Internet Censorship

What is censorship?

Censorship, the suppression of words, images, or ideas that are "offensive," happens

whenever some people succeed in imposing their personal political or moral values on others.

Censorship can be carried out by the government as well as private pressure groups. Censorship by the government is unconstitutional.

– The American Civil Liberties Union

What is censorship?

- Key points:

- Censorship in general is a non-technical problem
  - Think banned books, suppression of news media etc.
- In the **United States** censorship is unconstitutional
  - Other countries?
  - Are we forcing Western values on other countries?
  - United Nations **Universal Declaration of Human Rights** provides some guidance of what speech should be protected globally
    - **E.g., political, minority religions, LGBT, etc.**

What is a network censor

- An entity that desires that some *identifiable* communication is *blocked* from being transmitted over the network
- *Without* the authority to compel the content provider to remove the content
- *Without* the authority to compel the client to install sof censor's choosing
- Requires that the censor act on *network traffic*

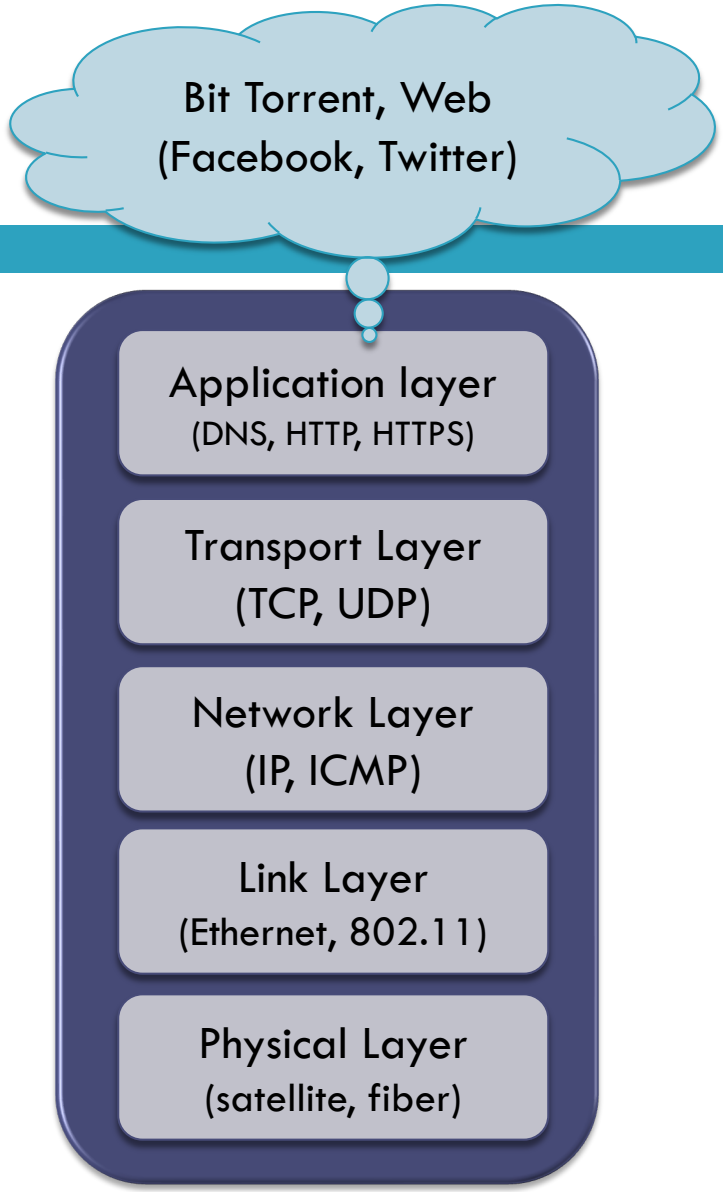
Image from Watch, Learn, Drive
http://watch-learn-drive.com/Learners_Online/New_places/Traffic_lights/TL_5.html

How to identify and block?

- Identification: The piece of information that allows the censor to identify content to be blocked is referred to as the *censorship trigger*
  - Example: IP address, hostname, URL, keywords etc.
- Blocking: The technical means used to restrict access to the content
  - Example: dropping packets, forging TCP RST packets or DNS responses
- In the next few slides we will be exploring censorship as it exploits different triggers and blocking mechanisms at different layers of the Internet Protocol stack.

Networking 101

- Protocols on the Internet divided into logical layers
- These layers work together to get traffic where it is going.
- Headers of upper layers encapsulate lower layer protocols
- A network censor can disrupt any layer!

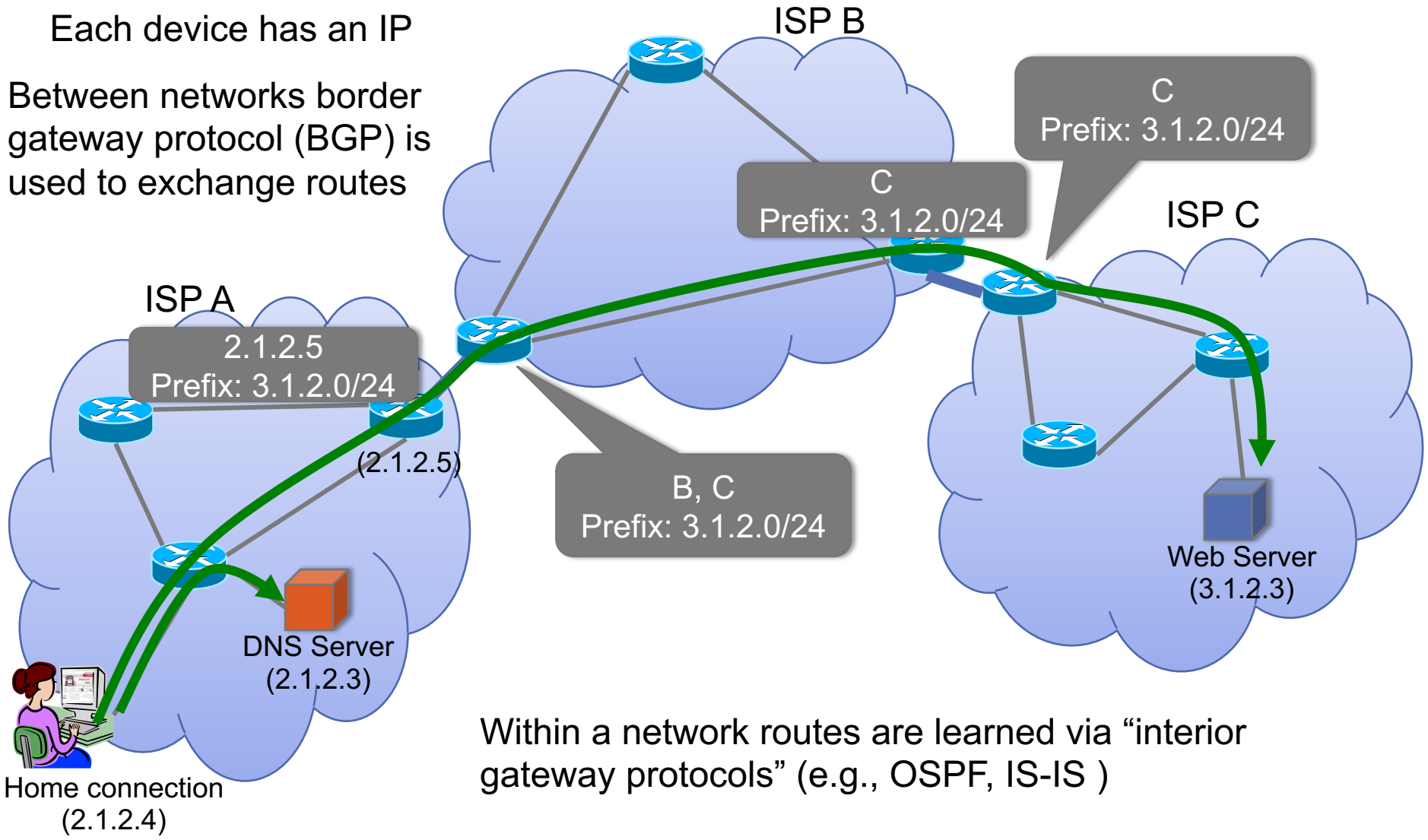


NETWORKING 101

So how does our traffic get where its going?

Each device has an IP

Between networks border gateway protocol (BGP) is used to exchange routes



Within a network routes are learned via “interior gateway protocols” (e.g., OSPF, IS-IS)

NETWORKING 101

…ok but humans don't request IP addr

… they want content!

