Come As You Are:

Helping Unmodified Clients Bypass Censorship with Server-Side Evasion

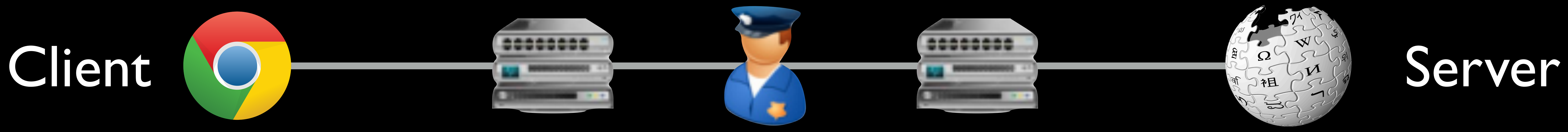
Kevin Bock

George Hughey, Louis-Henri Merino, Tania Arya,
Daniel Liscinsky, Regina Pogosian, Dave Levin

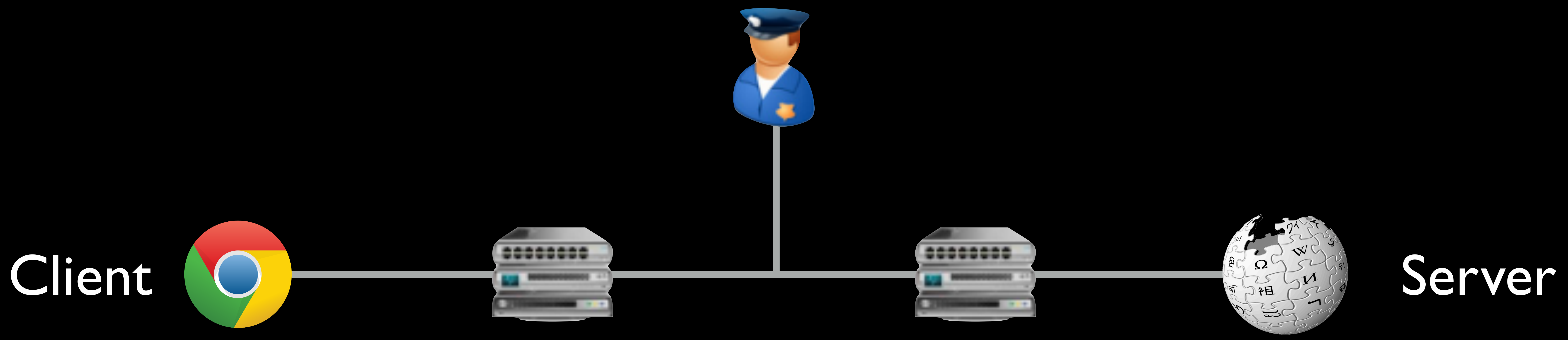
UNIVERSITY OF MARYLAND

ANRP
Applied Networking Research Prize

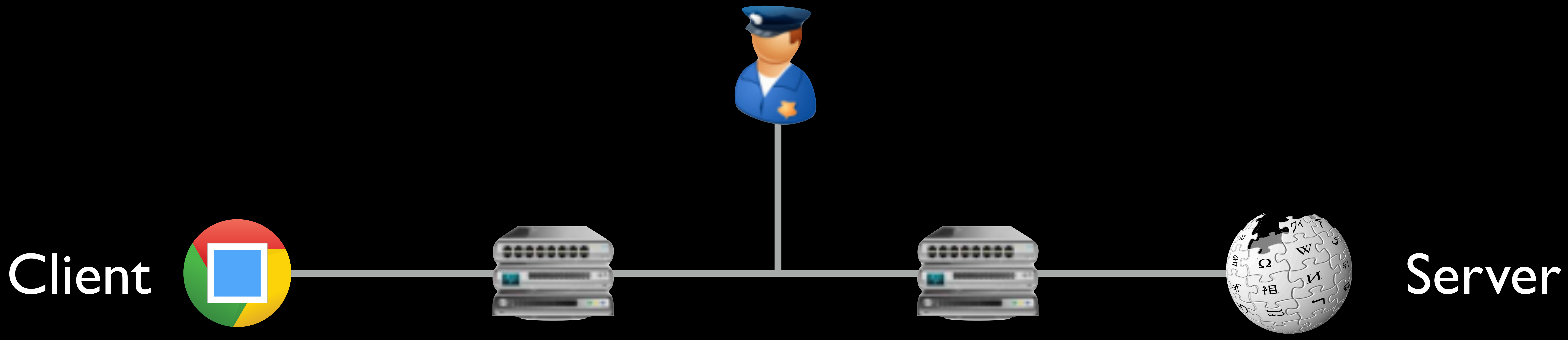
In-network censorship by nation-states



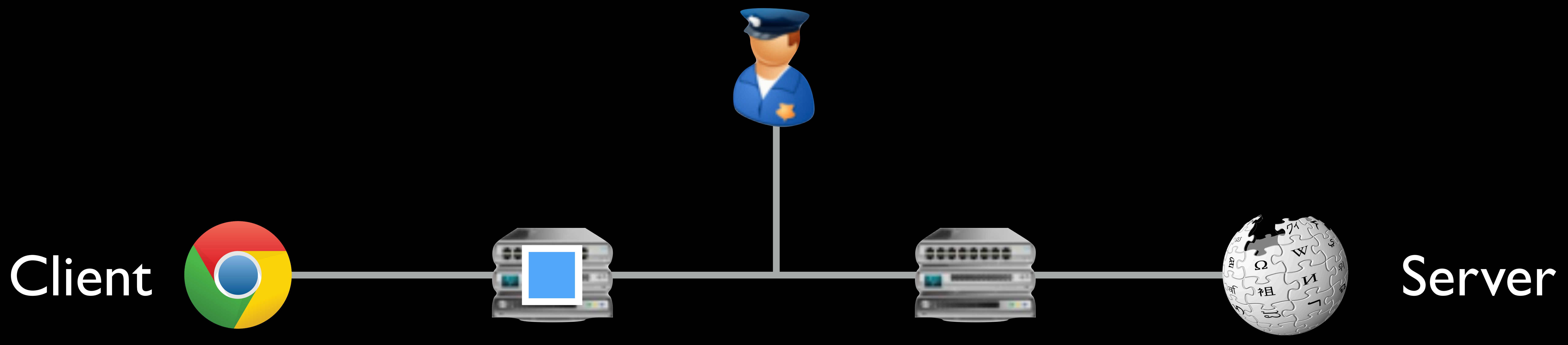
In-network censorship by nation-states



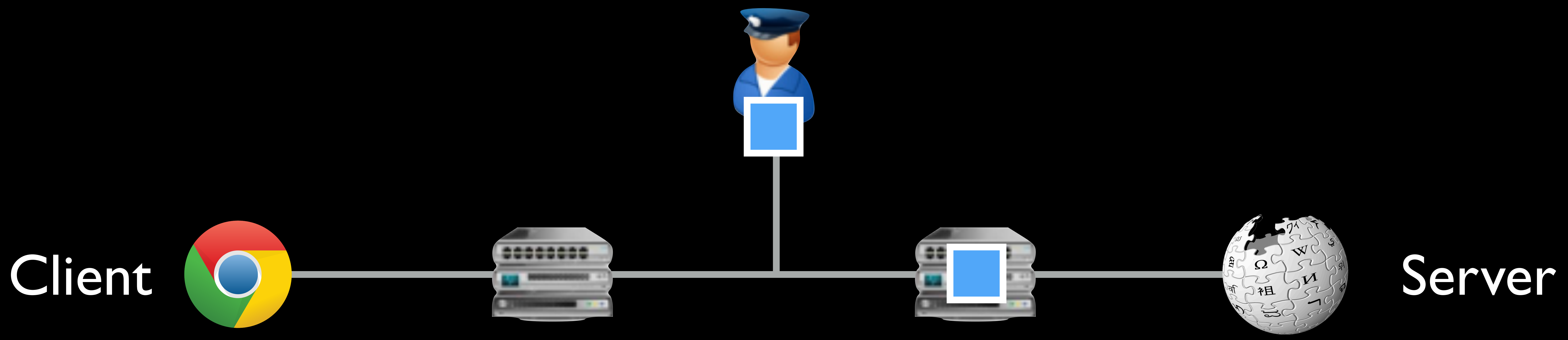
In-network censorship by nation-states



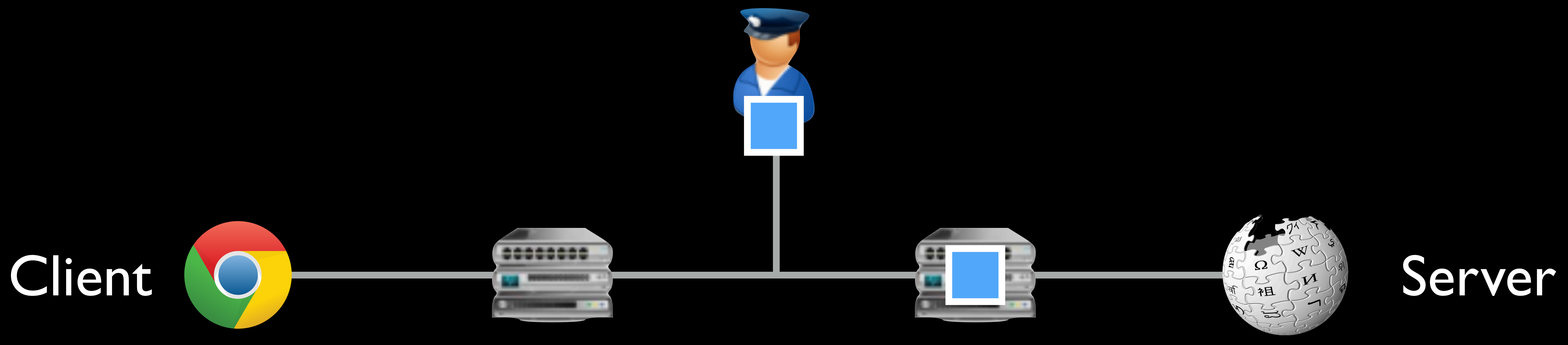
In-network censorship by nation-states



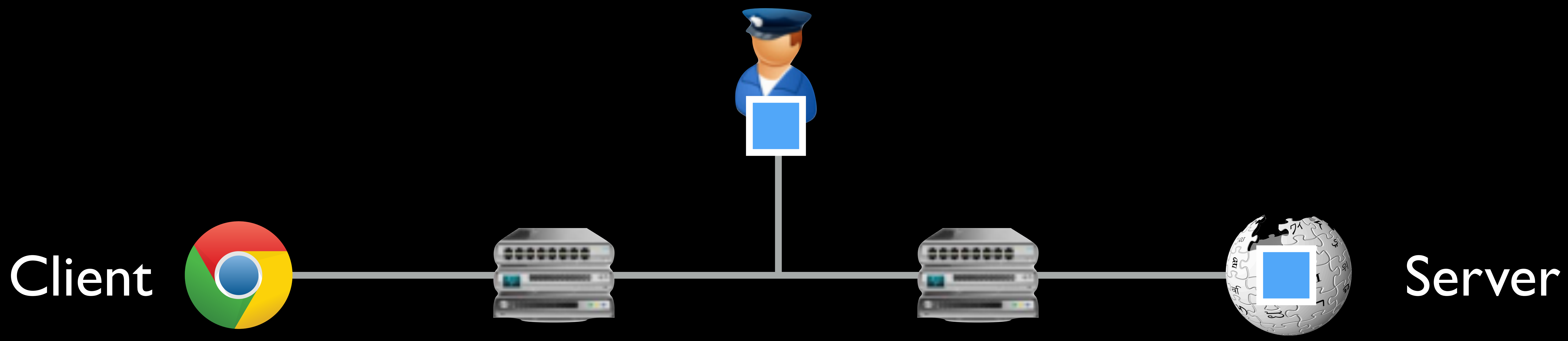
In-network censorship by nation-states



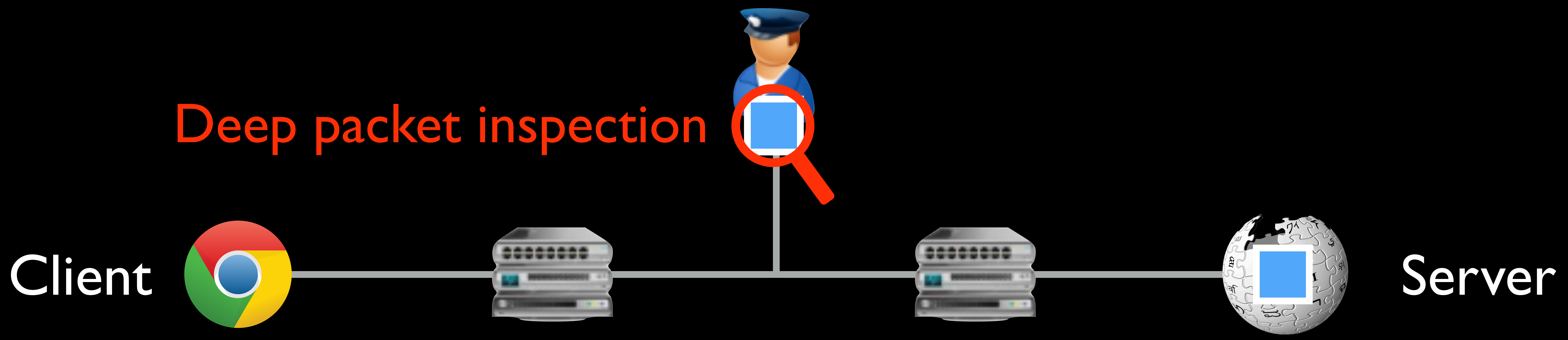
In-network censorship by nation-states



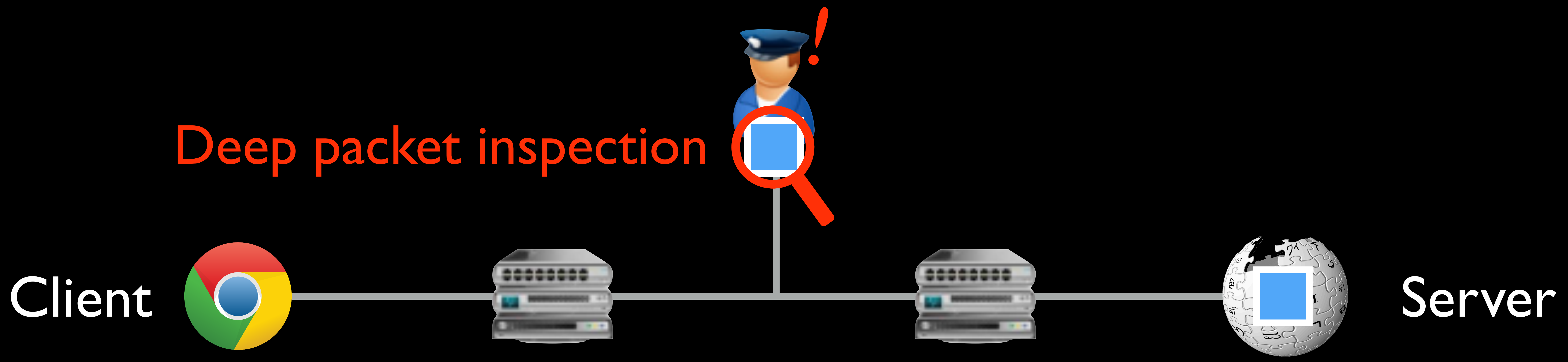
In-network censorship by nation-states



In-network censorship by nation-states



In-network censorship by nation-states

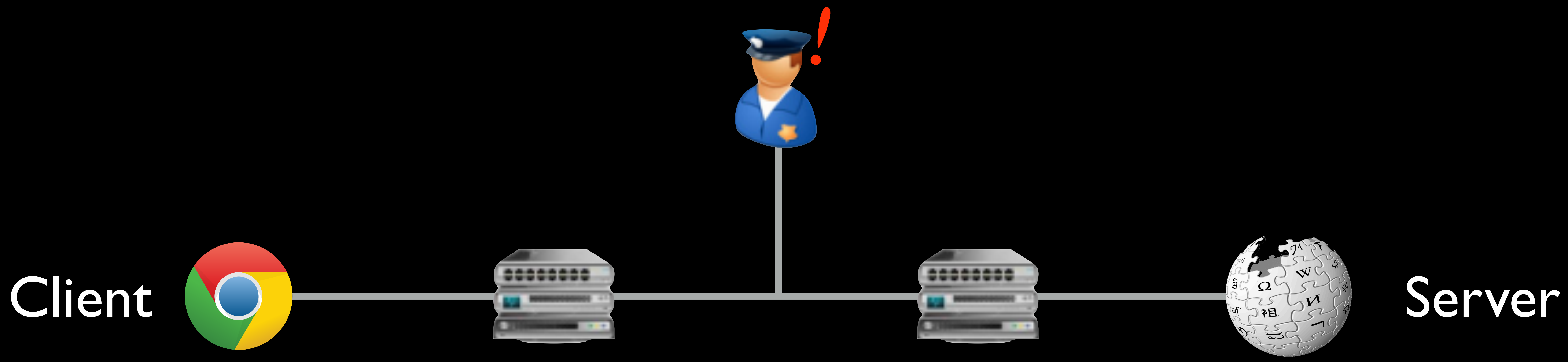


Deep packet inspection

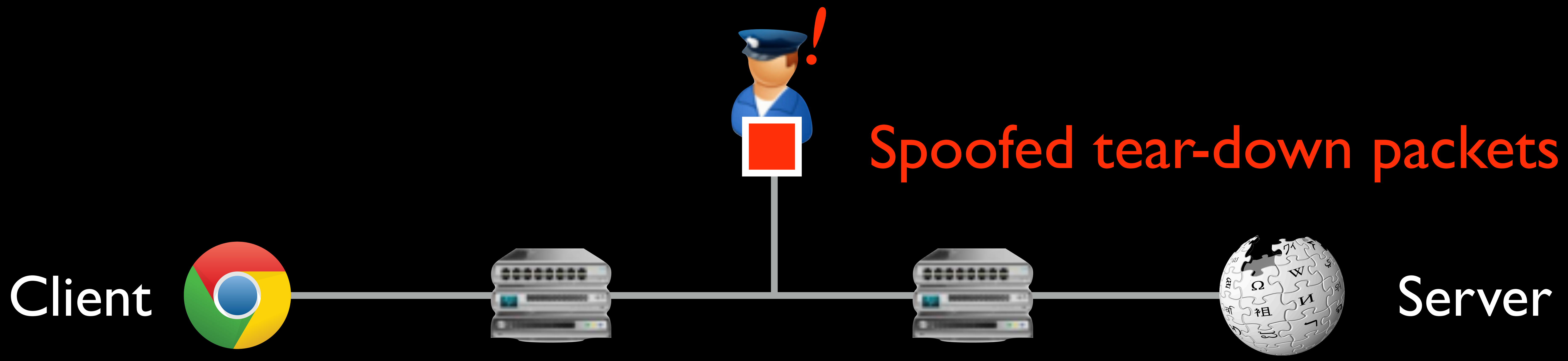
Client

Server

In-network censorship by nation-states



In-network censorship by nation-states

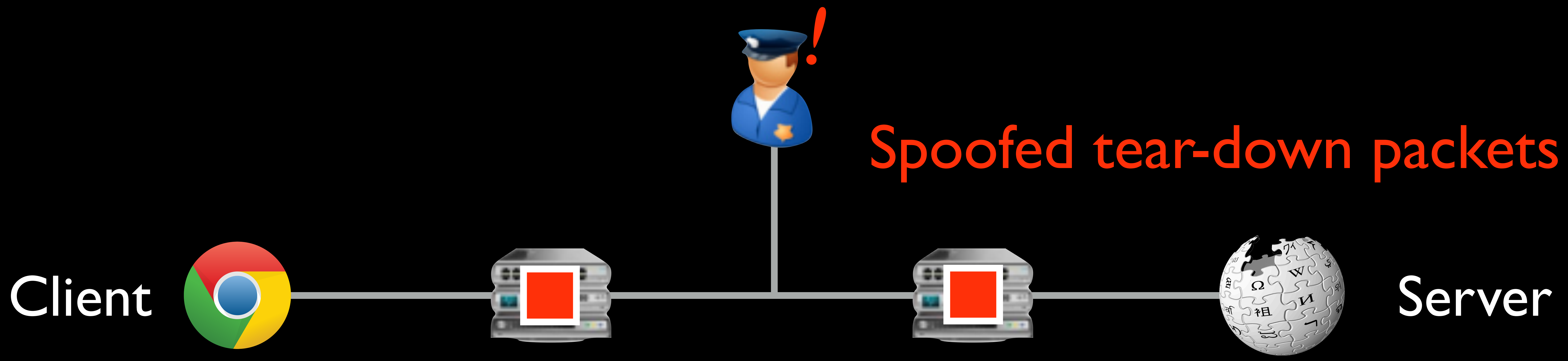


Client

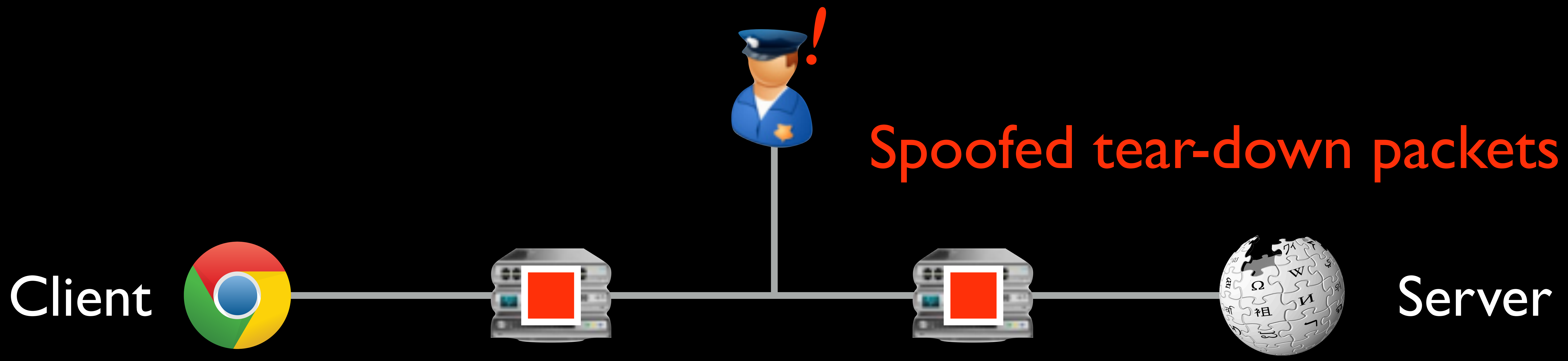
Spoofed tear-down packets

Server

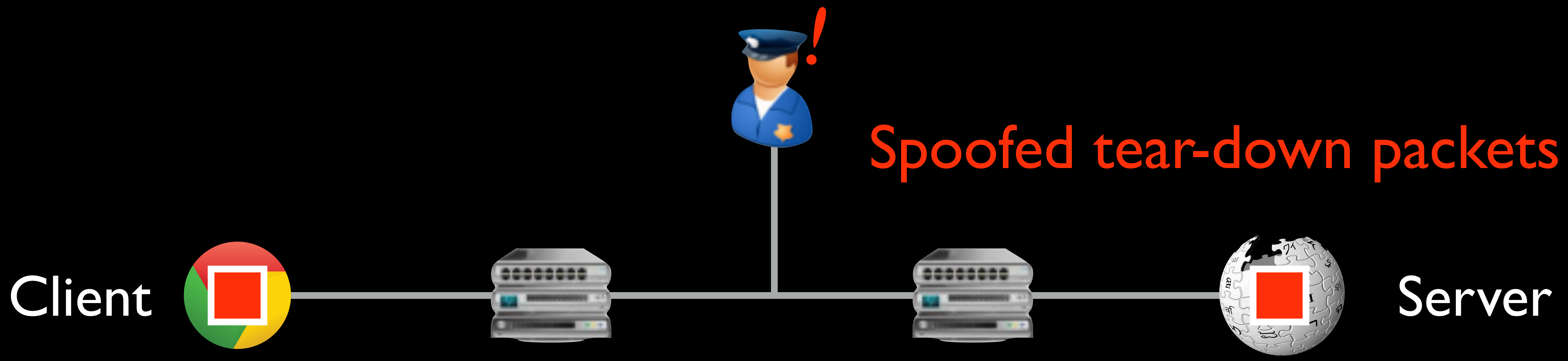
In-network censorship by nation-states



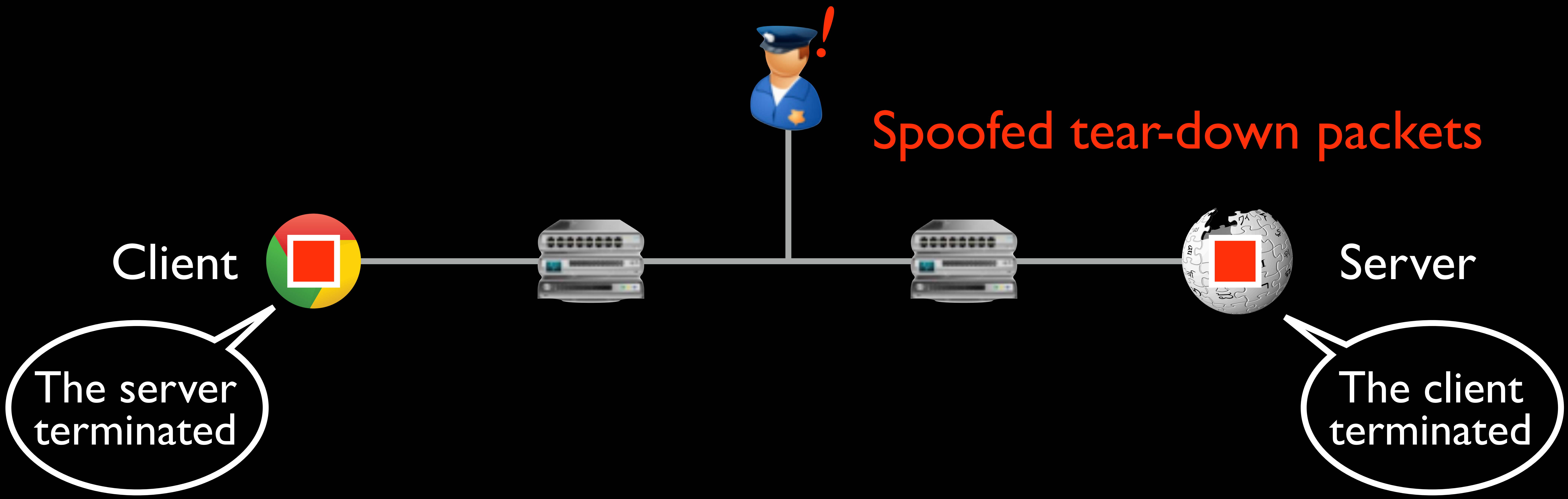
In-network censorship by nation-states



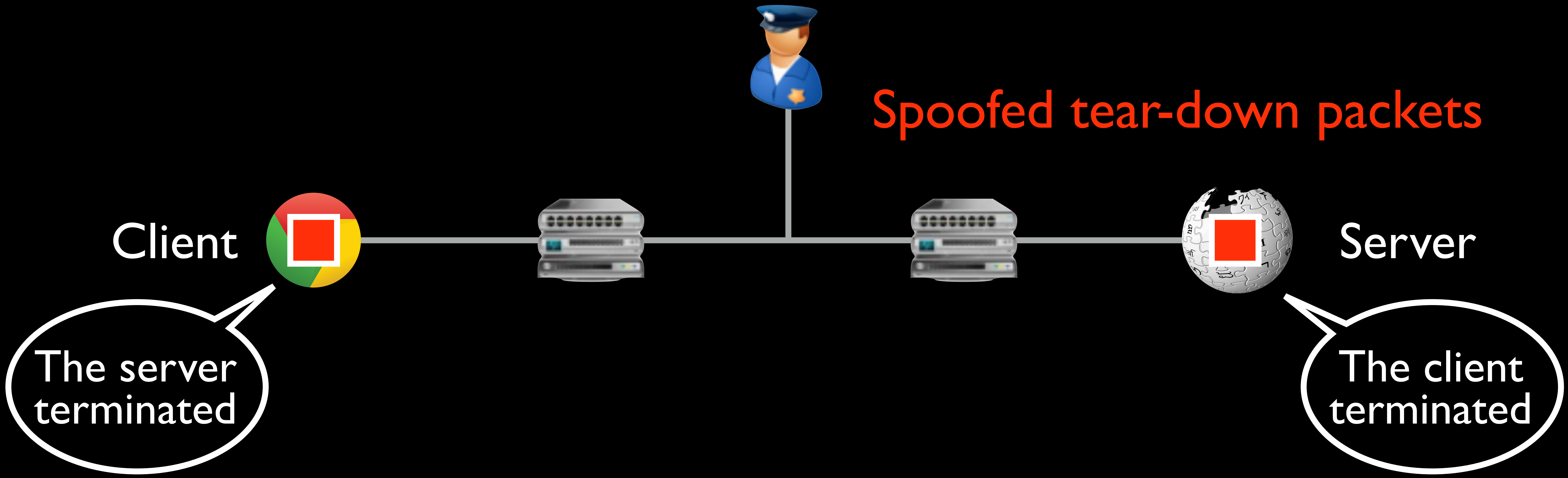
In-network censorship by nation-states



In-network censorship by nation-states

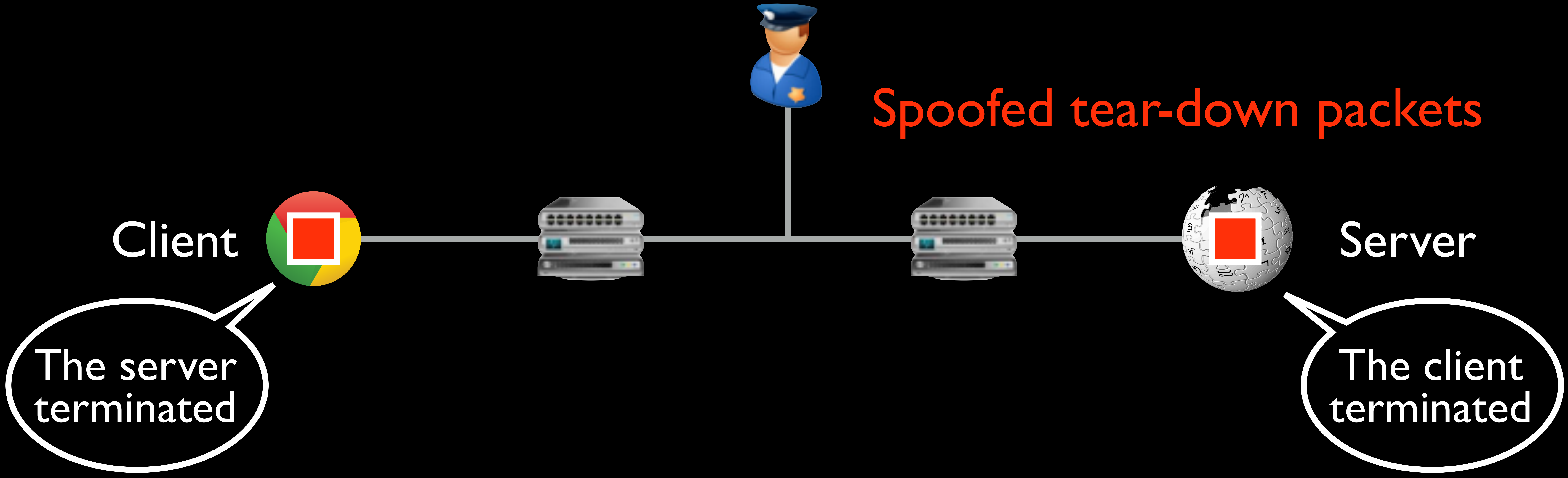


In-network censorship by nation-states



Injecting tear-down packets

In-network censorship by nation-states

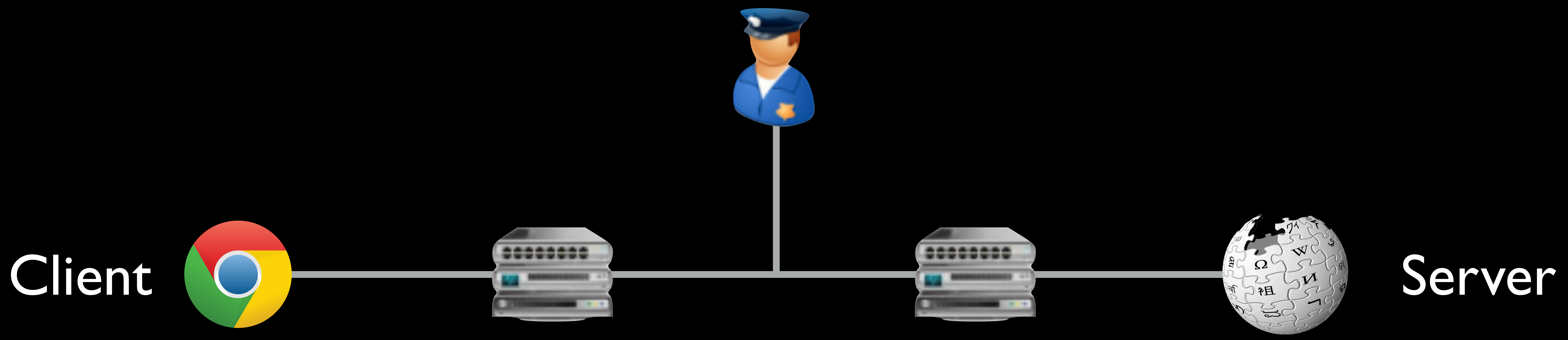


Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

In-network censorship by nation-states



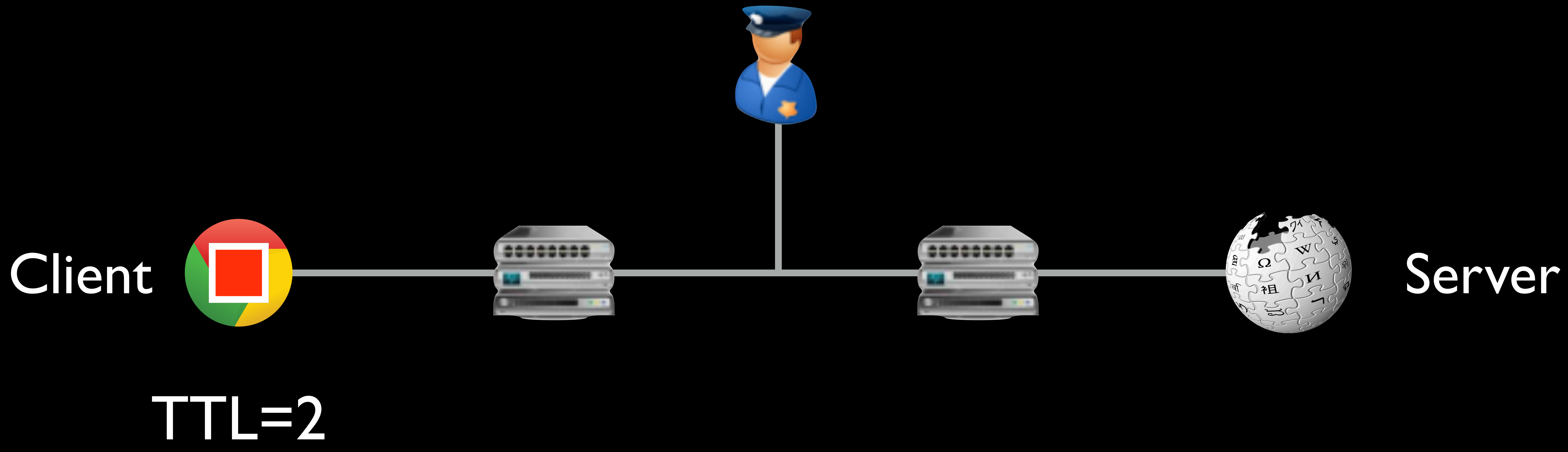
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



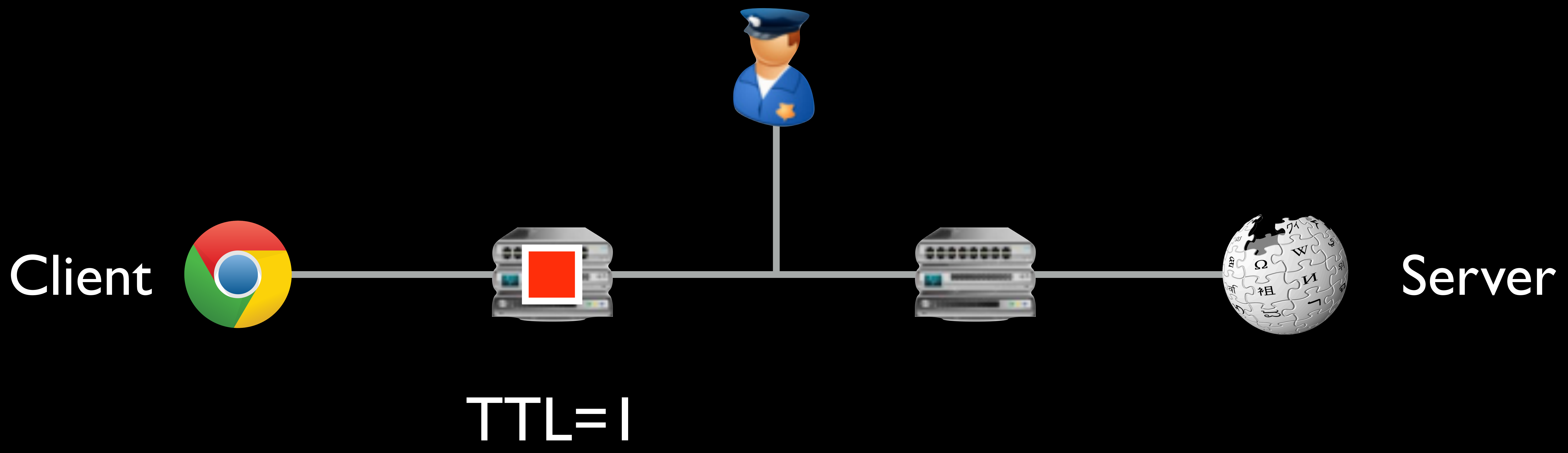
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



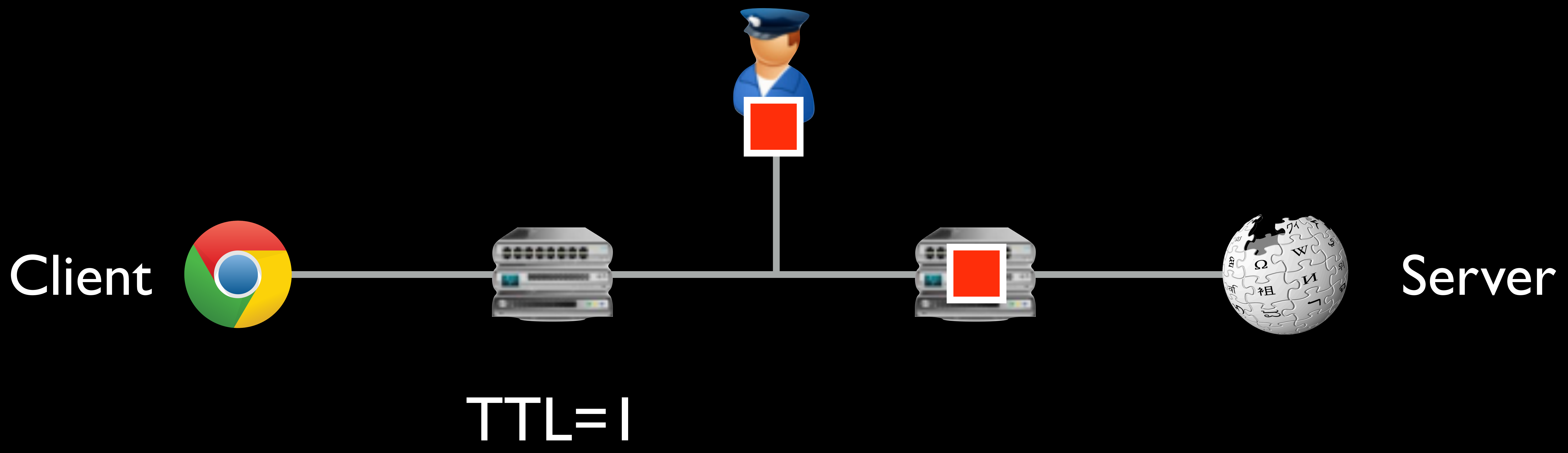
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



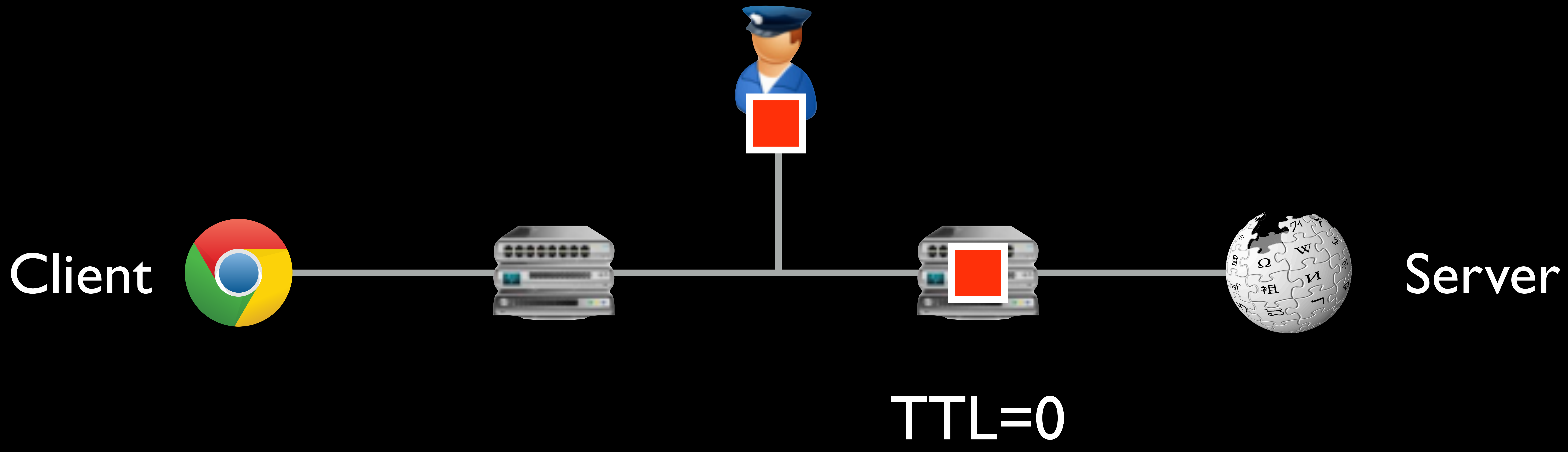
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



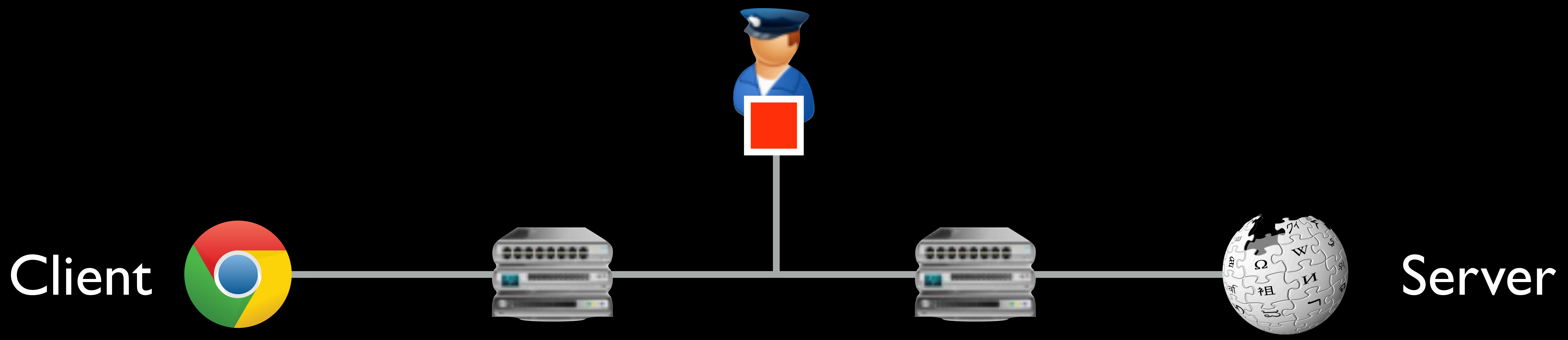
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



