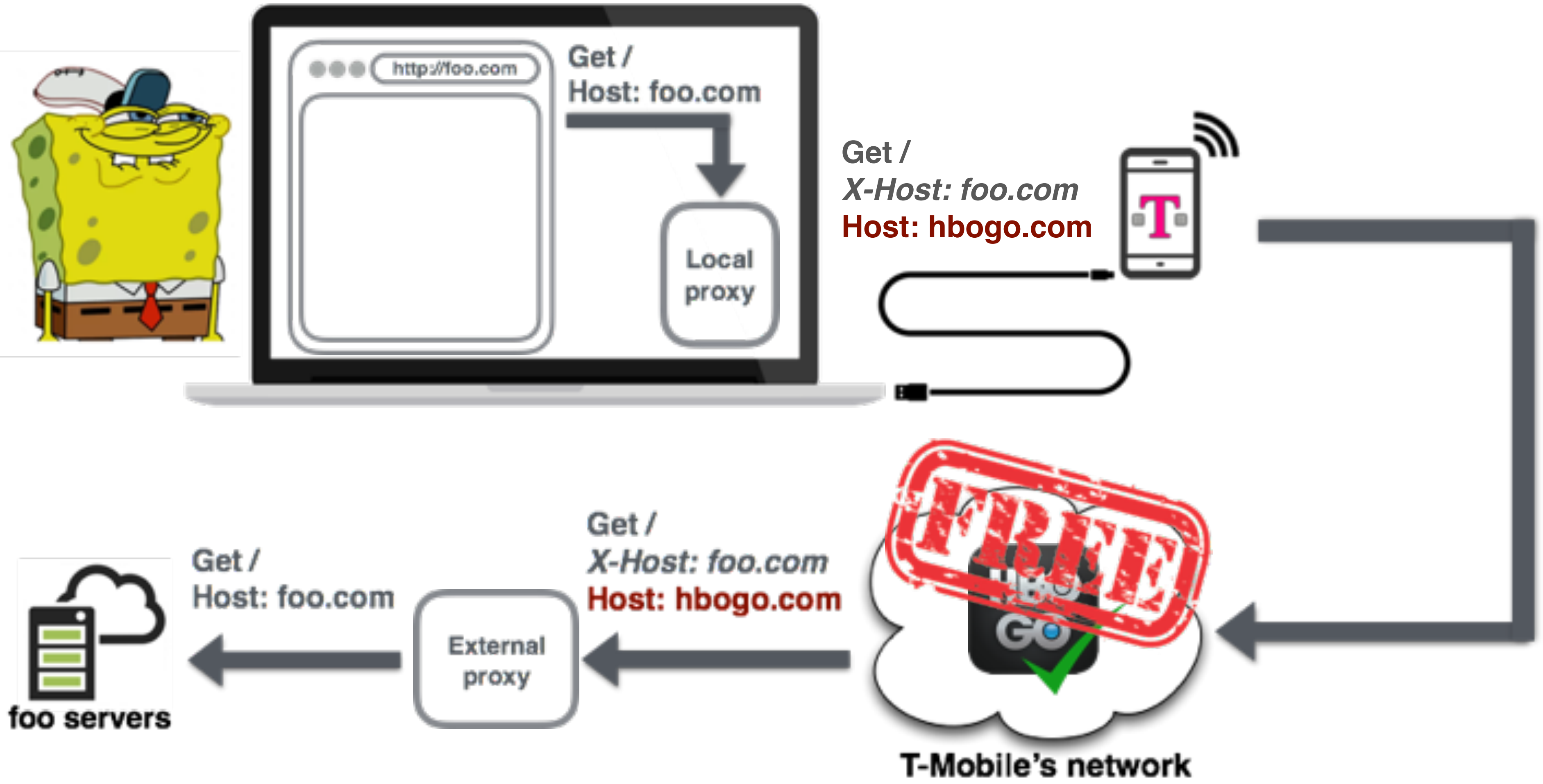
What are unintentional consequences?

Free riding on T-Mobile



What are unintentional consequences?

Free riding on T-Mobile



What can users do about this?

Axiom:
Middleboxes necessarily ***infer*** end-to-end state
using ***incomplete*** information

What can users do about this?

Axiom:
Middleboxes necessarily ***infer*** end-to-end state
using ***incomplete*** information

Hypothesis:
It is possible to ***systematically identify and violate assumptions*** used in inference, ***unilaterally*** at transport/network layer

What can users do about this?

Axiom:
Middleboxes necessarily ***infer*** end-to-end state using ***incomplete*** information

Hypothesis:
It is possible to ***systematically identify and violate assumptions*** used in inference, ***unilaterally*** at transport/network layer

Our approach:
Build a system that **automatically**, **efficiently** does this, to enable user control over impact of policies

- Evade censorship
- Select policies applied to traffic
- Overhead is ~ one header (10s of B) per flow, sometimes zero

Conclusion

Lack of **transparency** and **control** over network policies

Empirical, practical approach can recover these properties

- Reverse engineer middleboxes
- Identify policies and their implications
- Exploit invalid assumptions to regain control over policies

Testbed, datasets, results available

http://dd.meddle.mobi

What do I want

How do I engage with policy in an impactful way?

- You know, besides giving the FCC ombudsperson my reports, scheduling multiple phone calls with him, agreeing on there being potentially actionable issues, and having him forward to “the commission”

Who wants to help test networks for differentiation?

- We have an app, python clients
- We love to collaborate

Which networks should we test?

Who wants to use our testbed? What do you want?

…and of course any other feedback/questions from you