HTTP STATUS 200
Content Length: 523
Content Type: text/html

```
<!DOCTYPE html>
<html lang="en" dir="ltr"
class="client-nojs">
<head>
<meta charset="UTF-8" /><title>Douglas
MacArthur - Wikipedia, the free
encyclopedia</title>
<meta name="generator"
content="MediaWiki 1.23wmf10" />
```

HTTP GET /wiki/Douglas_MacArthur HTTP 1.1
Host: en.wikipedia.org

ISP

54.238

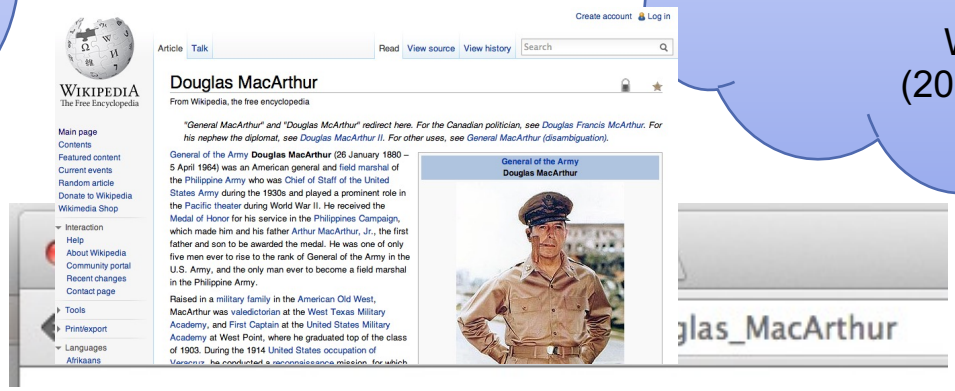
SYN

(2.1.2.5)

DNS Server
(2.1.2.3)

Web Server
(208.80.154.238)

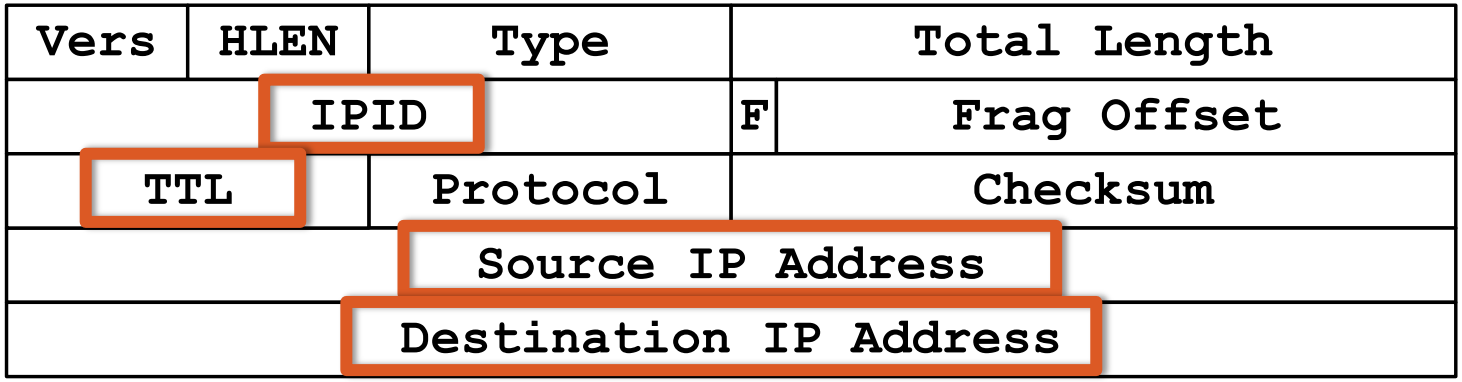
Home connection
(2.1.2.4)



MANY OPPORTUNITIES TO CENSOR

- **Block IP addresses**
  - IP layer
- **Block hostnames**
  - DNS (application layer)
- **Disrupt TCP flows**
  - TCP (transport layer)
  - Many possible triggers
- **Disrupt HTTP transfers**
  - HTTP (application layer)

INTERNET PROTOCOL 101



Relevant fields:

IPID: set by the sender of the IP packet. Some OSes increment globally for each IP packet generated by the host; some maintain per flow counters, use a constant or random values.

TTL: counter gets decremented by each hop on the path until it reaches 0 and an ICMP Time Exceeded Message is generated. Useful for probing/locating censors.

Source IP: IP of the sender of this packet

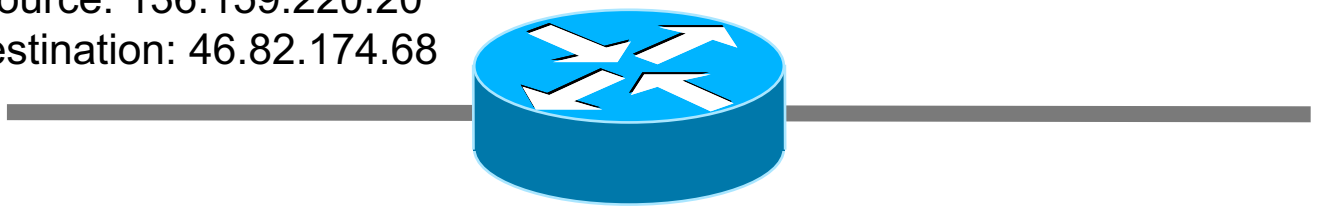
Destination IP: IP of the recipient of this packet

IP-BASED BLOCKING

Option 1: Configure routers using an access control list (ACL) to drop traffic to a given IP address.

Source: 136.159.220.20
Destination: 46.82.174.68

This is an example of in-path blocking (censor can **remove** packets)



Drop traffic to:
8.7.198.45
203.98.7.65
46.82.174.68
59.24.3.173
93.46.8.89

Image from Watch, Learn, Drive
http://watch-learn-drive.com/Learners_Online/New_places/Traffic_lights/TL_5.html

IP-BASED BLOCKING

Option 1: Configure routers using an access control list (ACL) to drop traffic to a given IP address.

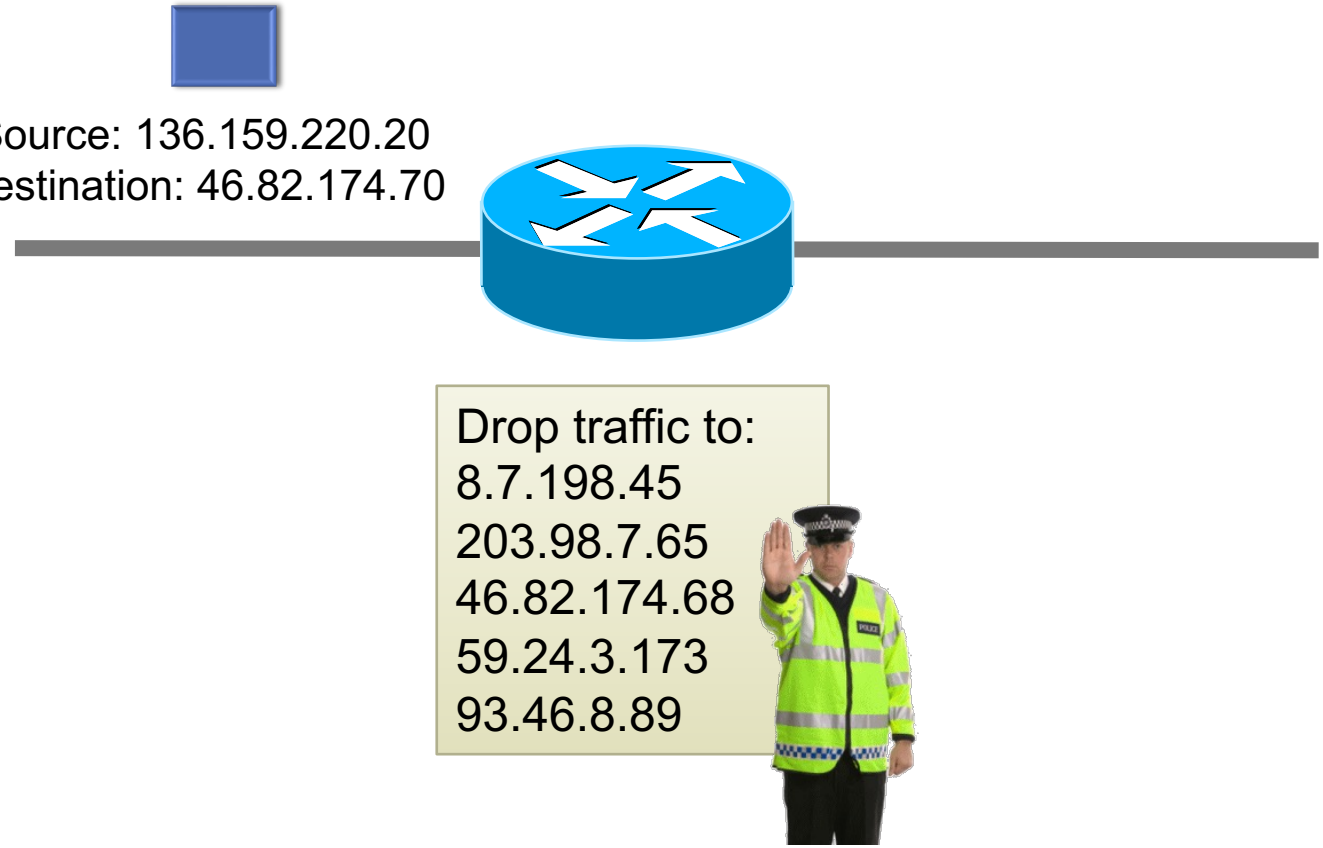


Image from Watch, Learn, Drive
http://watch-learn-drive.com/Learners_Online/New_places/Traffic_lights/TL_5.html