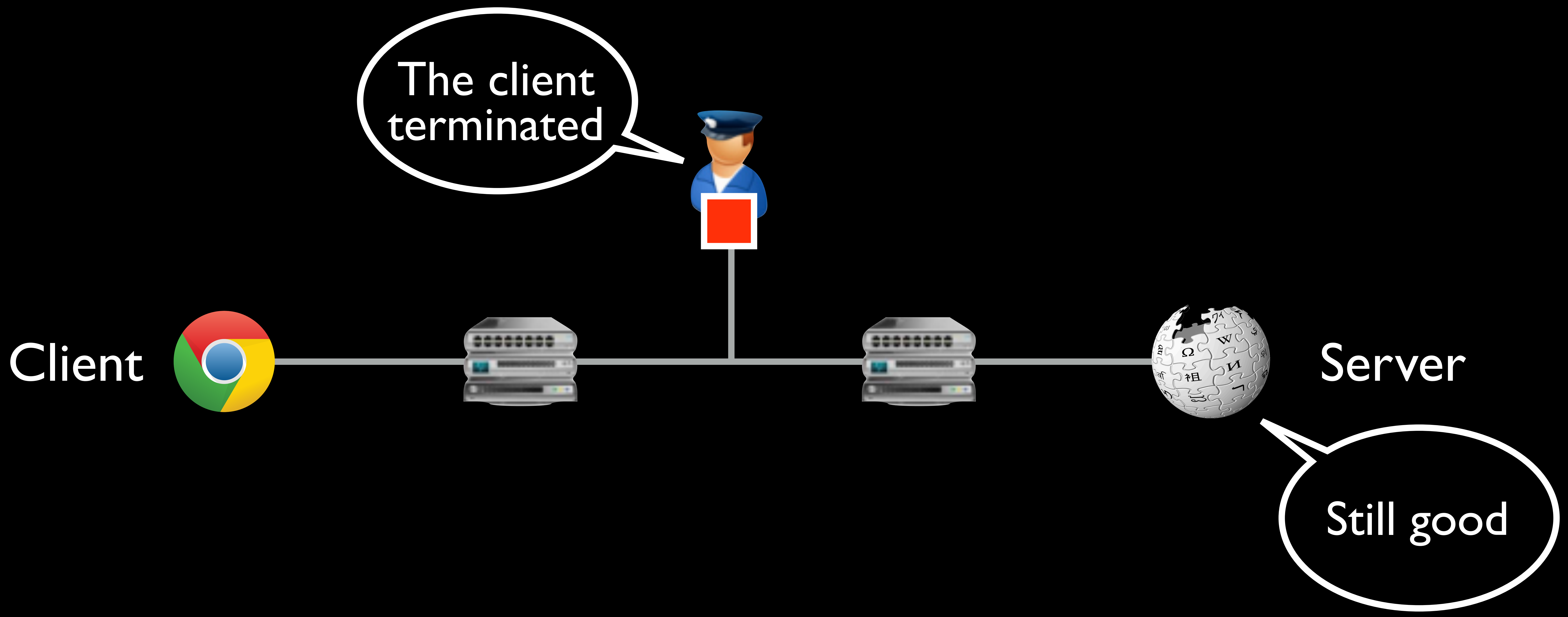
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



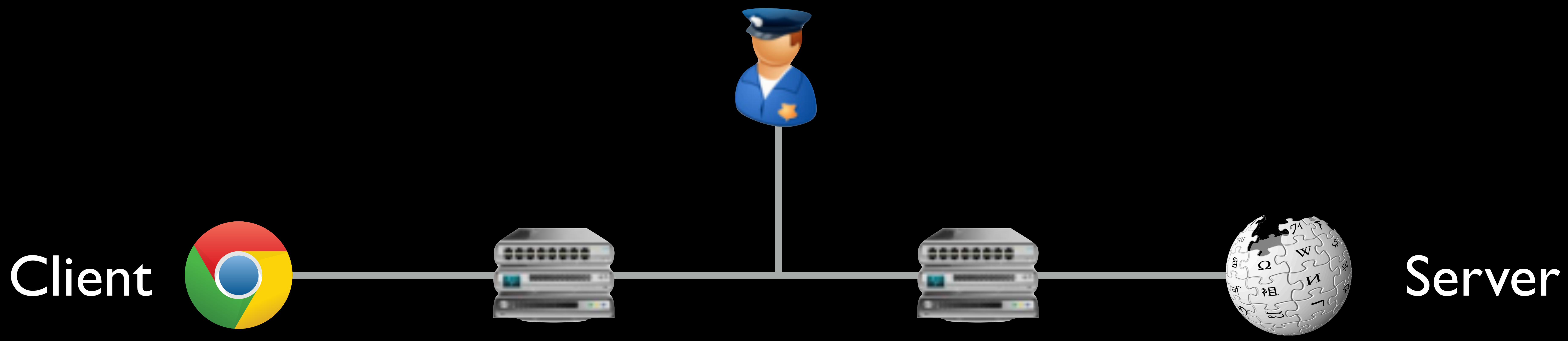
Injecting tear-down packets

Requires *per-flow state*

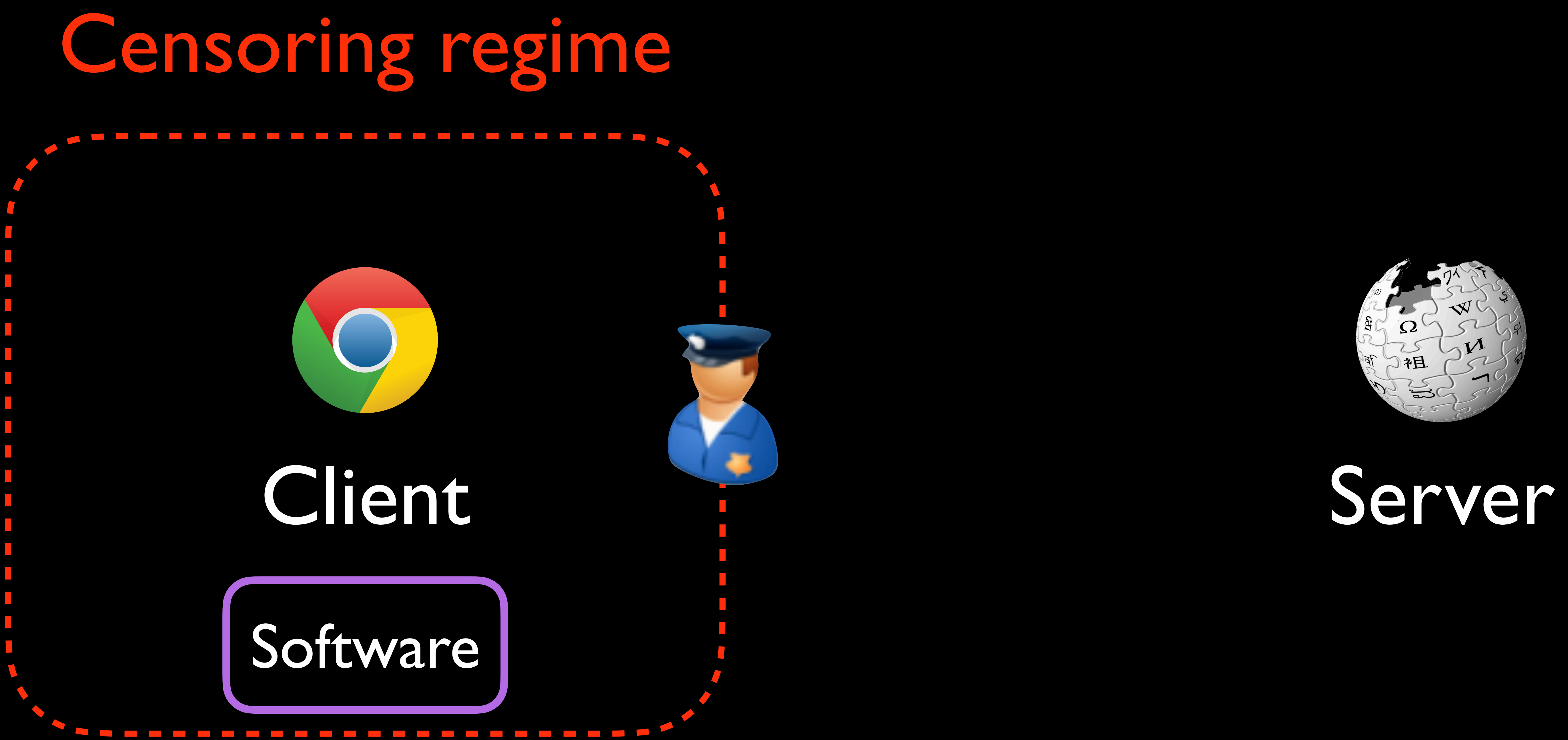
Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

Evasion has always involved the client



Evasion has always involved the client



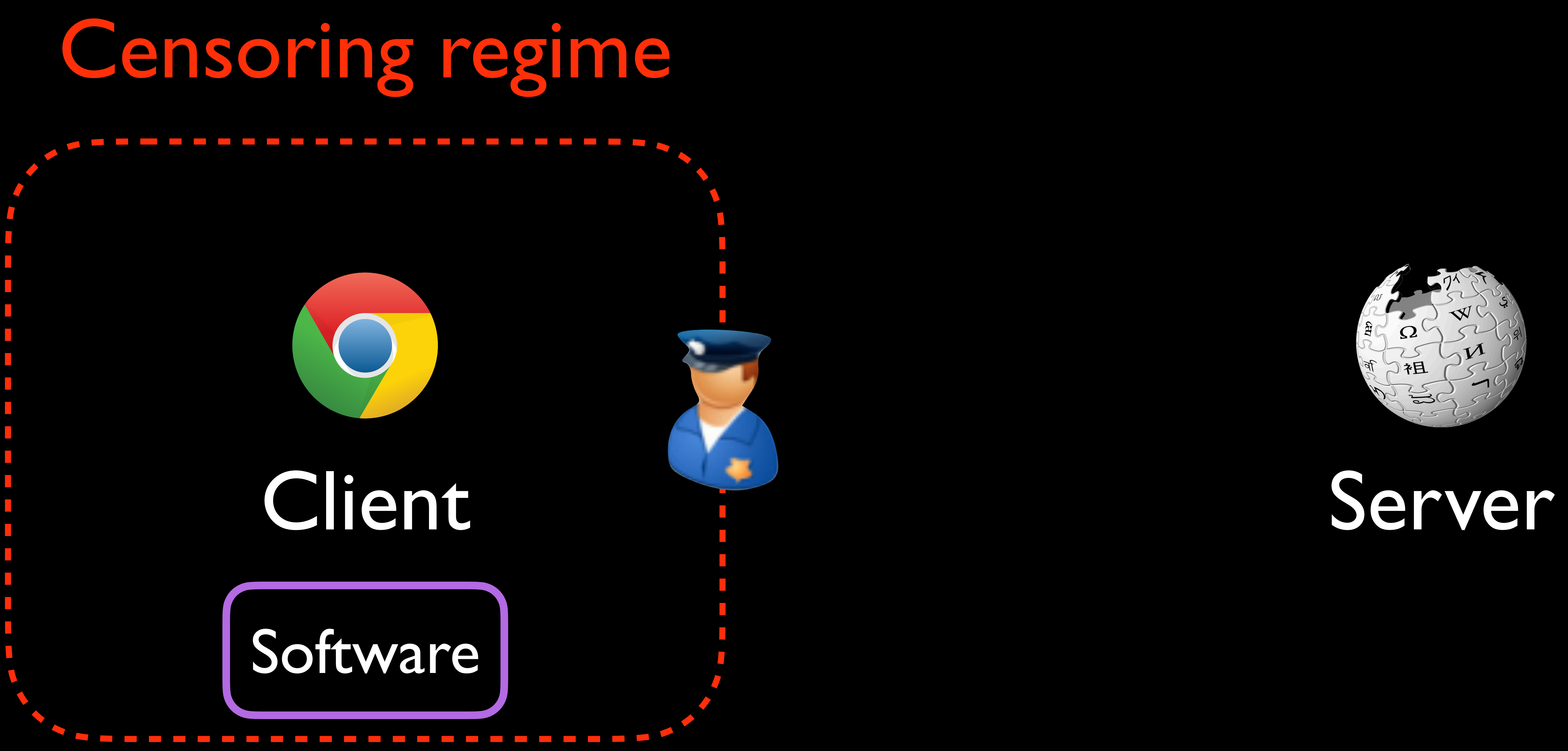
Censoring regime

Client

Software

Server

Evasion has always involved the client



Censoring regime

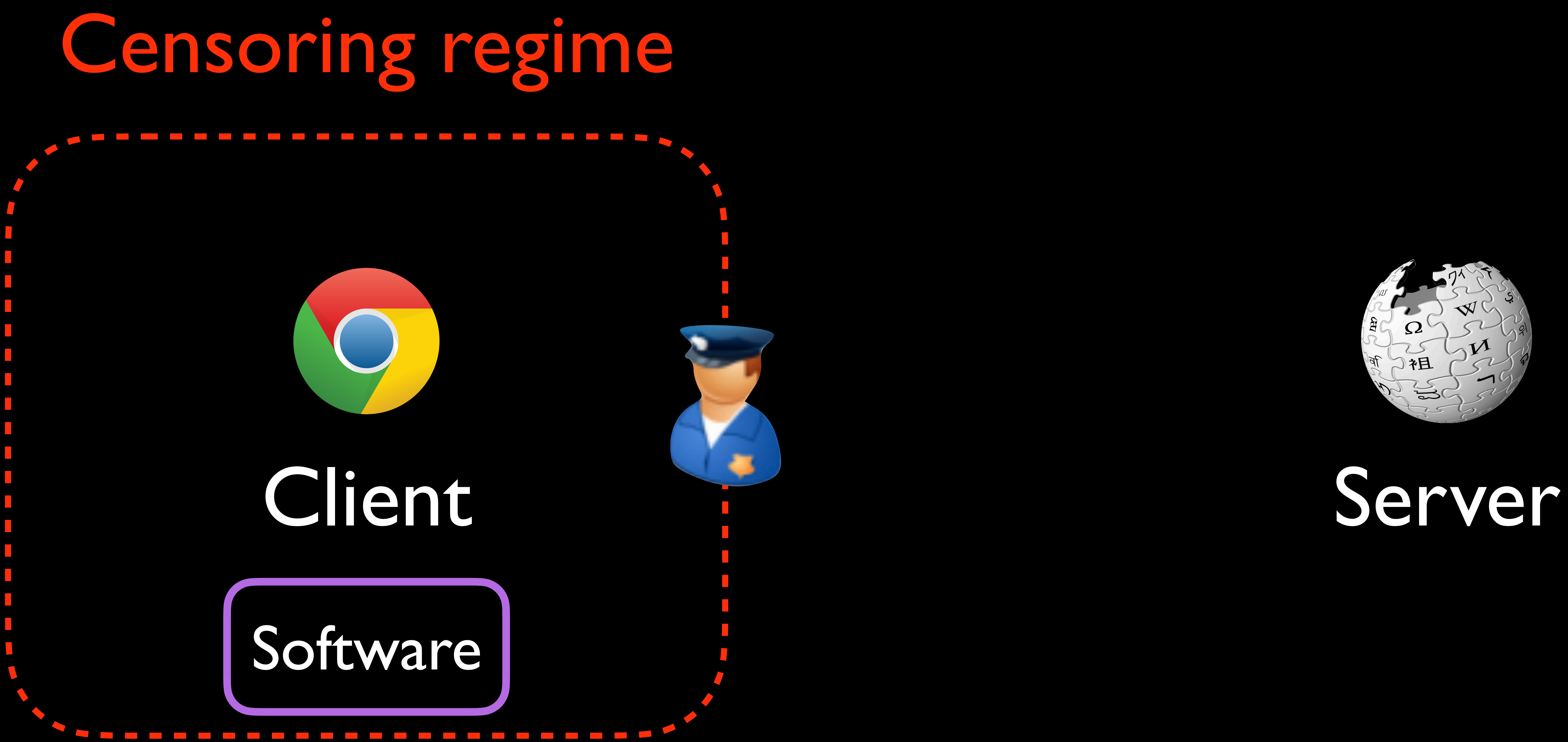
Client

Software

Server

Installing software can pose risks to the user

Evasion has always involved the client



Censoring regime

Client

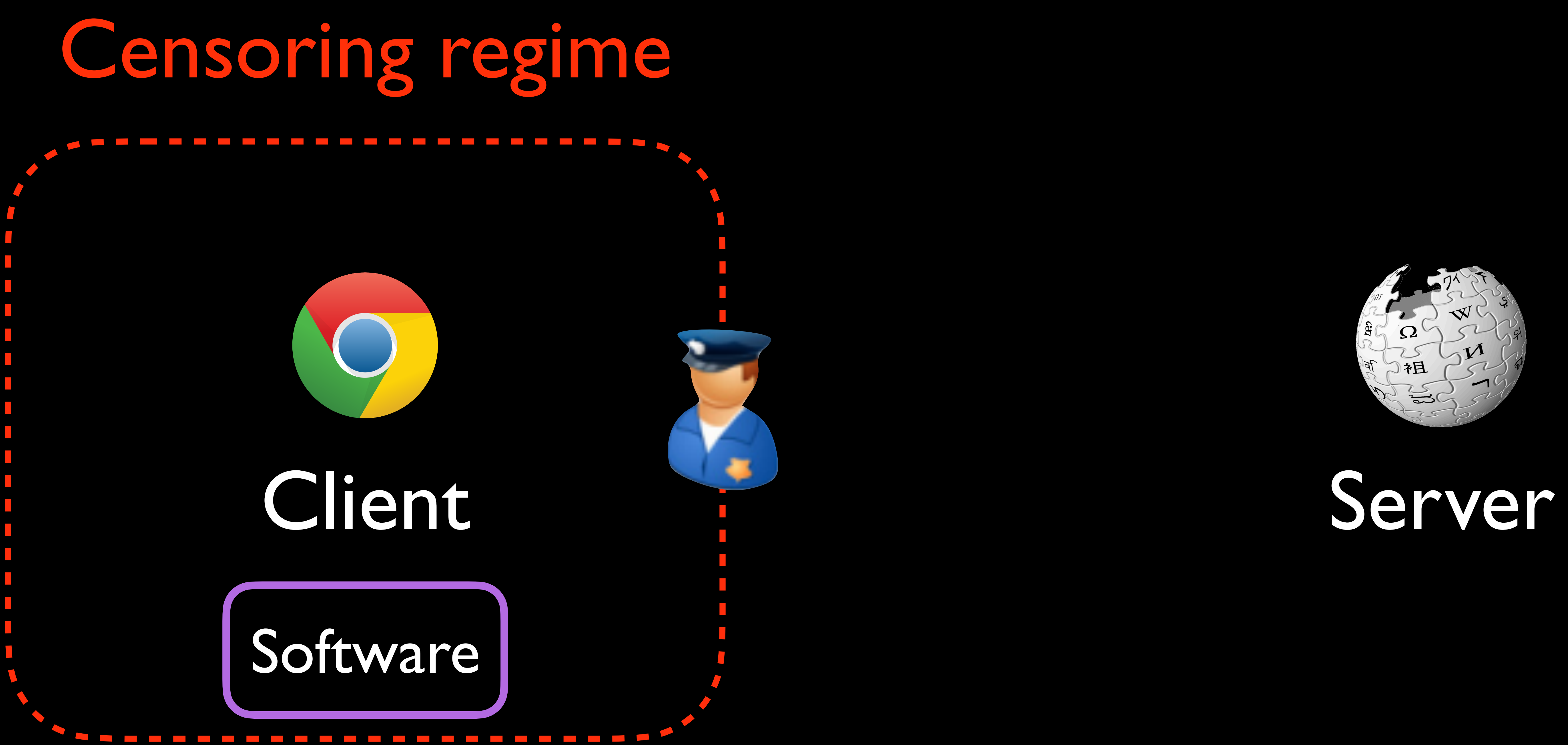
Software

Server

Installing software can
pose risks to the user

Cannot help users who do not
know they are censored

Ideally, servers could help



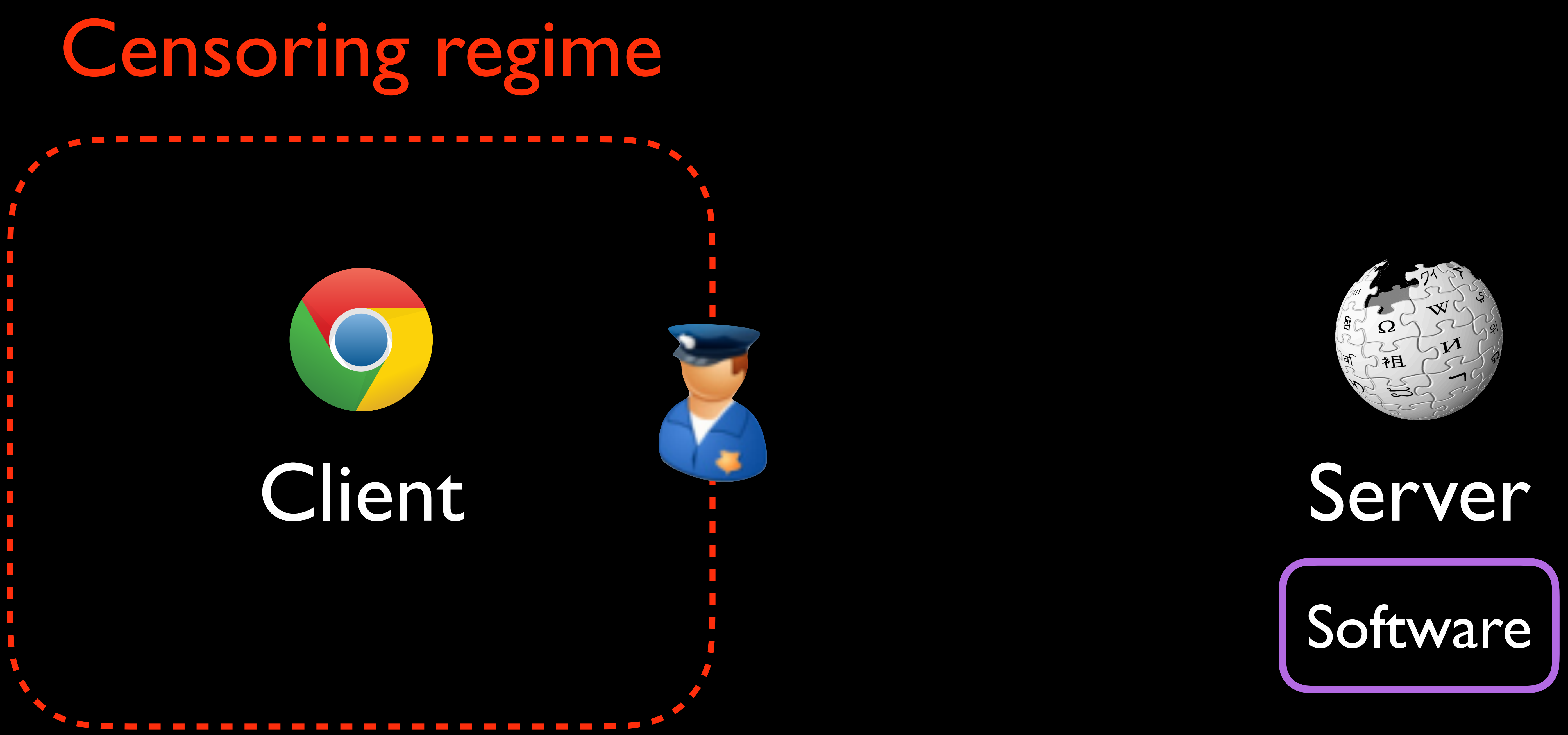
Censoring regime

Client

Software

Server

Server-side evasion



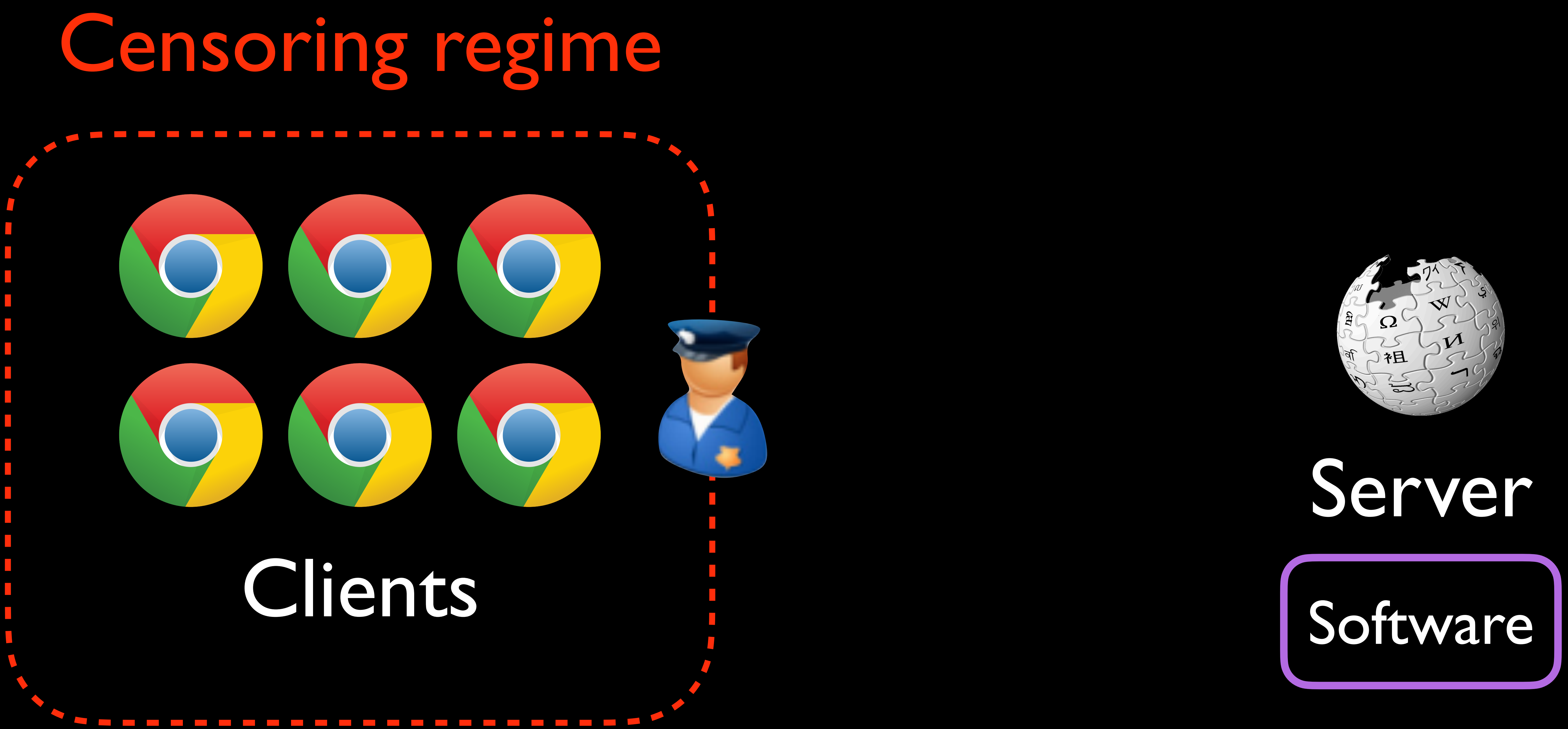
Censoring regime

Client

Server

Software

Server-side evasion



Censoring regime

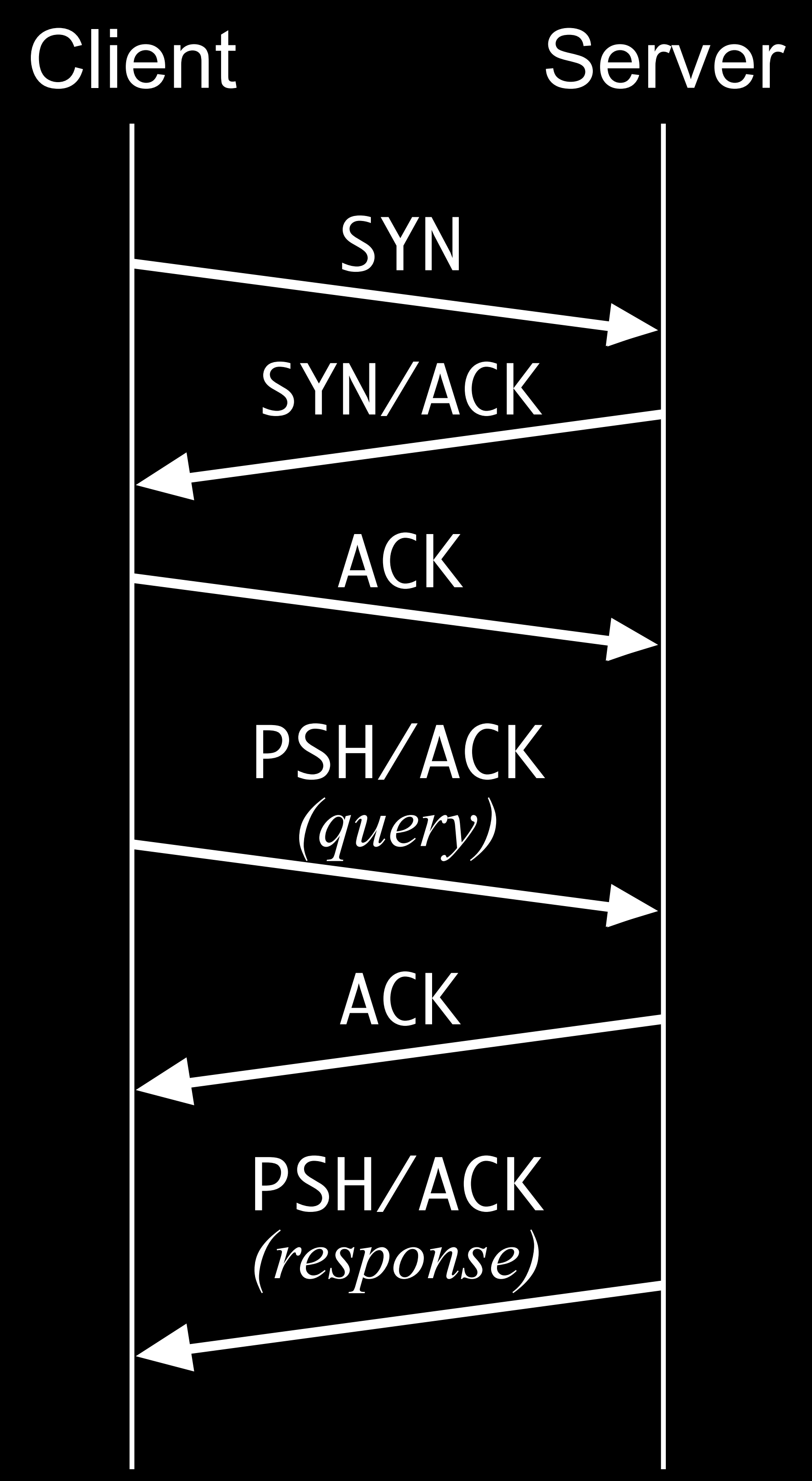
Clients

Server

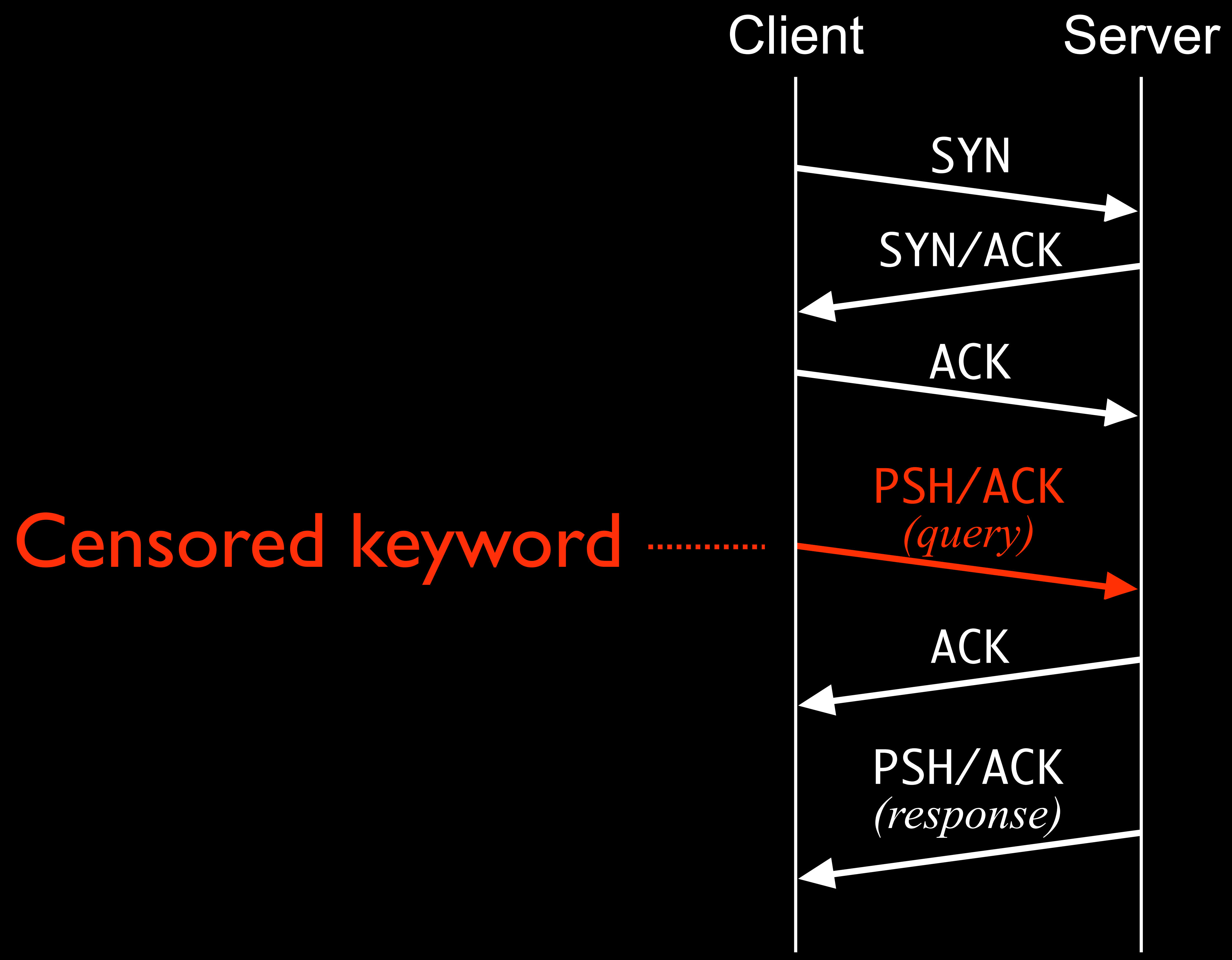
Software

Potentially broadens reachability
without *any* client-side deployment

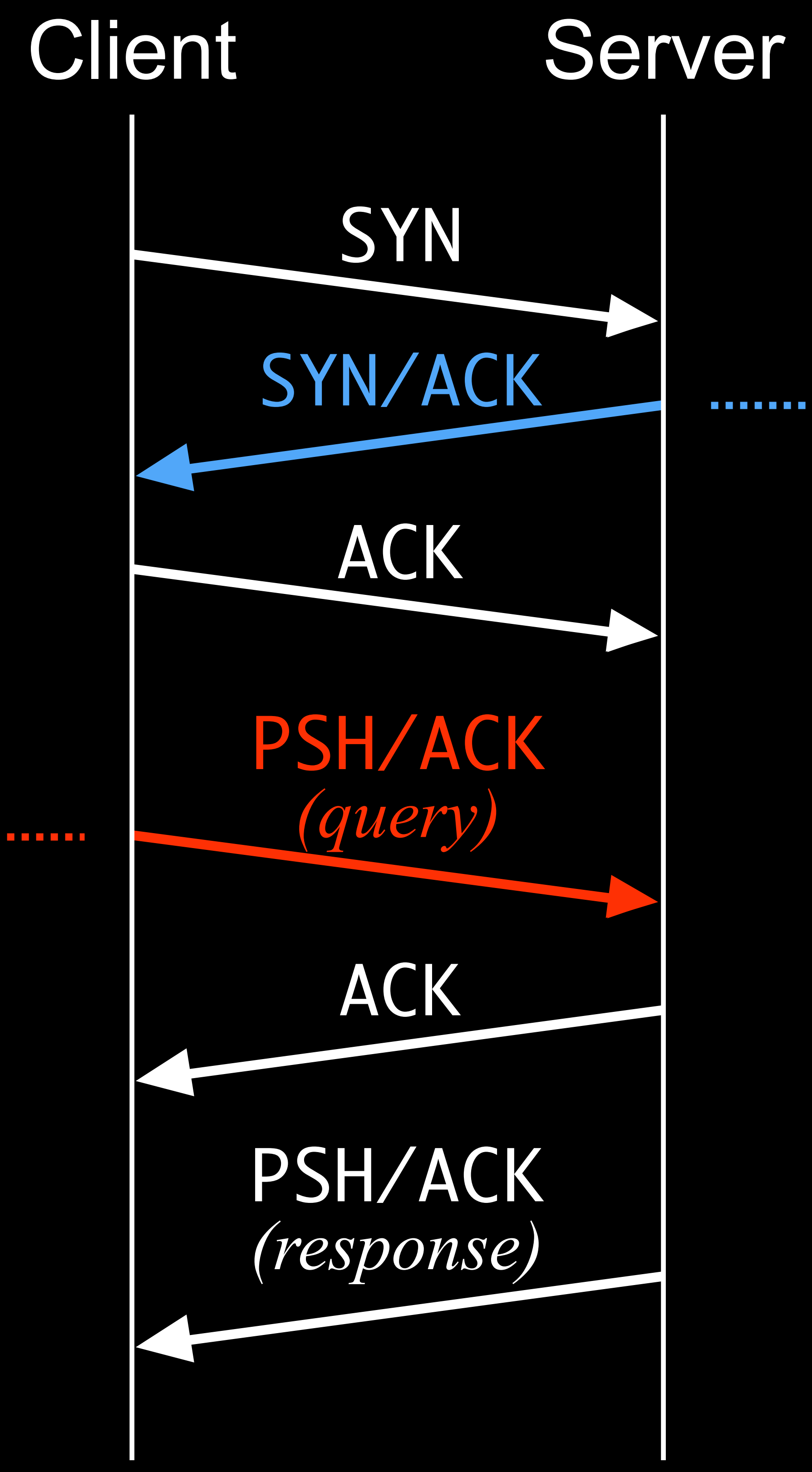
Server-side evasion “shouldn’t” work



Server-side evasion “shouldn’t” work



Server-side evasion "shouldn't" work



All a server does before client is censored

Censored keyword

This paper

Server-side evasion is possible

- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW’s resynchronization state
- “Multibox Theory”

This paper

Server-side evasion is possible

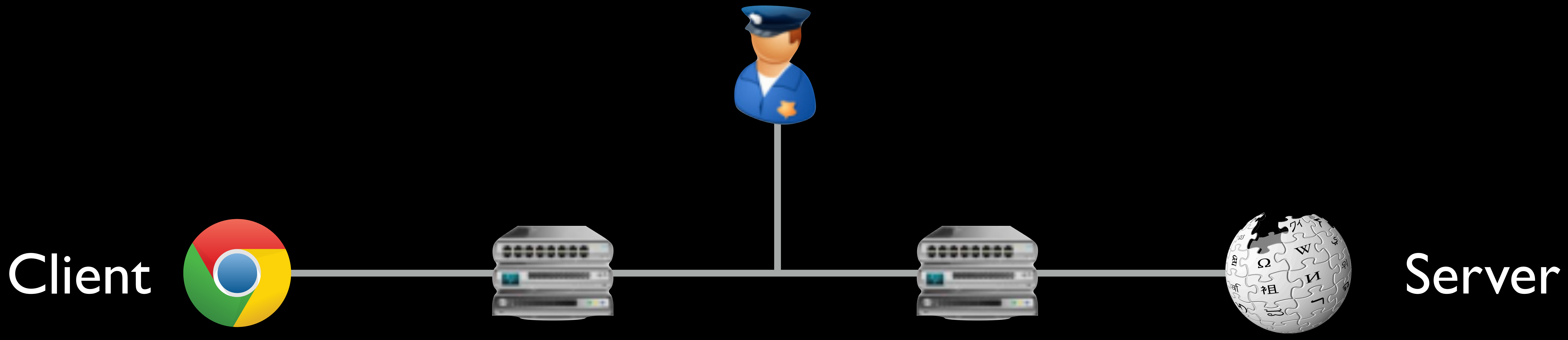
- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW's resynchronization state
- "Multibox Theory"

Geneva

Genetic Evasion



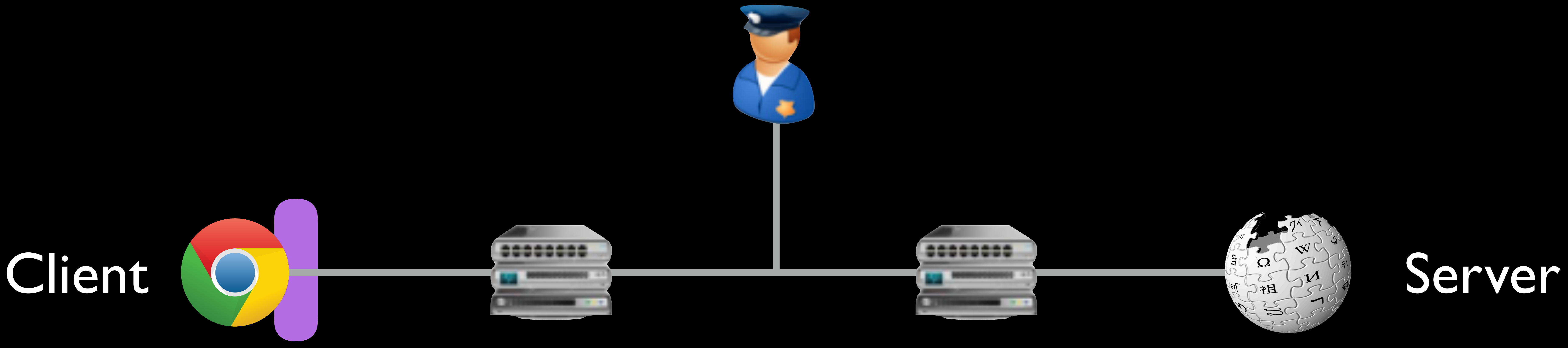
Client

Server

Bock et al. CCS'19

Geneva

Genetic Evasion



Geneva runs strictly at one side

Manipulates packets to and from the client

Bock et al. CCS'19

Geneva

Genetic Evasion

Manipulates packets to and from the client

Duplicate

Tamper

Fragment

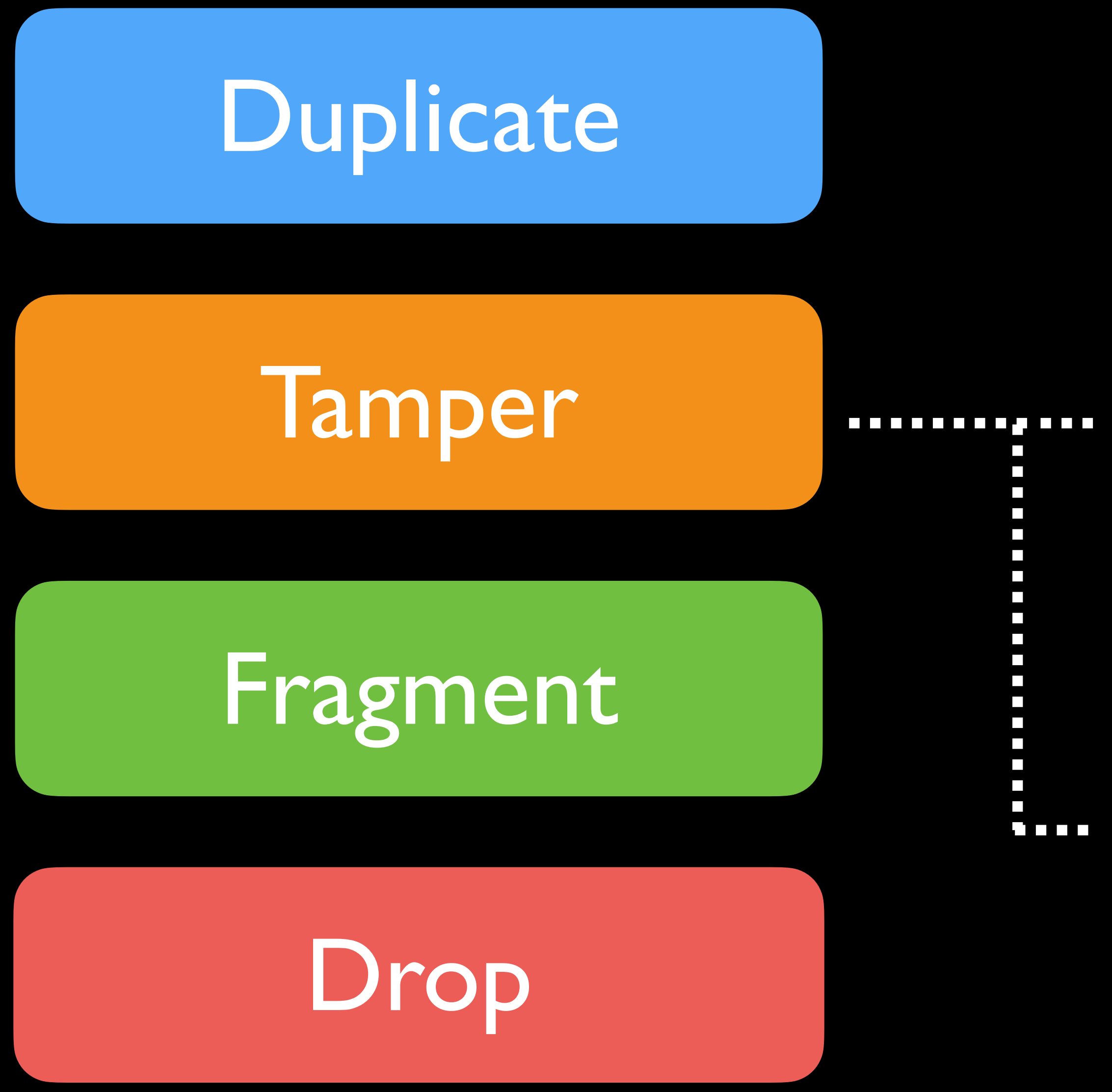
Drop

Bock et al. CCS'19

Geneva

Genetic Evasion

Manipulates packets to and from the client



Duplicate

Tamper

Fragment

Drop

Alter or corrupt
any TCP/IP header field

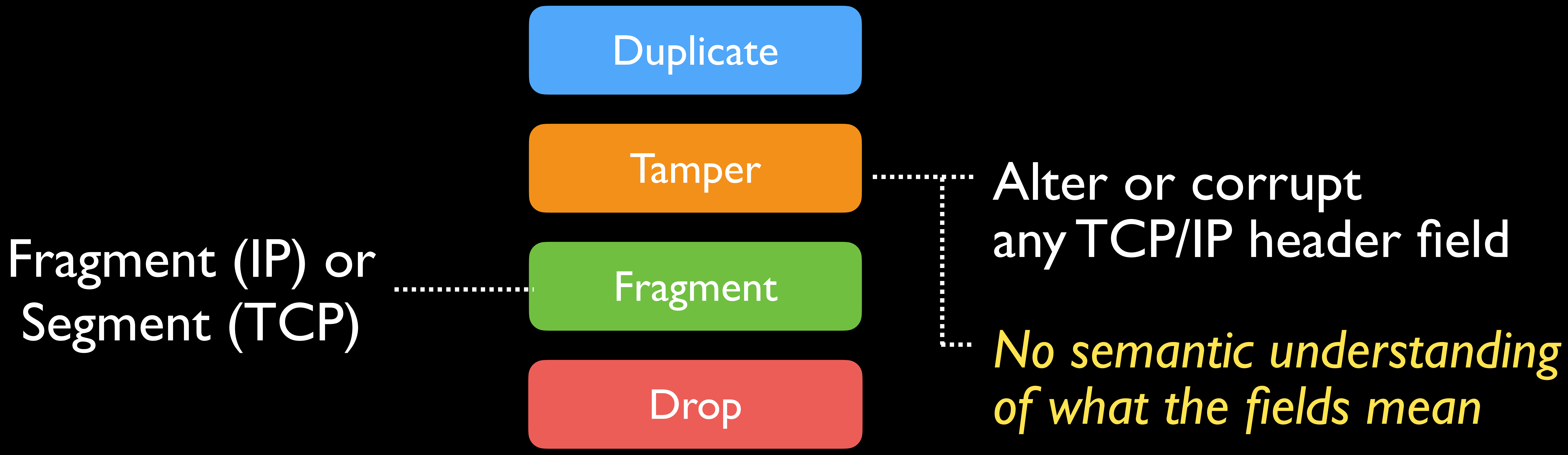
*No semantic understanding
of what the fields mean*

Bock et al. CCS'19

Geneva

Genetic Evasion

Manipulates packets to and from the client



Duplicate

Tamper

Fragment

Drop

Fragment (IP) or
Segment (TCP)

Alter or corrupt
any TCP/IP header field

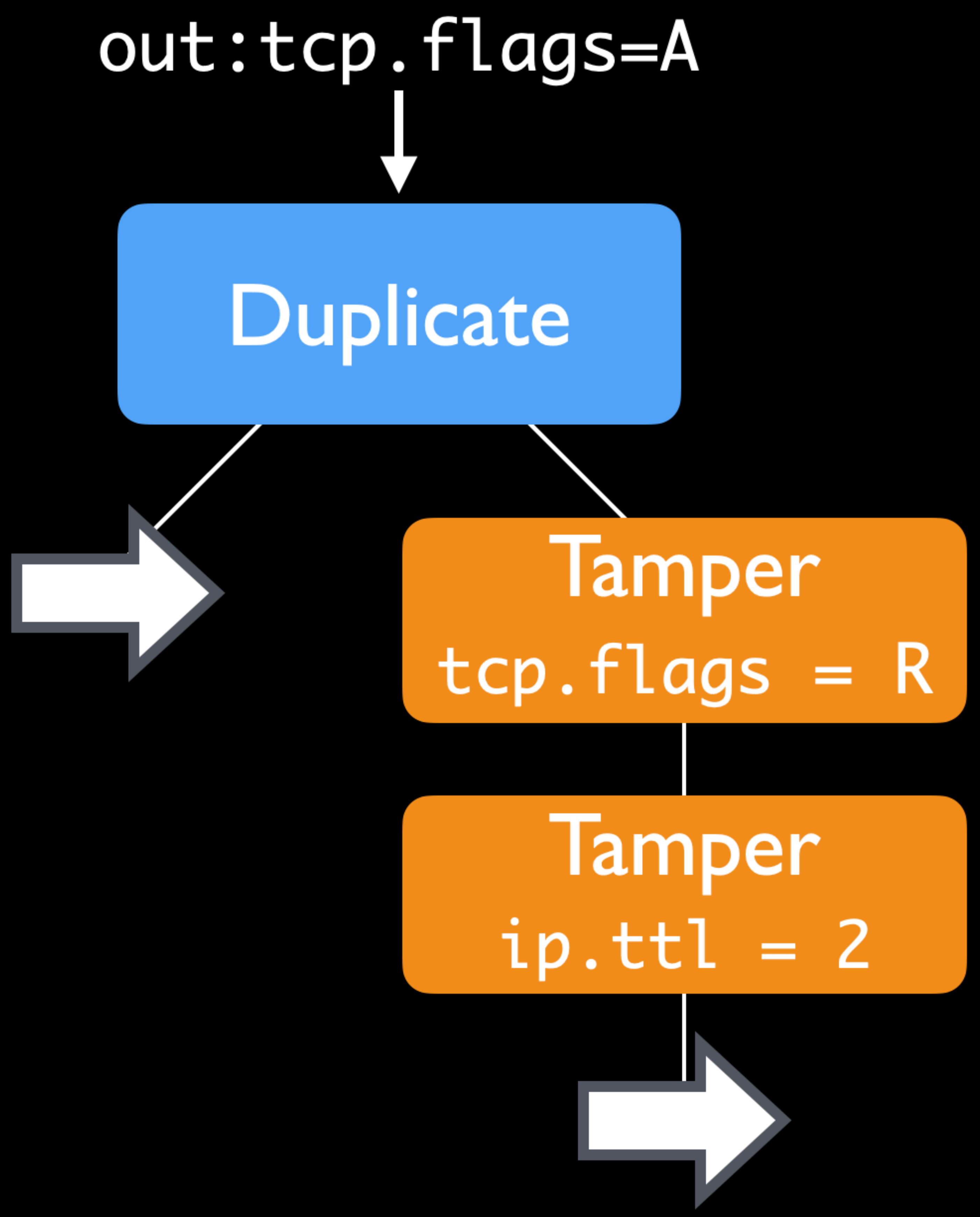
*No semantic understanding
of what the fields mean*