IP-BASED BLOCKING

- **Advantages (for the censor)**
  - Quick and easy to configure
  - Routers have efficient techniques for IP matching
- **Disadvantages**
  - Need to know the IP
    - Easily evadable!
  - High collateral damage: IP != Web host
    - Noticeable if high profile site is hosted on the same system
    - 60% of Web servers are hosted with 10,000 or more other Web servers (Shue et al. 2007)
  - Location of the censor can be determined from within the censored network
    - Just need to **traceroute** to the blocked IP (use TCP port 80 SYNs in case ACL is selective).
  - Can determine location from censored host as well
    - Assuming ICMP Time Expired messages are blocked.

IP-BASED BLOCKING

Option 2: Use BGP to block IPs

Februa

Tube

YouTube

an

om

Multinet Pakistan

Corrigendum- Most Urgent

GOVERNMENT OF PAKISTAN
PAKISTAN TELECOMMUNICATION AUTHORITY
ZONAL OFFICE PESHAWAR
Plot-11, Sector A-3, Phase-V, Hayatabad, Peshawar.
Ph: 091-9217279- 5829177 Fax: 091-9217254
www.pta.gov.pk

NWFP-33-16 (BW)/06/PTA February ,2008

Subject: **Blocking of Offensive Website**

Reference: This office letter of even number dated 22.02.2008.

I am directed to request all ISPs to immediately block access to the following website

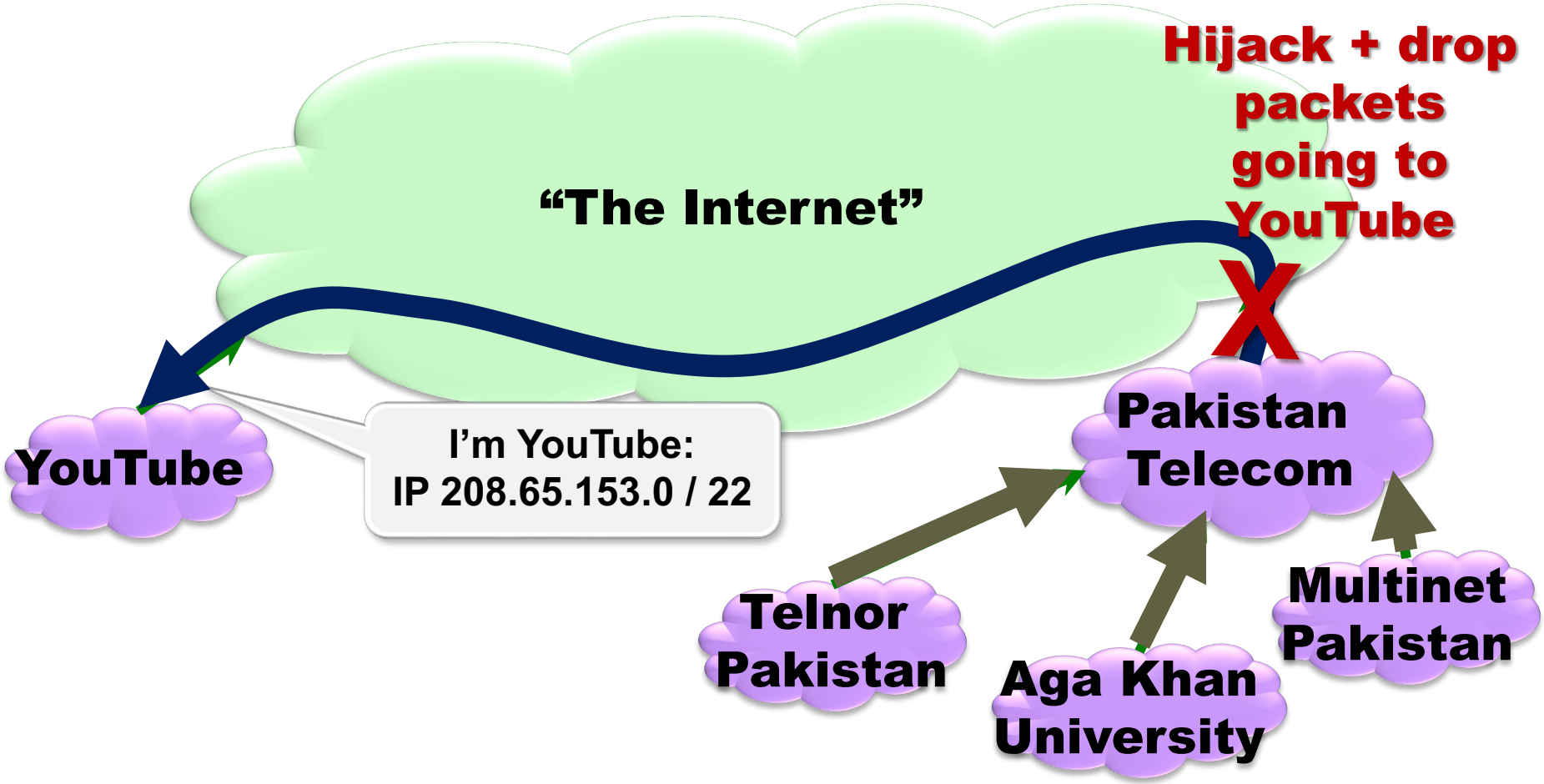
URL: http://www.youtube.com/watch?v=o3s8jtvvg00

IPs: 208.65.153.238, 208.65.153.253, 208.65.153.251

Compliance report should reach this office through return fax or at email peshawar@pta.gov.pk today please.