Bock et al. CCS'19

Geneva

Genetic Evasion

Manipulates packets to and from the client

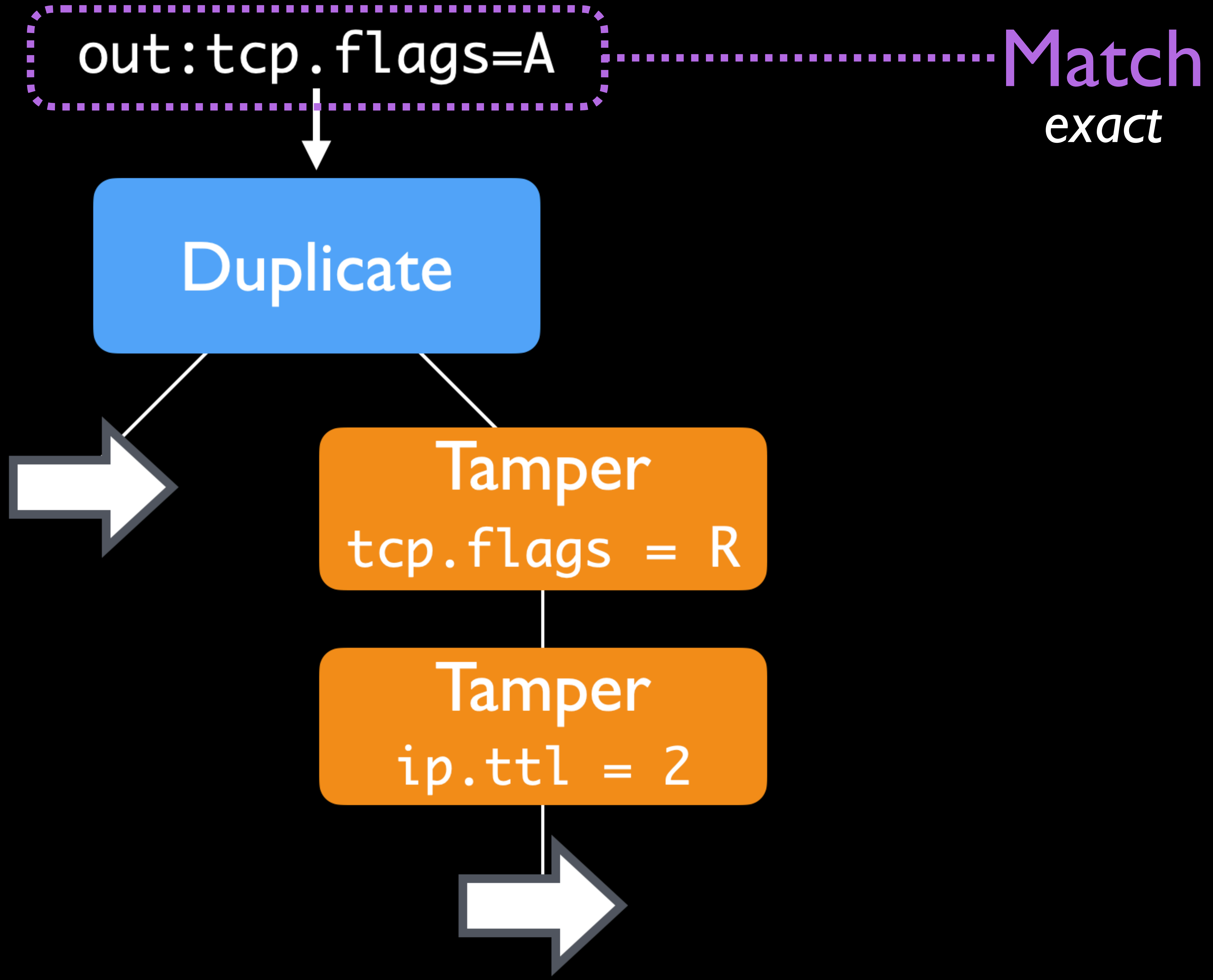


Bock et al. CCS'19

Geneva

Genetic Evasion

Manipulates packets to and from the client

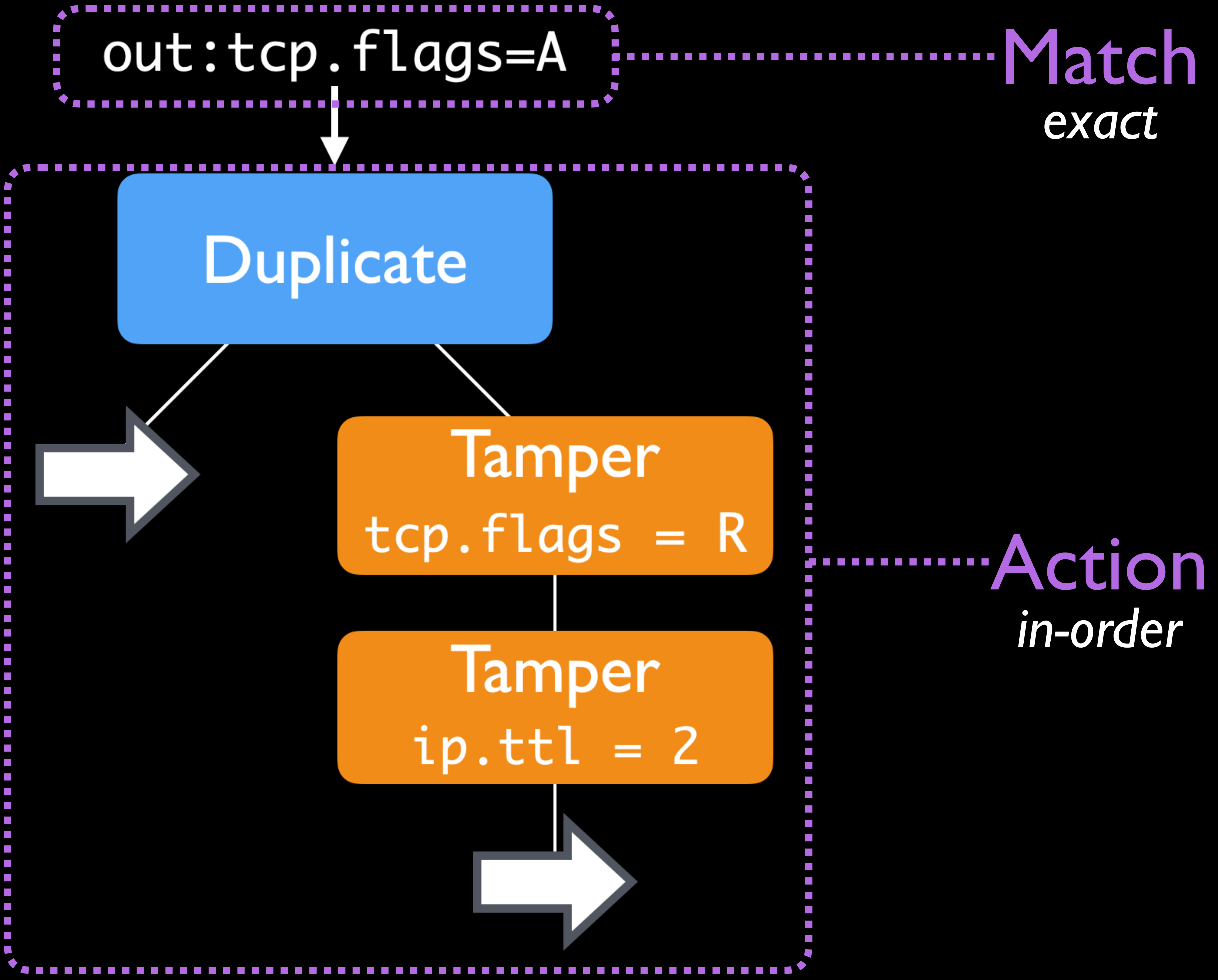


Bock et al. CCS'19

Geneva

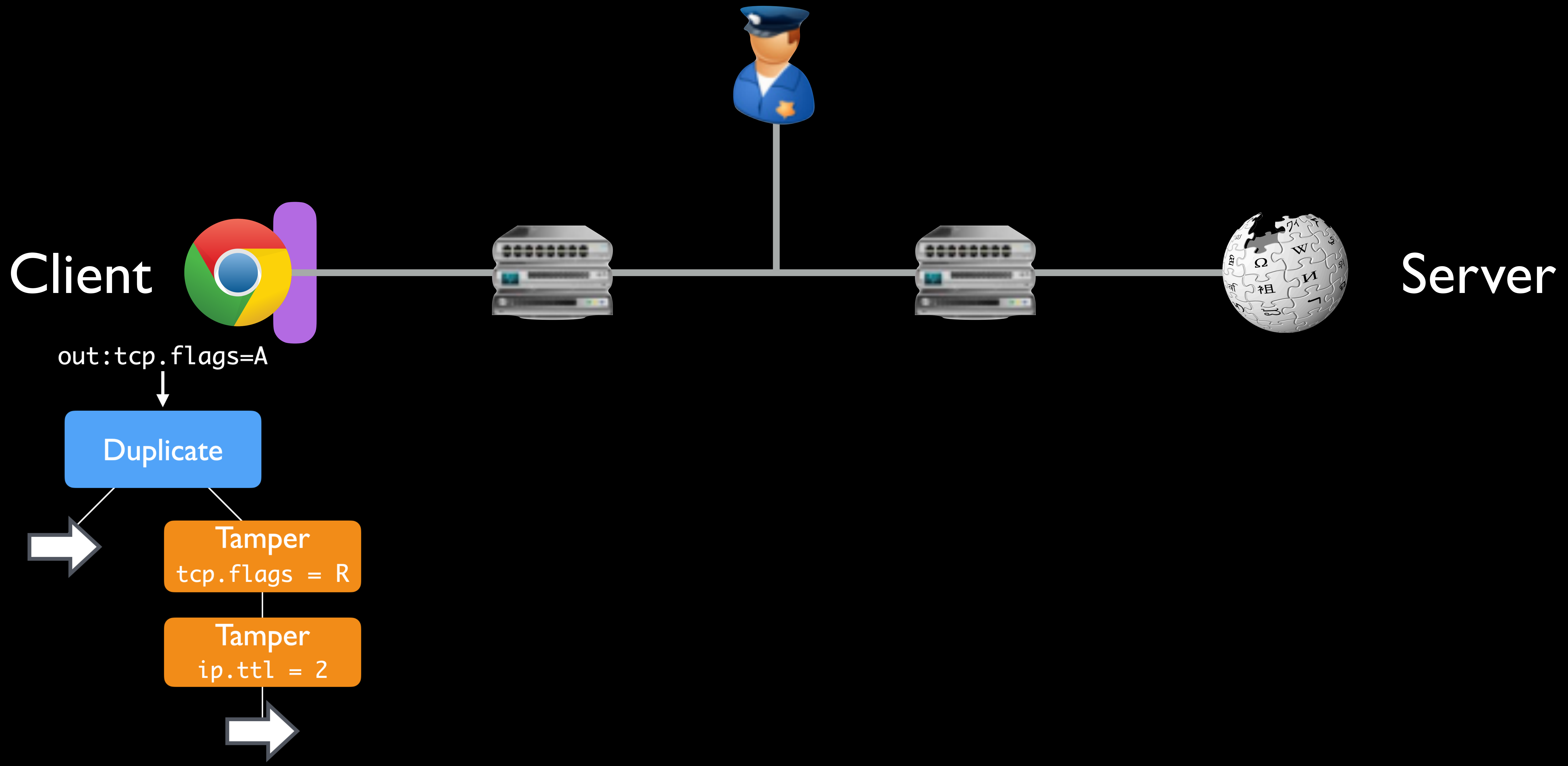
Genetic Evasion

Manipulates packets to and from the client



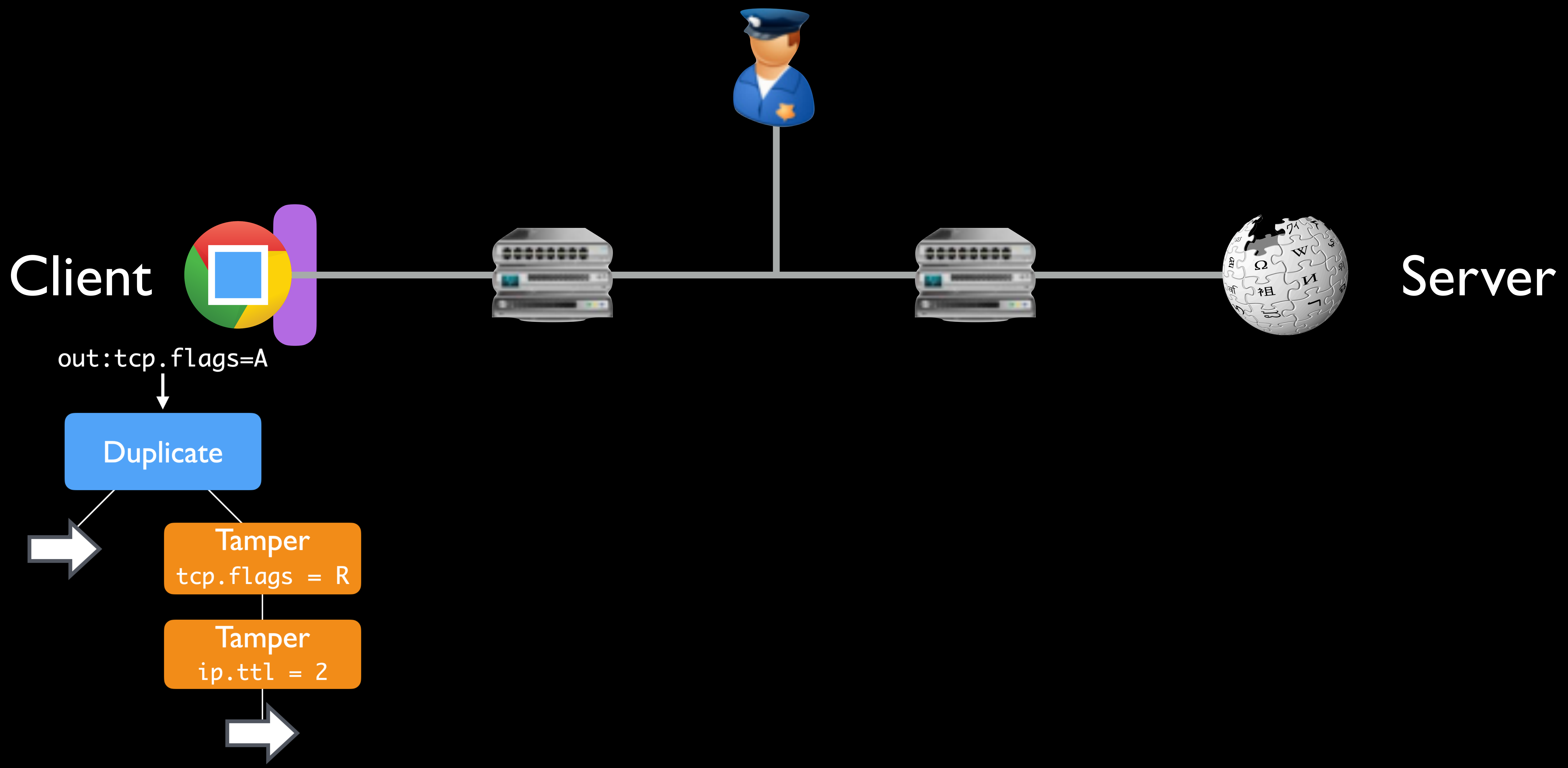
Bock et al. CCS'19

Running a Strategy



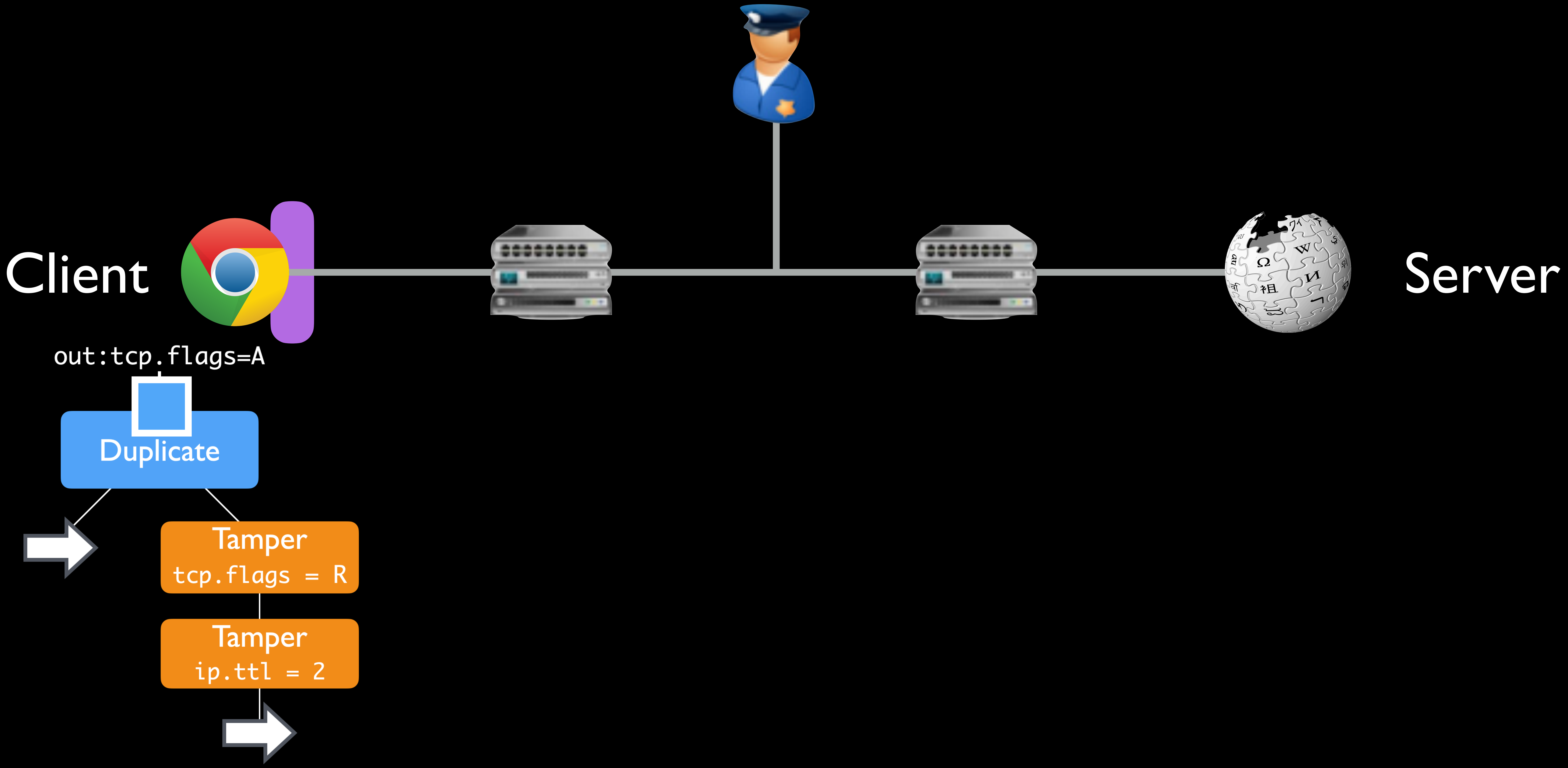
Bock et al. CCS'19

Running a Strategy



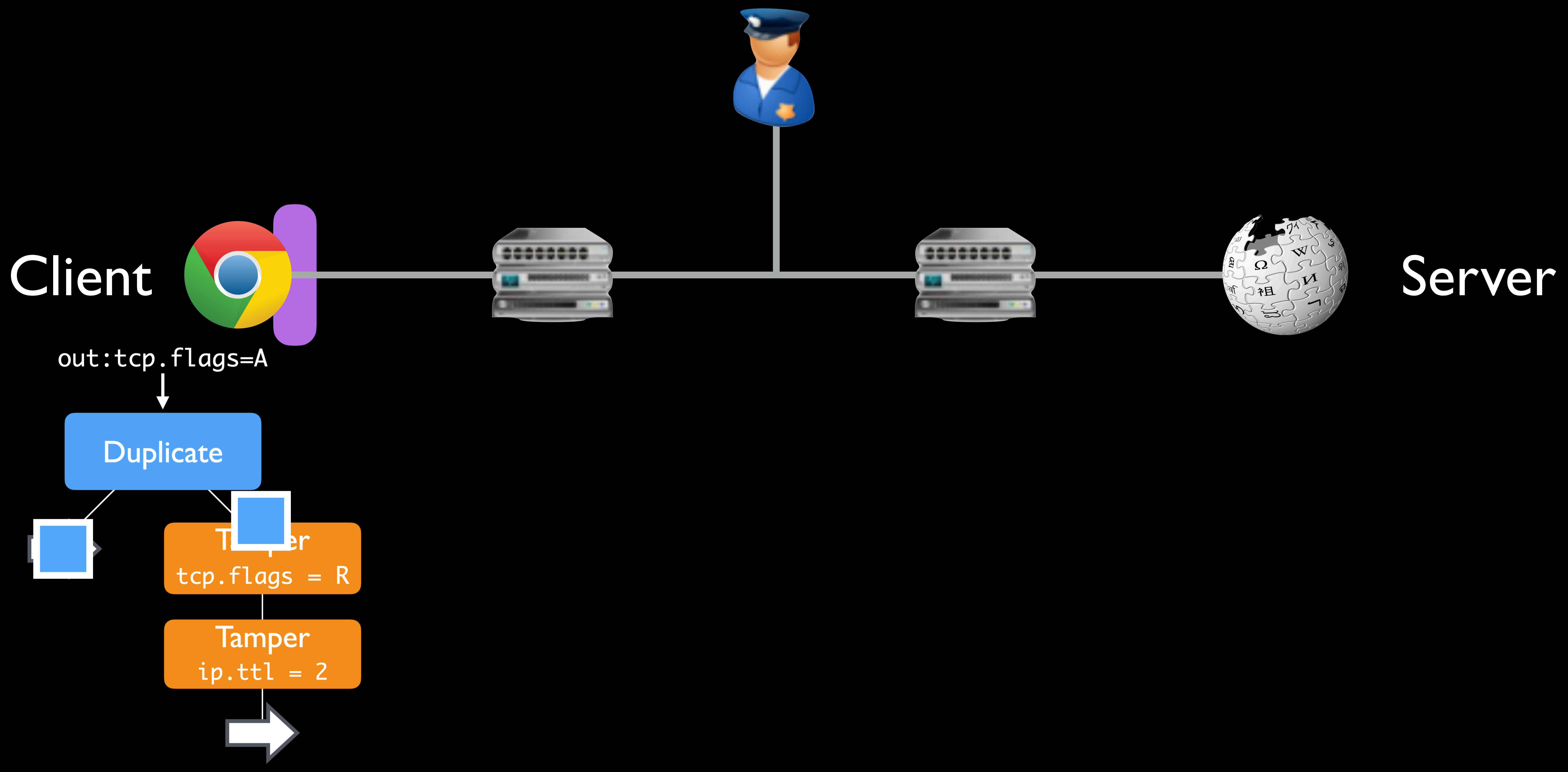
Bock et al. CCS'19

Running a Strategy



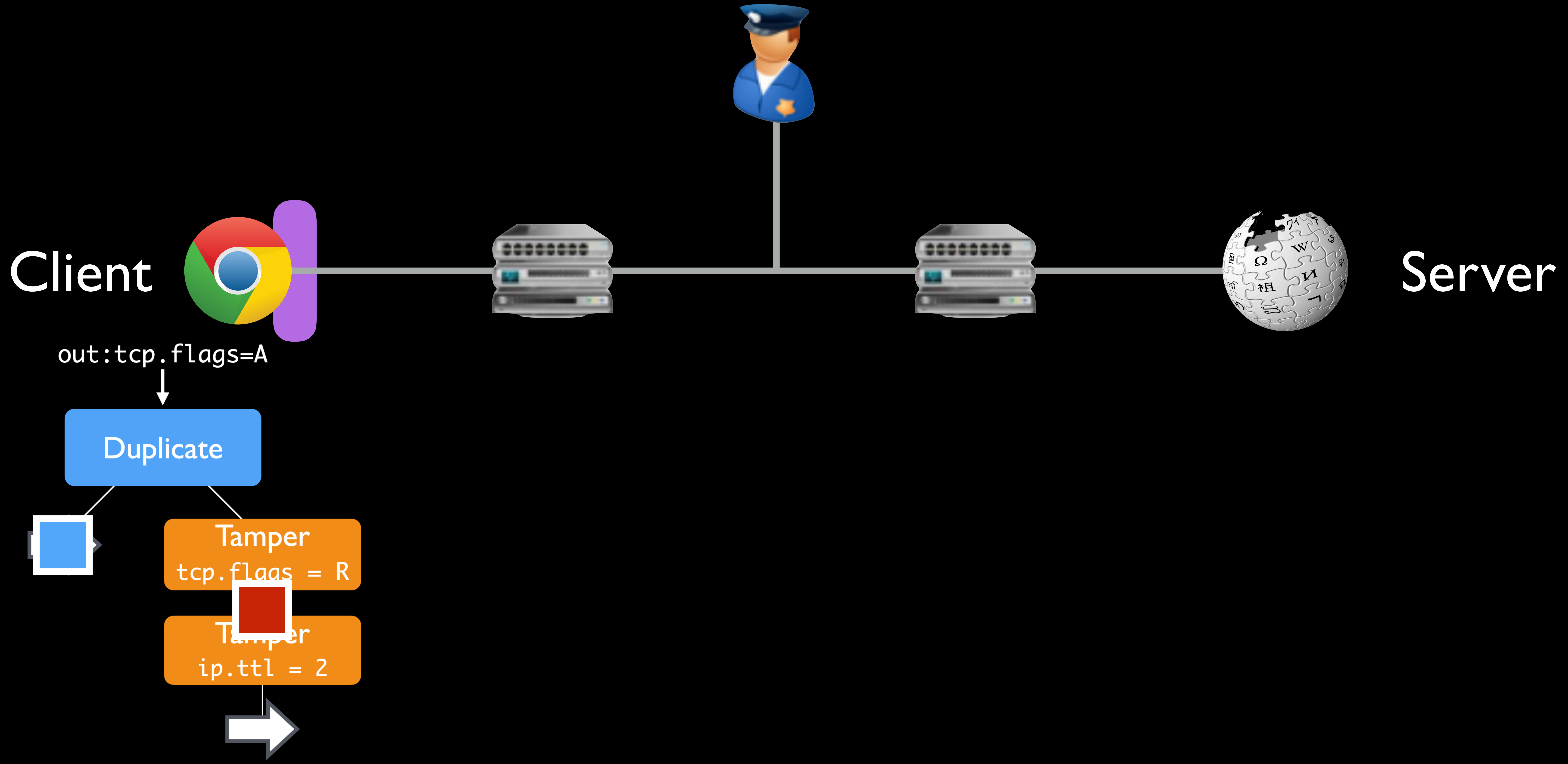
Bock et al. CCS'19

Running a Strategy



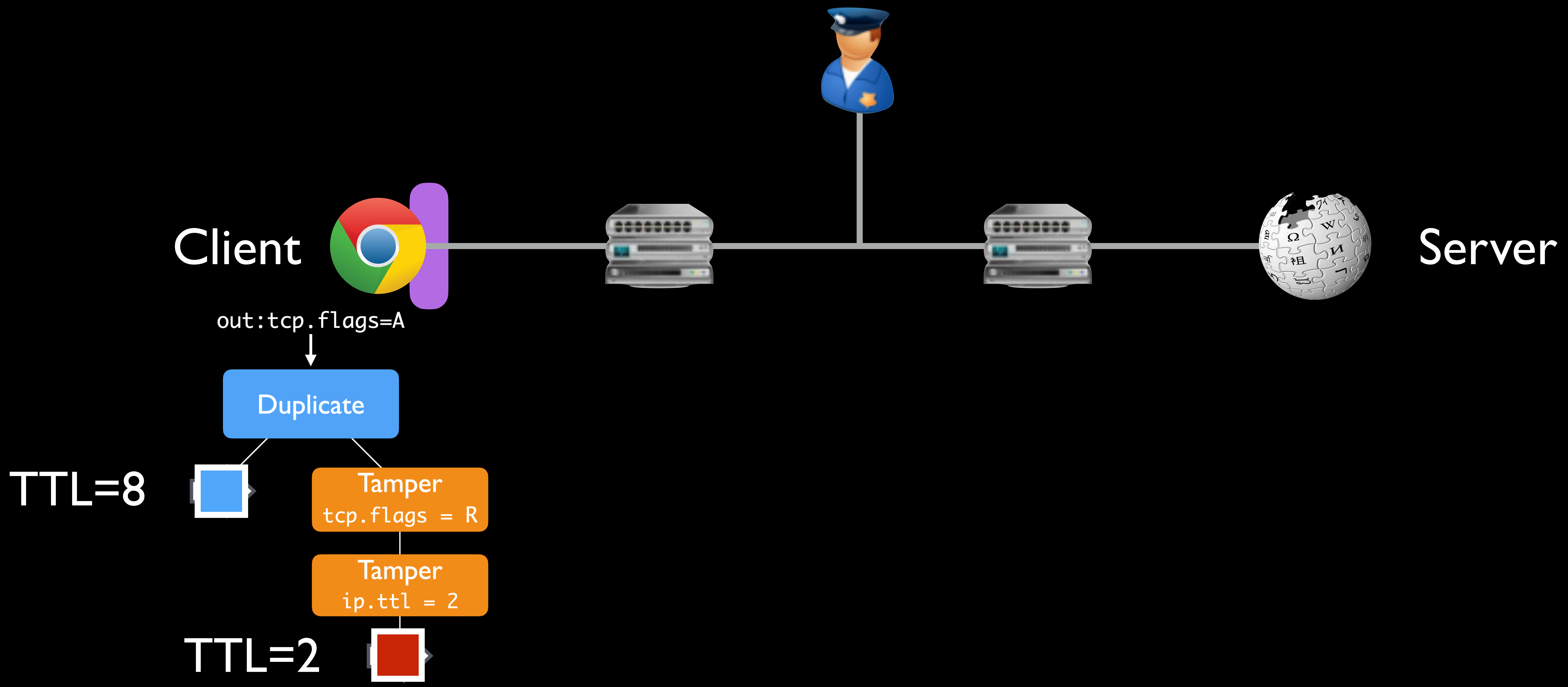
Bock et al. CCS'19

Running a Strategy



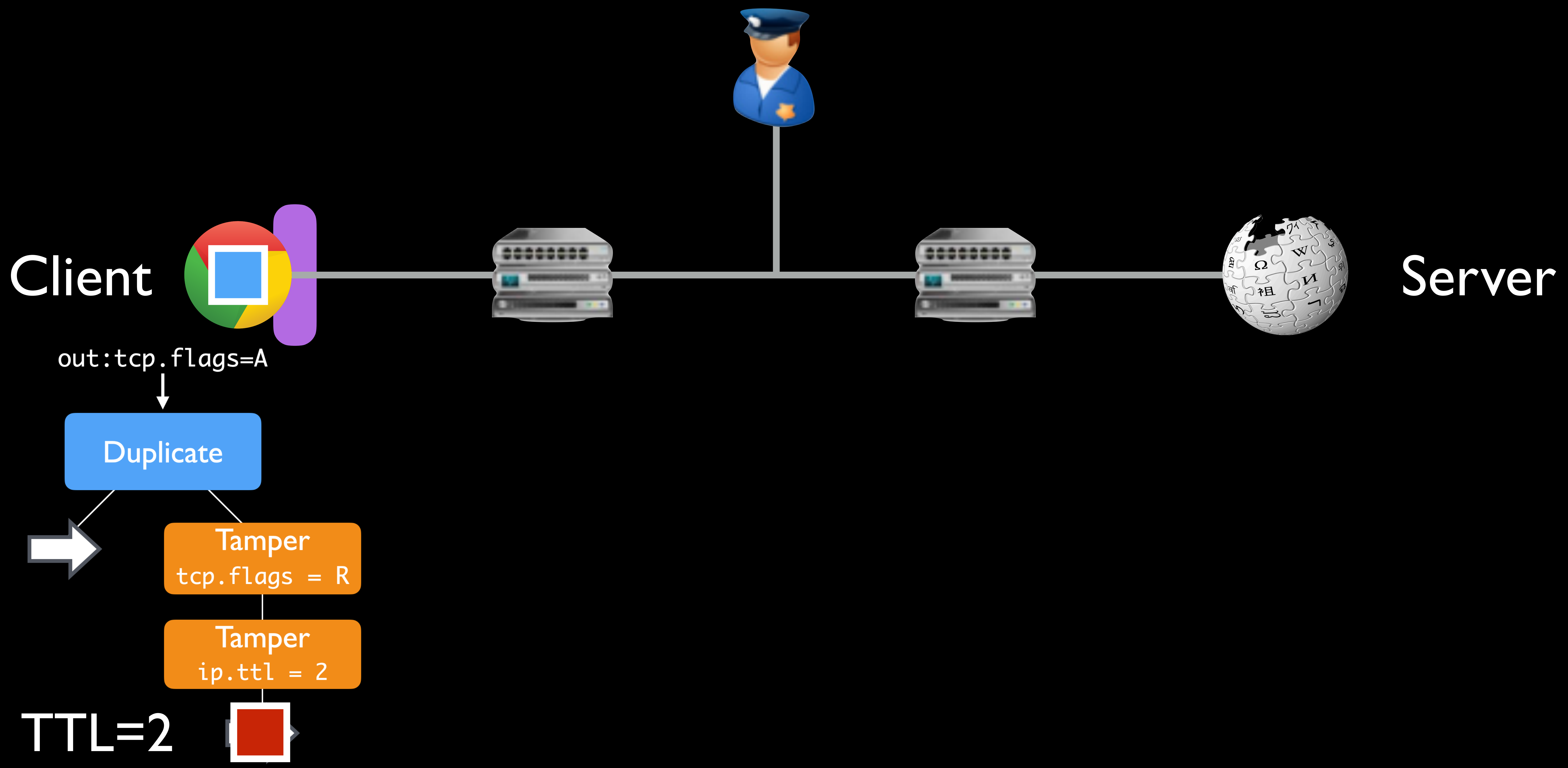
Bock et al. CCS'19

Running a Strategy



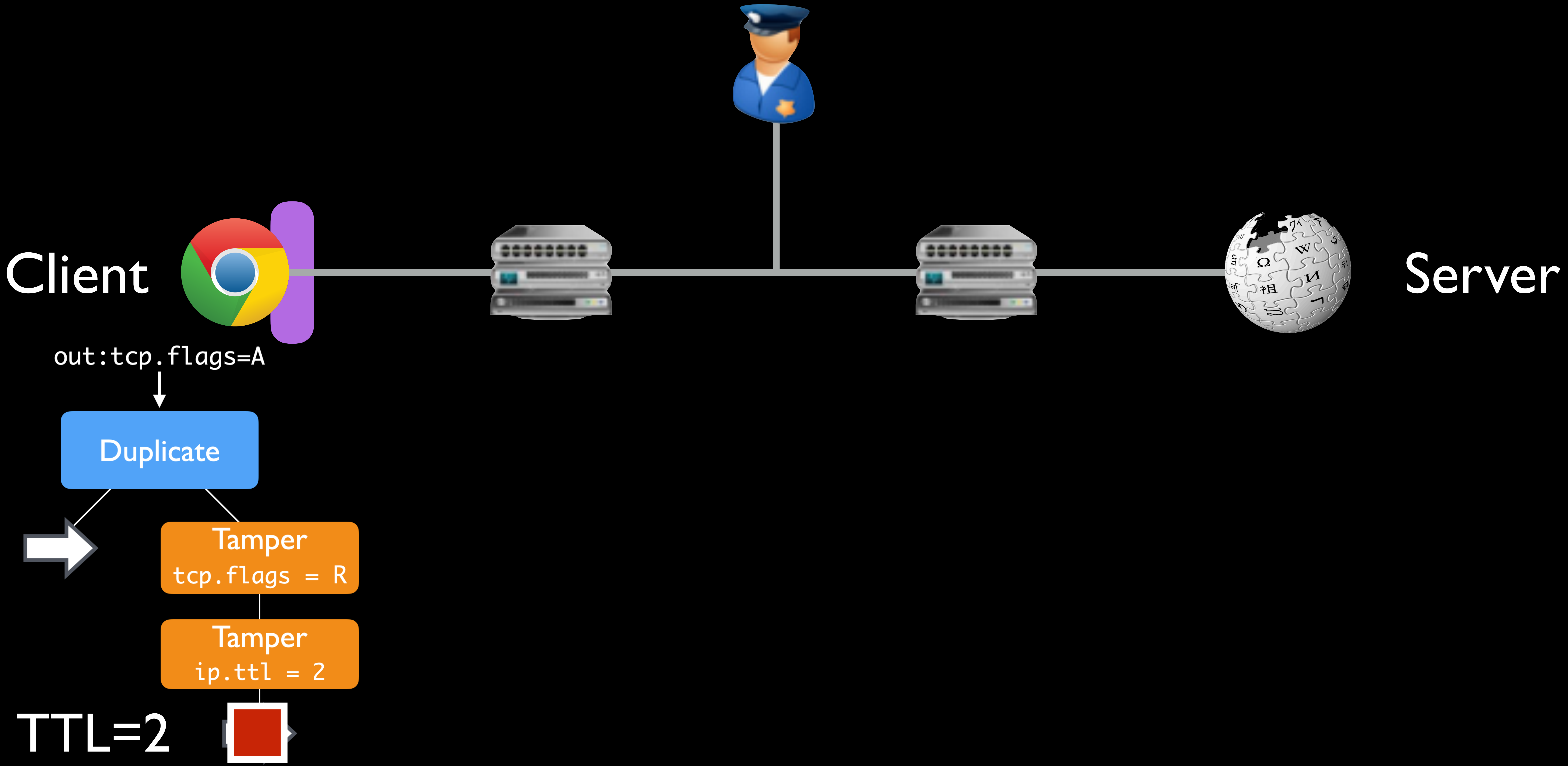
Bock et al. CCS'19

Running a Strategy



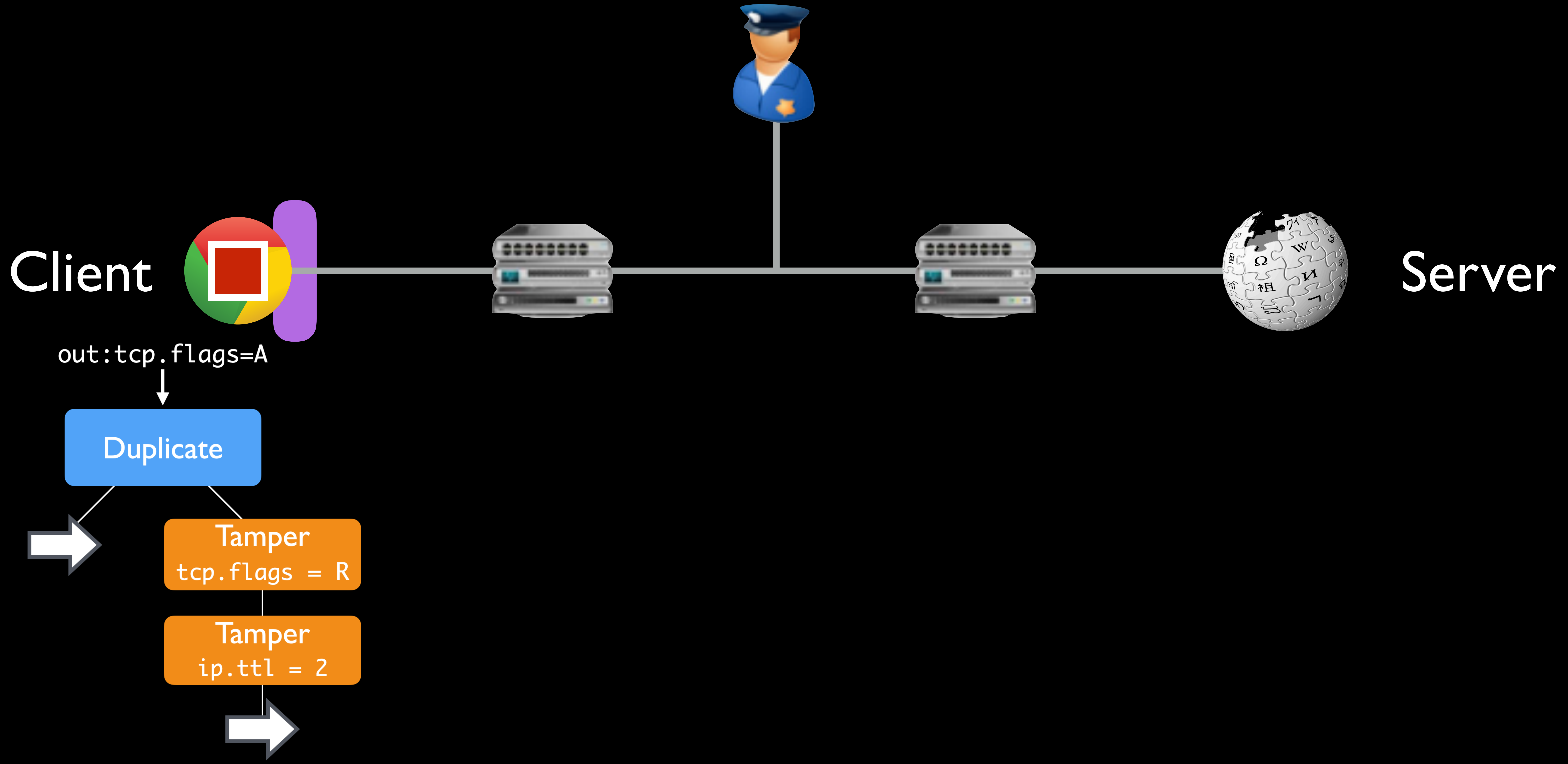
Bock et al. CCS'19

Running a Strategy



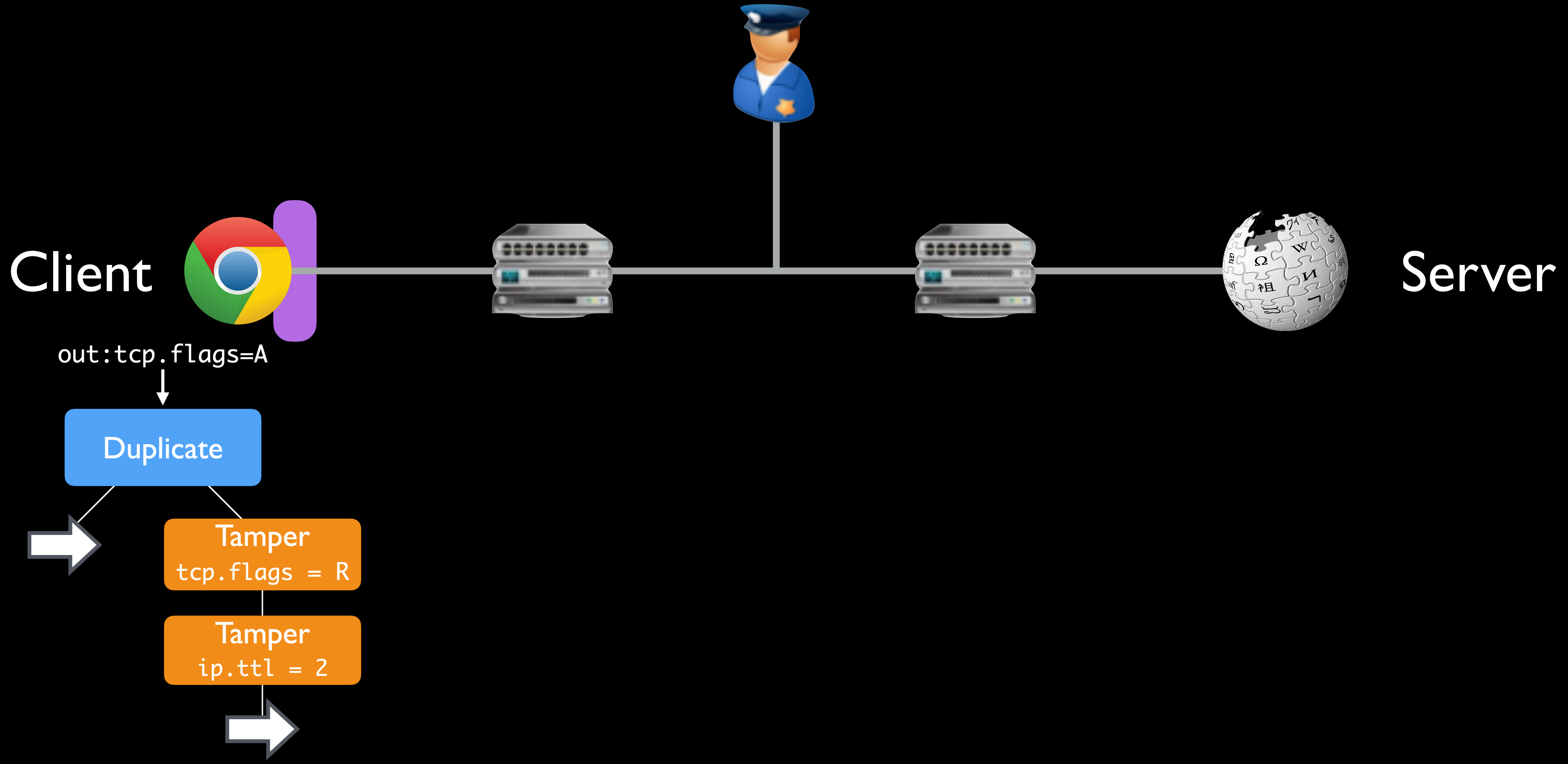
Bock et al. CCS'19

Running a Strategy



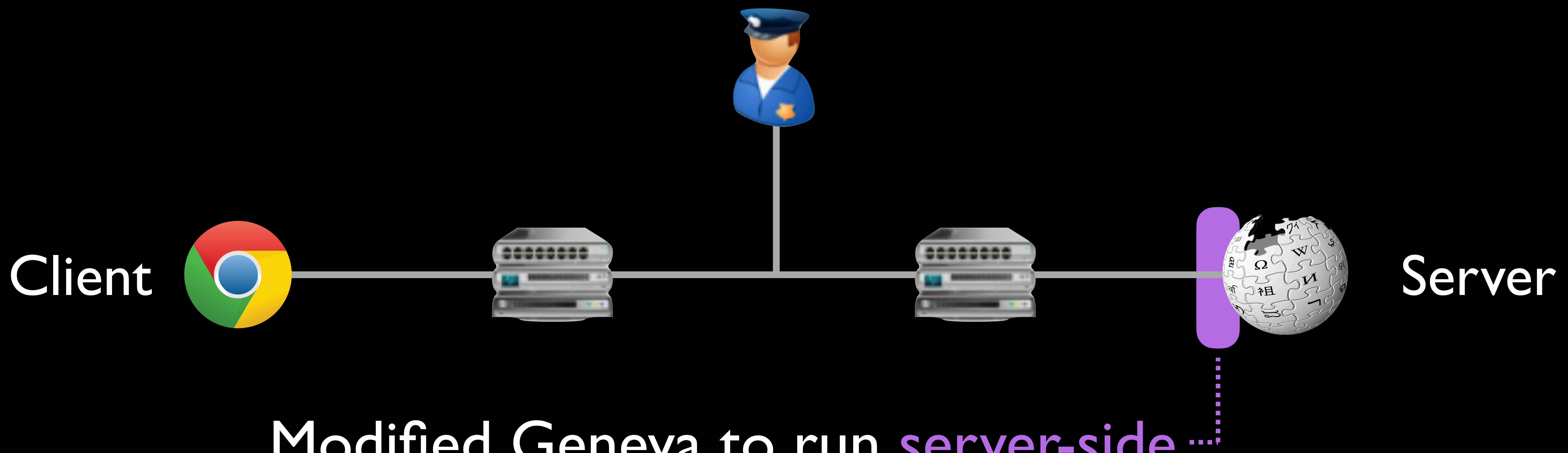
Bock et al. CCS'19

Running a Strategy



Bock et al. CCS'19

This paper: Server-side Geneva



Modified Geneva to run server-side

Deployed against real world censors

Results against real censors

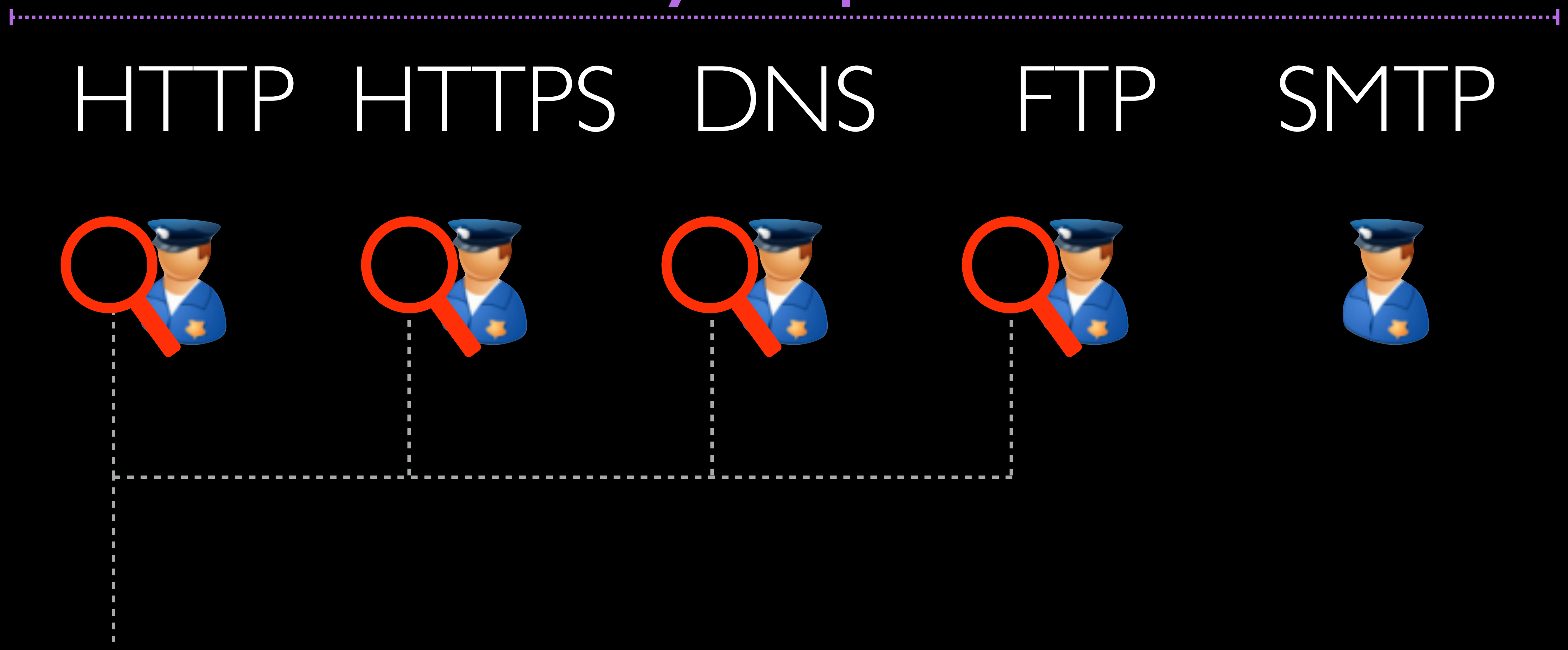
Diversity of protocols

HTTP	HTTPS	DNS	FTP	SMTP
👮	👮	👮	👮	👮

Results against real censors

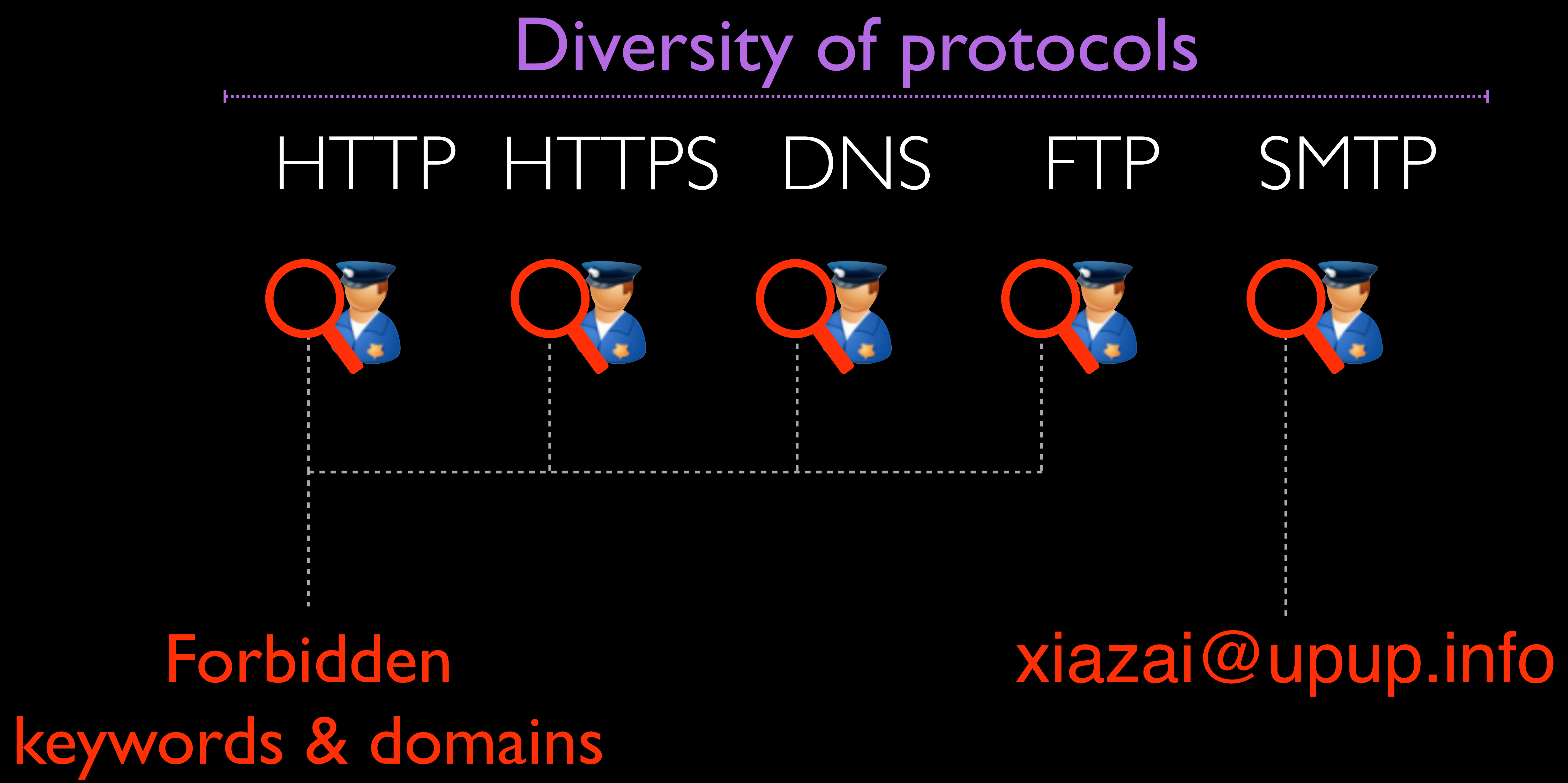
Diversity of protocols

HTTP HTTPS DNS FTP SMTP



Forbidden
keywords & domains

Results against real censors



Diversity of protocols

HTTP HTTPS DNS FTP SMTP

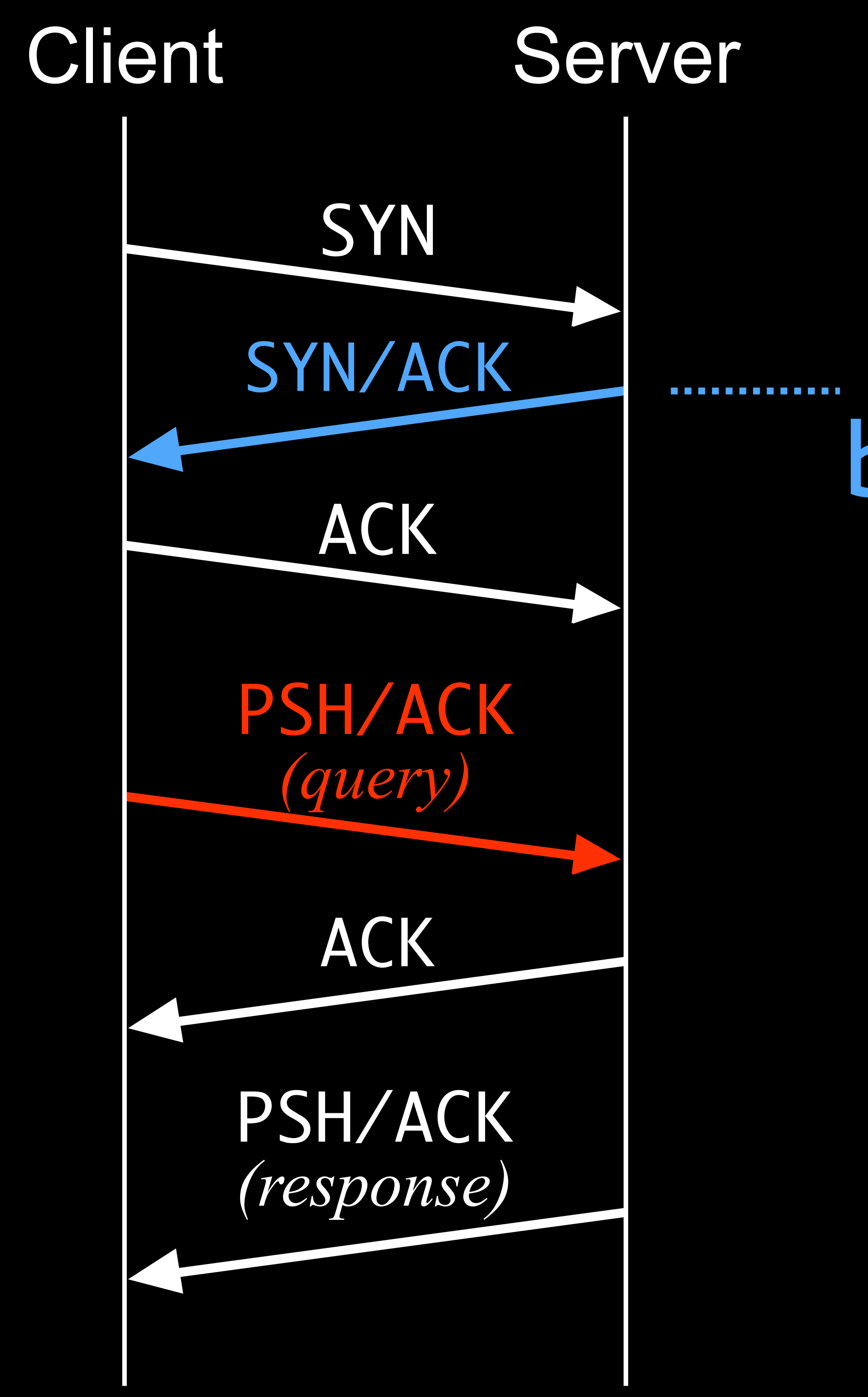
Forbidden keywords & domains

xiazai@upup.info

Results against real censors

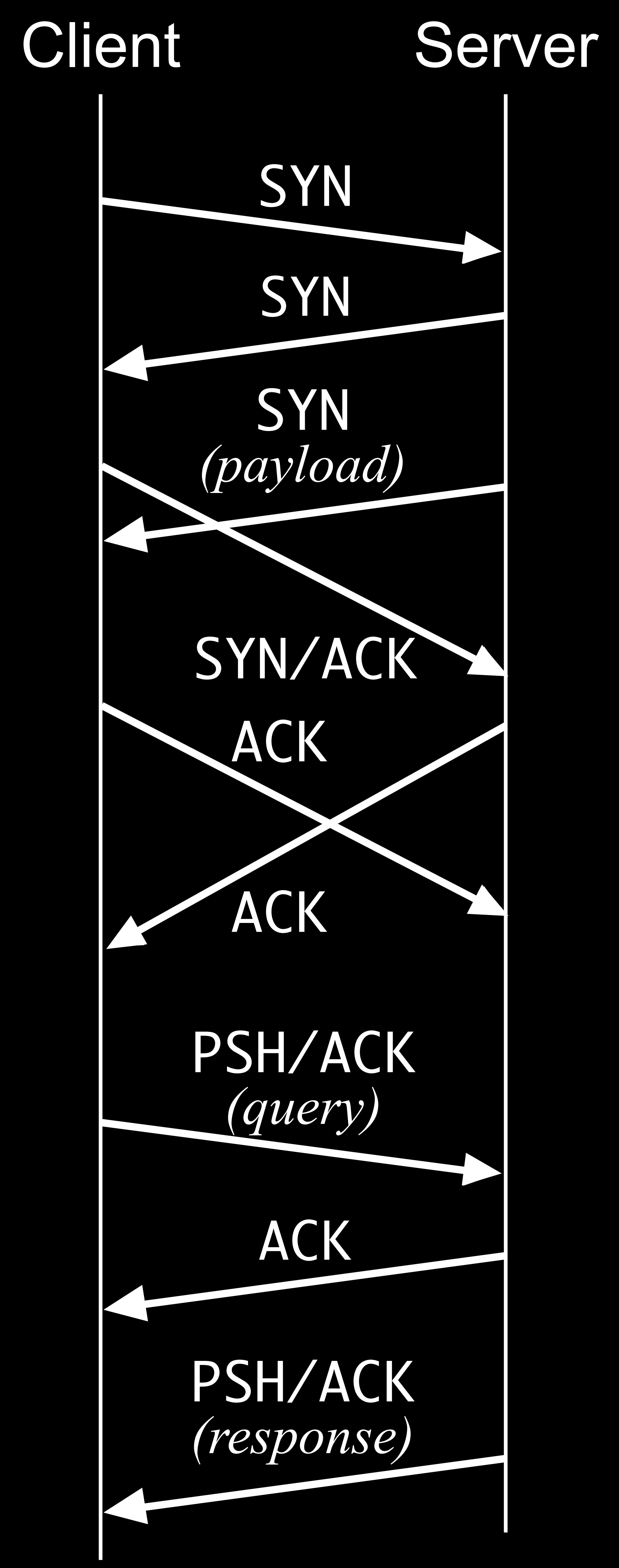
Diversity of censors		Diversity of protocols				
		HTTP	HTTPS	DNS	FTP	SMTP
Injects TCP RSTs	China	👮	👮	👮	👮	👮
Injects & blackholes	Iran	👮	👮	👮*		
Injects & blackholes	Kazakhstan	👮	👮			
Injects a block page	India	👮	👮			

Server-side evasion “shouldn’t” work

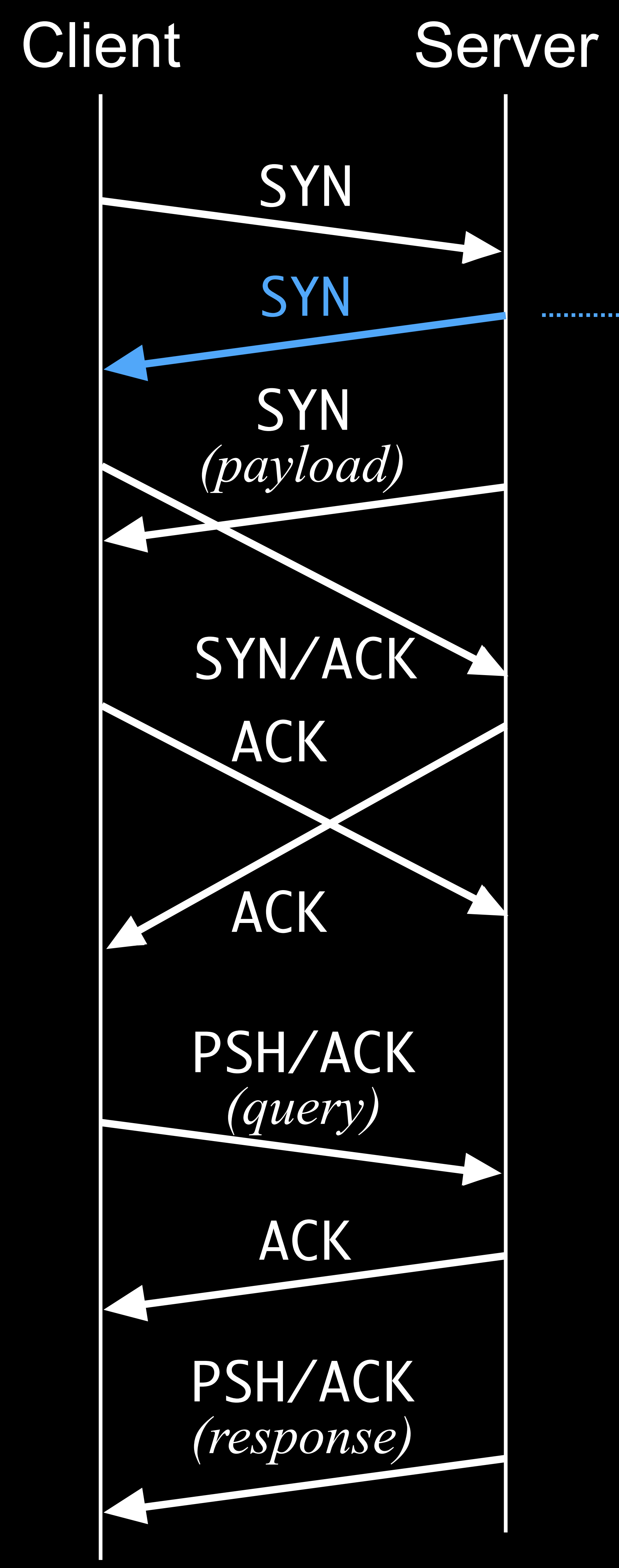


All a server does before client is censored

A successful server-side evasion strategy

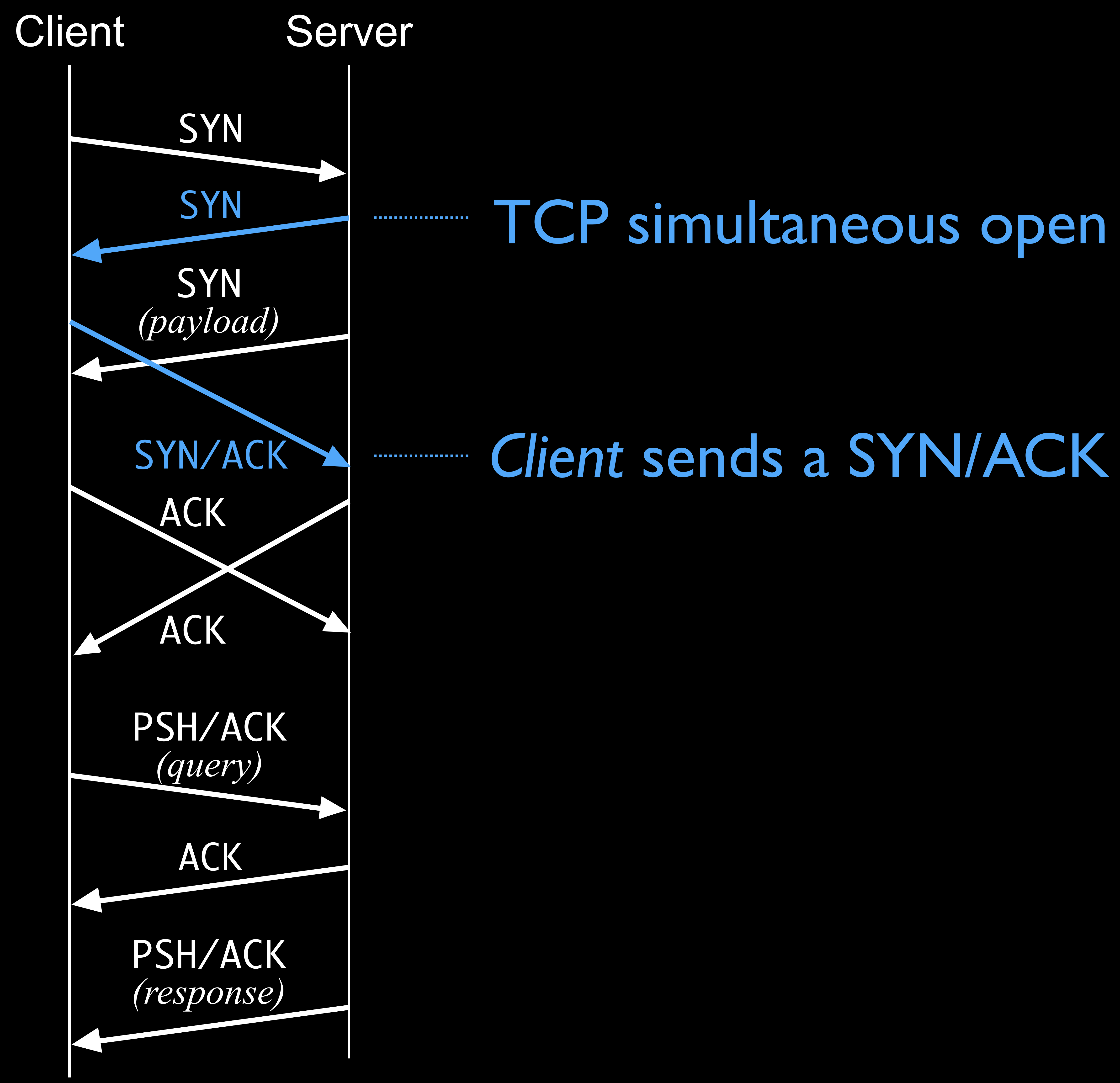


A successful server-side evasion strategy



TCP simultaneous open

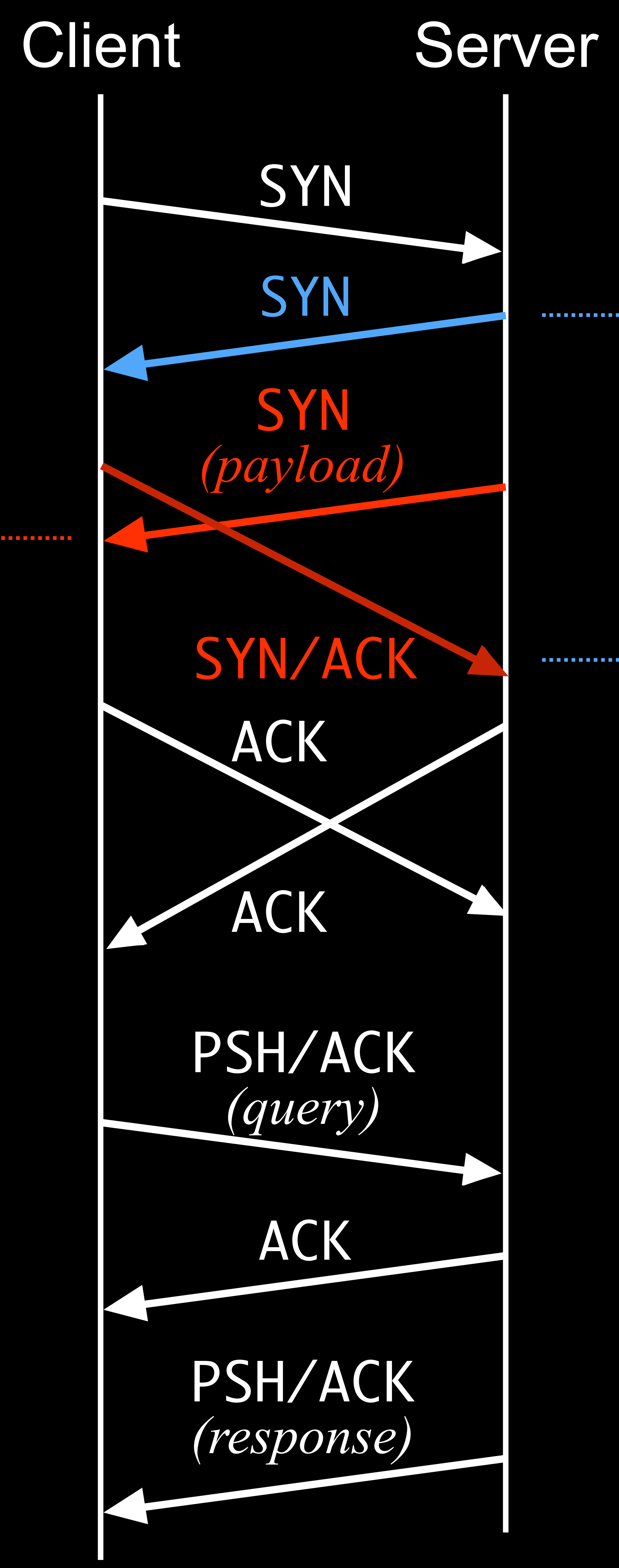
A successful server-side evasion strategy