IP-BASED BLOCKING

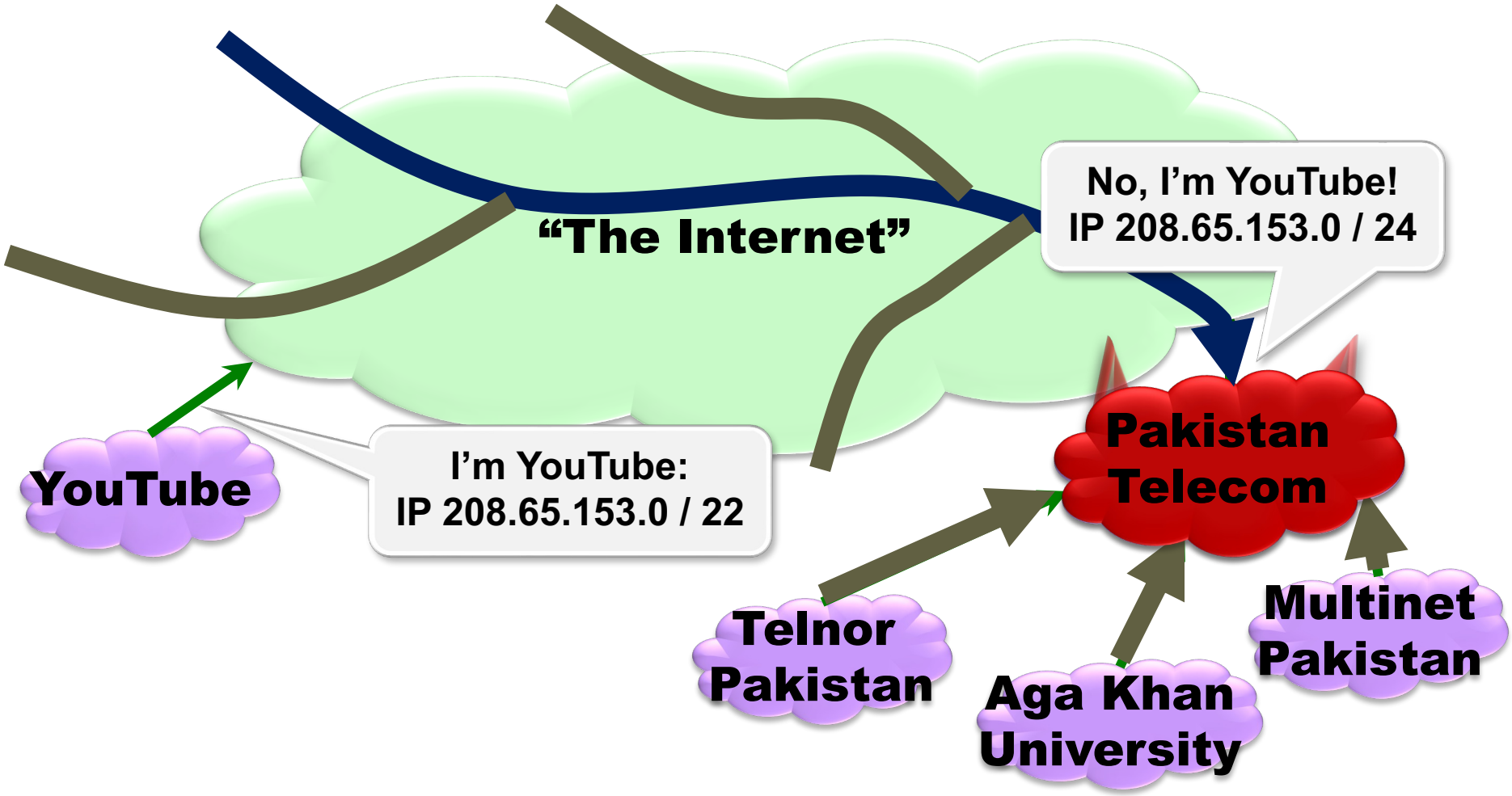
Here's what should have happened....



Block your own customers.

IP-BASED BLOCKING

But here's what Pakistan ended up doing...



WHY WAS THE PAKISTAN INCIDENT SO BAD?

- **They announced a *more specific* prefix**
  - BGP routing is based on longest prefix match
- **There is no global route authentication in place!**
  - ISPs ***should*** filter announcements from their customers that are clearly wrong
  - (As an ISP you should know what IP address space is in use by your customers)
  - In reality this is harder than it seems ☹

IP-BASED BLOCKING

Option 2: BGP route poisoning

- **Instead of configuring router ACLs, just advertise a bogus route**
  - Causes routers close to the censor to route traffic to the censor, which just drops the traffic
- **How to detect this type of censorship?**
  - BGP looking glass servers in the impacted region
  - Sometimes global monitors as well …
- **Challenges**
  - Can cause **international** collateral damage!
  - Will block all content on a given prefix
    - Could announce a /32 to get a single address but most ISPs will not propagate beyond a /24

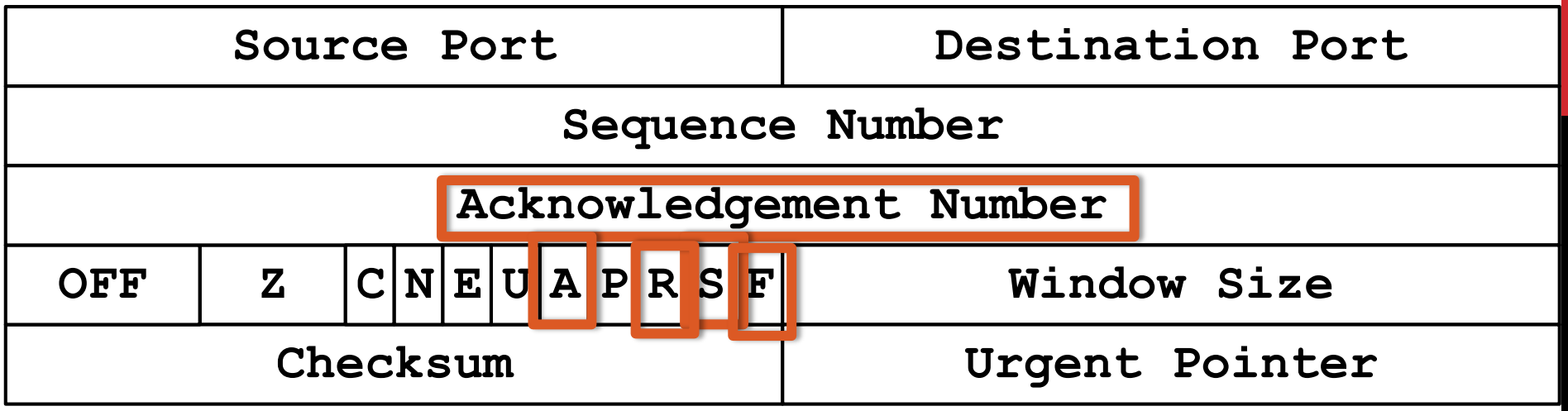
KNOWN USERS OF IP-BASED BLOCKING

- **Pakistan using IP-based blocking for YouTube address ranges**
  - Can interfere with other Google services
- **China**
  - Some reports of IP blocking
  - Many URLs redirected to small set of IP-addresses, possibly this is the set used for ACLs
- **UK**
  - Uses IP blocking of the Pirate Bay's IP address
- **Australia**
  - IP blocking for Melbourne Free University IPs (precise motivation unclear…)
  - https://www.eff.org/deeplinks/2013/04/australian-networks-censor-community-education-site
- **In general, too much collateral damage of IP-based blocking.**

OVERVIEW

- **Block IP addresses**
  - IP layer
- **Disrupt TCP flows**
  - TCP (transport layer)
  - Many possible triggers
- **Block hostnames**
  - DNS (application layer)
- **Disrupt HTTP transfers**
  - HTTP (application layer)

TCP: TRANSMISSION CONTROL PROTOCOL



TCP is used for reliable, in-order communication

- **Connection established using a "three-way handshake"**
- **All data is ACKnowledged**
  - If no ACK is received packets will be resent
- **Connection *normally* closed with a FIN (finish) packet**
  - Indicates that **this side** has no more information to send
- **Connections can also be closed with a RST (reset) packet**
  - Indicates a problem: **both sides** should stop communicating
  - Some software makes liberal use of RSTs.

WHY INJECT TCP RESET PACKETS?

- **A TCP Reset (RST) tells the other side of the connection:**
  - There will be **no more data from this source** on this connection
  - This source will not accept any more data, **so no more data should be sent**
- **Once a side has decided to abort the connection, the only subsequent packets sent on this connection may be RSTs in response to data.**
- **Once a side accepts a RST it will treat the connection as aborted**
- **… but RSTs are quite common, 10-15% of ALL flows are terminated by a RST rather than a FIN**
  - For HTTP, it can be over 20%: Web servers/browsers often time out with RST instead of FIN
  - Thus we cannot treat RSTs as “adversarial”

TYPES OF CENSORS

- **Last time we discussed IP blocking via ACLs which is an example of an *in-path* censor.**
- **Censors can also operate *on-path*: a wiretap, (intrusion detection system (IDS), deep packet inspection (DPI)) + attached network connection**
  - Censor can see **all** the packets
  - Censor can **add** their own packets through **packet injection**
  - Censor **cannot** remove packets
- **Can censor:**
  - DNS requests (by injecting bogus replies)
  - Web requests to given hosts (including HTTPS)
  - Web requests over HTTP for forbidden content
    - Latter two possible via injecting **TCP RST packets!**

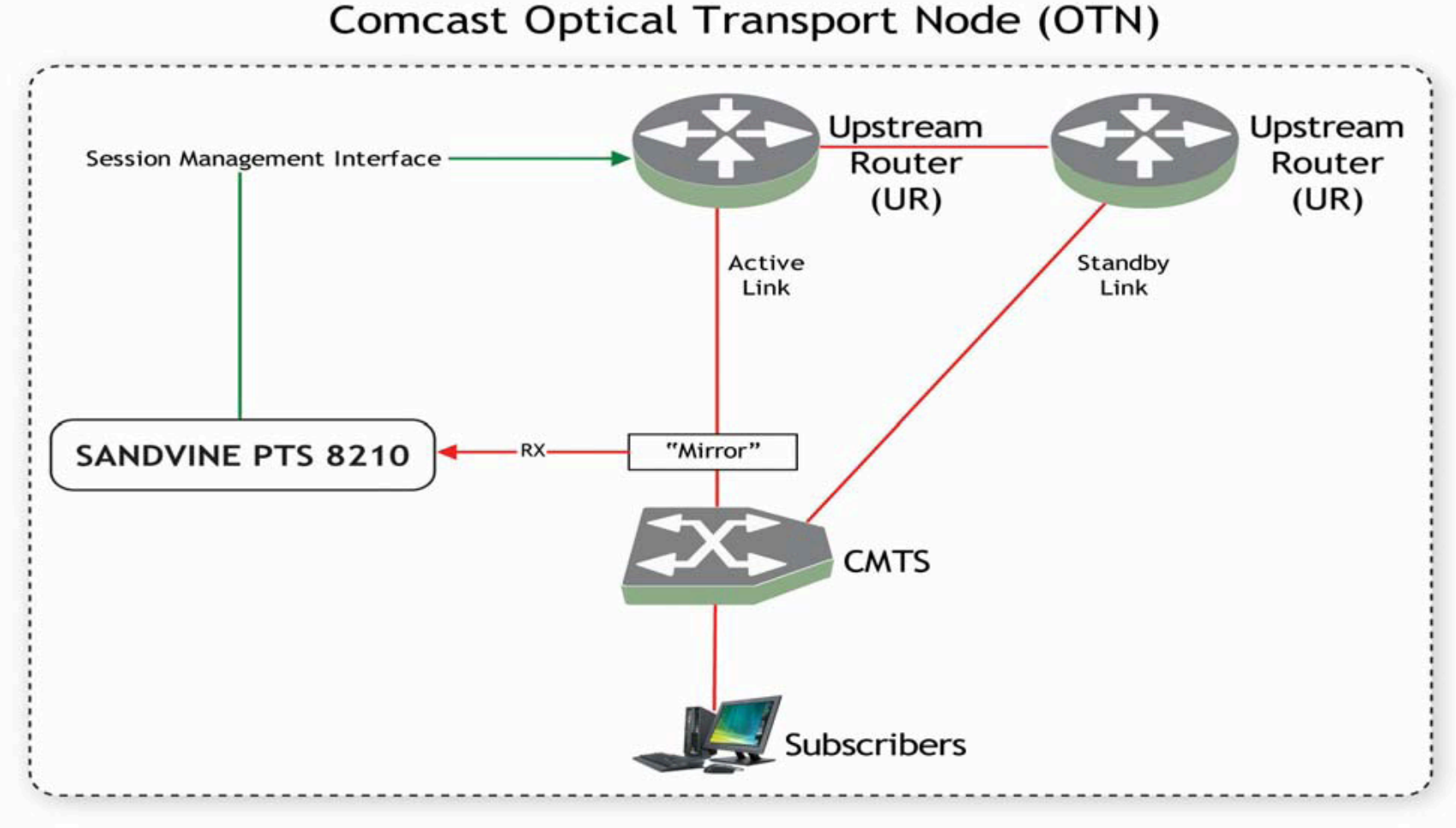
LIMITATIONS OF ON-PATH CENSORS

- On-path censorship can't censor the ***last*** message in a flow
  - ➡ Since by the time the censor decides to block it, the message has passed
    - ➡ Thus can't censor DNS replies, only requests
    - ➡ And many web replies as well
  - ➡ But ***can*** block HTTPs sites based on certificates
    - ➡ The certificate handshake process gives enough time to block actual data
- Often limited to simple blocking, not a notice page/ image replacement
  - ➡ Often a censor ***wants*** the user to know (s)he was censored
- And if it can't keep up, censored material can get through
  - ➡ Which ***may*** be unacceptable

WHY ON-PATH CENSORS?

- **In-path device must process the traffic**
  - If they fail, they fail closed (connection gone!)
- **On-path devices are safer**
  - Tapping a link is “safe” (in network operator terms)
  - Easy to parallelize (just mirror traffic to more filters)
  - Less disruptive to install and use
- **Limitations:**
  - Can’t censor single replies
  - Censorship is **always detectable**
    - Censor cannot perfectly mimic the other endpoint.

ON PATH CENSOR EXAMPLE



DETECTING ON-PATH CENSORSHIP

Not only is the act of censorship detectable, the *mechanism*, is detectable

- **Since censor creates new packets but can't remove existing packets**
- **Since the injected packets can be identified, fingerprinting is also possible.**

Using packets which trigger censorship but with a short TTL can *localize* the censor in the network

- **Leads to tricky cross-layer network measurements (easier with DNS)**

Detection limitation: Can only detect an on-path censor when it is active

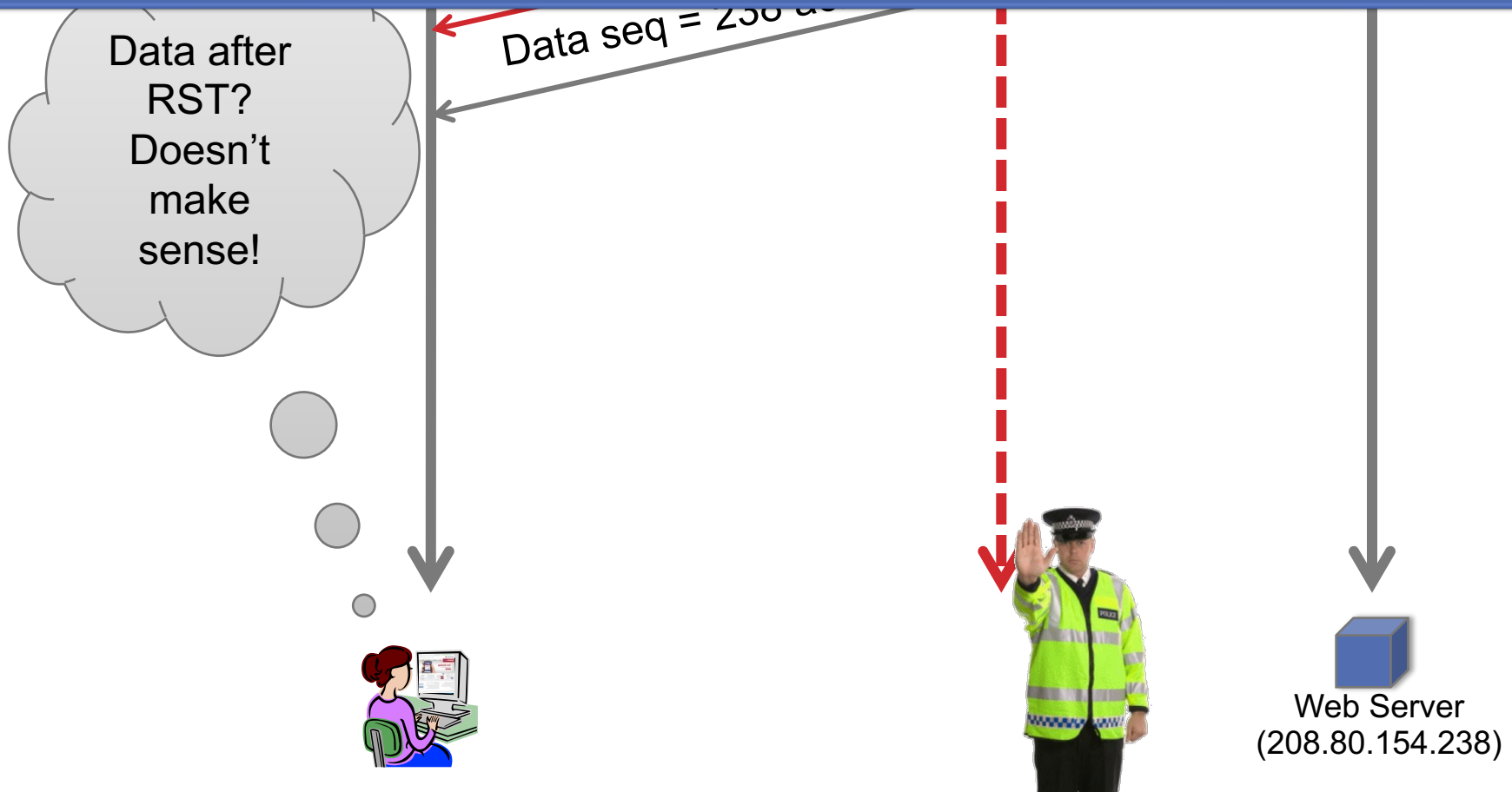
- **A censor which doesn't create an effect on measured traffic is not detectable**
- **E.g., DPI used for surveillance**