TCP simultaneous open

Client sends a SYN/ACK

A successful server-side evasion strategy

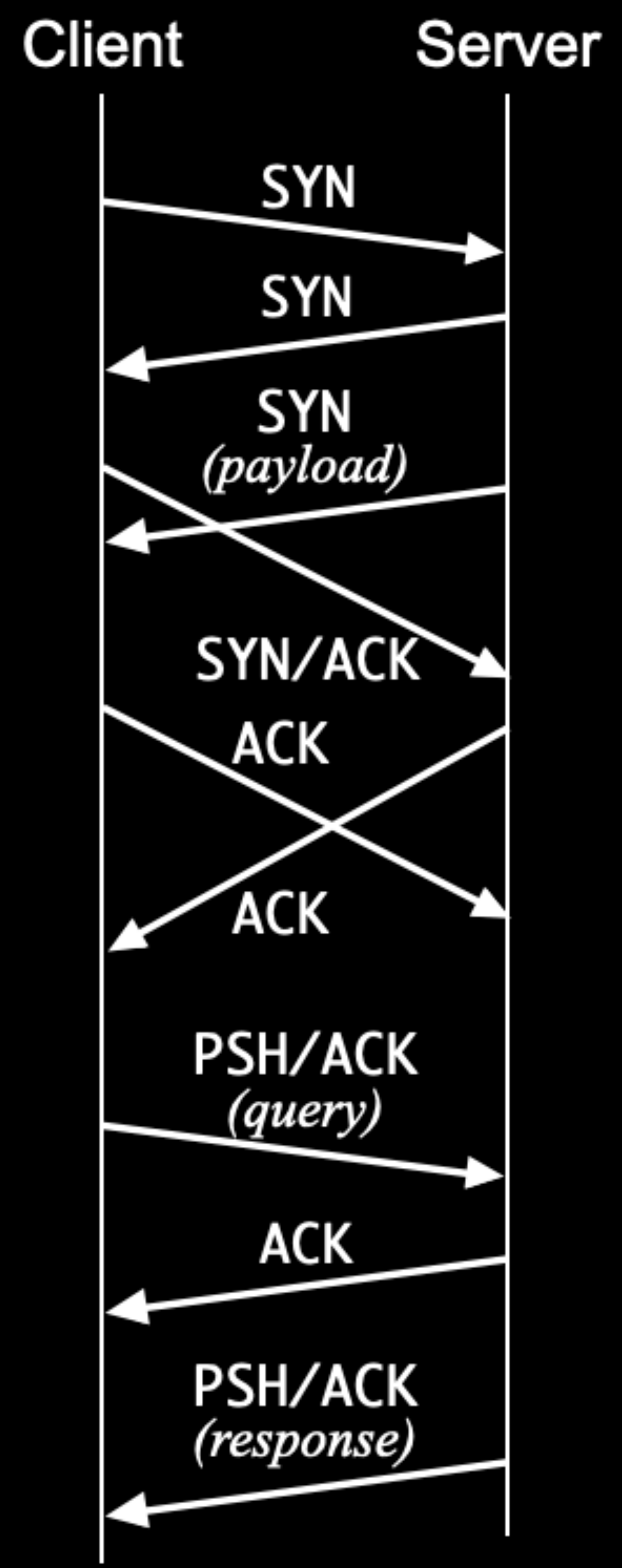


TCP simultaneous open

Censor de-synchronizes

Client sends a SYN/ACK

A successful server-side evasion strategy



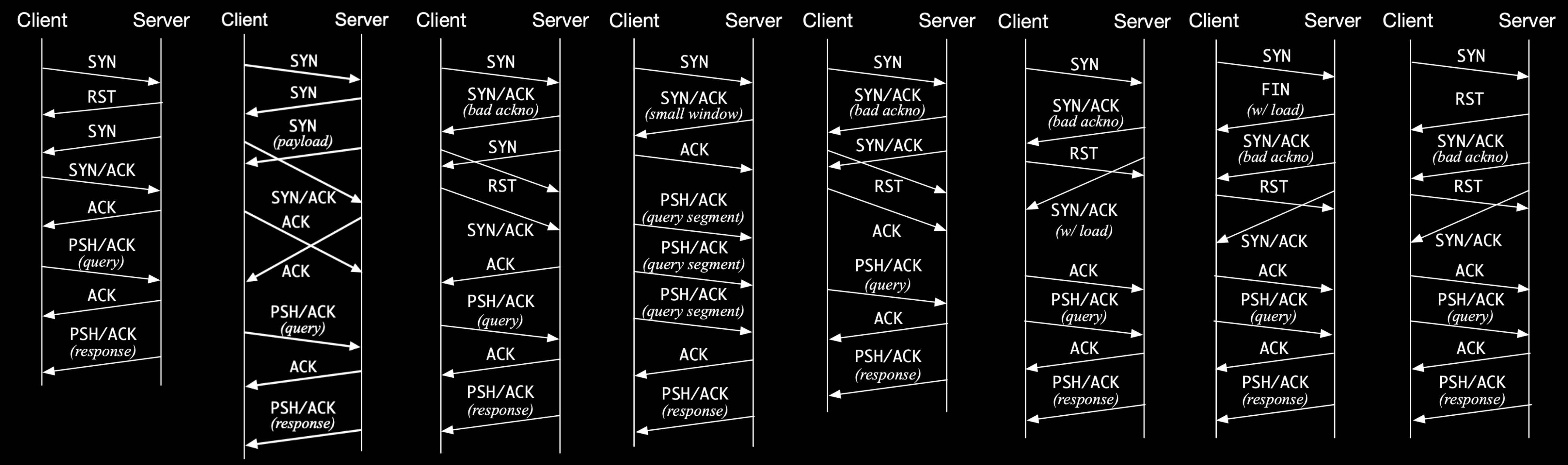
Success rates

DNS	89%
FTP	36%
HTTP	54%
HTTPS	55%
SMTP	70%

Server-side evasion strategies

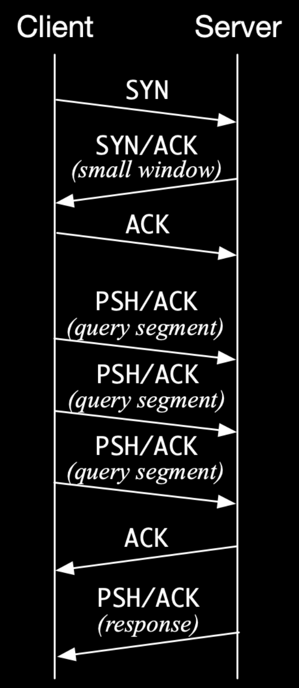
China

8 strategies



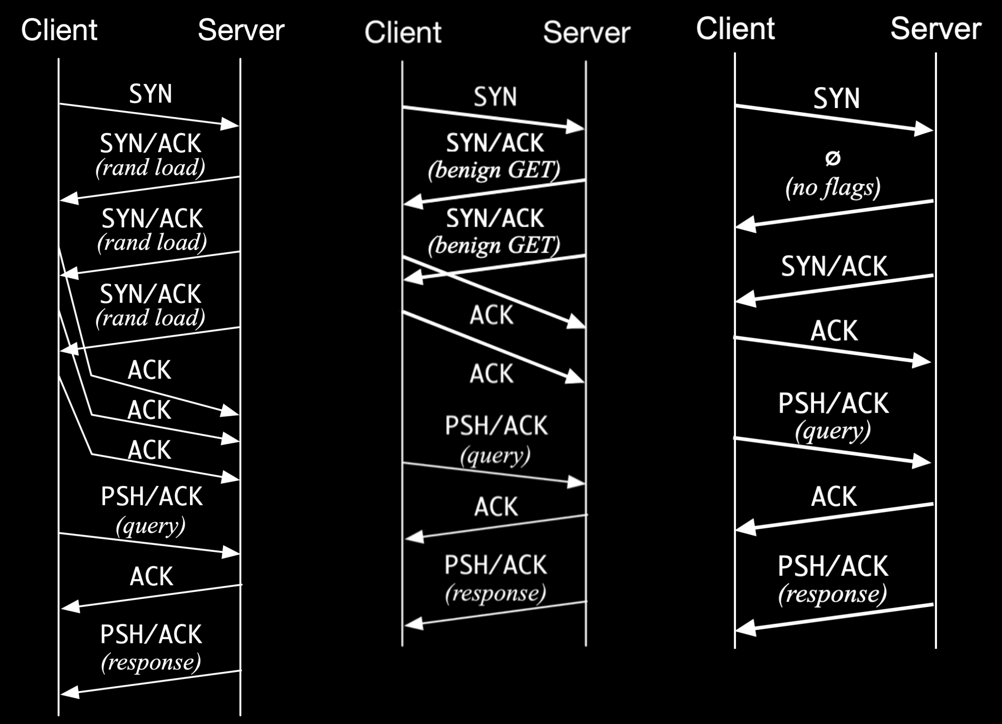
Iran/India

1 strategy



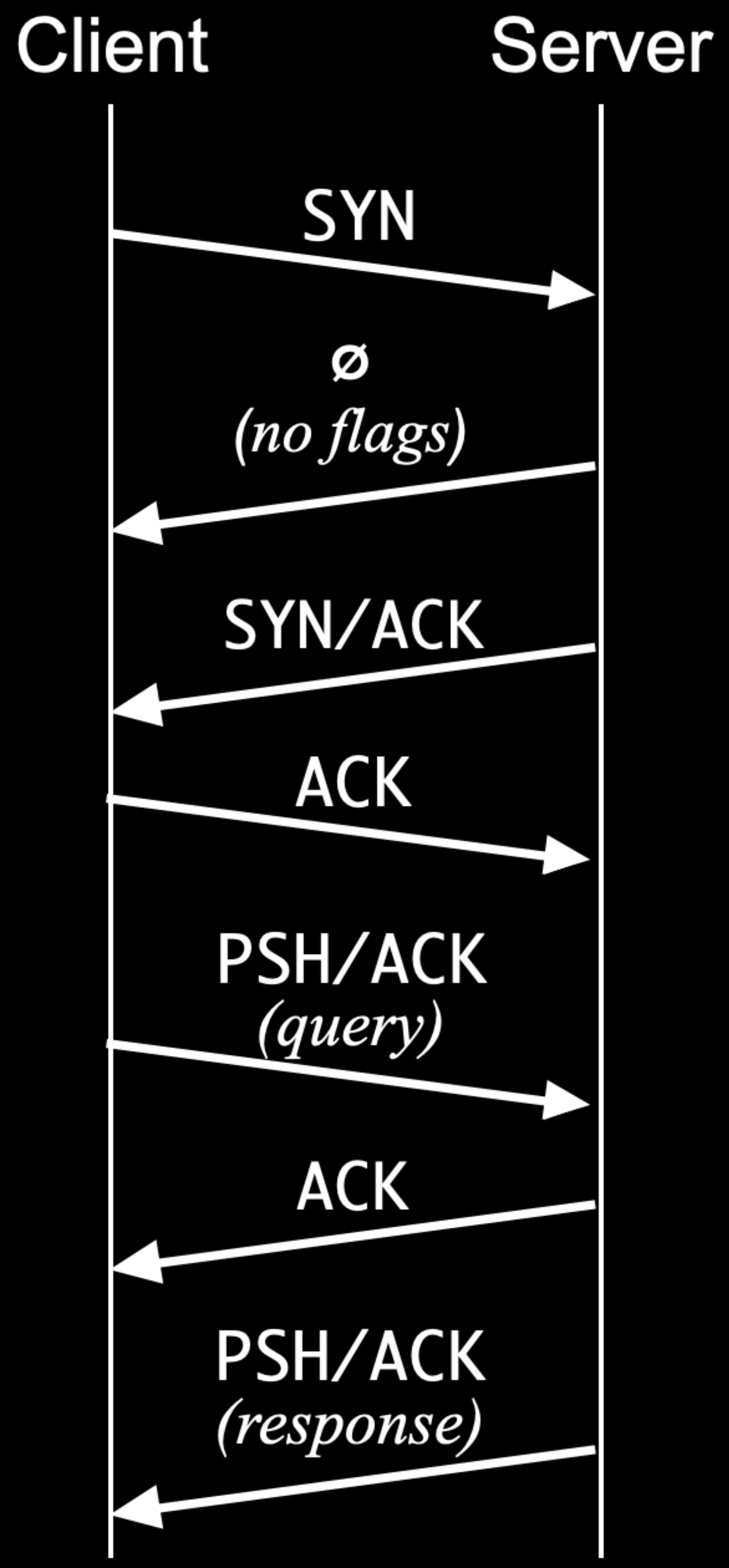
Kazakhstan

3 strategies



Server-side evasion results

NULL TCP Flags

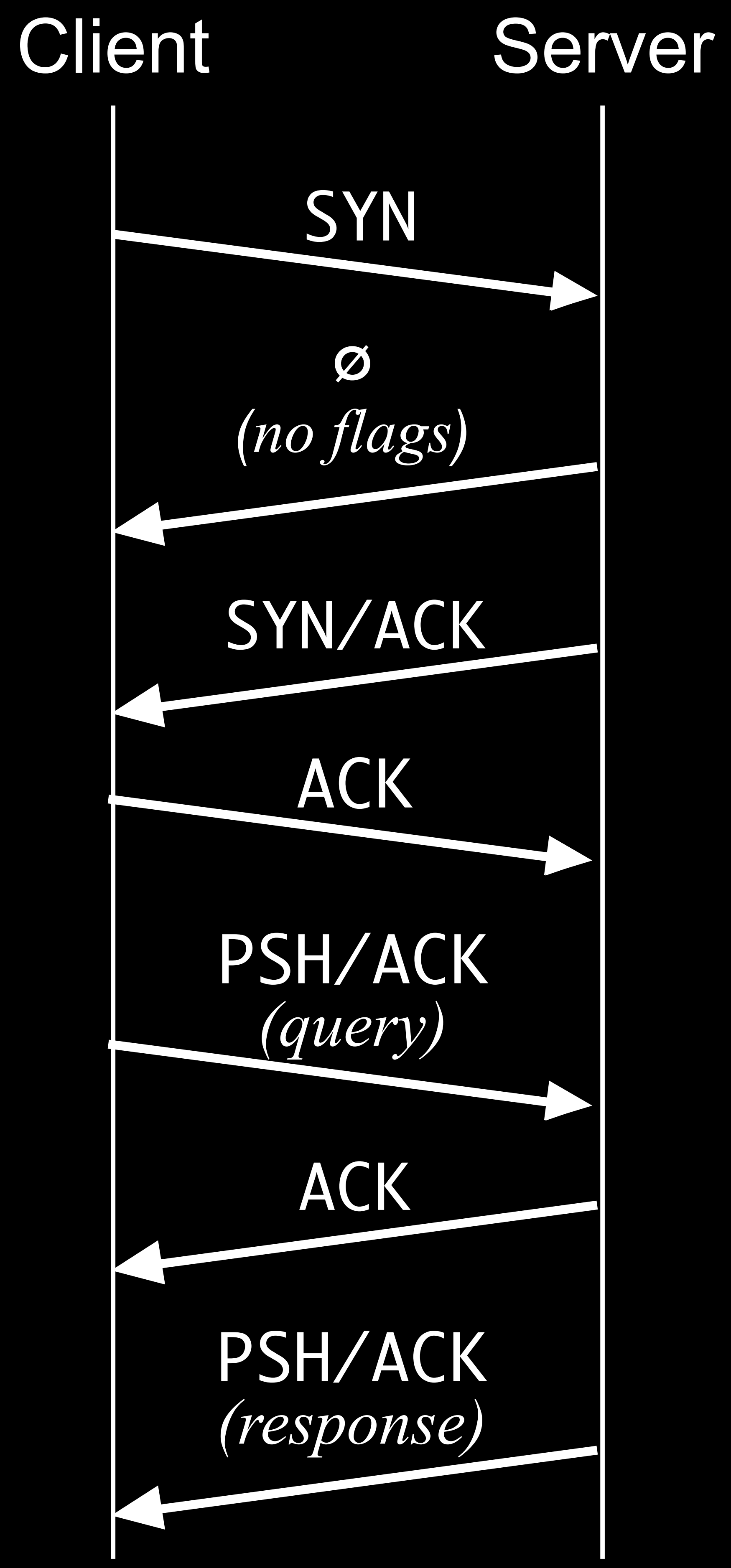


Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags

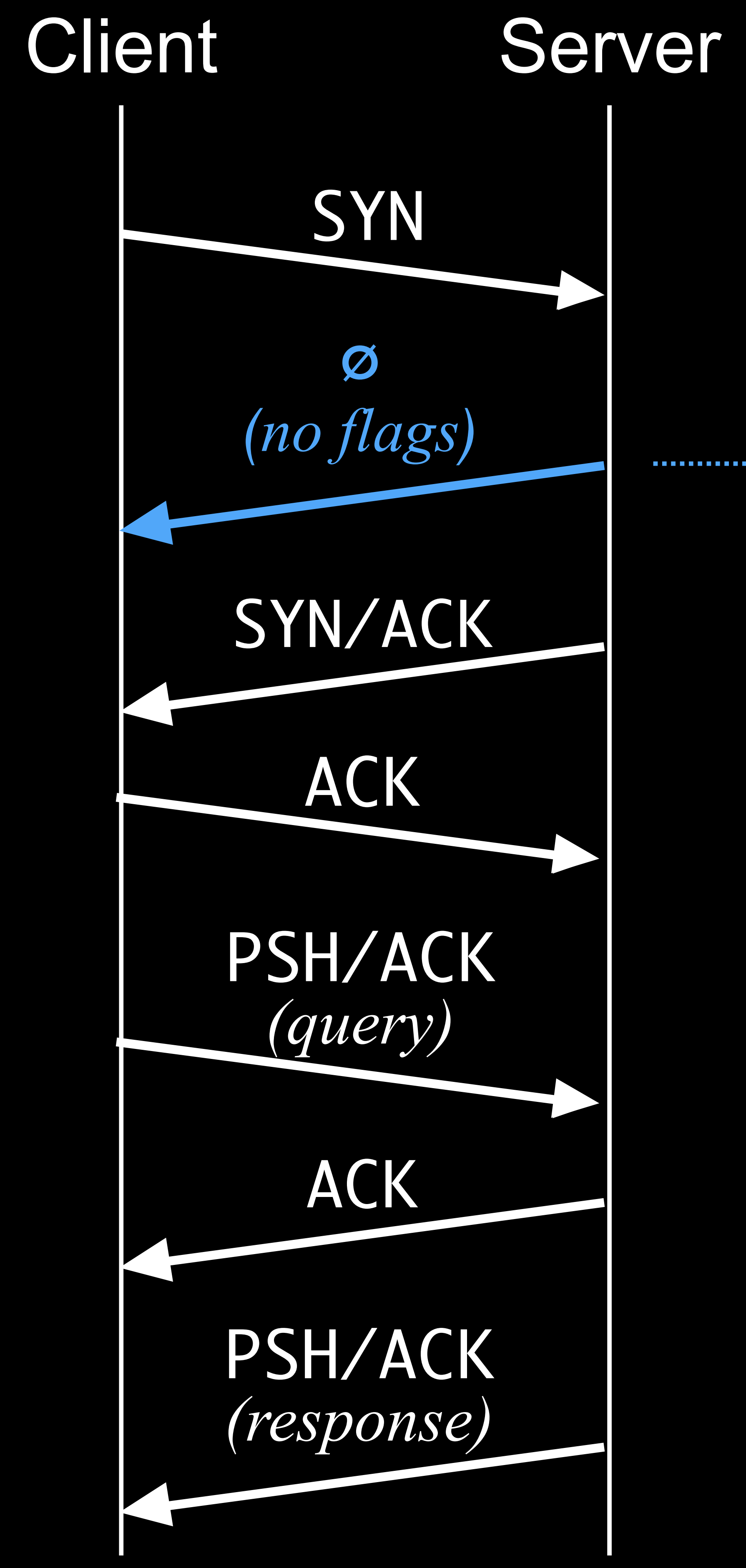


Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags



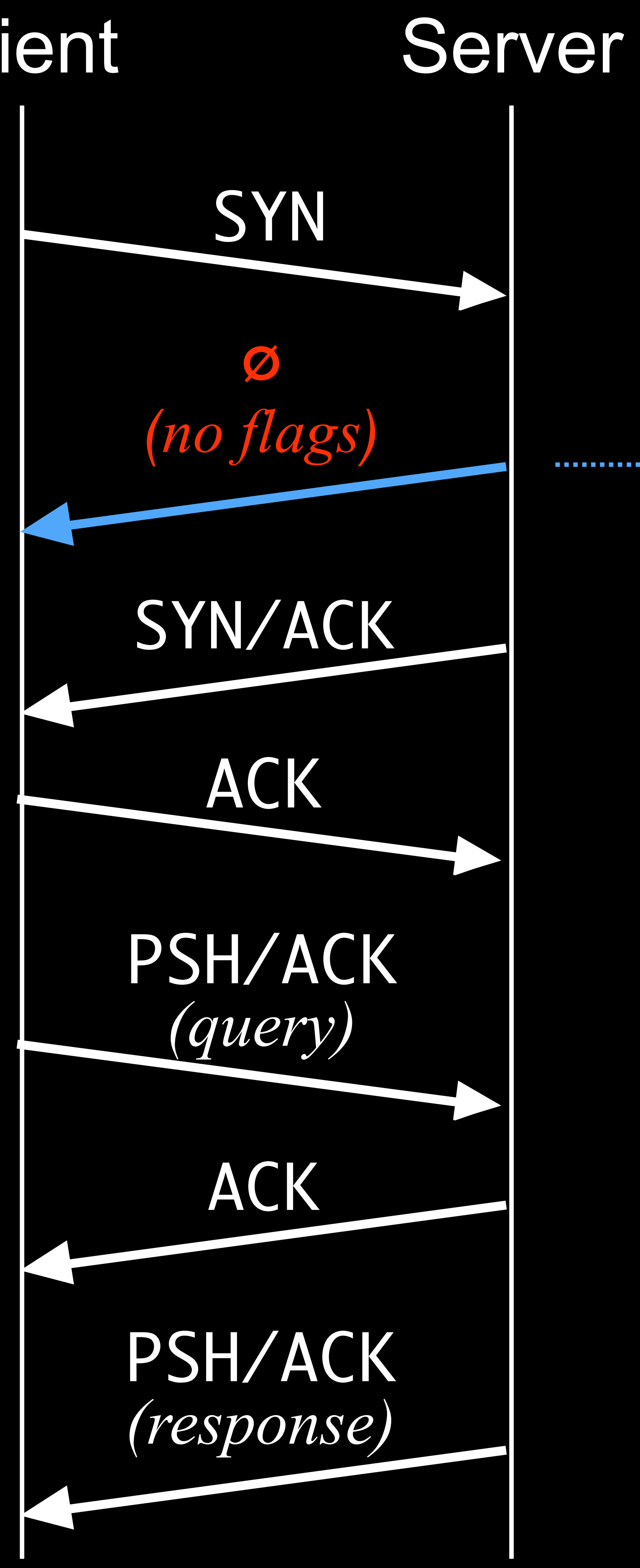
Server sends a packet with no TCP flags set

Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags



Censor can't handle unexpected flags

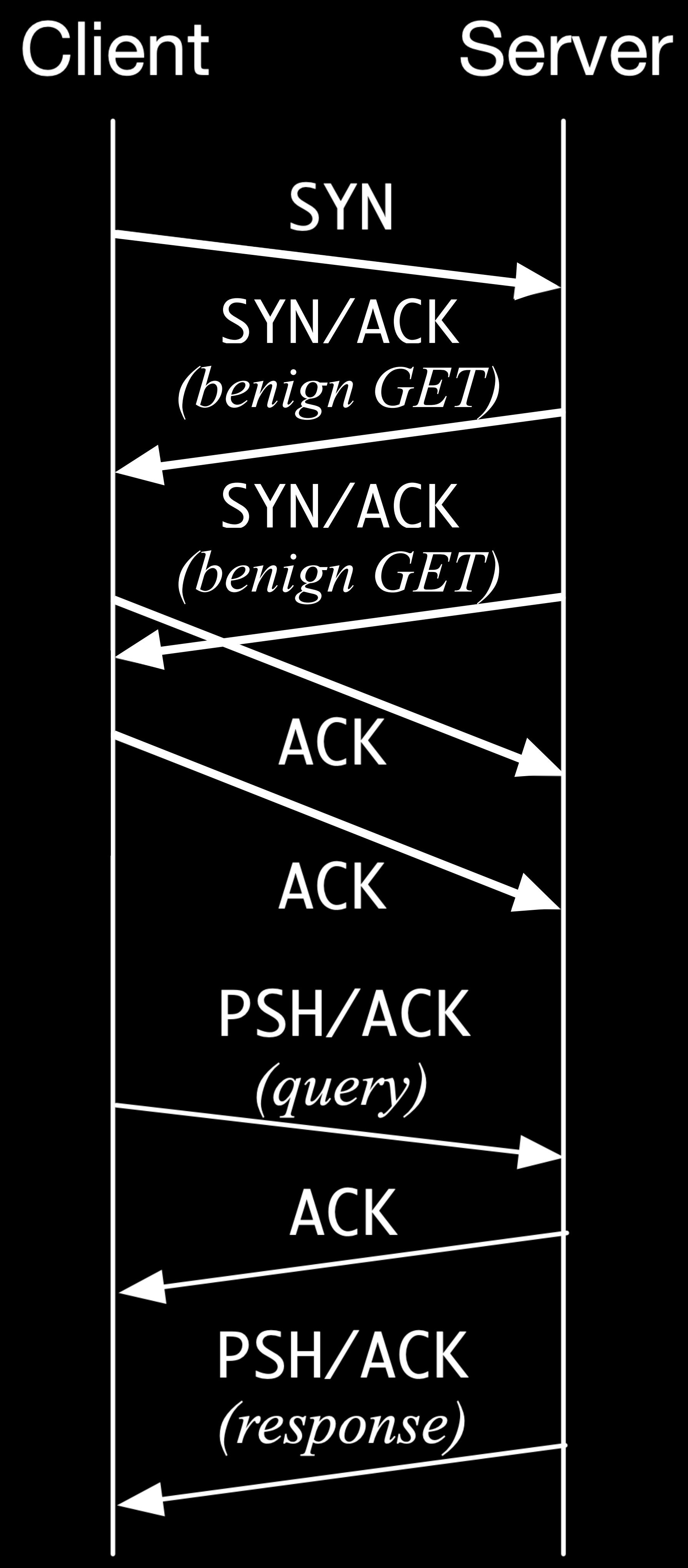
Server sends a packet with no TCP flags set

Success rates

HTTP	100%

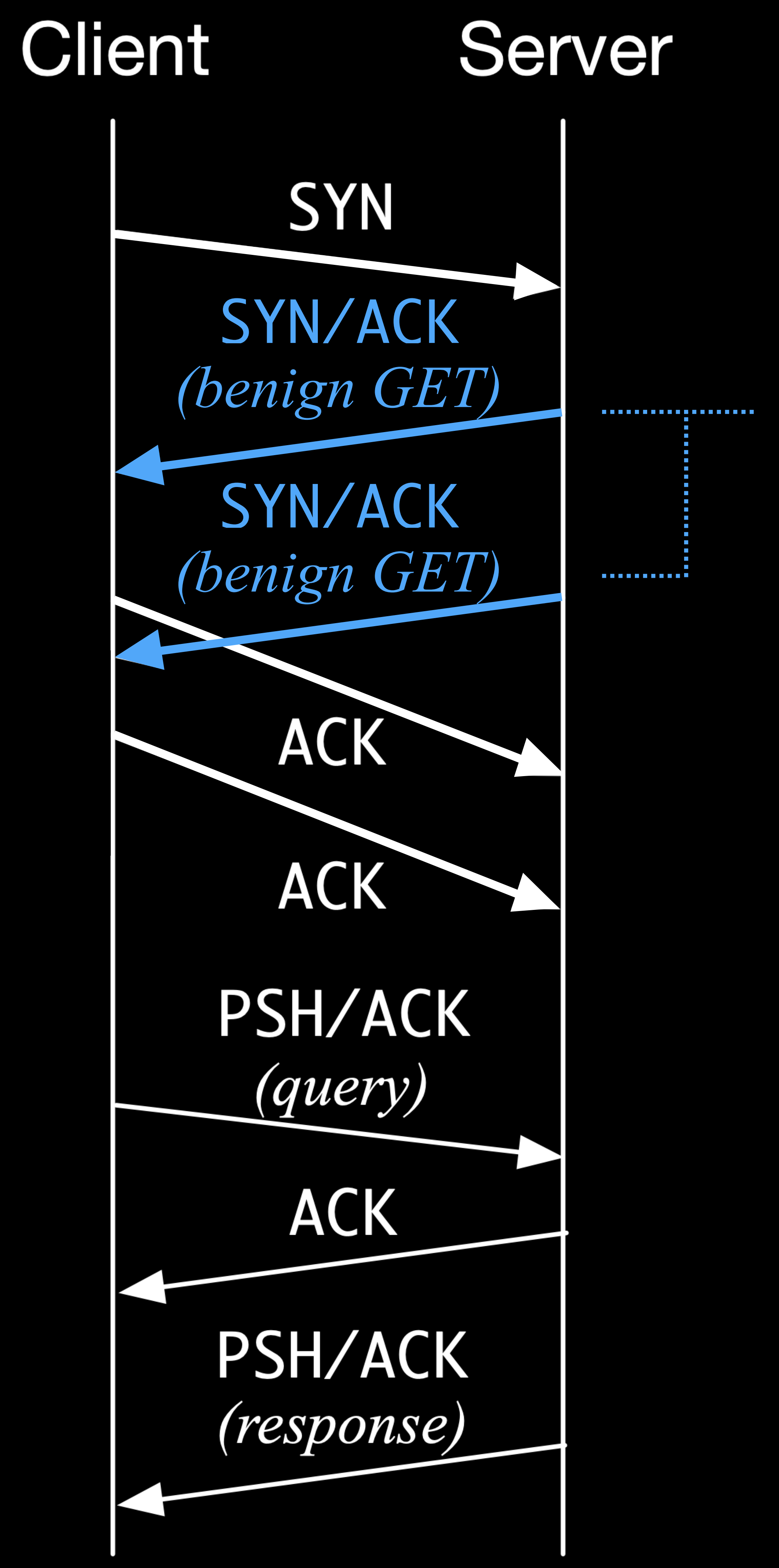
Server-side evasion results

Double benign-GETs



Server-side evasion results

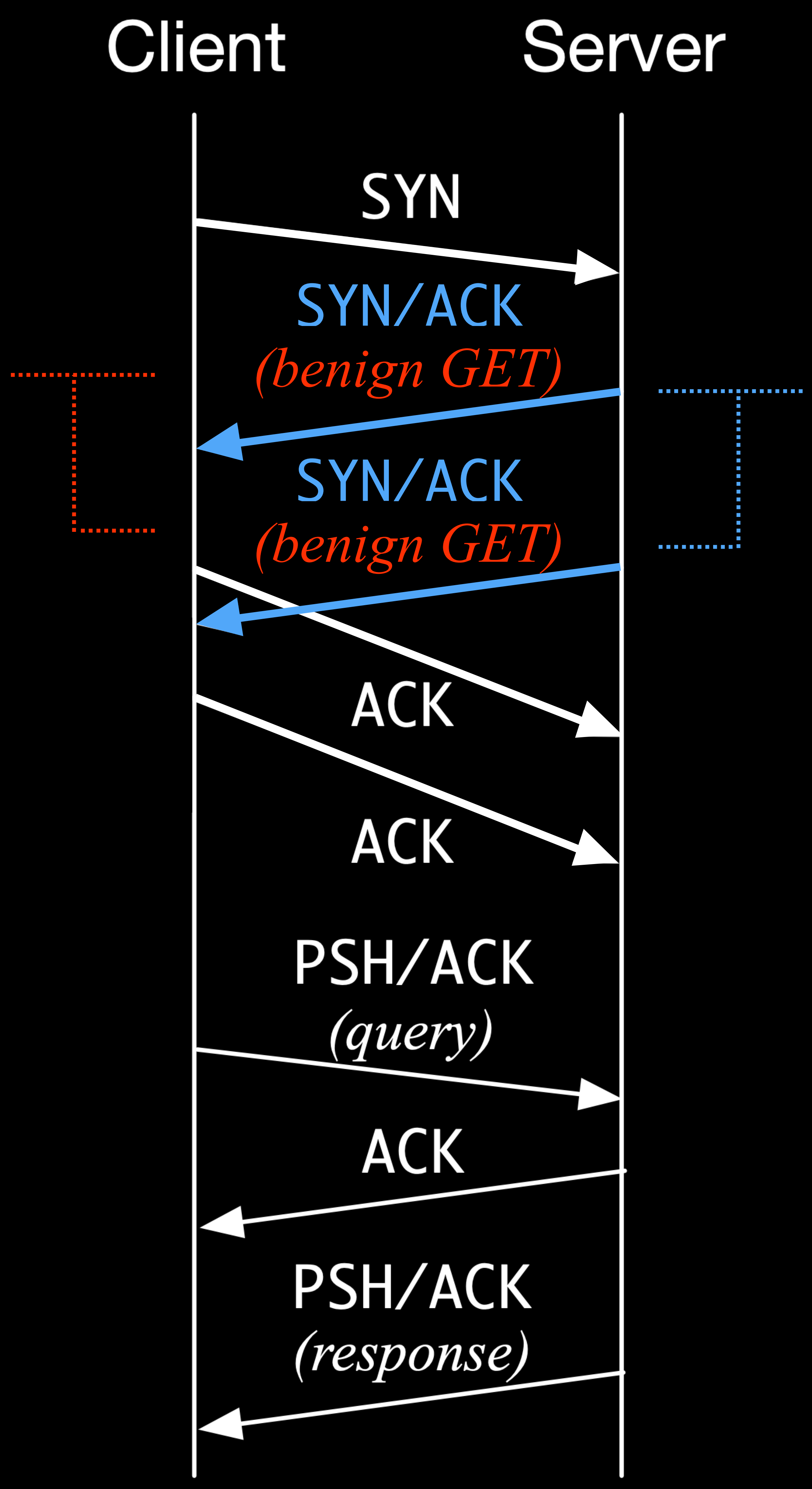
Double benign-GETs



Server sends uncensored GETs inside *two* SYN/ACKs

Server-side evasion results

Double benign-GETs

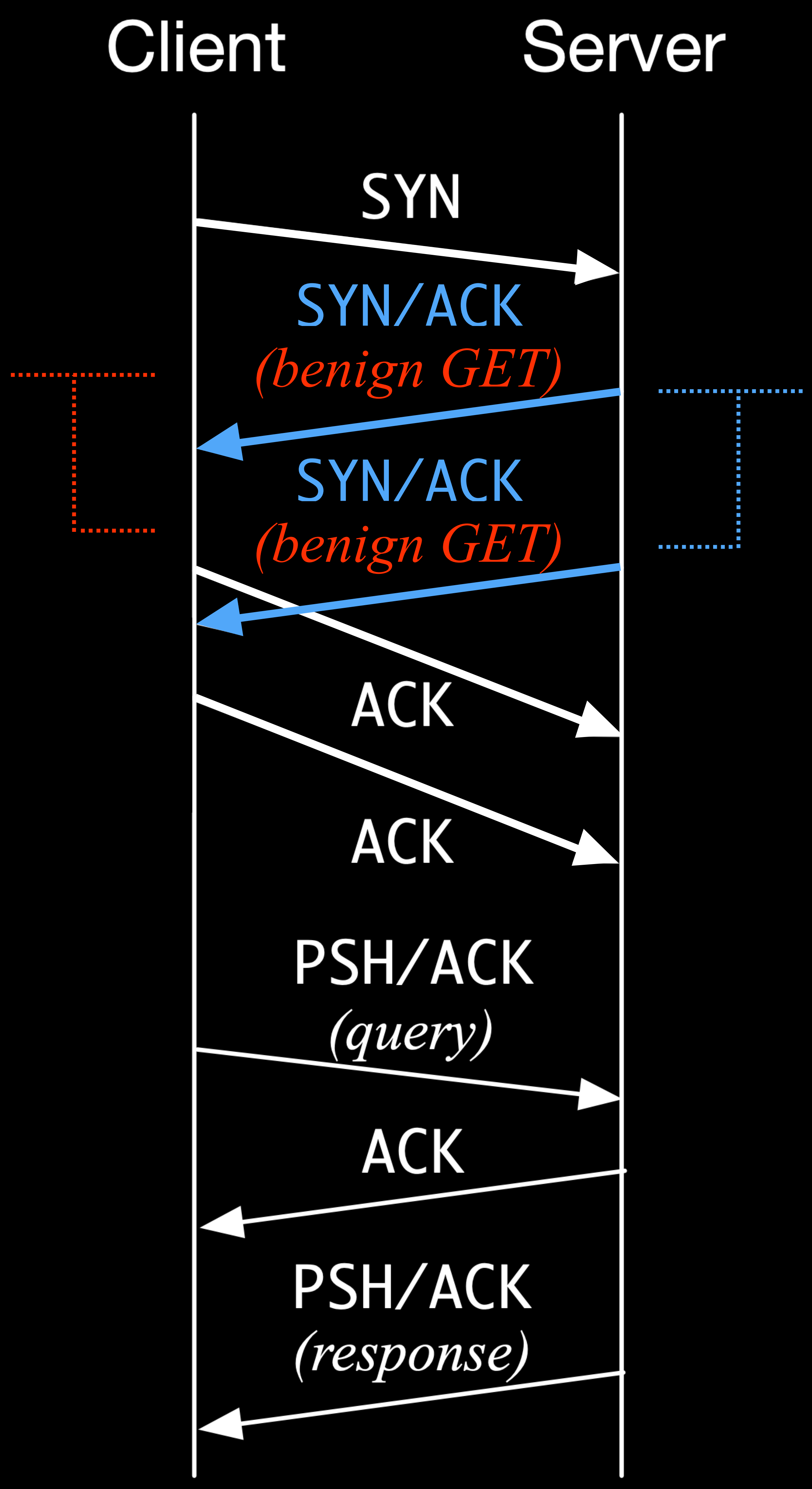


Censor confuses connection direction

Server sends uncensored GETs inside *two* SYN/ACKs

Server-side evasion results

Double benign-GETs



Censor confuses connection direction

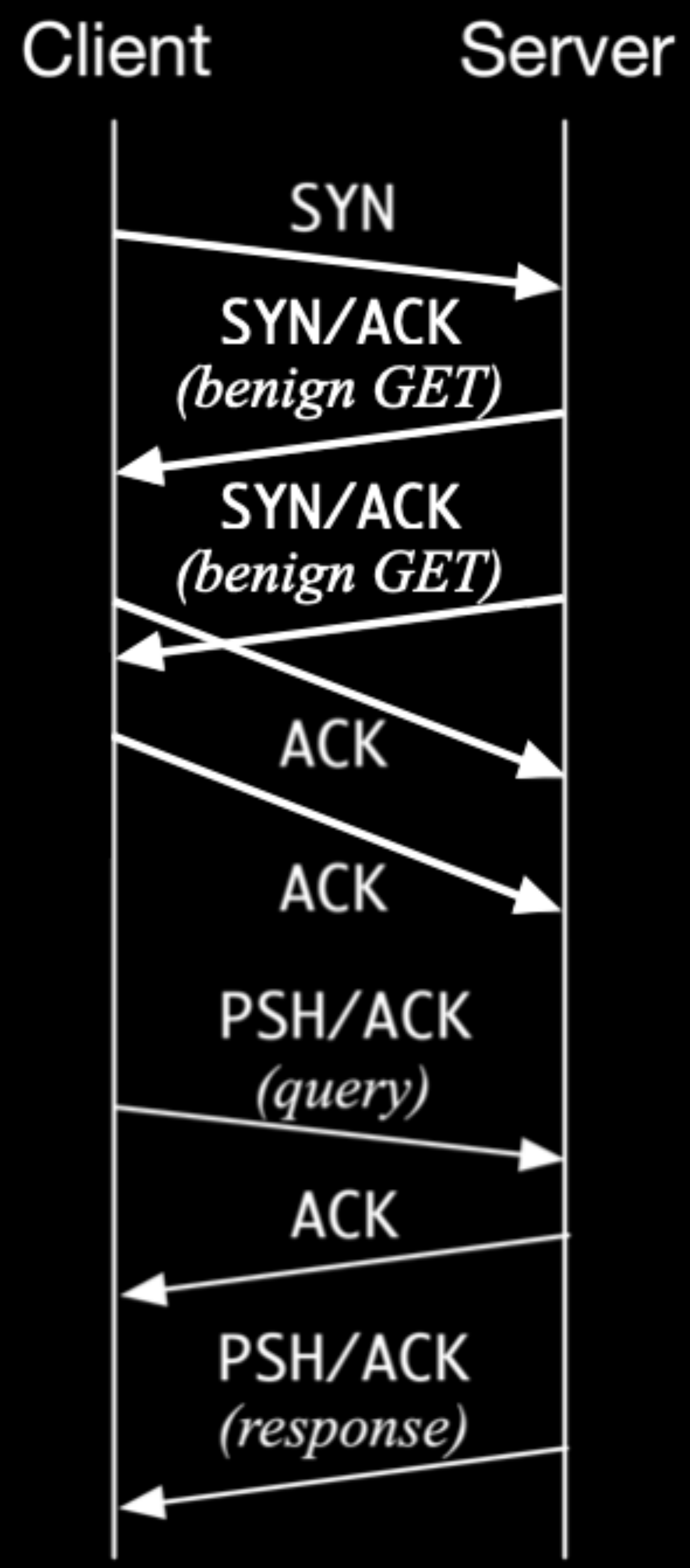
Server sends uncensored GETs inside *two* SYN/ACKs

Success rates

HTTP 100%

Server-side evasion results

Double benign-GETs



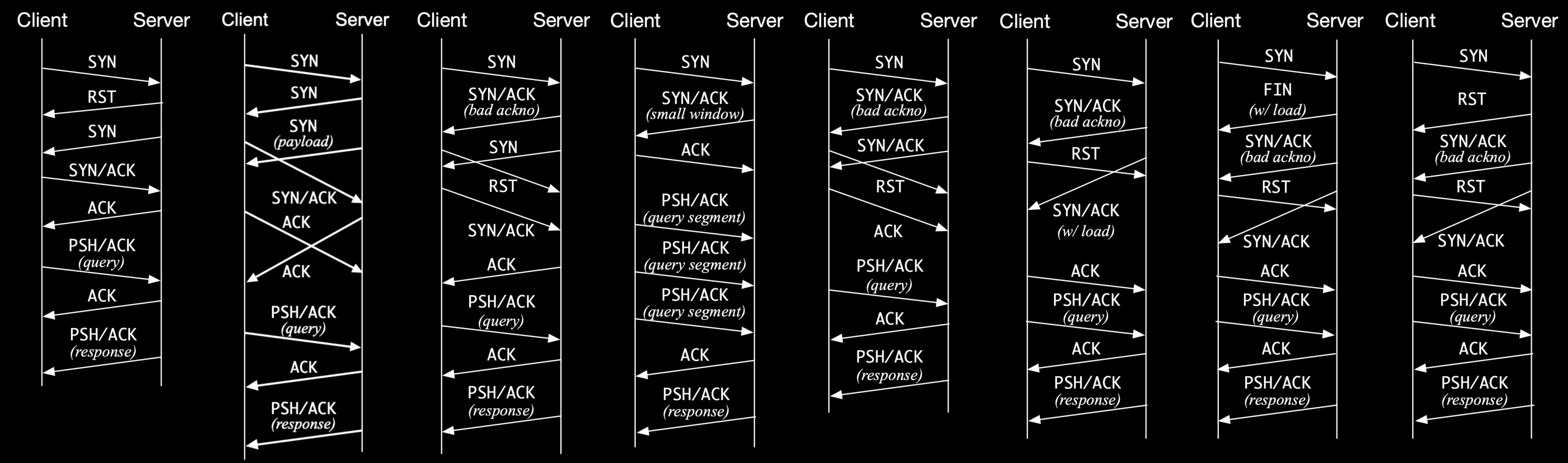
Success rates

HTTP	100%

Server-side evasion strategies

China

8 strategies



Client Server
SYN
RST
SYN
SYN/ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
SYN
SYN *(payload)*
SYN/ACK
ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
SYN/ACK *(bad ackno)*
SYN
RST
SYN/ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
SYN/ACK *(small window)*
ACK
PSH/ACK *(query segment)*
PSH/ACK *(query segment)*
PSH/ACK *(query segment)*
ACK
PSH/ACK *(response)*

Client Server
SYN
SYN/ACK *(bad ackno)*
SYN/ACK
RST
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

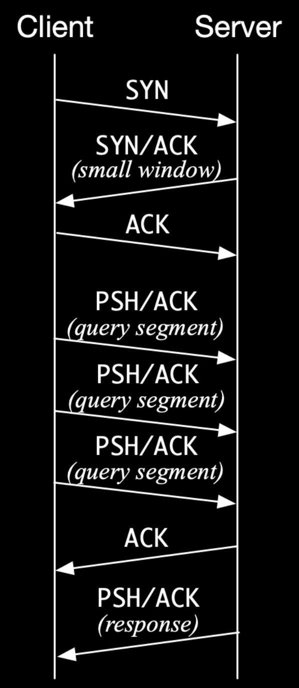
Client Server
SYN
SYN/ACK *(bad ackno)*
RST
SYN/ACK *(w/ load)*
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
FIN *(w/ load)*
SYN/ACK *(bad ackno)*
RST
SYN/ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
RST
SYN/ACK *(bad ackno)*
RST
SYN/ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Iran/India

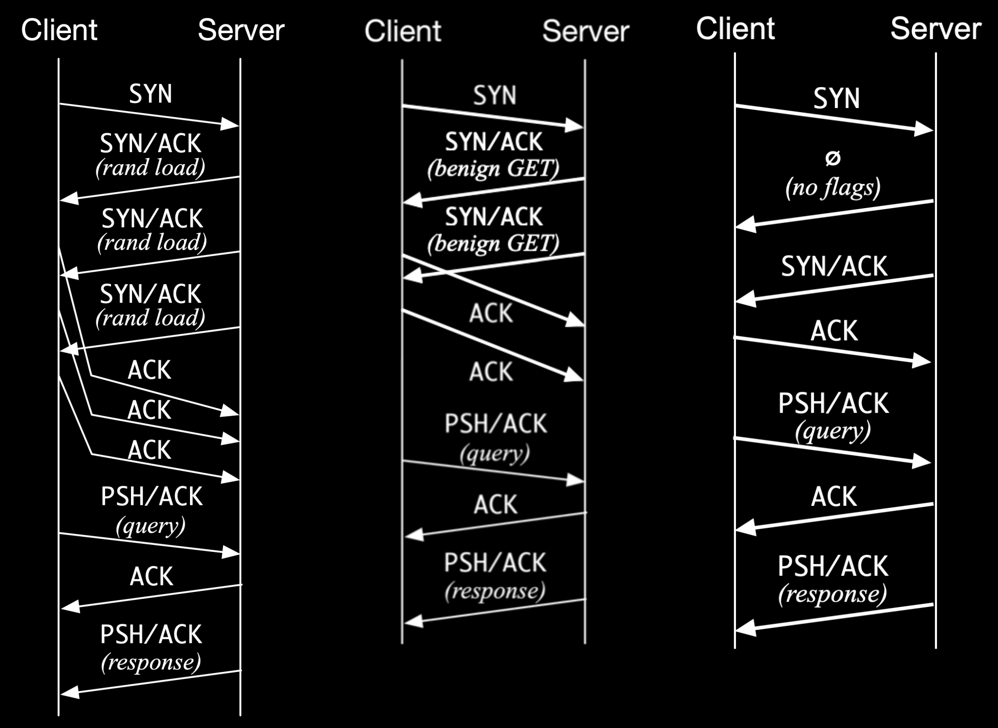
1 strategy



Client Server
SYN
SYN/ACK *(small window)*
ACK
PSH/ACK *(query segment)*
PSH/ACK *(query segment)*
PSH/ACK *(query segment)*
ACK
PSH/ACK *(response)*

Kazakhstan

3 strategies



Client Server
SYN
SYN/ACK *(rand load)*
SYN/ACK *(rand load)*
SYN/ACK *(rand load)*
ACK
ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

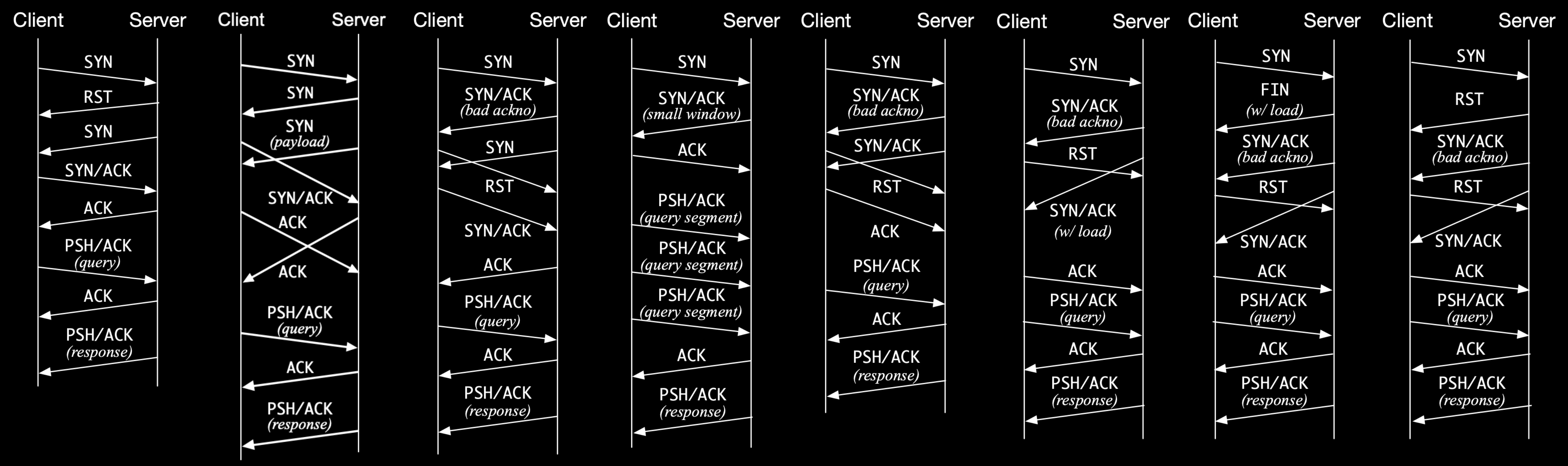
Client Server
SYN
SYN/ACK *(benign GET)*
SYN/ACK *(benign GET)*
ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Client Server
SYN
ø *(no flags)*
SYN/ACK
ACK
PSH/ACK *(query)*
ACK
PSH/ACK *(response)*

Server-side evasion strategies

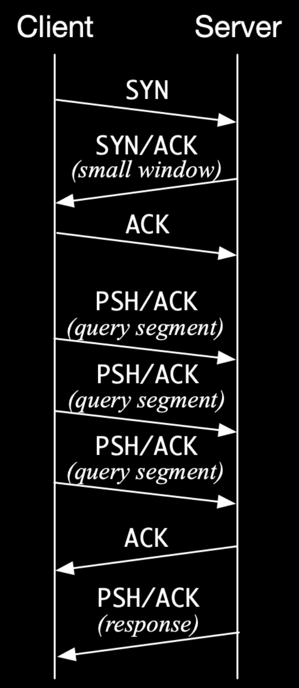
China

8 strategies



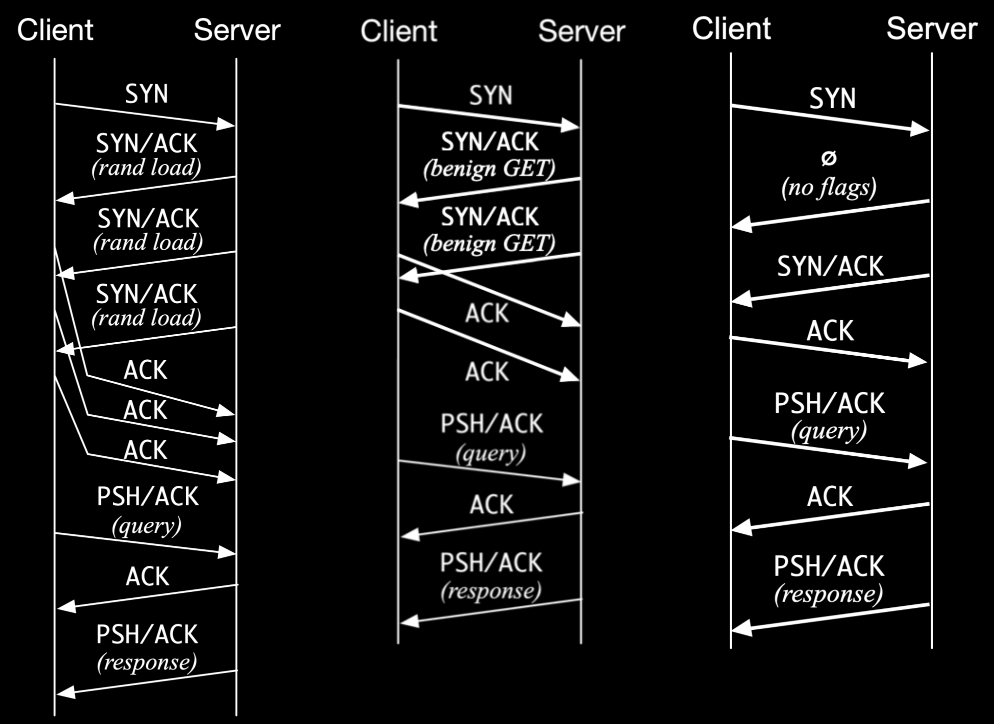
Iran/India

1 strategy



Kazakhstan

3 strategies



None of these require *any* client-side deployment

Come as you are

Windows XP
Windows 7
Windows 8.1
Windows 10
Server 2003
Server 2008
Server 2013
Server 2018