RACE CONDITIONS: DATA AFTER RESET

- **TCP packets are tracked by sequence numbers**
  - The next packet's sequence number should be the previous packet's sequence number plus the packet length
- **What is a sender is still sending data when the RST is injected?**
  - The receiver will see both a reset and a subsequent data packet, where the packet's sequence number + length > the reset packet's sequence number

RACE CONDITIONS: DATA AFTER RESET

Such a packet arrangement is out of specification
No TCP stack should generate such a sequence! It would imply that the stack decided to abort the connection yet keep sending anyway

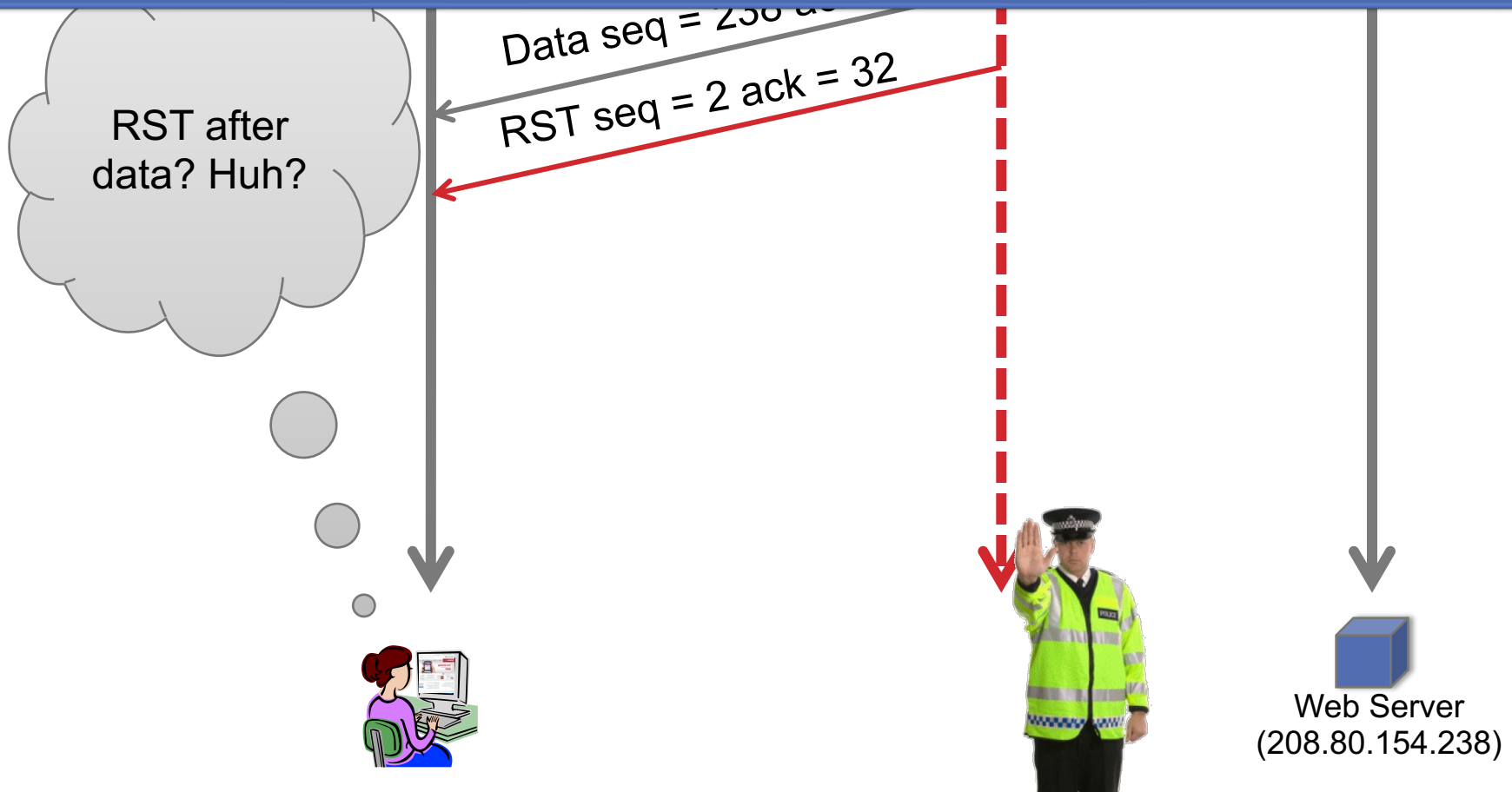


RACE CONDITIONS: RESET AFTER DATA

- **What if the reset injector is just slow?**
  - It takes time to determine that a flow should be blocked…
  - … in the mean time traffic is flying by!
- **Result is a reset after data race condition**
  - Reset packet appears after the data packet
  - Reset's sequence number is less than the data packet's sequence number plus its length

RACE CONDITIONS: RESET AFTER DATA

This is also out of specification
Why would a TCP stack do a retroactive abort?
Worse, such resets **should be ignored** by the receiver:
The received reset is “out-of-window”

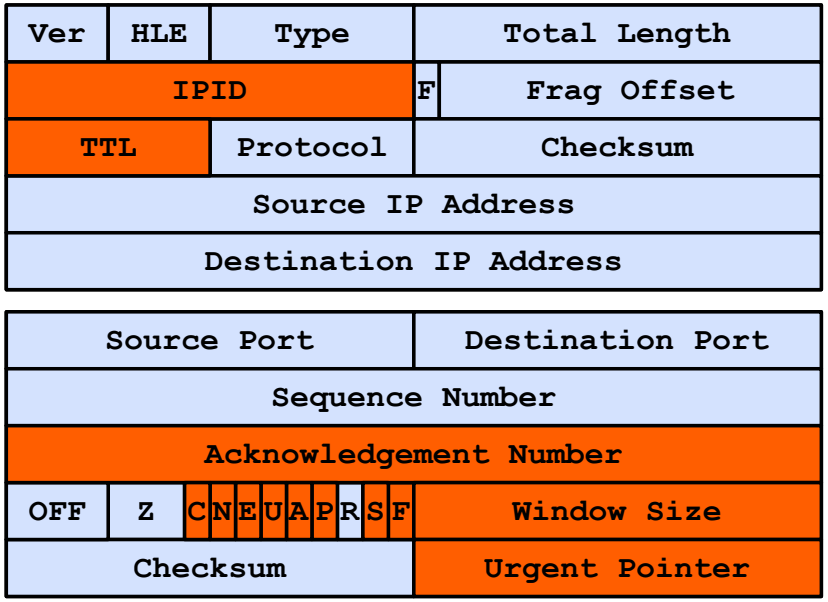


BUILDING A RELIABLE RST INJECTOR ENABLES DETECTION

- **Thus a reliable packet injector must anticipate the reset after data condition**
  - Instead of sending one reset it needs to send multiple resets with increasing sequence number
- **This is detectable as a "reset sequence change condition"**
  - An end host should never generate such resets as the host can always generate an in-sequence reset
- **An unreliable injector can only be detected when a race condition occurs**
- **A *reliable* injector *always* can be detected.**

FINGERPRINTING RST INJECTORS

- Injectors have significant freedom when constructing packets
  - ➡ Only some fields are checked when a Reset packet is received



- Correlations in a database of injectors allowed the creation of fingerprints of injectors and identifying common sources for injected packets
  - ➡ TTL, IPID, ACK value, and additional flags

CAN WE JUST IGNORE THESE RSTS?

- **As of 2006, yes *but* both ends of the connection need to ignore the RSTs.**

- **Client cannot do it unilaterally.**
  - Injectors will just send RSTs to the server and the client

REMEMBER … RSTS ARE A MECHANISM

They don't tell us anything about what *triggers* the mechanism

- **Some clues ..**
  - When the RST is sent
    - Before the HTTP GET
    - After the HTTP GET
  - Still not definitive
- **Need purpose build experiments**
  - Run tests towards your own server
  - Put blocked keyword in host name
  - … in HTML body content

OVERVIEW

- **Block IP addresses**
  - IP layer
- **Disrupt TCP flows**
  - TCP (transport layer)
  - Many possible triggers
- **Block hostnames**
  - DNS (application layer)
- **Disrupt HTTP transfers**
  - HTTP (application layer)

DOMAIN NAME SYSTEM (DNS)

- The rest of the network runs on IP addresses
  - ➡ But people run on ***names***
- DNS: the Domain Name System
  - ➡ A hierarchical, distributed database used to convert names to addresses
- Three major pieces
  - ➡ The Client's "Stub Resolver"
  - ➡ The ISP's (or third party) "Recursive Resolver"
  - ➡ The Authority Servers

HOW CAN WE BLOCK DNS?