OS X 10.14
OS X 10.15

iOS 13.3

Android 10

Centos 6
Centos 7

Ubuntu 12.04
Ubuntu 14.04
Ubuntu 16.04
Ubuntu 18.04

This paper

Server-side evasion is possible

- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW's resynchronization state
- "Multibox Theory"

This paper

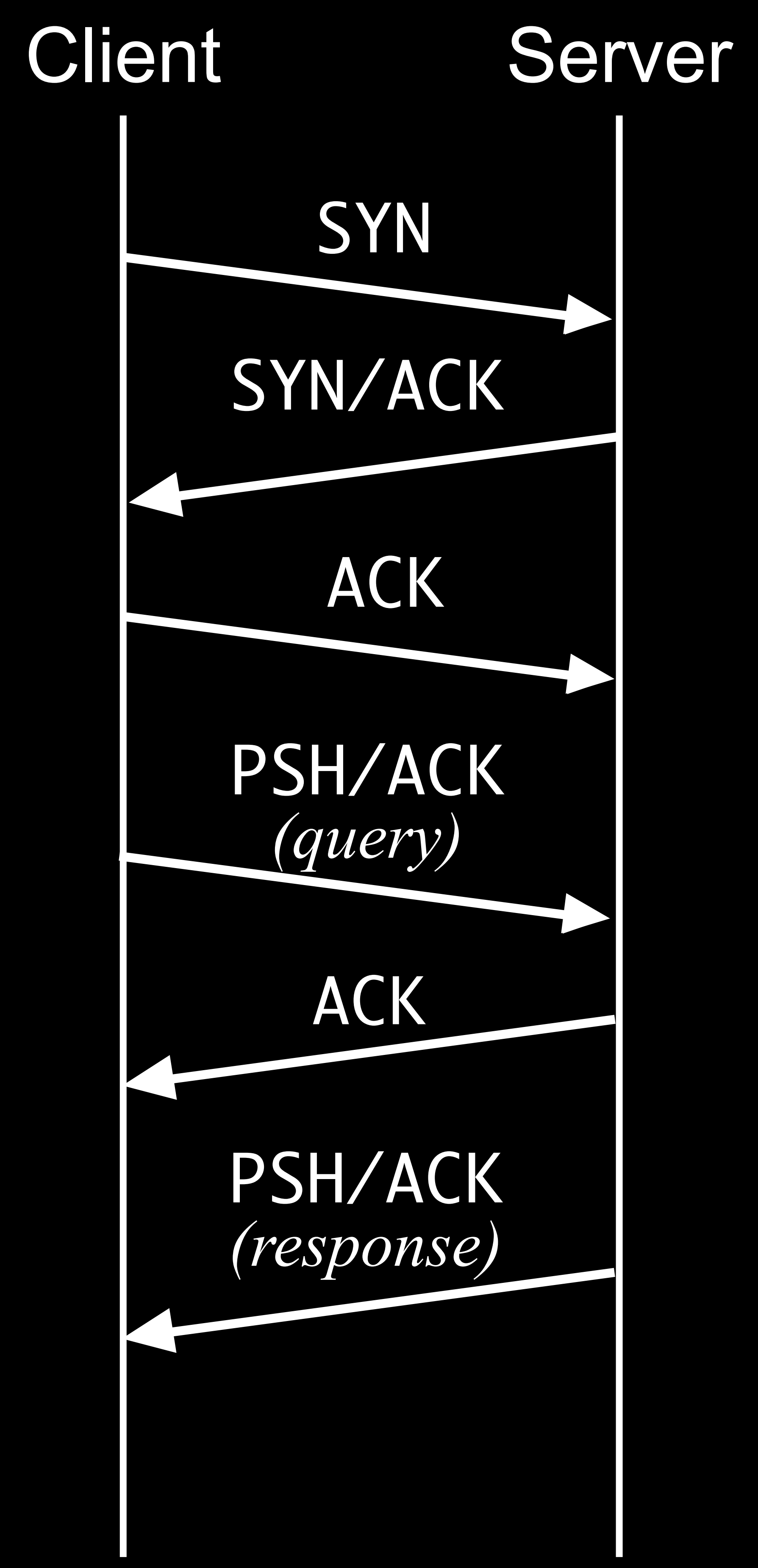
Server-side evasion is possible

- For every country and protocol we tested
- Artifact-evaluated, open-source tool

New insights into how censors work

- GFW's resynchronization state
- "Multibox Theory"

🇨🇳 Resynchronization State

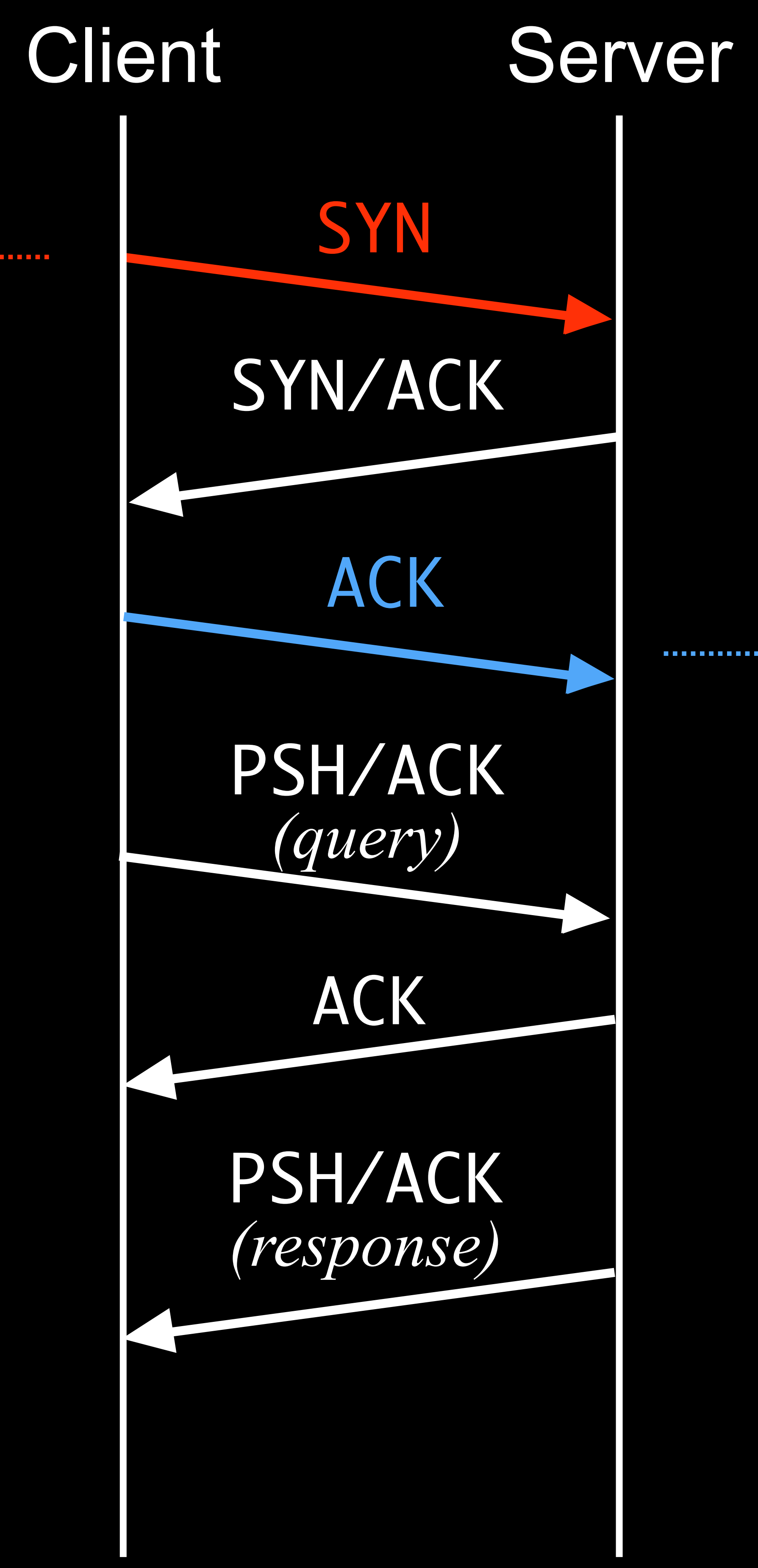


Censoring middleboxes

tolerant to packet loss

🇨🇳 Resynchronization State

If middleboxes misses a packet

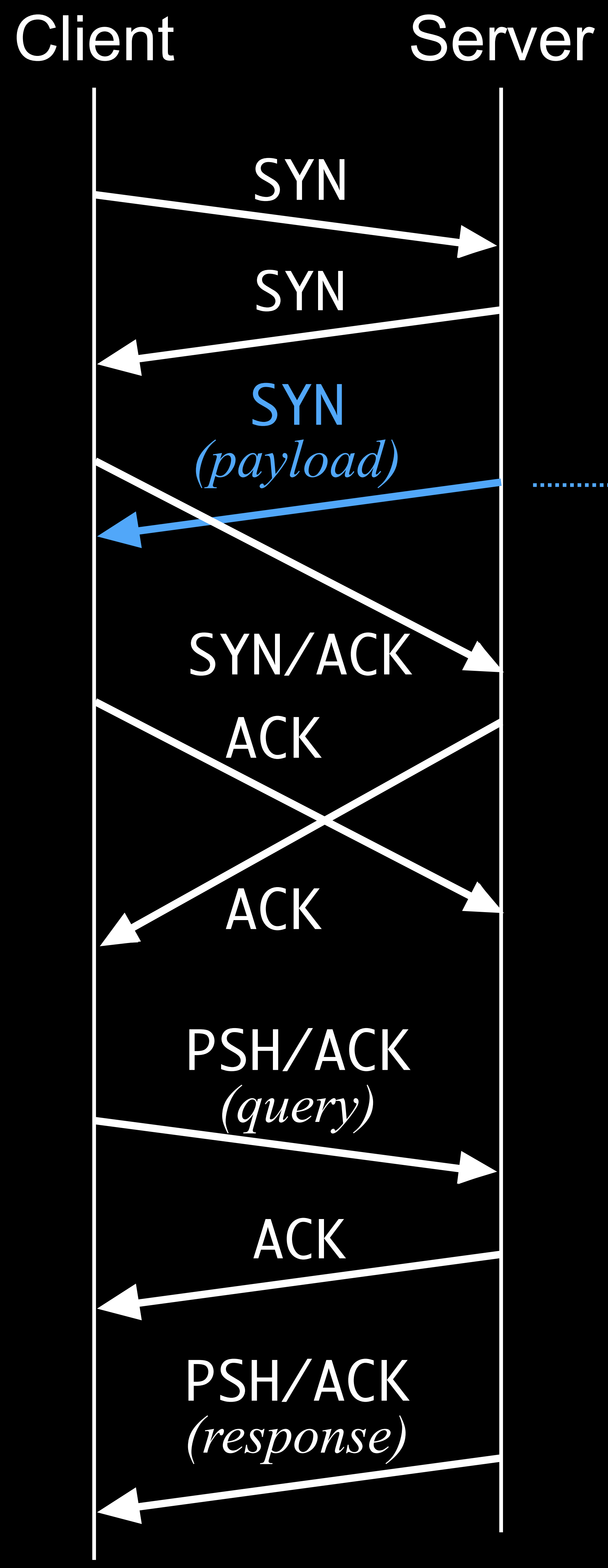


Censor can *resynchronize* its state

Censoring middleboxes tolerant to packet loss

Resynchronization State

Simultaneous-open-based desynchronization

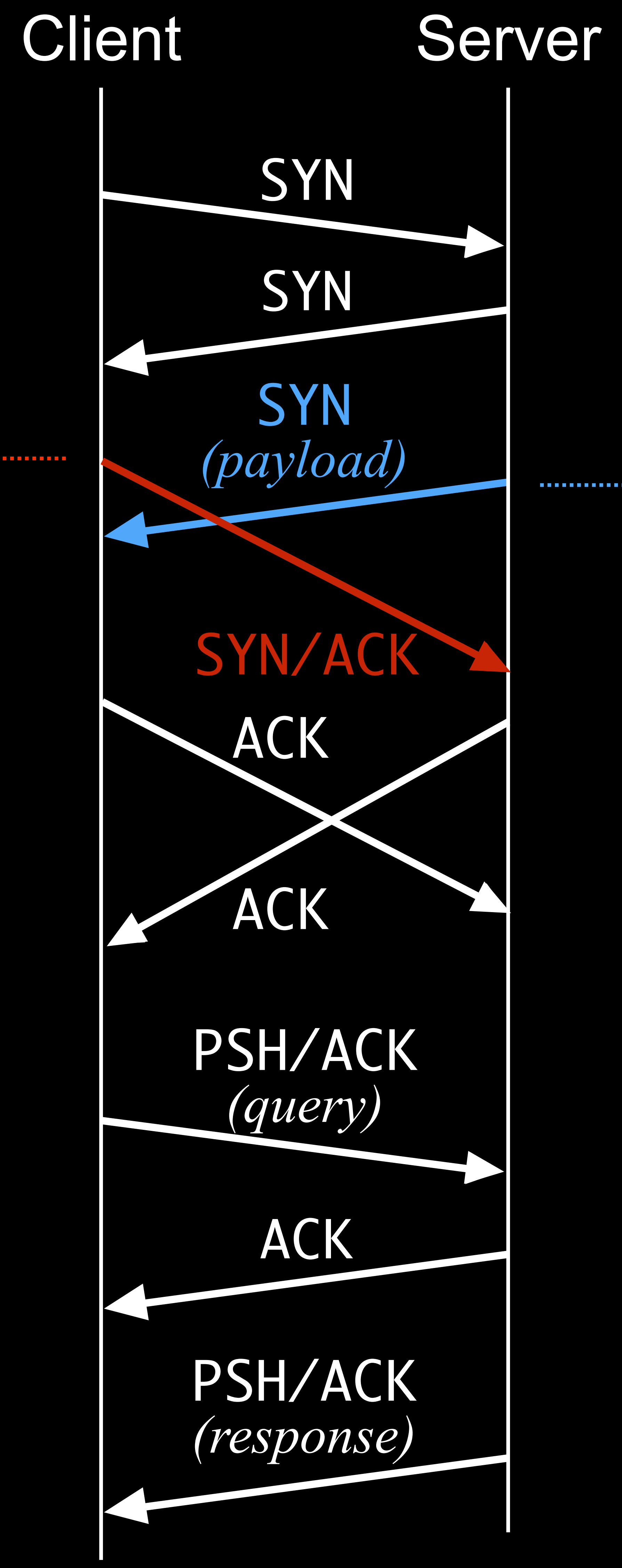


Payload from server triggers resynchronization

Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client



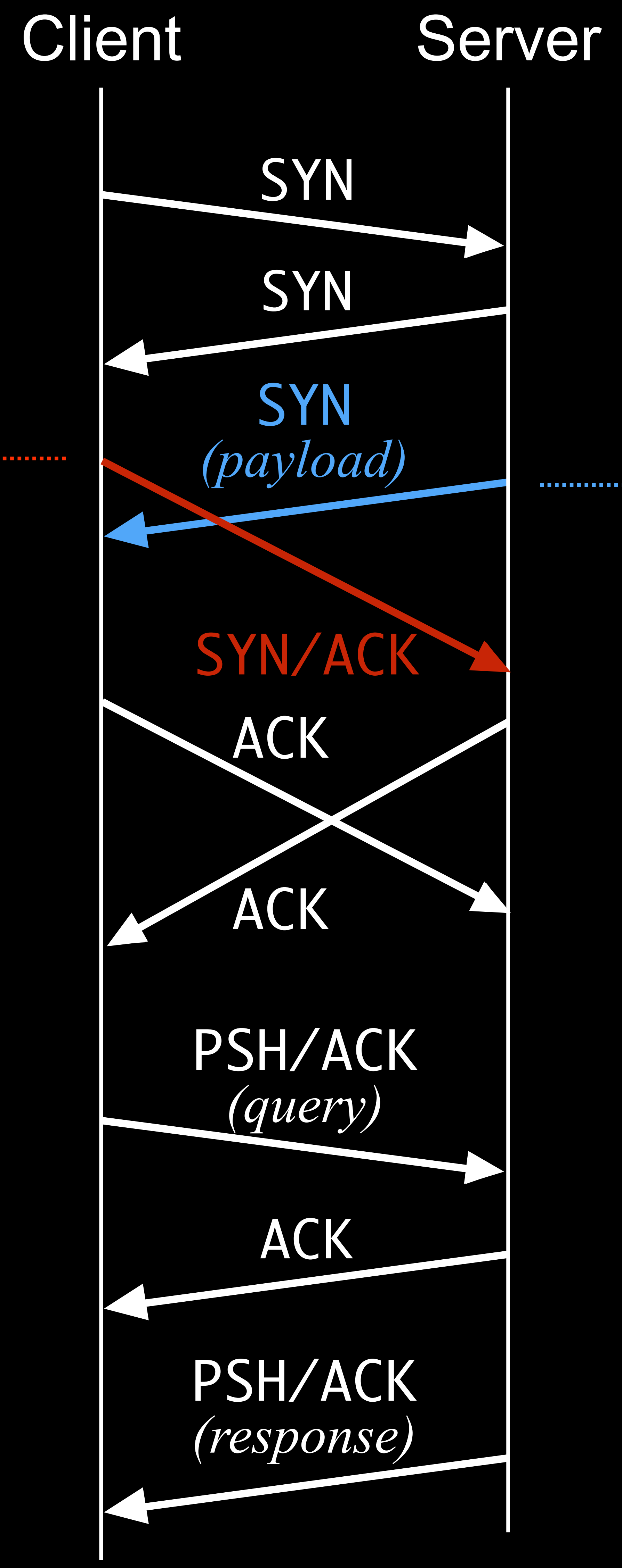
Payload from server triggers resynchronization

Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

…but does not properly increment ISN



Payload from server triggers resynchronization

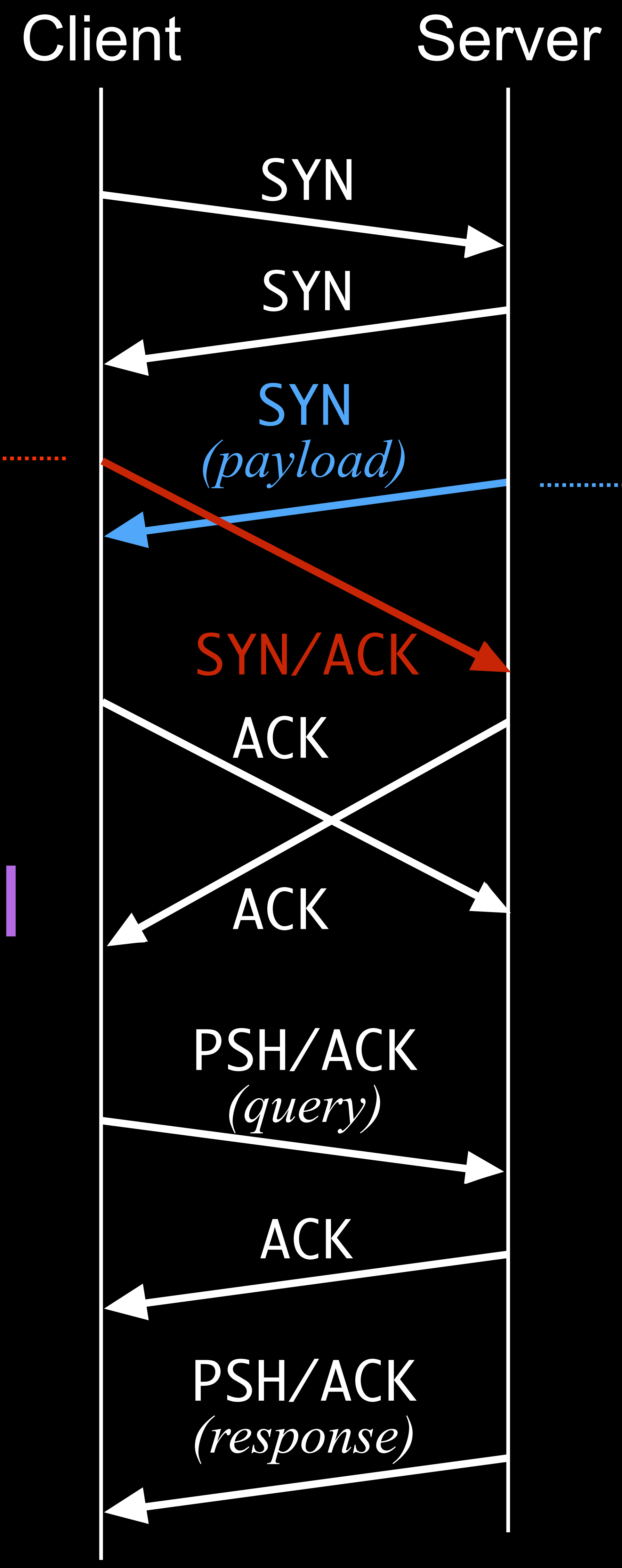
Resynchronization State

🇨🇳 Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

…but does not properly increment ISN

Off-by-1 bug in the Great Firewall



Payload from server triggers resynchronization

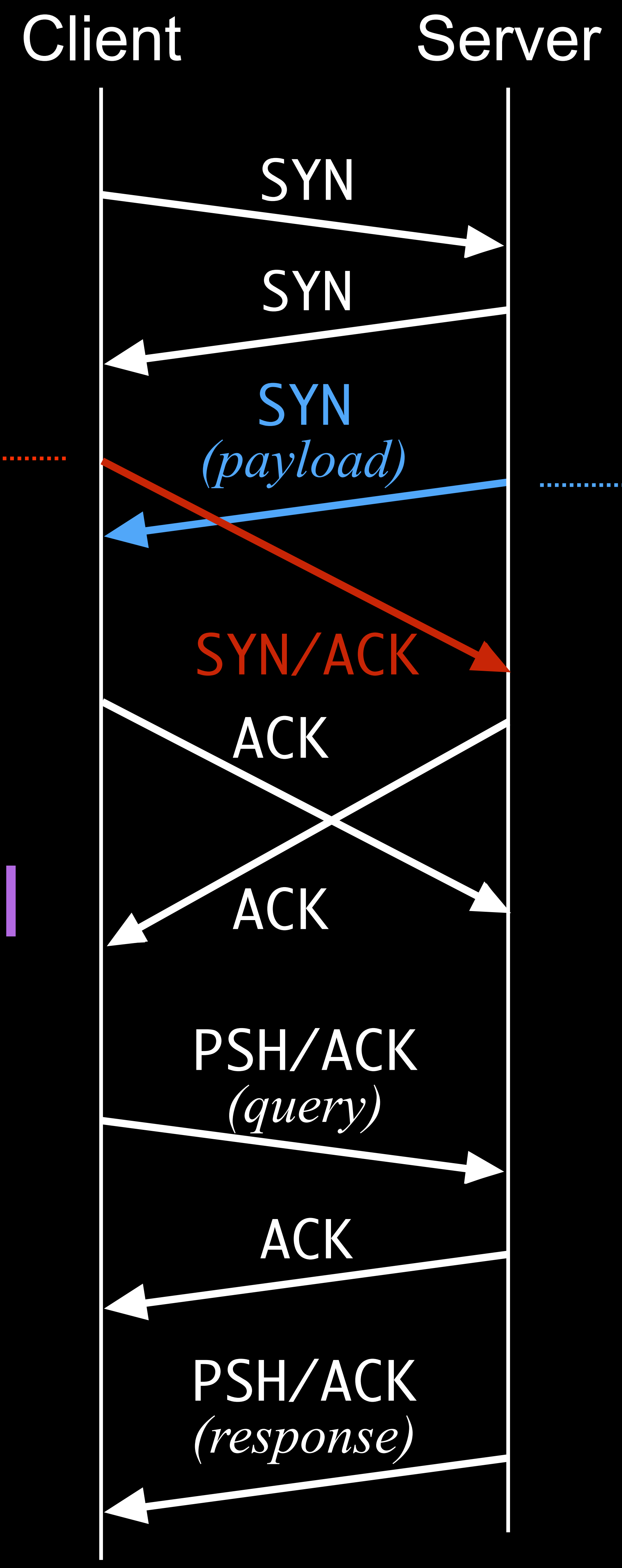
Resynchronization State

Simultaneous-open-based desynchronization

Resynchronizes on SYN/ACK from the client

…but does not properly increment ISN

Off-by-1 bug in the Great Firewall



Payload from server triggers resynchronization

DNS	89%
FTP	36%
HTTP	54%
HTTPS	55%
SMTP	70%

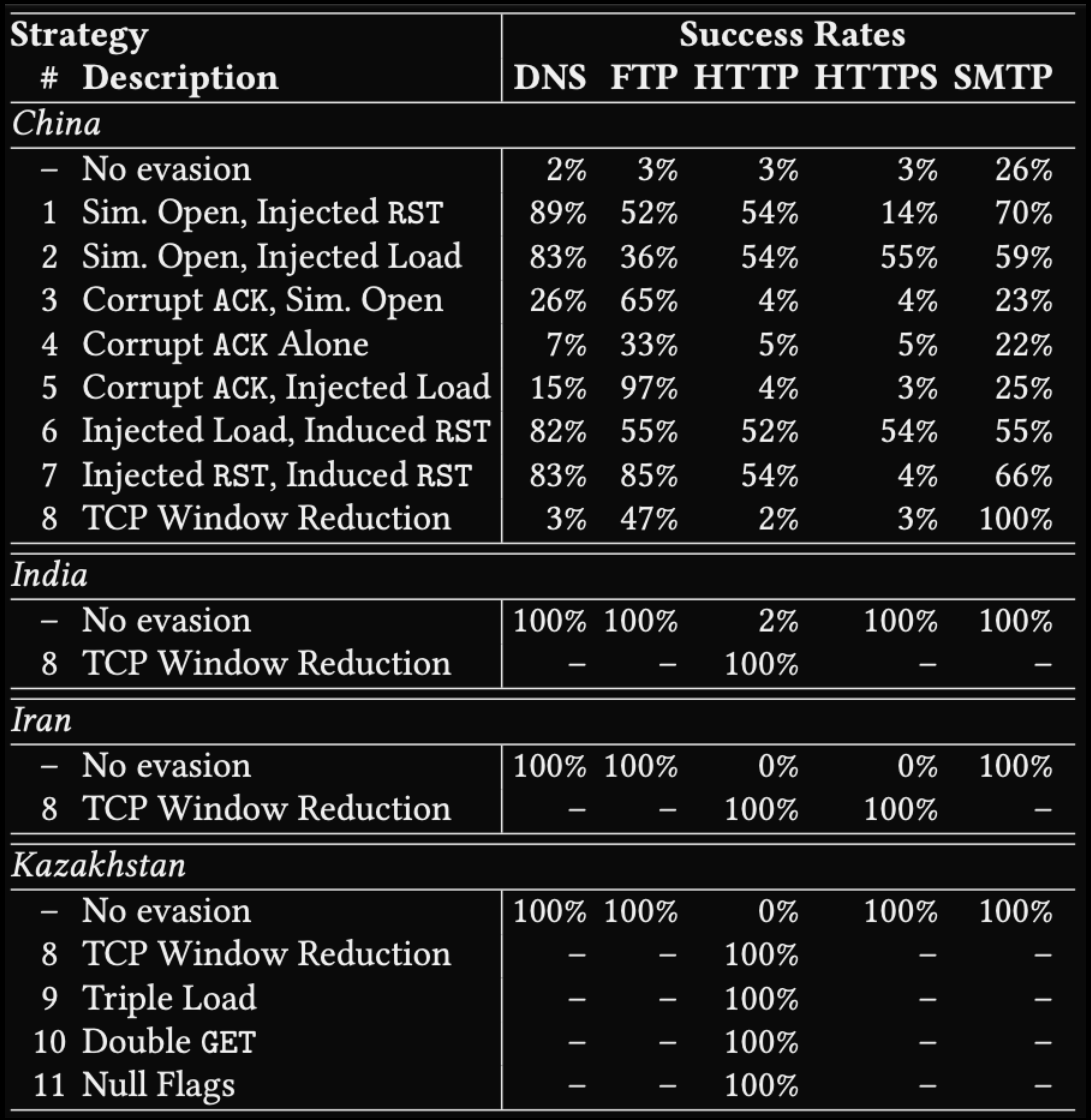
Resynchronization State

GFW resynchronizes differently
depending on protocol

GFW Resynchronizes on the next:

FTP	*Client packet if SYN+ACK has a bad ack number*
All but HTTPS	*Client packet if server sends a RST*
All protocols	*ACK packet if server sends non-SYN+ACK with a payload*

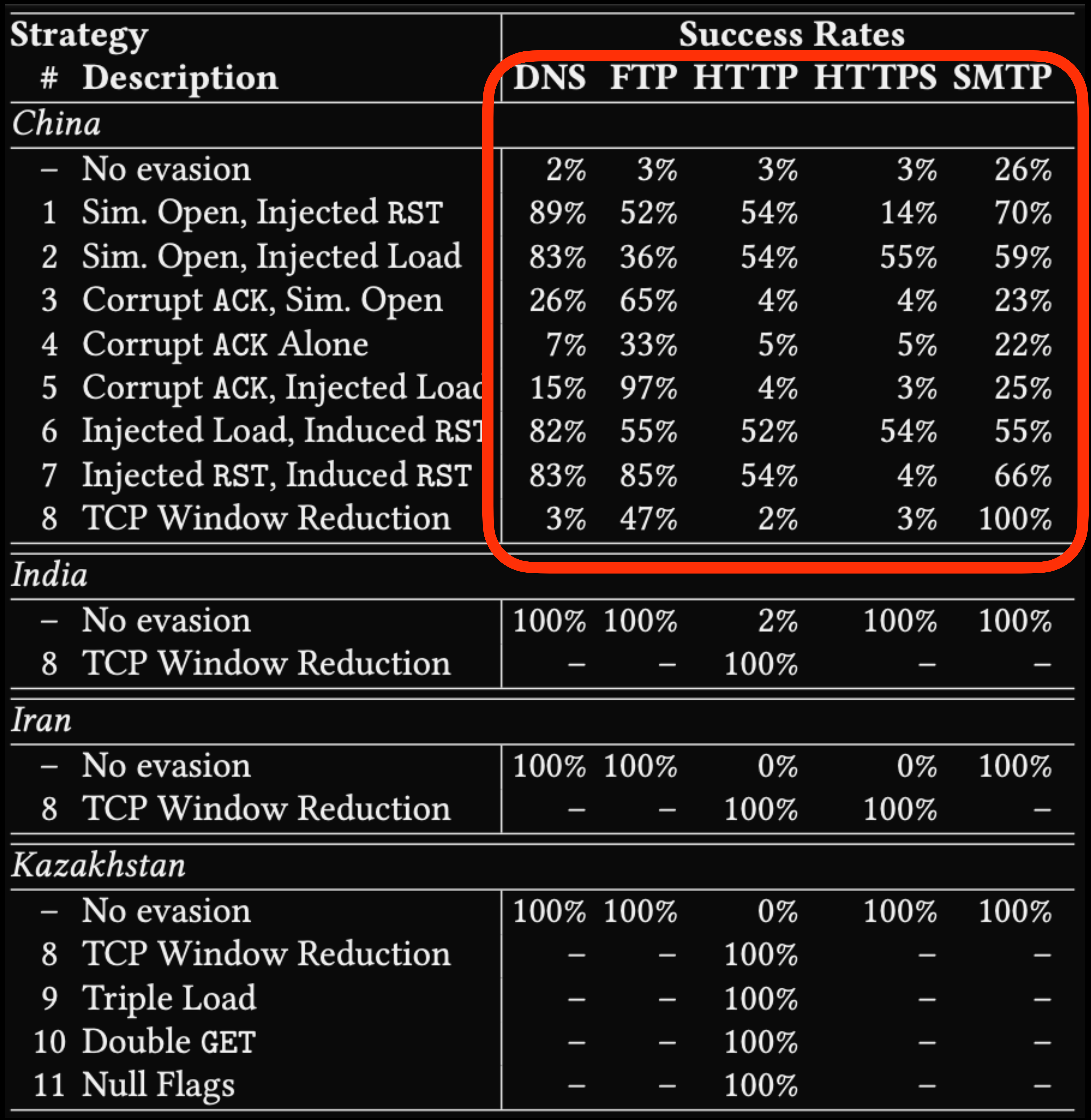
New Model for Chinese Censorship



Strategy		Success Rates				
#	Description	DNS	FTP	HTTP	HTTPS	SMTP
China						
–	No evasion	2%	3%	3%	3%	26%
1	Sim. Open, Injected RST	89%	52%	54%	14%	70%
2	Sim. Open, Injected Load	83%	36%	54%	55%	59%
3	Corrupt ACK, Sim. Open	26%	65%	4%	4%	23%
4	Corrupt ACK Alone	7%	33%	5%	5%	22%
5	Corrupt ACK, Injected Load	15%	97%	4%	3%	25%
6	Injected Load, Induced RST	82%	55%	52%	54%	55%
7	Injected RST, Induced RST	83%	85%	54%	4%	66%
8	TCP Window Reduction	3%	47%	2%	3%	100%
India						
–	No evasion	100%	100%	2%	100%	100%
8	TCP Window Reduction	–	–	100%	–	–
Iran						
–	No evasion	100%	100%	0%	0%	100%
8	TCP Window Reduction	–	–	100%	100%	–
Kazakhstan						
–	No evasion	100%	100%	0%	100%	100%
8	TCP Window Reduction	–	–	100%	–	–
9	Triple Load	–	–	100%	–	–
10	Double GET	–	–	100%	–	–
11	Null Flags	–	–	100%	–	–

All of the server-side strategies operate strictly during the TCP 3-way handshake

New Model for Chinese Censorship



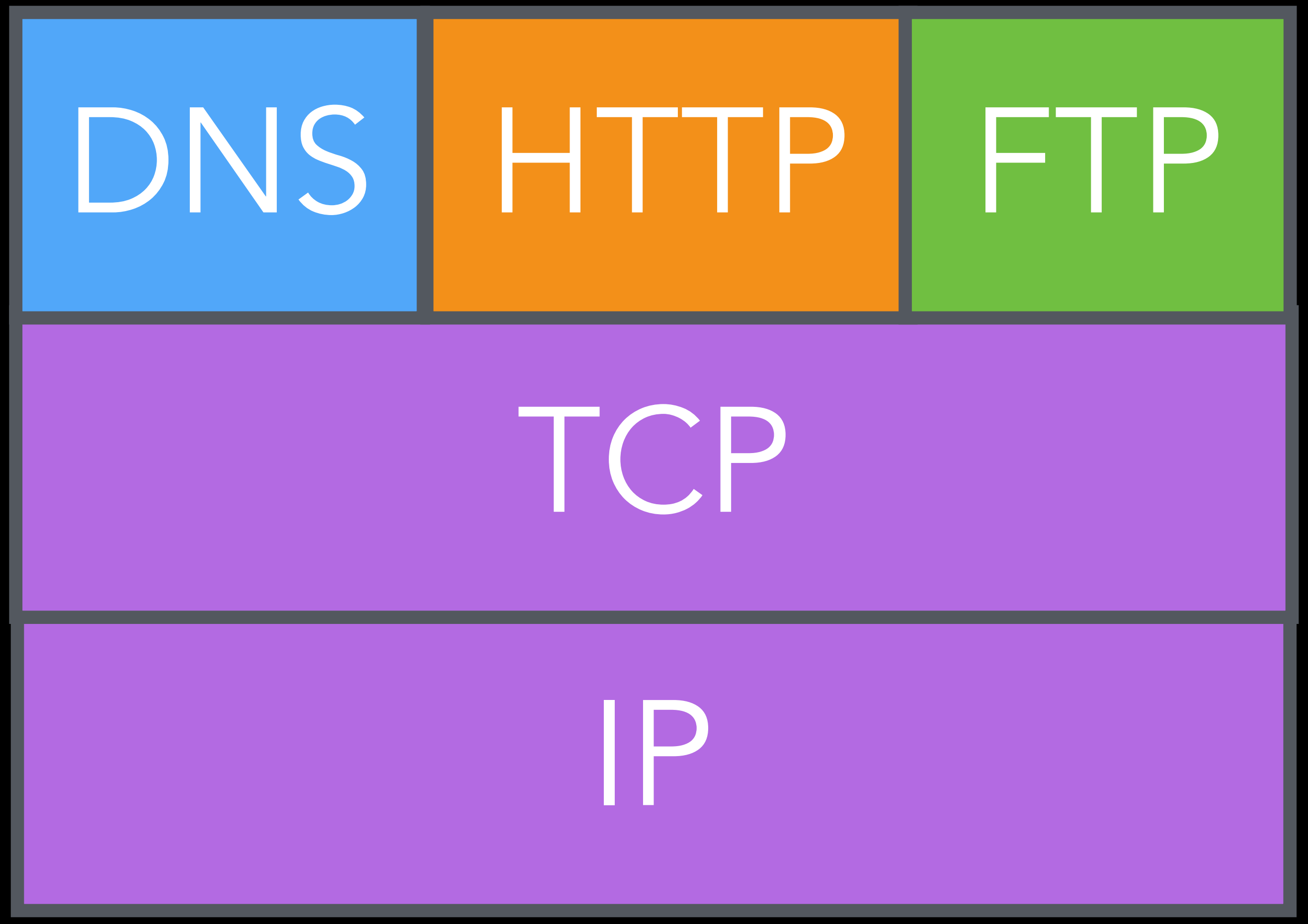
Strategy		Success Rates				
#	Description	DNS	FTP	HTTP	HTTPS	SMTP
China						
–	No evasion	2%	3%	3%	3%	26%
1	Sim. Open, Injected RST	89%	52%	54%	14%	70%
2	Sim. Open, Injected Load	83%	36%	54%	55%	59%
3	Corrupt ACK, Sim. Open	26%	65%	4%	4%	23%
4	Corrupt ACK Alone	7%	33%	5%	5%	22%
5	Corrupt ACK, Injected Load	15%	97%	4%	3%	25%
6	Injected Load, Induced RST	82%	55%	52%	54%	55%
7	Injected RST, Induced RST	83%	85%	54%	4%	66%
8	TCP Window Reduction	3%	47%	2%	3%	100%
India						
–	No evasion	100%	100%	2%	100%	100%
8	TCP Window Reduction	–	–	100%	–	–
Iran						
–	No evasion	100%	100%	0%	0%	100%
8	TCP Window Reduction	–	–	100%	100%	–
Kazakhstan						
–	No evasion	100%	100%	0%	100%	100%
8	TCP Window Reduction	–	–	100%	–	–
9	Triple Load	–	–	100%	–	–
10	Double GET	–	–	100%	–	–
11	Null Flags	–	–	100%	–	–

All of the server-side strategies operate strictly during the TCP 3-way handshake

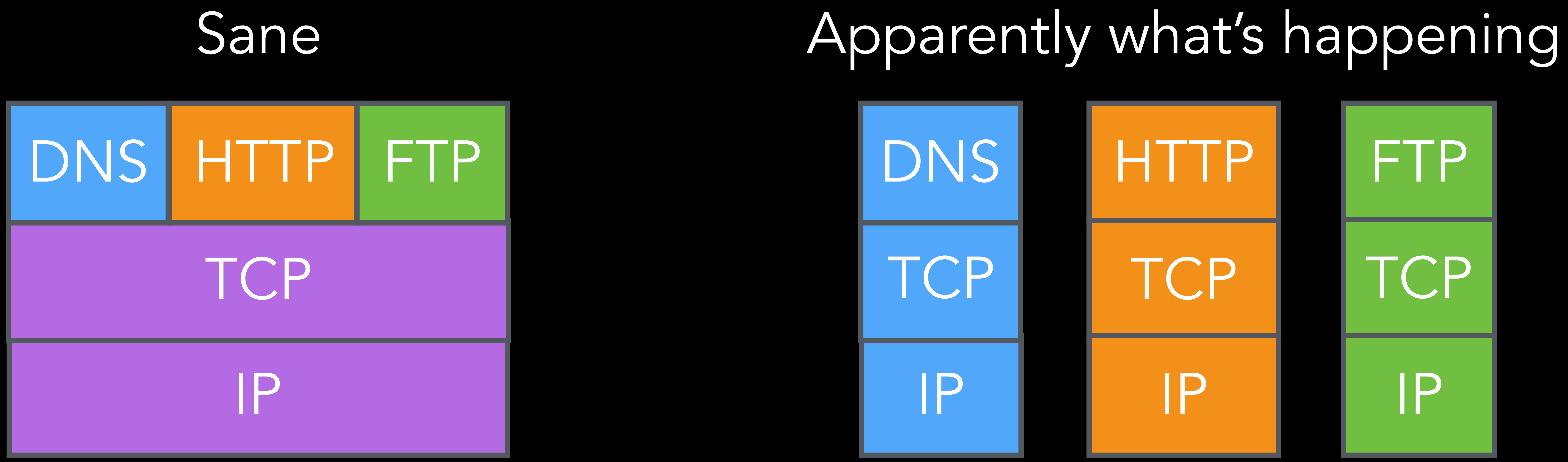
So why are different applications affected differently in China?

New Model for Chinese Censorship

Sane



New Model for Chinese Censorship



Sane

DNS | HTTP | FTP

TCP

IP

Apparently what's happening

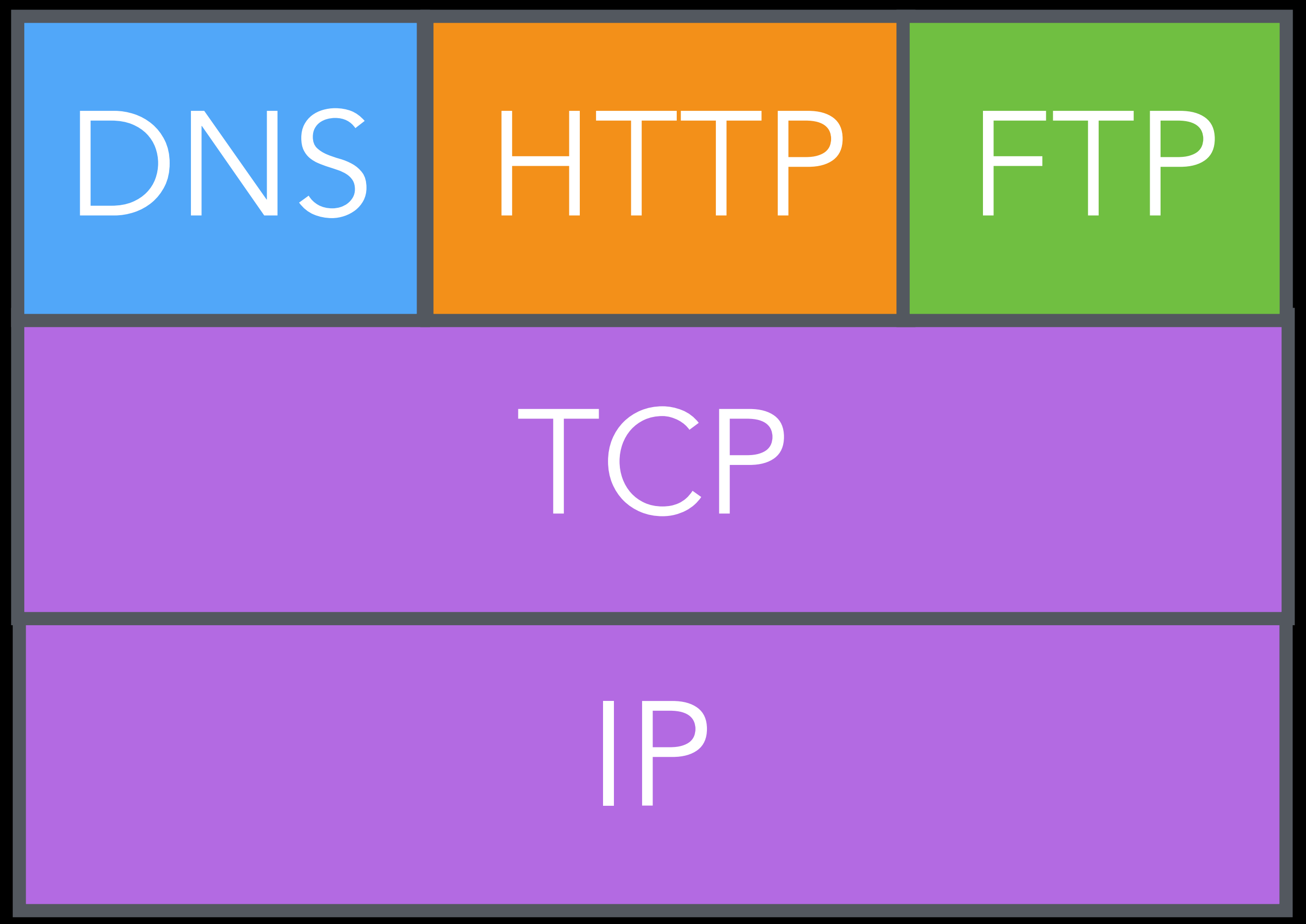
DNS
TCP
IP

HTTP
TCP
IP

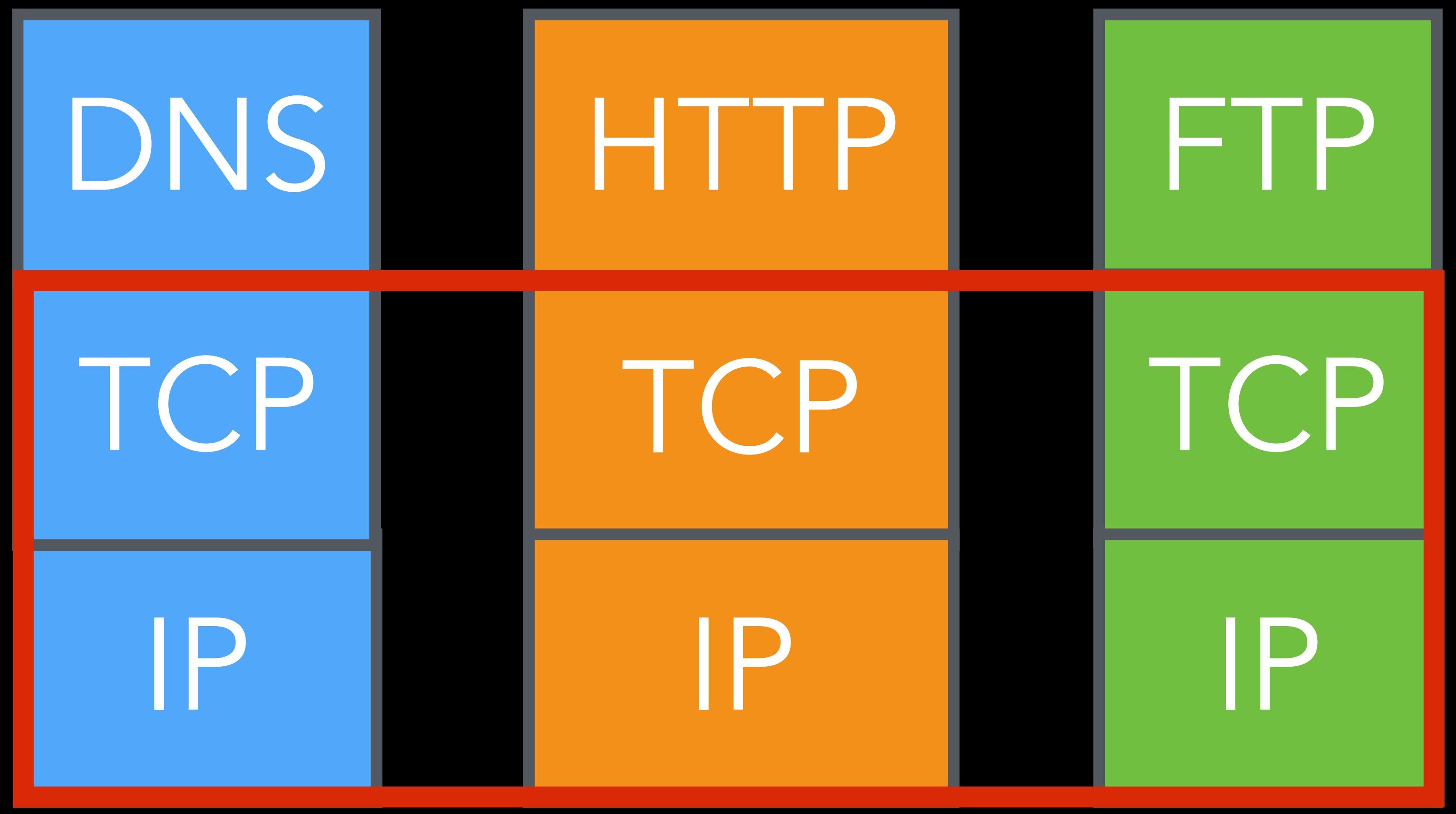
FTP
TCP
IP

New Model for Chinese Censorship

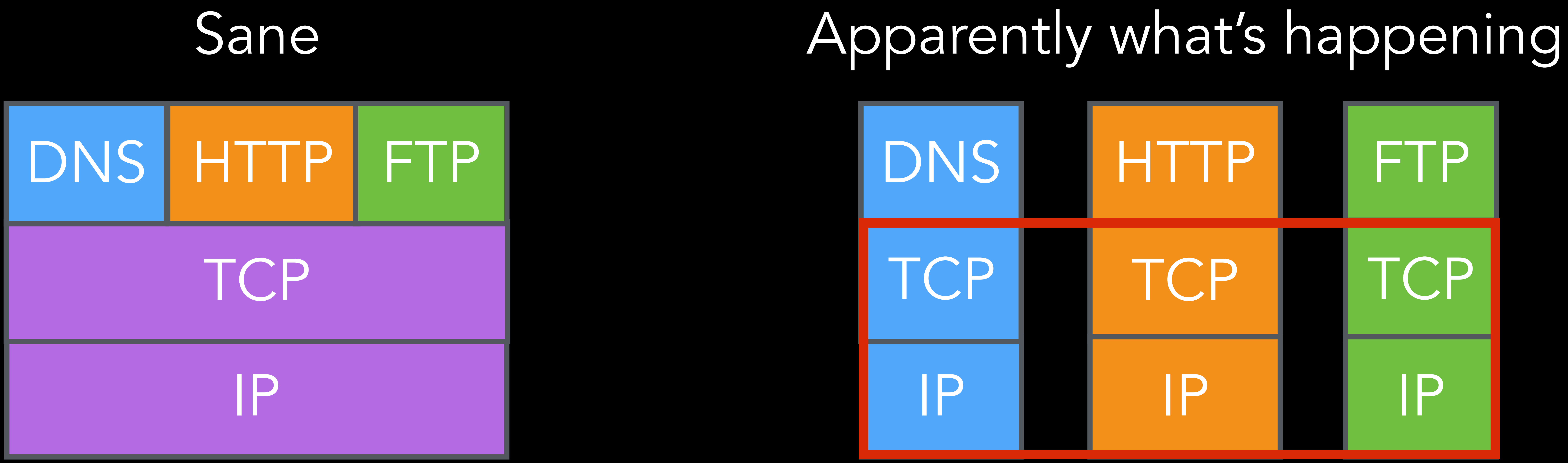
Sane



Apparently what’s happening

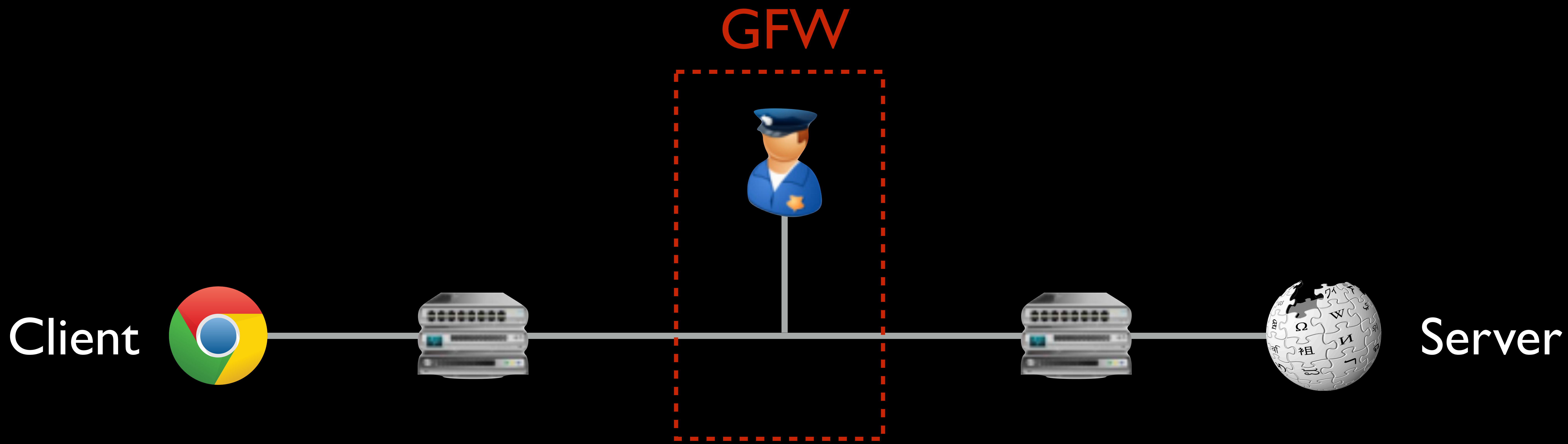


New Model for Chinese Censorship

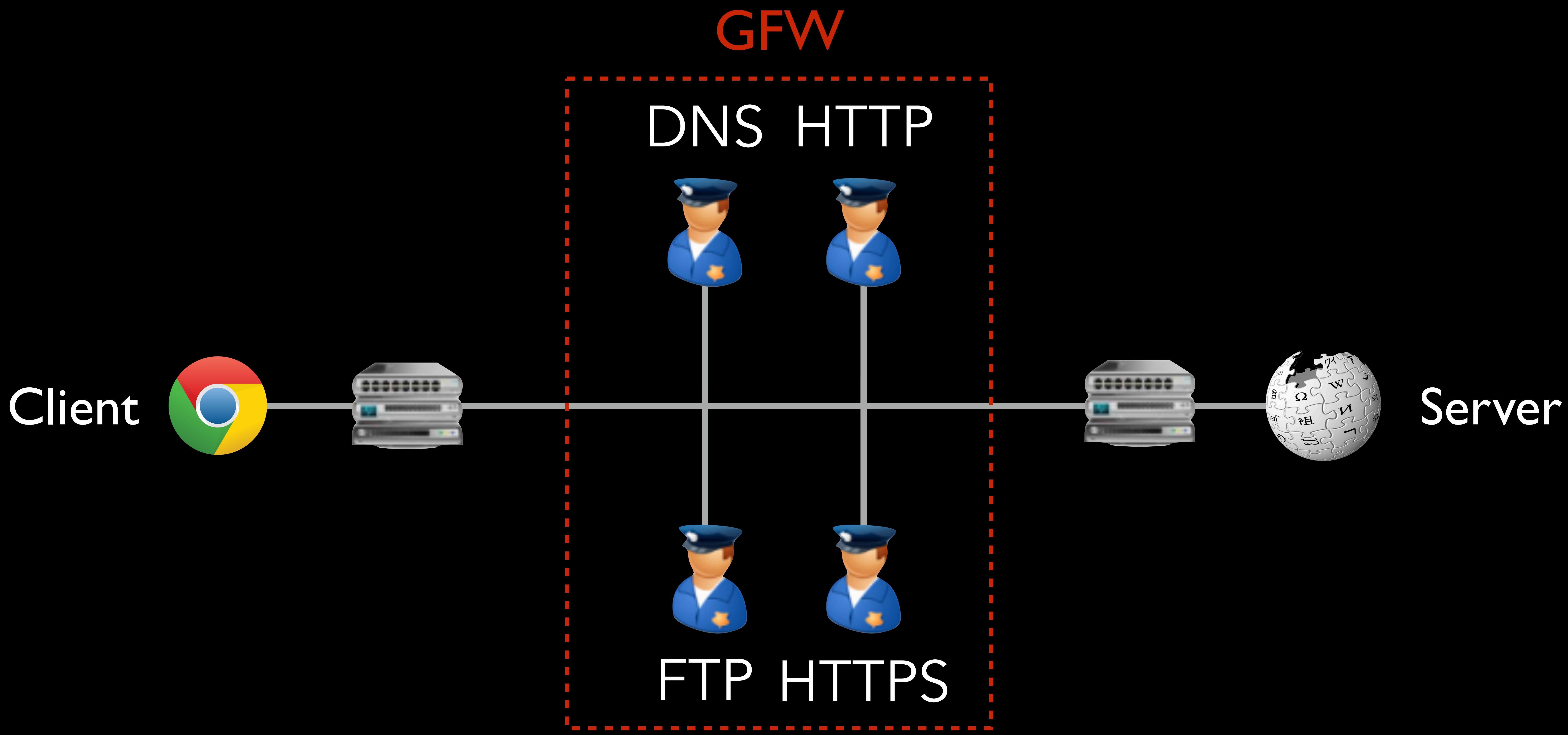


Results suggest GFW is running
multiple censoring middleboxes in parallel

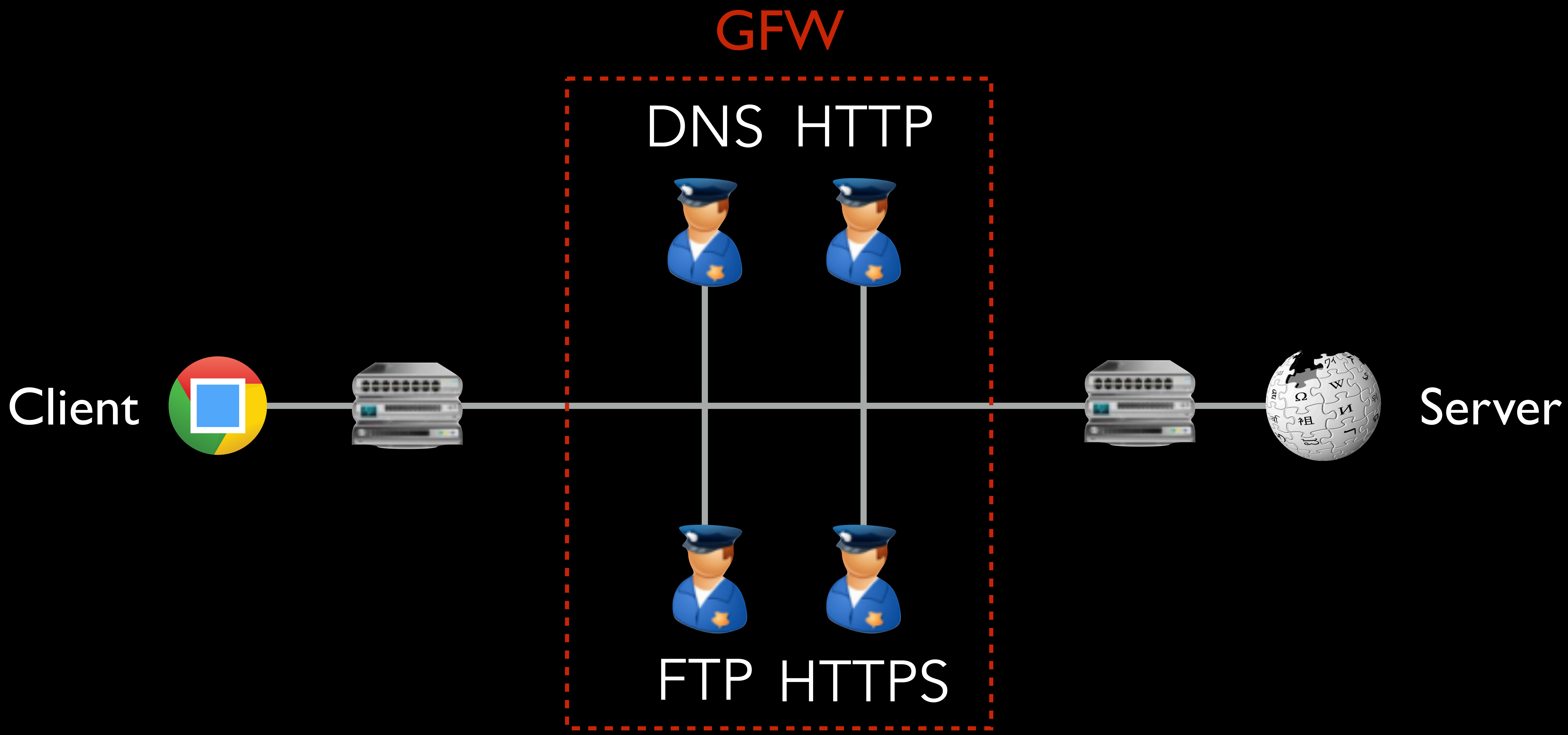
Multi-box theory



Multi-box theory

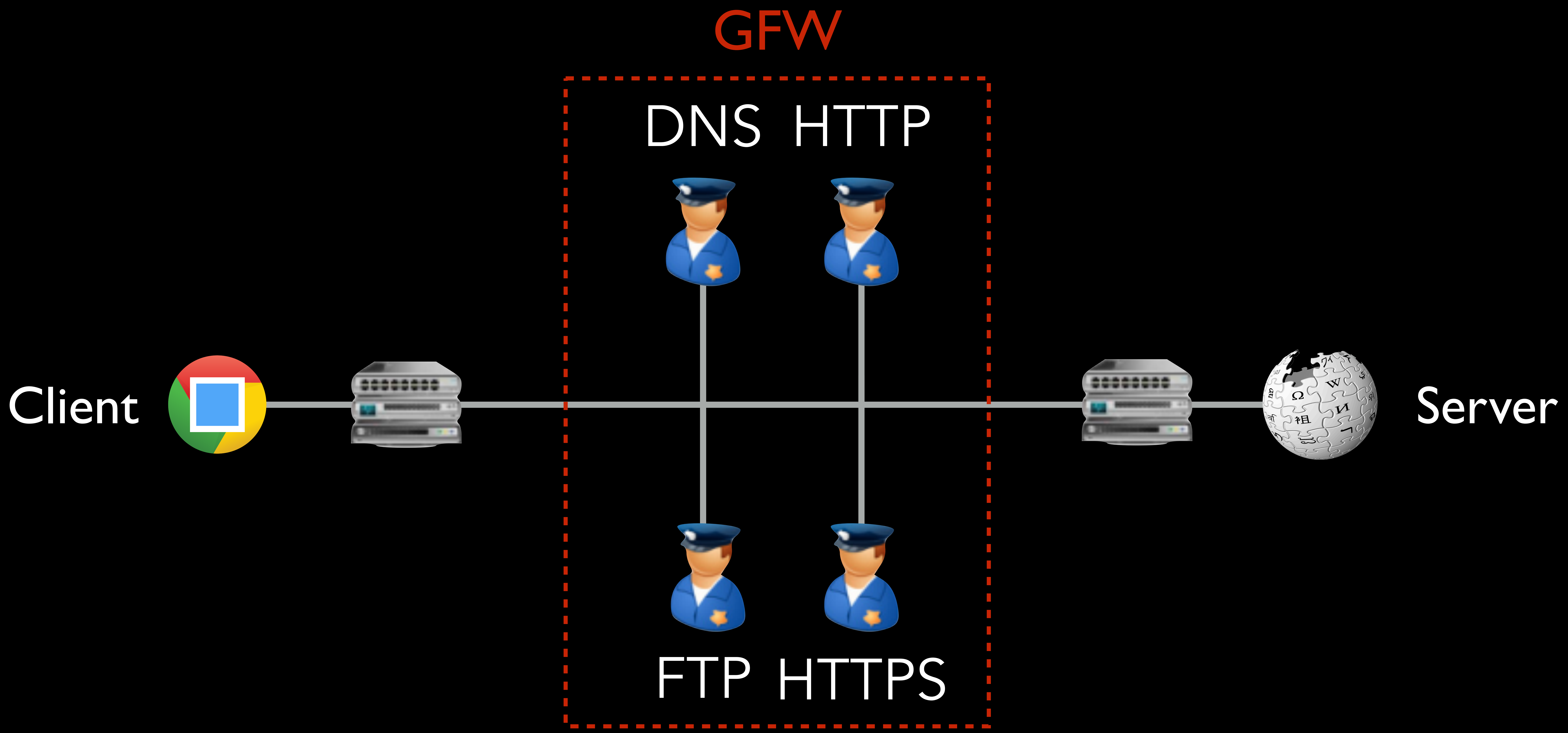


Multi-box theory



How does the censor know which one to apply to a connection?