A few things to keep in mind …

- **No cryptographic integrity of DNS messages**
  - DNSSEC proposed but not widely implemented
- **Caching of replies means leakage of bad DNS data can persist**

BLOCKING DNS NAMES

- Can the censor pressure the registrar?

➡ Name blocked, forever

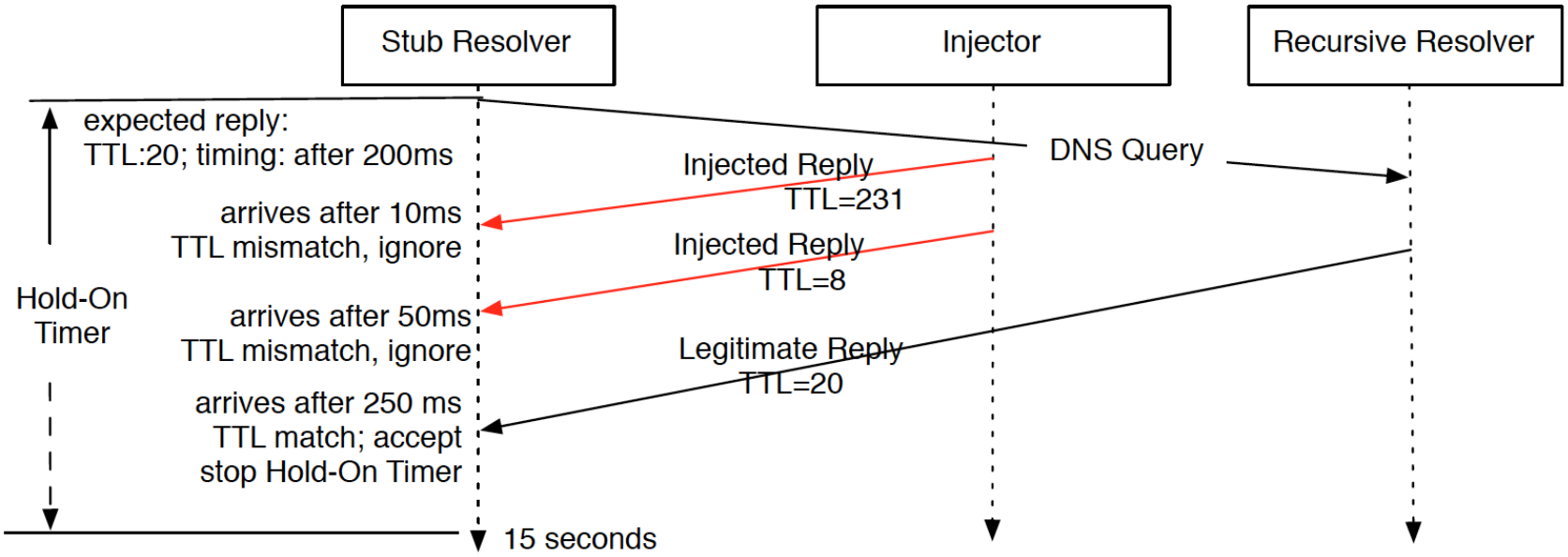
BLOCKING DNS NAMES

- Can the censor pressure the ISPs?
  - ➡ Just force an entry in the recursive resolver to poison results...
- Clients can trivially evade by using alternate DNS services
  - ➡ But this does require both client changes
  - ➡ And ISPs must not block third-party DNS queries
    - ➡ Collateral damage if you do so, so generally not a good idea
- Initially used by ISPs in the UK to block The Pirate Bay...

BLOCKING DNS NAMES

- **Option A: get ISP resolver on board**
  - (Previous slide)
- **Option B: On-path packet injection similar to TCP Resets**
  - Can be mostly countered with DNS-hold-open:
    - Don’t take the first answer but instead wait for up to a second
  - Generally reliable when using an out of country recursive resolve
    - E.g., 8.8.8.8
  - Can be **completely** countered by DNS-hold-open + DNSSEC
    - Accept the first DNS reply **which validates**

HOLD-ON IN ACTION



OVERVIEW

- **Block IP addresses**
  - IP layer
- **Disrupt TCP flows**
  - TCP (transport layer)
  - Many possible triggers
- **Block hostnames**
  - DNS (application layer)
- **Disrupt HTTP transfers**
  - HTTP (application layer)

NETWORKING 101: HTTP

- **HTTP (Hyper Text Transfer Protocol) is what most people think of when they talk about "the web"**
- **Client-server request/response protocol**
  - Client requests "I want file X from host Y that is on this server"
  - Server replies
- **Content can be *any*** filetype
- E.g. "HyperText Markup Language" (HTML) pages
  - Embedded programs (JavaScript, Flash, etc) which run on the browser
- ***No*** cryptographic integrity

HTTPS ADDS ENCRYPTION

- **The TLS (Transport Layer Security) protocol**
  - Sits “between” TCP and HTTP
- **Uses cryptographic certificates to authenticate the server**
  - One of ~300 entities vouch for (or vouch for someone who vouches for) the server
    - Who do you trust? CNNIC? US DOD? Your browser trusts them...
    - These days, however, fake certs get noticed: Certificate pinning in Google Chrome, certificate observatories and notaries, etc...
- **Without a fake certificate, the data is cryptographically protected**
  - But *does not* protect the TCP control messages
  - And *does not* protect against traffic analysis: Certificate effectively asserts what is the hostname! Also watching dataflow can often infer content

OK … SO WHERE ARE WE NOW?

- **We've so far talked about a bunch of different blocking techniques**
  - Packet filtering/BGP manipulation
  - Injecting RSTs
  - Injecting DNS replies
- **Those can all be used to block HTTP (and other types of content)**
- **Our focus now: proxies and blocking mechanisms that act specifically on HTTP traffic.**

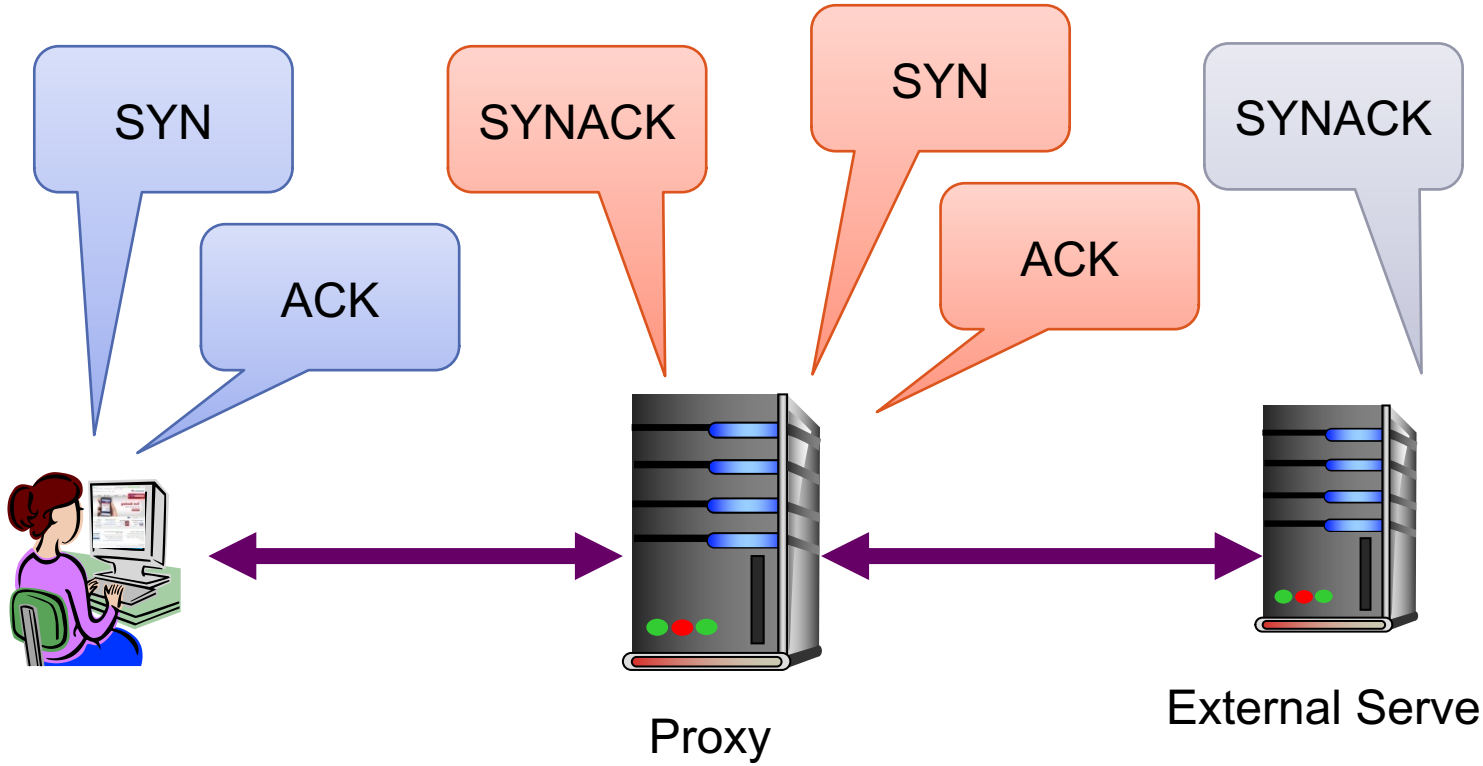
IN-PATH CENSORSHIP

- **Rather than sitting as a wiretap, actually intercept all traffic**
  - Now the censor can *remove* undesired packets
- Two possible mechanisms:
  - Flow Terminating
  - Flow Rewriting
- Two possible targets:
  - Partial Proxying
  - Complete Proxying

FLOW TERMINATING PROXIES

- The proxy sits in the middle and answers the TCP connection
  - ➡ Then creates a connection to the final destination
    - ➡ Either with the proxy's own IP address or the user's IP address
      - ➡ The former trivial to detect: do a web connection and a "non-web" connection: if the IPs are different, there is a proxy
- Common operating mode for most web-censor products
  - ➡ Easy to implement, reliable, and well understood
  - ➡ Standard option for tools like Squid etc...

FLOW TERMINATING



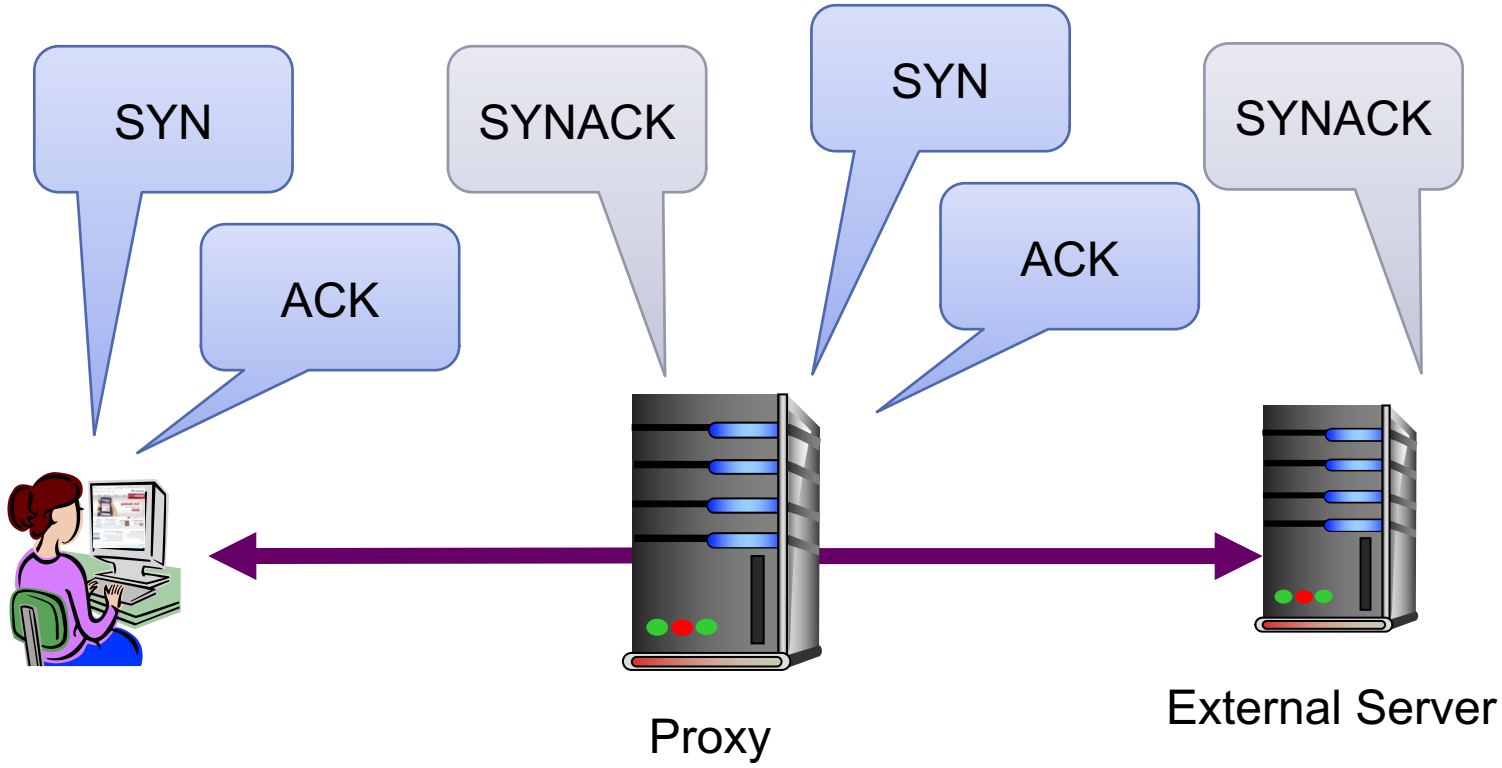
Two separate TCP connections.
Buys the censor some time to process content.
No worry about having to match state because the proxy is the end point (from client's point of view)

External Server might see client IP, might see Proxy IP

FLOW REWRITING PROXIES

- By default, simply pass packets
  - ➡ When objectionable content discovered, replace with something else
- Harder to implement:
  - ➡ Need to build custom TCP-manipulating software
- ***Very*** hard to detect:
  - ➡ Only the act of censorship itself is detectable
- I personally don't know of any products doing this for censorship rather than "intrusion prevention"
  - ➡ Only advantage over TCP-terminating proxies is detectability and ***perhaps*** performance, at the cost of significant complexity
    - ➡ Can someone who's tested Iranian censorship talk to me at the break?

FLOW REWRITING



PARTIAL VS. COMPLETE PROXYING

- Complete proxying:
  - ➡ All customer traffic passes through the proxy
  - ➡ Can also act as a web cache or content transcoder to improve performance
    - ➡ Assuming, that is, that the cache is implemented correctly
- Partial proxying:
  - ➡ Targeted IPs are routed to the proxy
    - ➡ E.g. by router ACLs or BGP
  - ➡ Traffic for those IPs passes through the proxy
    - ➡ Far safer: proxy failures don't cause overall failures
    - ➡ Far more load-tolerant: complete proxies take a lot of compute
  - ➡ But limited in scope:
    - ➡ Useful for targeting content when the sites are known

DETECTING AND USING PARTIAL PROXIES

- Biggest known user is probably the UK “child porn” filtering
  - ➡ Best known incident when a single Wikipedia image was (mis)classified
    - ➡ All traffic to Wikipedia to the UK for major ISPs came from 2 proxy IPs, which triggered the abusive edit detector
    - ➡ http://en.wikipedia.org/wiki/Internet_Watch_Foundation_and_Wikipedia
- Can be used as an oracle to find hosts of interest
  - ➡ Richard Clayton: www.cl.cam.ac.uk/~rnc1/cleanfeed.pdf
    - ➡ Basic concept is to send SYNs to suspected address ranges with a slightly-too-short TTL
      - ➡ Censor responds to those, allowing identification of censored IPs
      - ➡ Then use reverse DNS/passive DNS to determine whats hosted on those IPs
  - ➡ Jeffrey Knockel’s technique should also work

DETECTING COMPLETE TERMINATING PROXIES

- Even when ***not*** censoring, complete flow-terminating proxies are readily detectable
  - ➡ A connection request to an invalid host
    - ➡ Proxy returns an acknowledgement immediately
      - ➡ Detection can be suppressed, but loses the benefits of caching
  - ➡ Traceroutes on TCP port 80 also shows anomalies
- A flow terminating proxy ***may*** also induce HTTP header changes
  - ➡ E.g. change the CapiTolizAtion on a specially constructed header

OVERVIEW

- **Block IP addresses**
  - IP layer
- **Disrupt TCP flows**
  - TCP (transport layer)
  - Many possible triggers
- **Block hostnames**
  - DNS (application layer)
- **Disrupt HTTP transfers**
  - HTTP (application layer)

- **Fingerprinting filtering products**