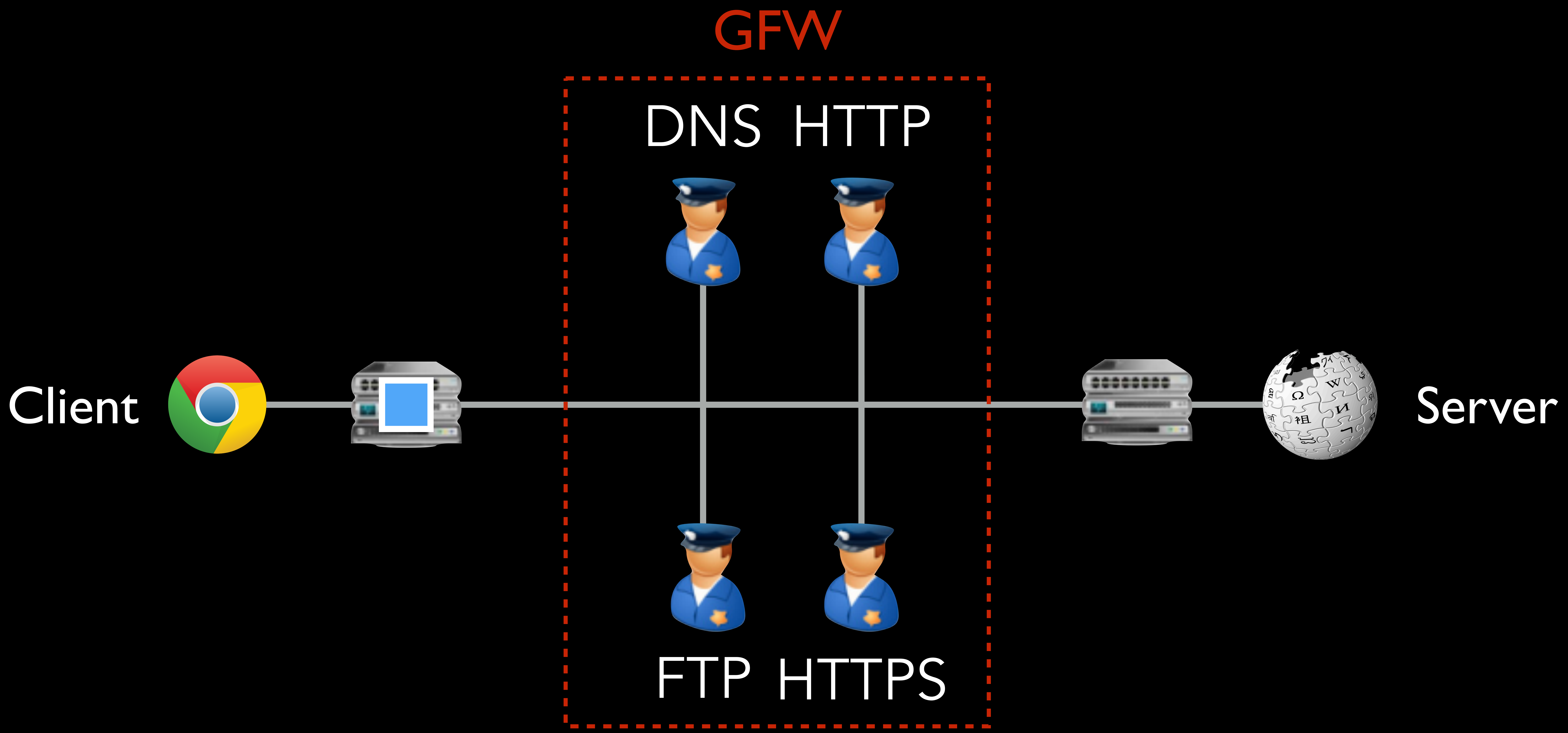
Multi-box theory



Not port number

Censors effectively on any port

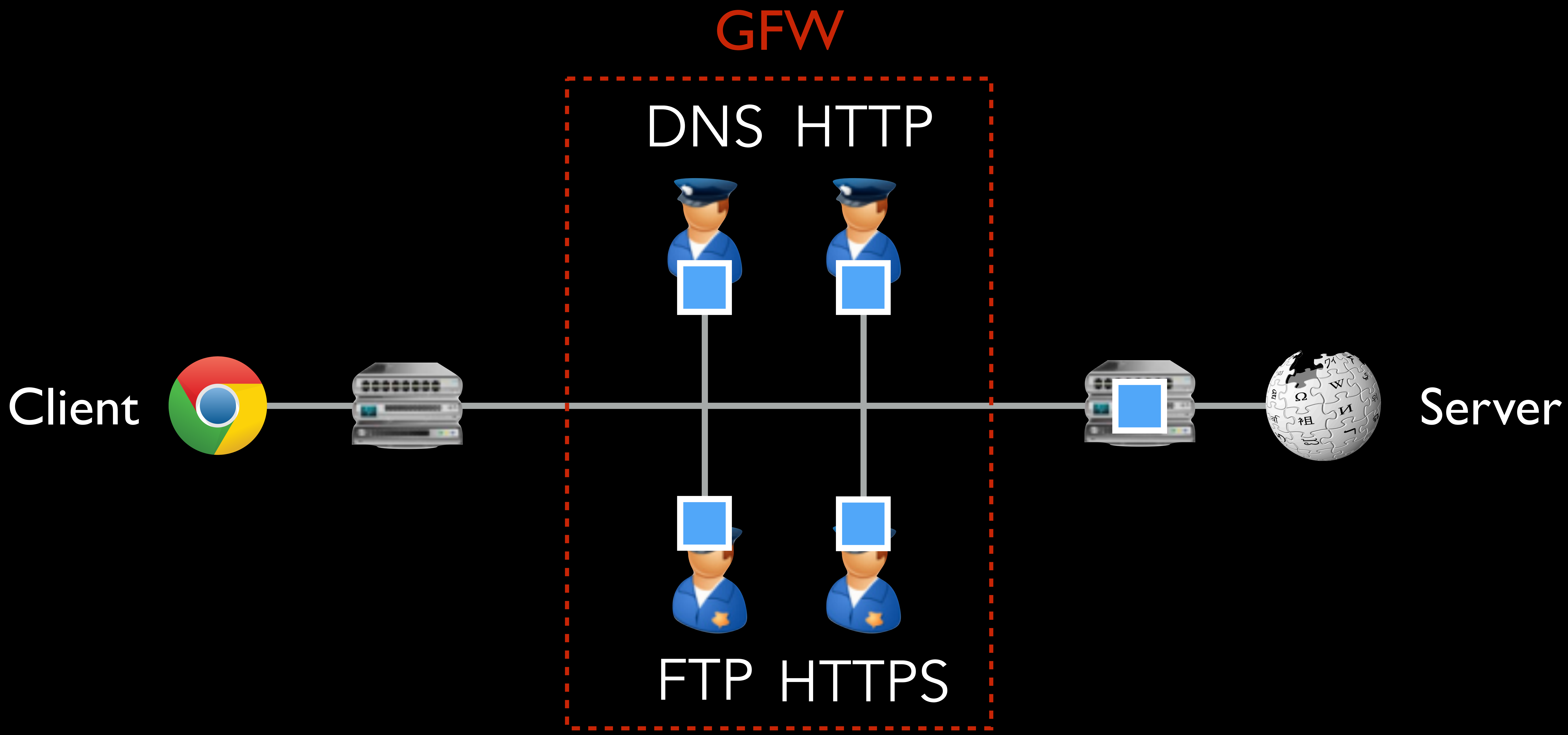
Multi-box theory



Not port number

Censors effectively on any port

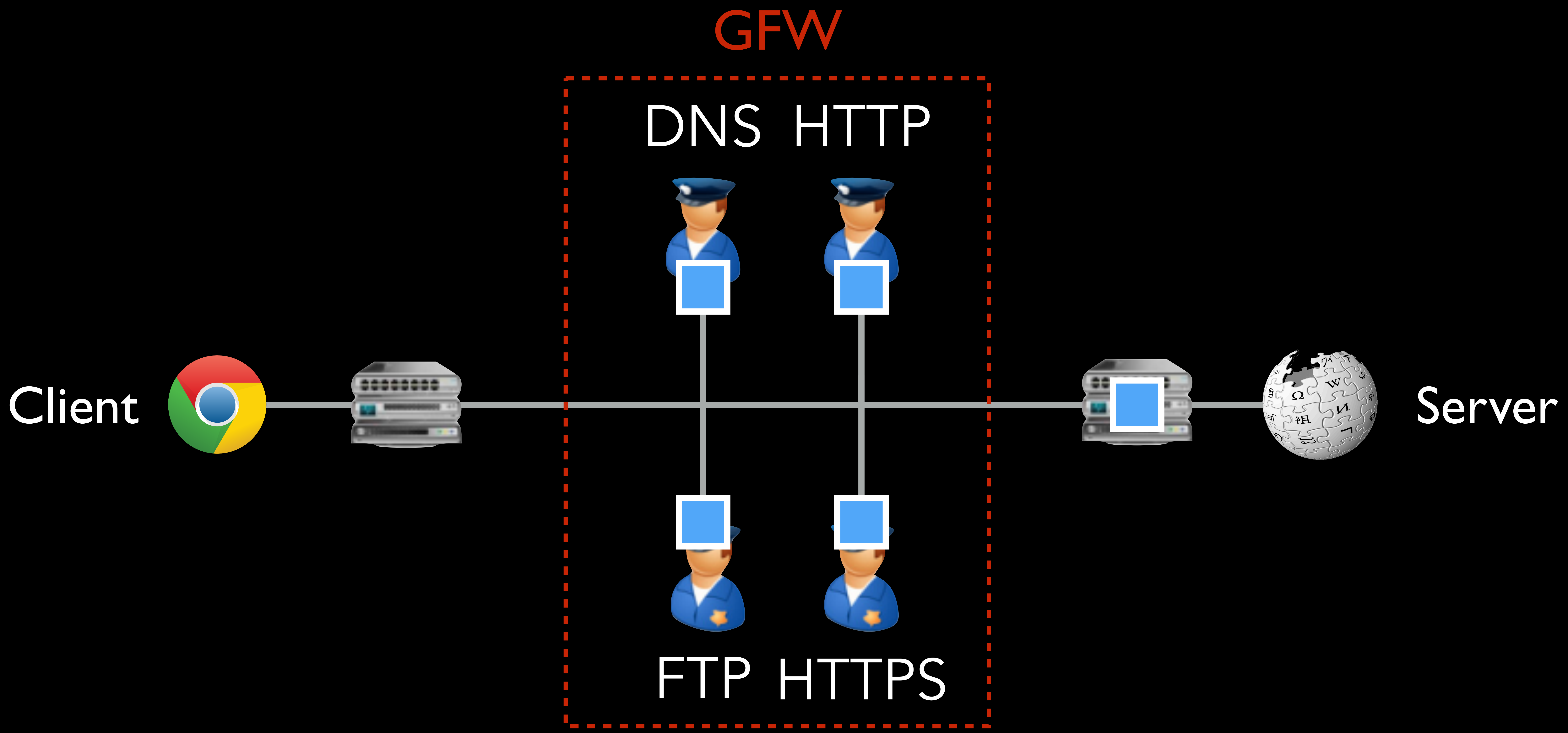
Multi-box theory



Not port number

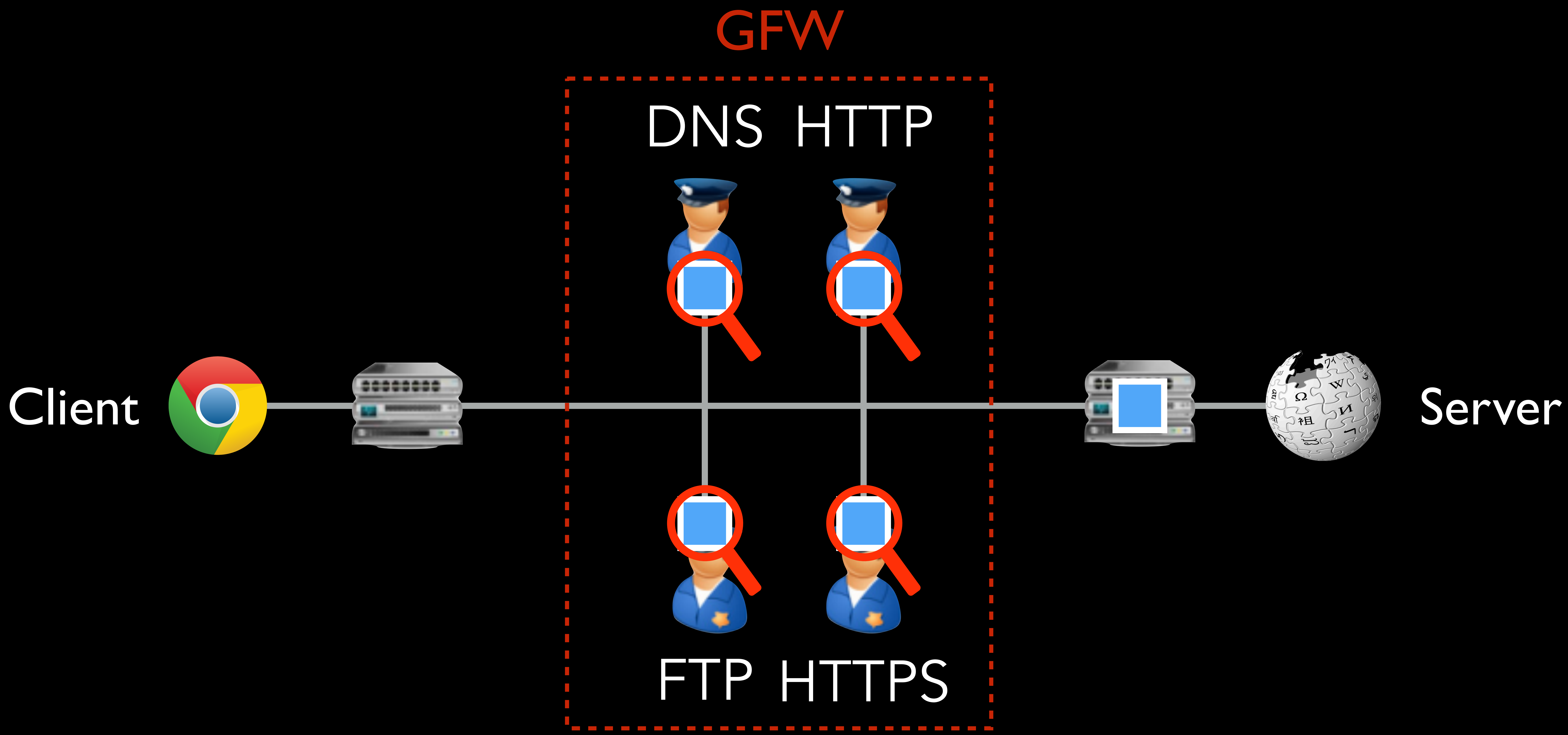
Censors effectively on any port

Multi-box theory



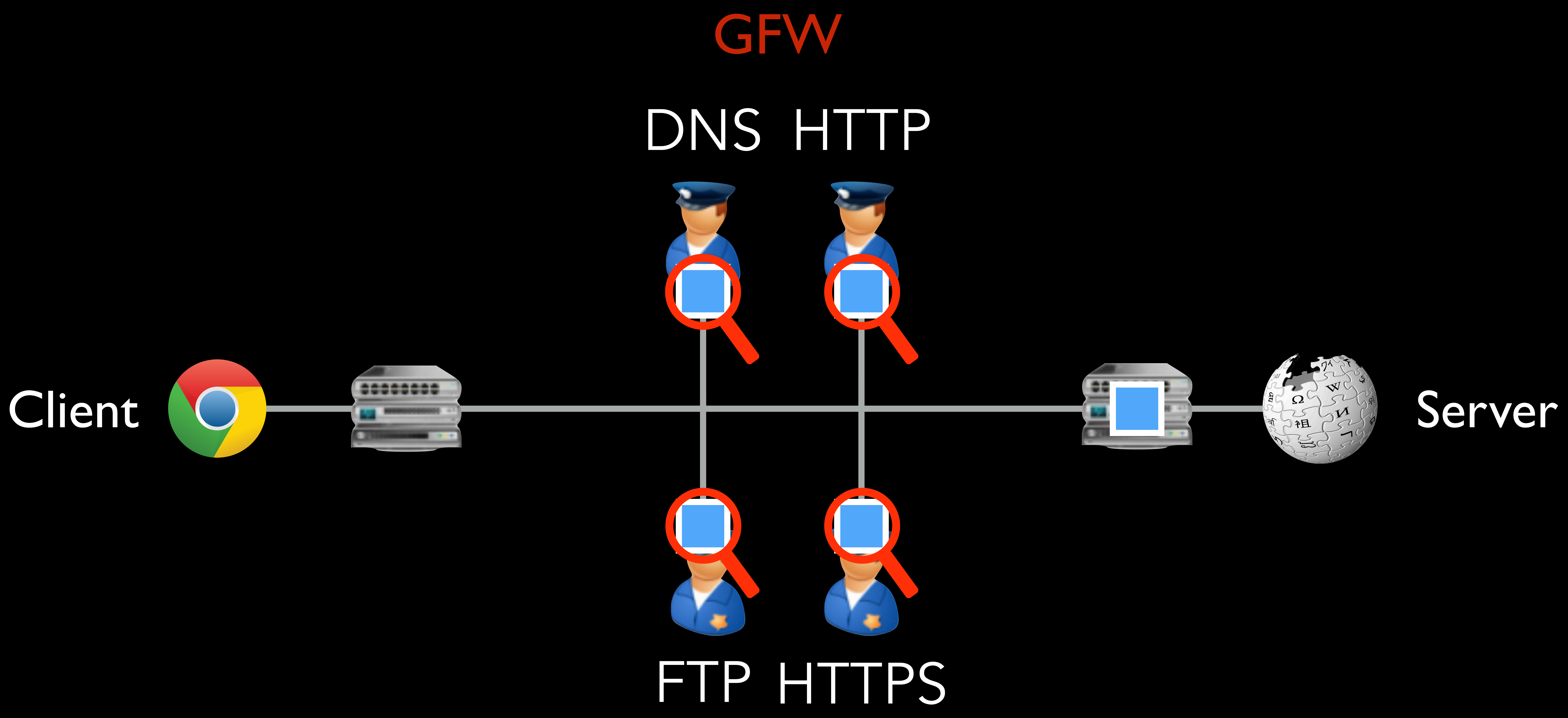
Applies protocol fingerprinting

Multi-box theory



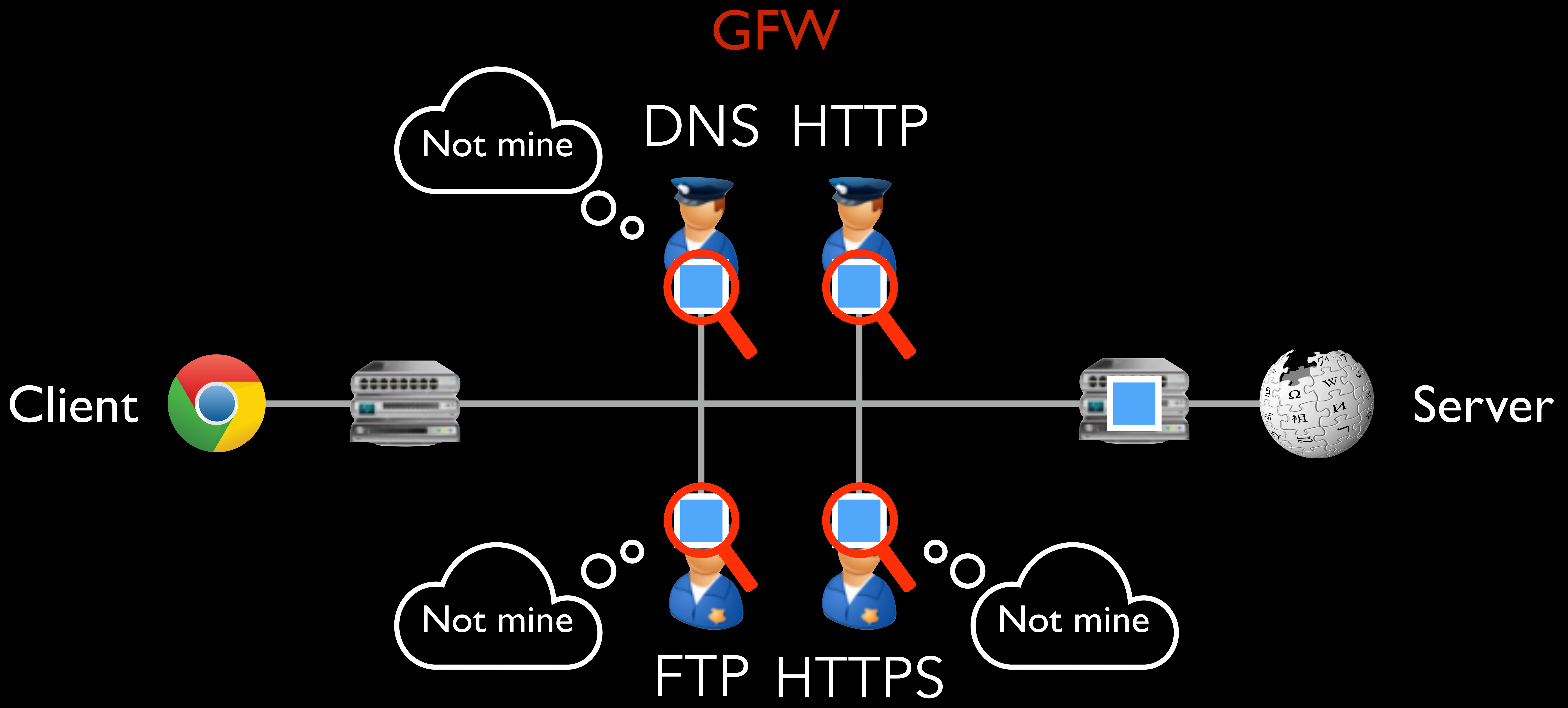
Applies protocol fingerprinting

Multi-box theory



Applies protocol fingerprinting

Multi-box theory



GFW

Not mine

DNS HTTP

Client

Server

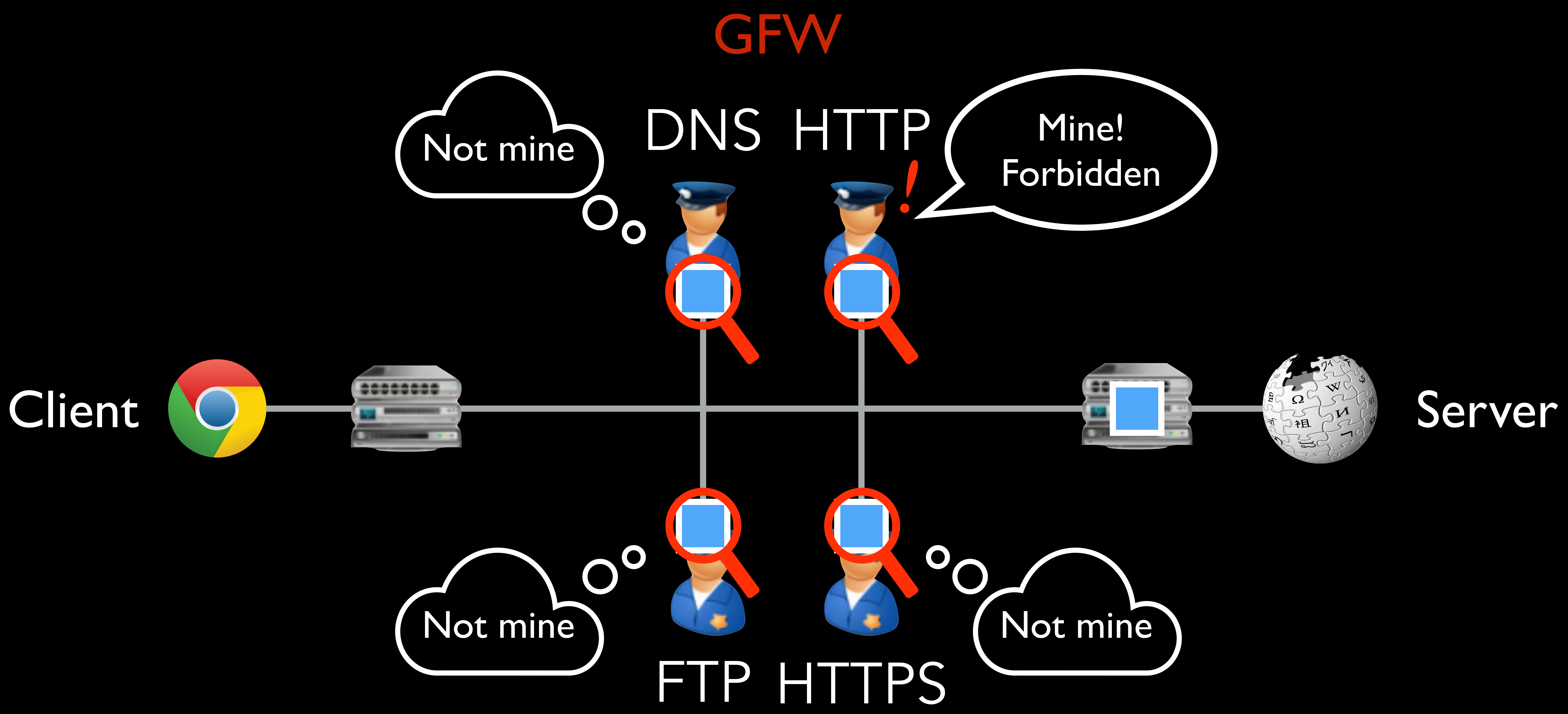
Not mine

Not mine

FTP HTTPS

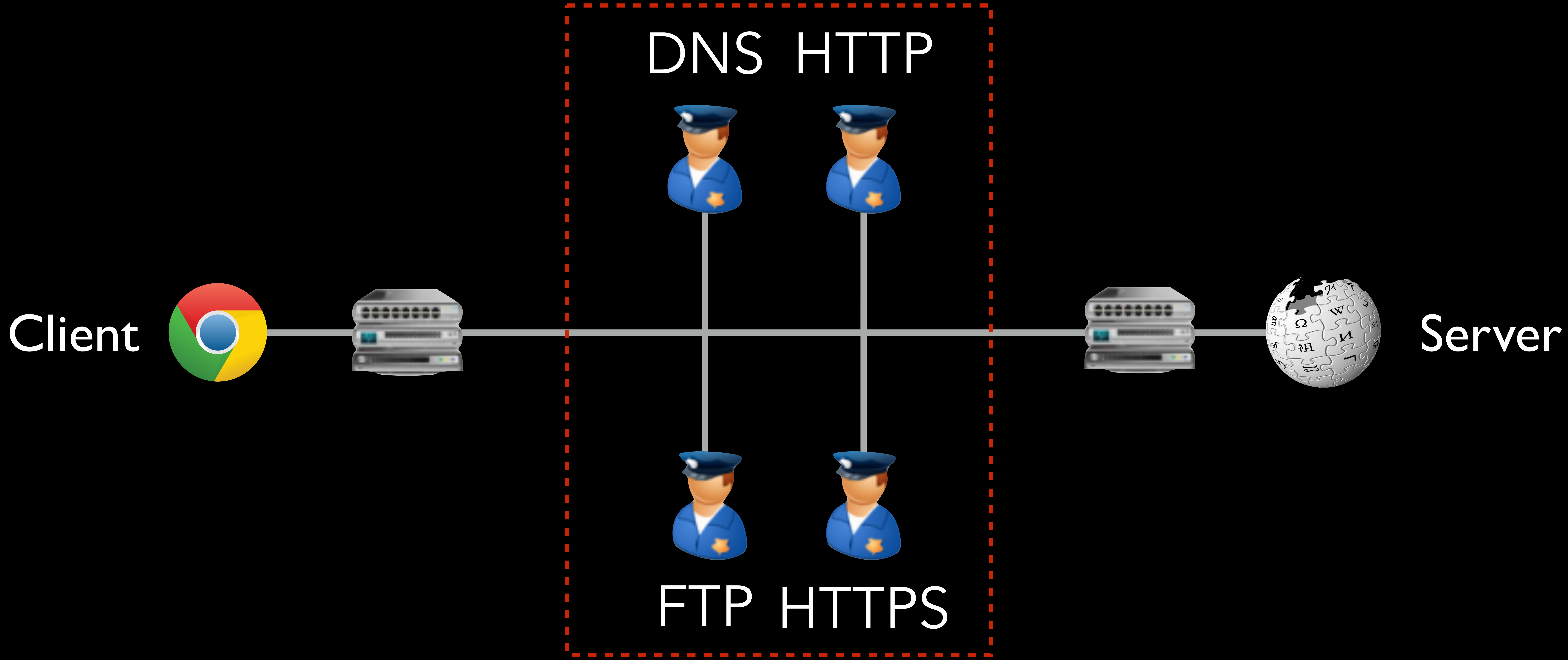
Applies protocol fingerprinting

Multi-box theory



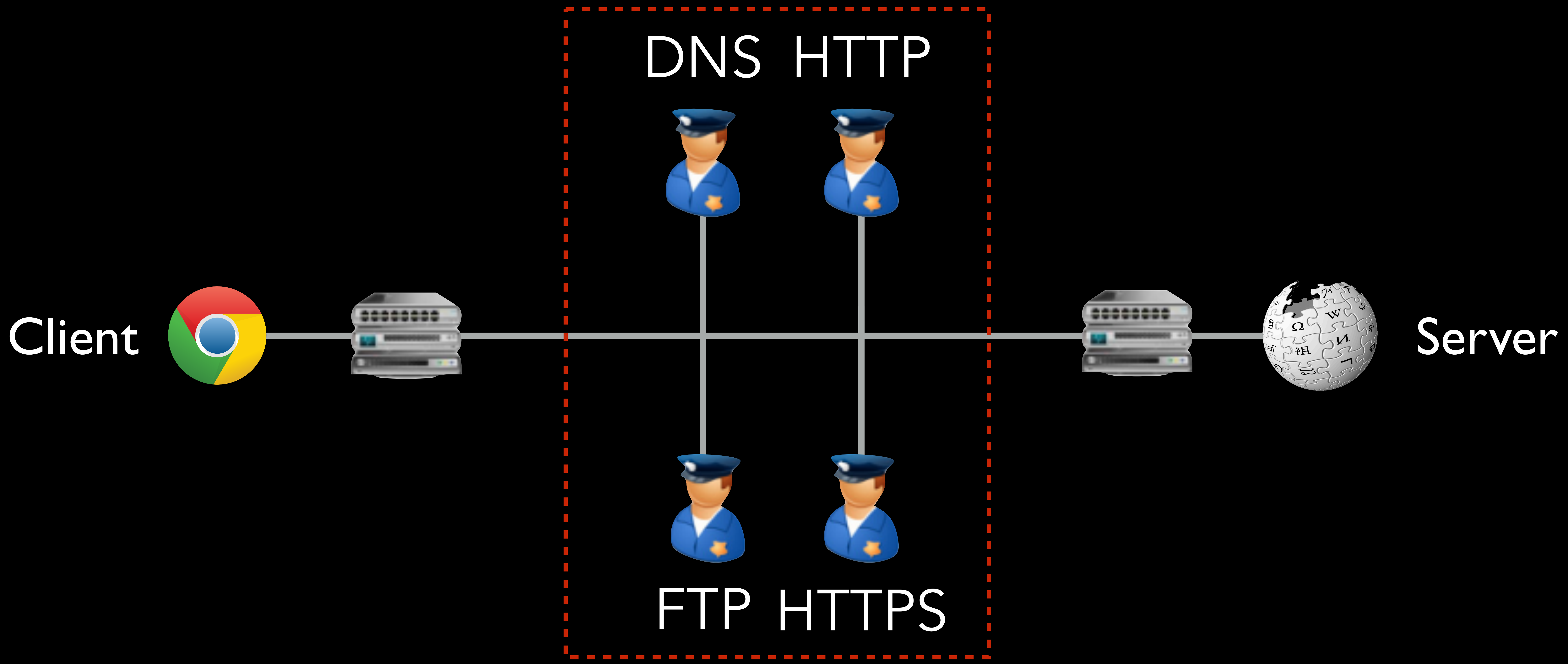
Applies protocol fingerprinting

Where are these middleboxes?



Used TTL-limited probes

Where are these middleboxes?

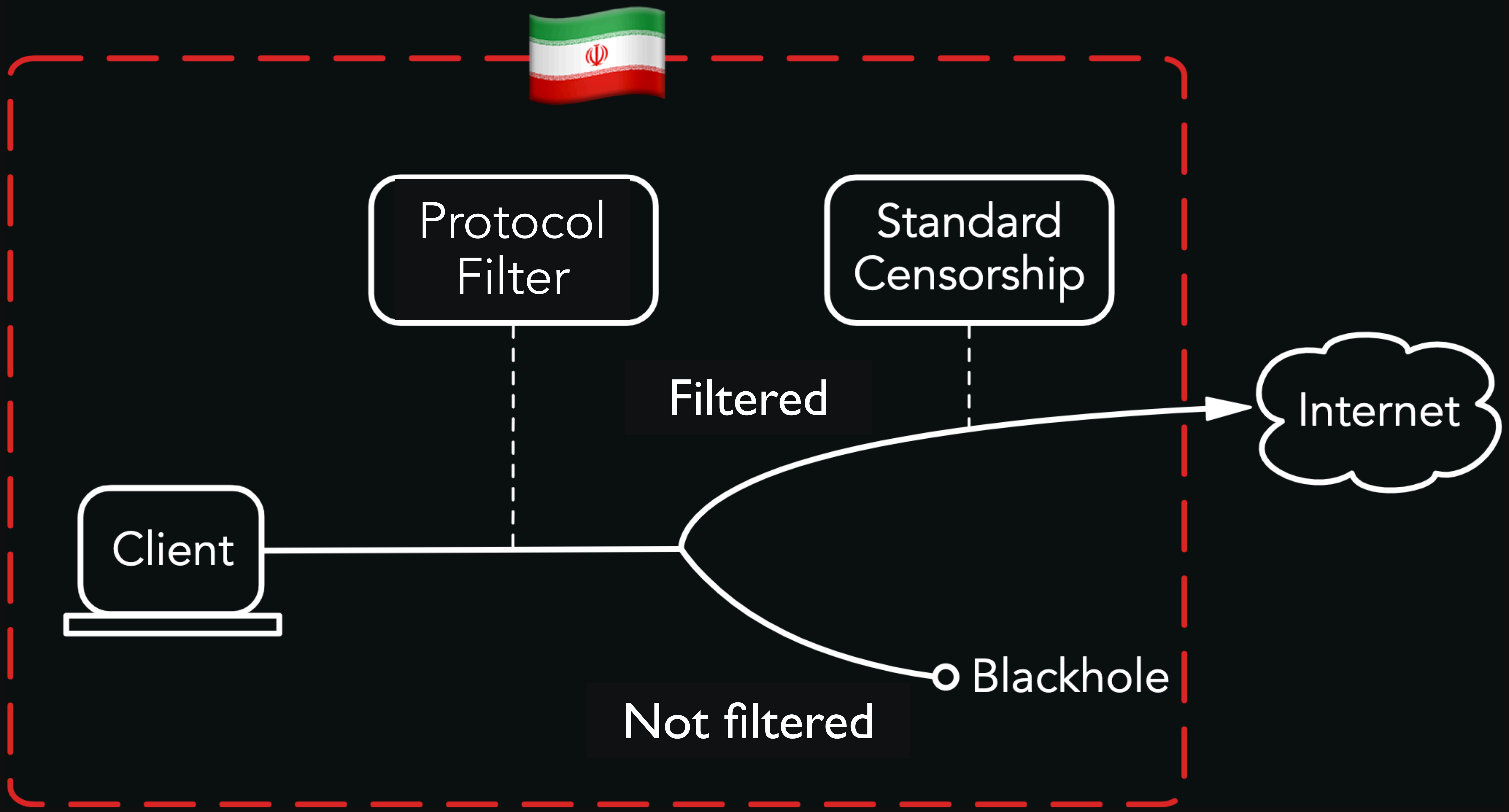


Used TTL-limited probes

Co-located at the network level

🇮🇷 Responsive to new censorship events

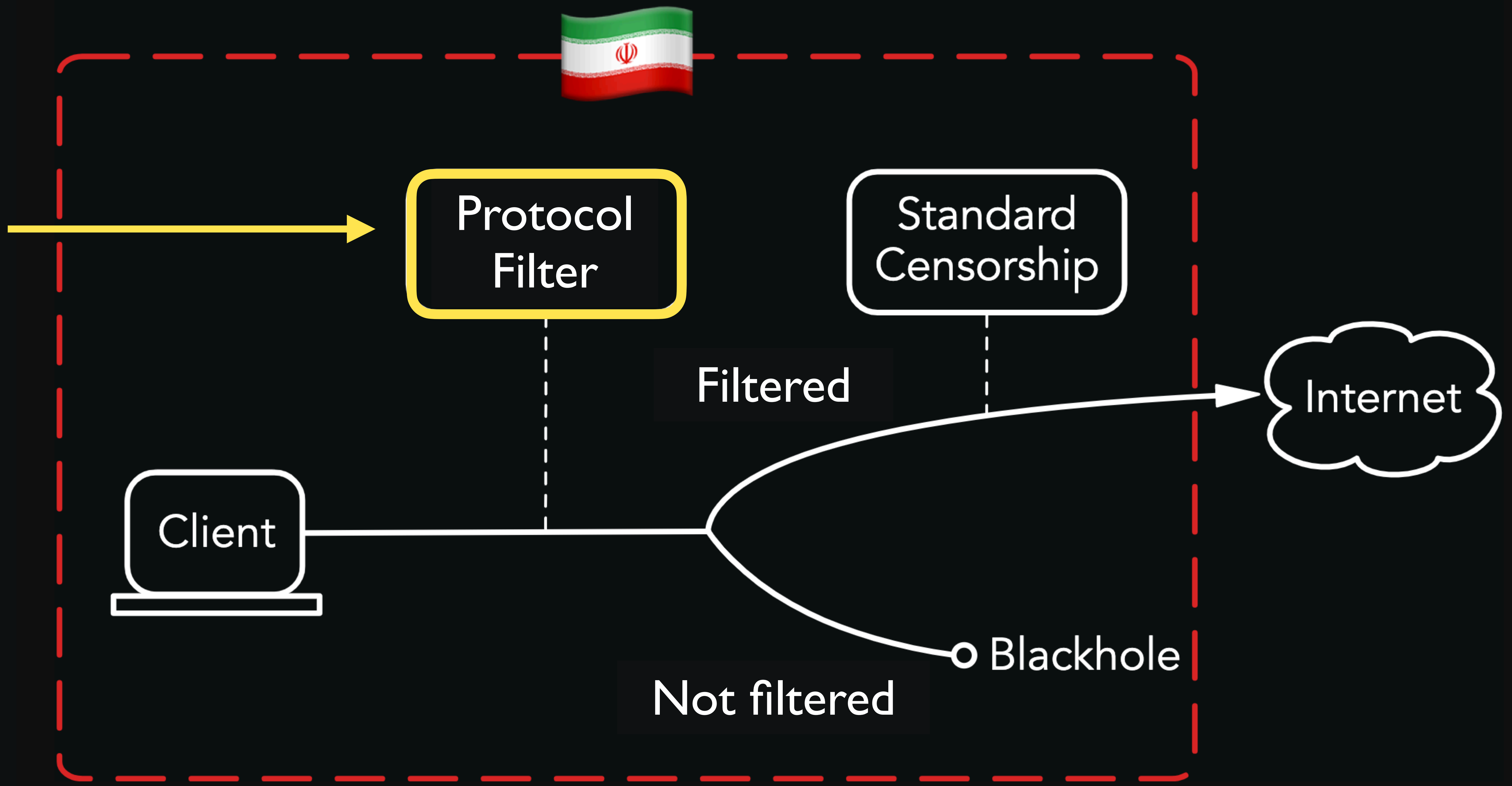
February 2020: Iran launched a new system: a protocol filter



🇮🇷 Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter

Censors connections that do not match protocol fingerprints

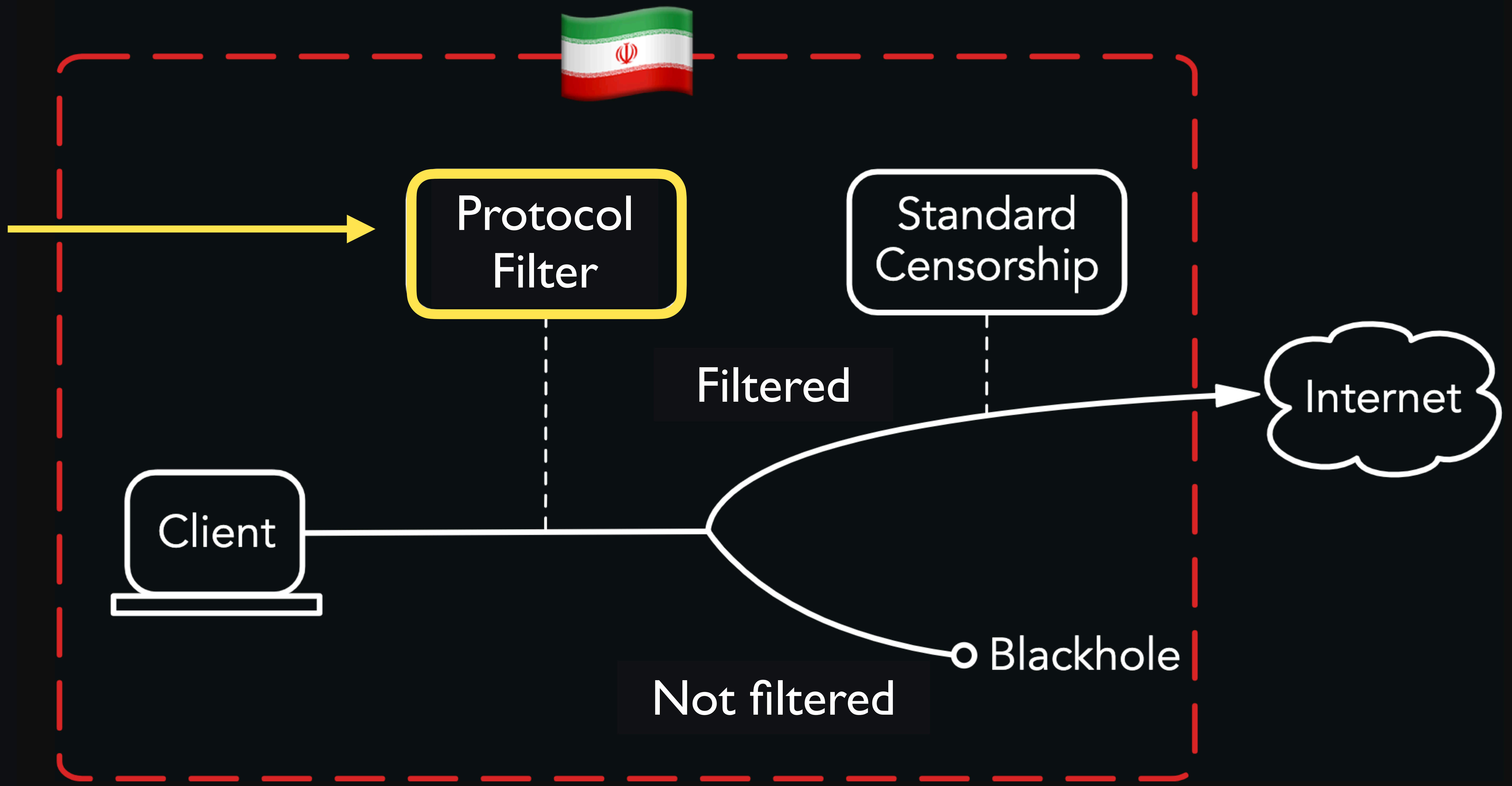


🇮🇷 Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter

Censors connections that do not match protocol fingerprints

Those that do match are then subjected to standard censorship

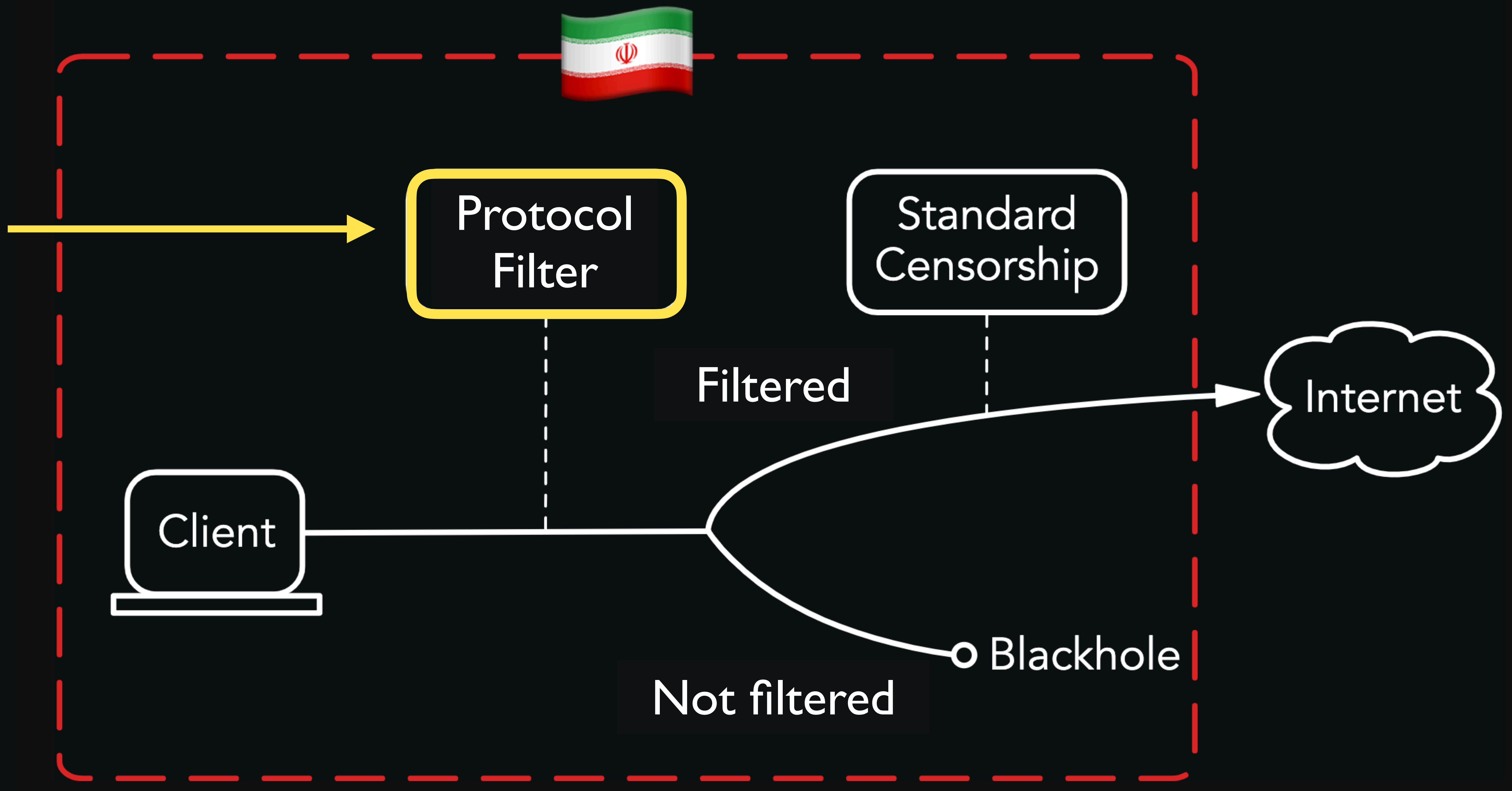


🇮🇷 Responsive to new censorship events

February 2020: Iran launched a new system: a protocol filter

Censors connections that do not match protocol fingerprints

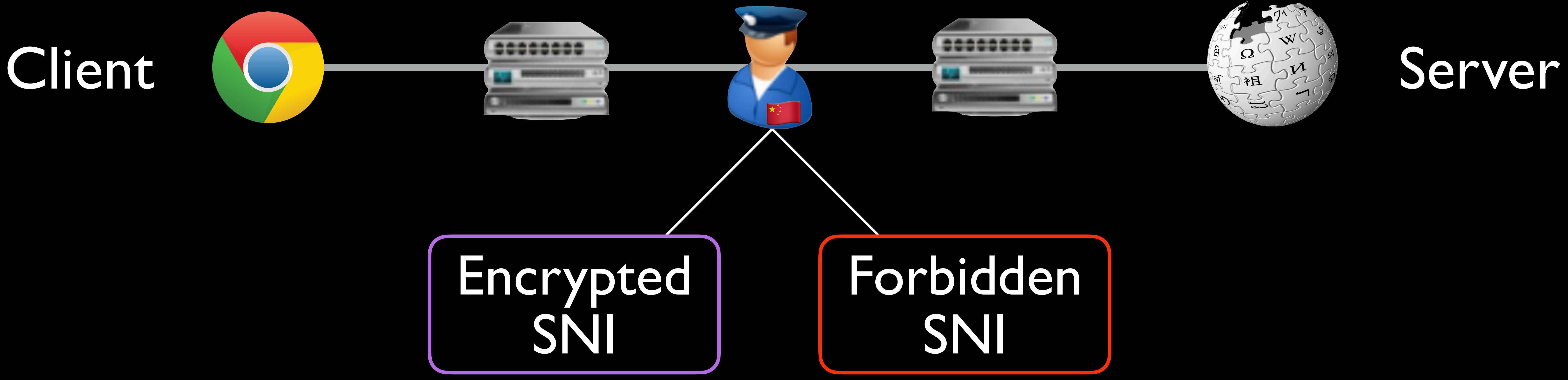
Those that do match are then subjected to standard censorship



Geneva discovered 4 strategies to evade Iran's filter

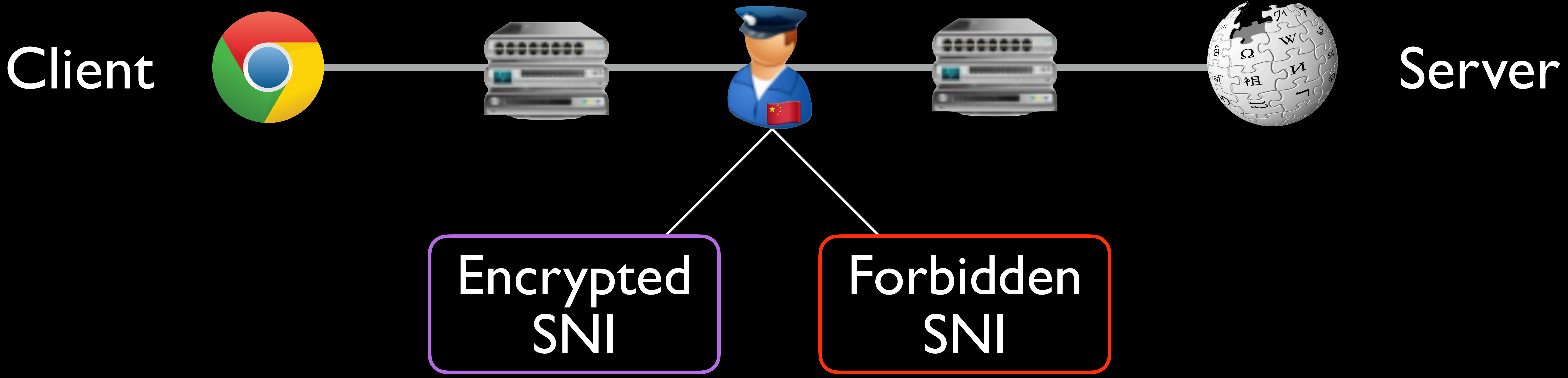
Responsive to new censorship events

July 29th 2020: China begins censoring the use of Encrypted SNI



Responsive to new censorship events

July 29th 2020: China begins censoring the use of Encrypted SNI



Client

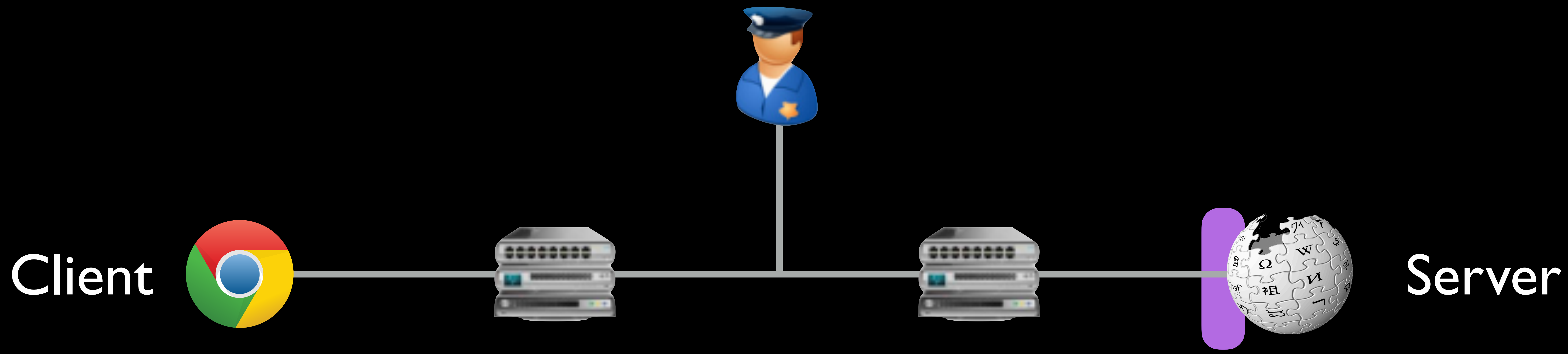
Server

Encrypted SNI

Forbidden SNI

Geneva discovered 6 strategies to evade ESNI censorship

Real world deployment



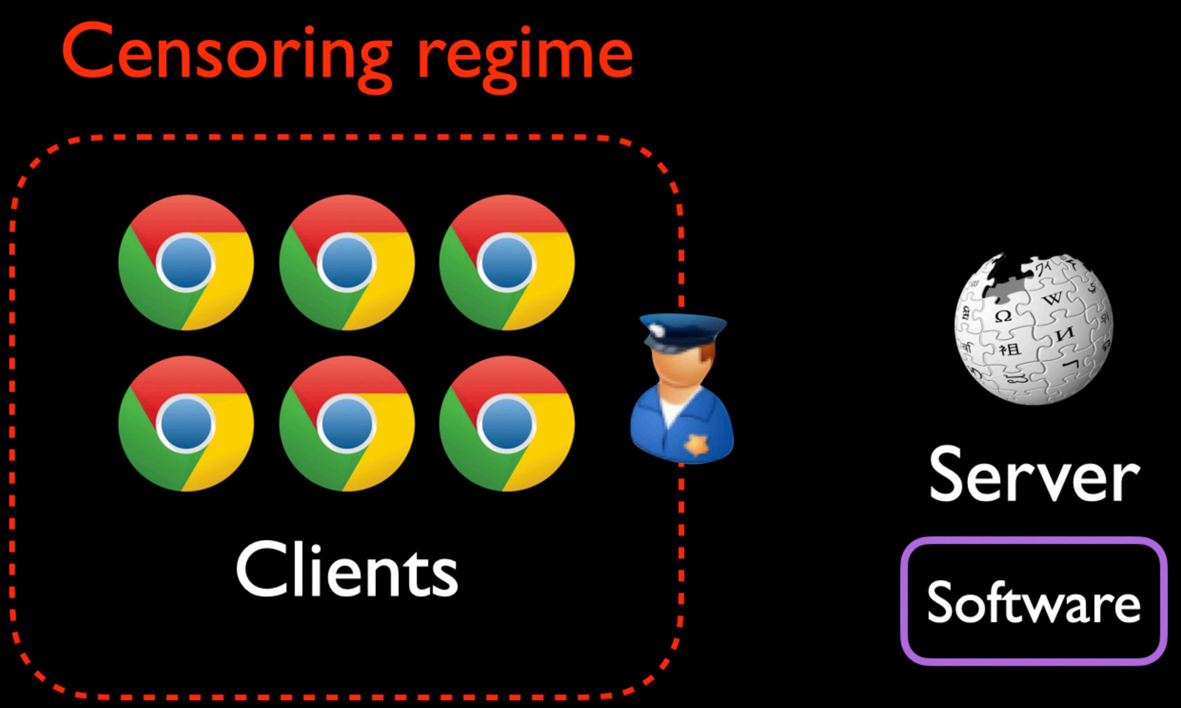
Assist in bootstrapping connections

Harden existing evasion protocols

Middleboxes create new possibilities

The good

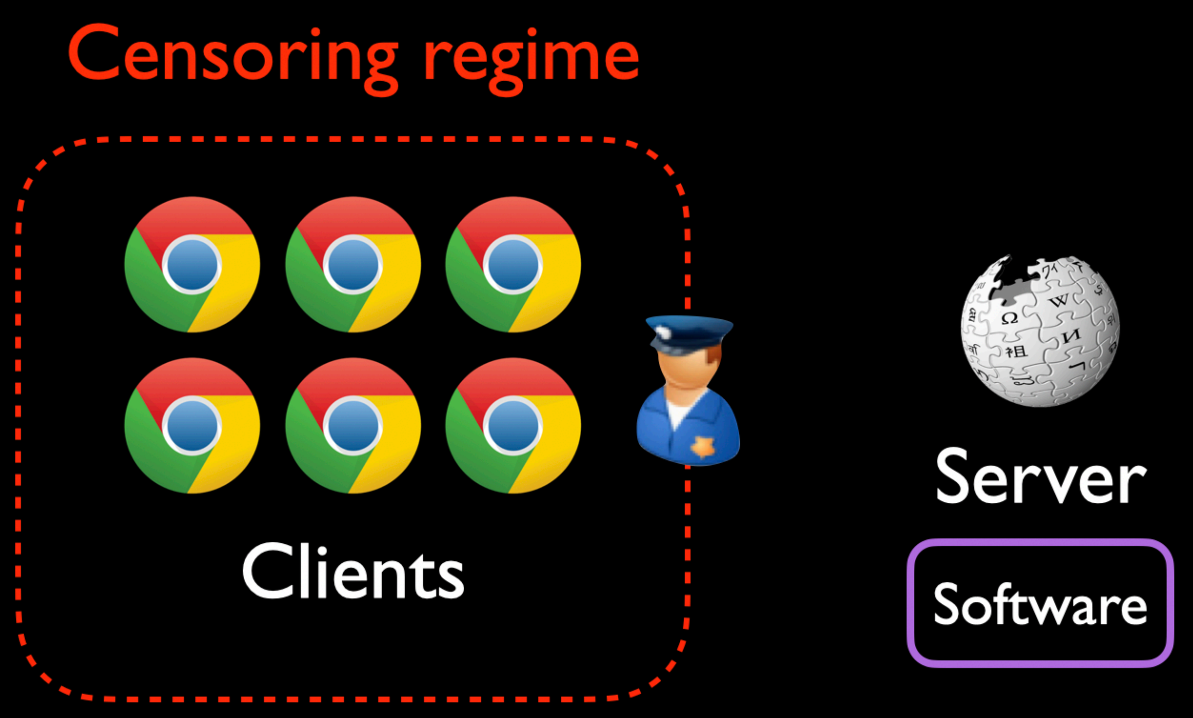
They make server-side evasion possible!



Middleboxes create new possibilities

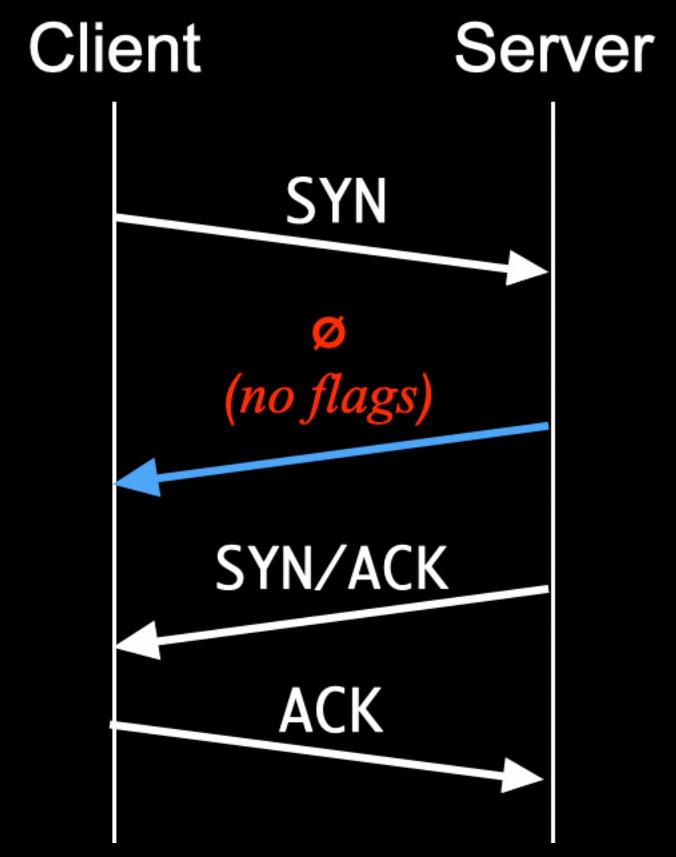
The good

They make server-side evasion possible!



The ugly

They have exploitable bugs and assumptions



Middleboxes create new possibilities

The good

They make server-side evasion possible!

The very bad

Middleboxes can be weaponized

The ugly

They have exploitable bugs and assumptions

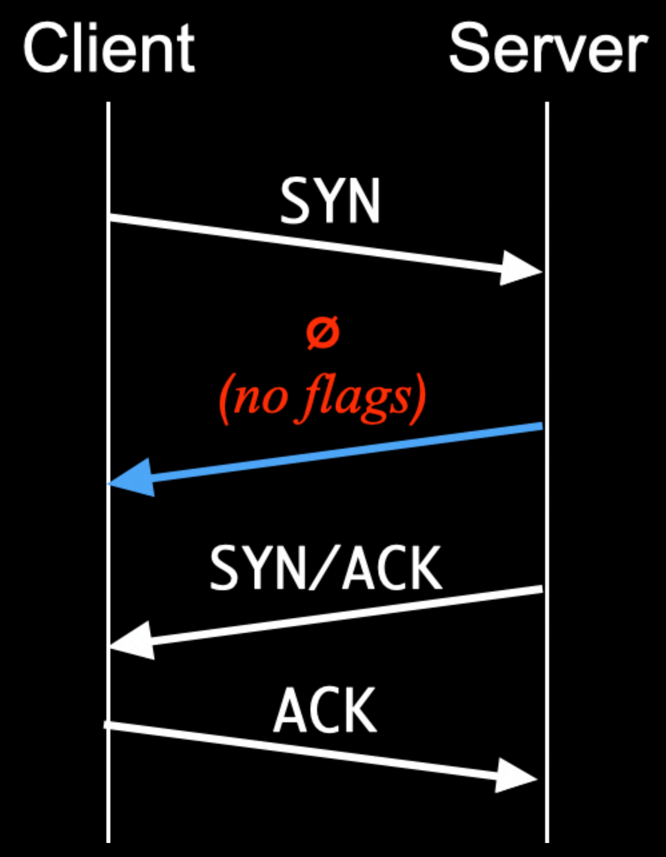
Censoring regime
Clients
Server
Software

Weaponizing Middleboxes for TCP Reflected Amplification

Kevin Bock* Abdulrahman Alaraj† Yair Fax* Kyle Hurley* Eric Wustrow† Dave Levin*
*University of Maryland †University of Colorado Boulder

USENIX Security '21

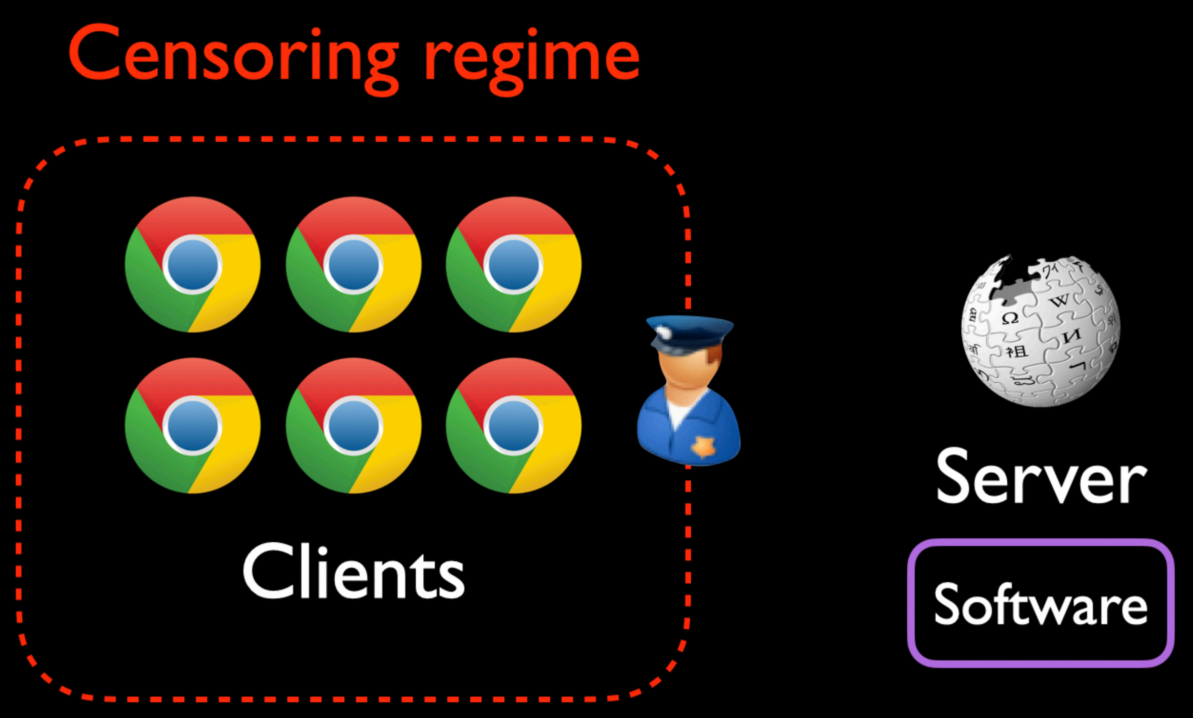
TCP-based reflected amplification



Middleboxes create new possibilities

The good

They make server-side evasion possible!



The very bad

Middleboxes can be weaponized

Weaponizing Middleboxes for TCP Reflected Amplification

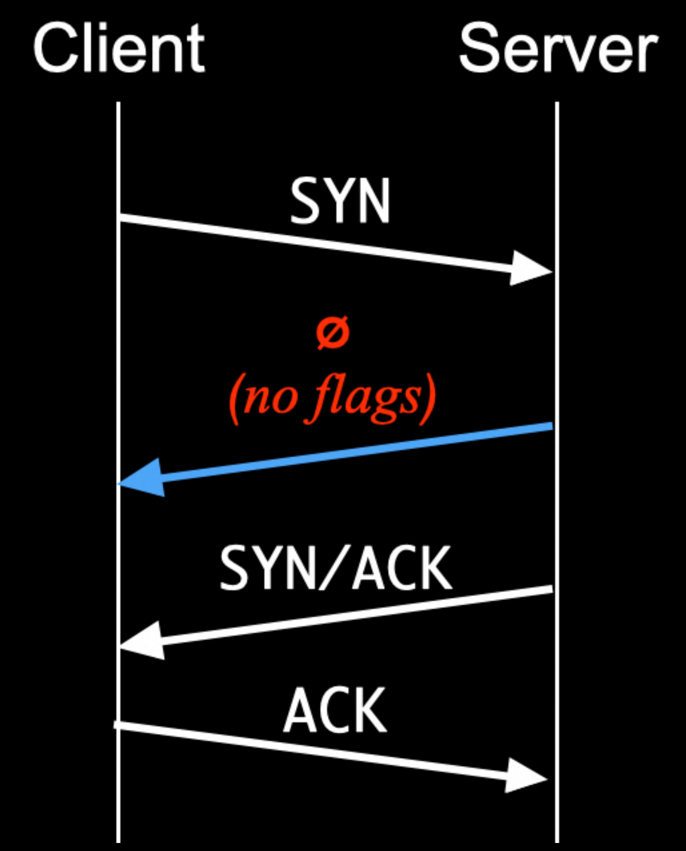
Kevin Bock* Abdulrahman Alaraj† Yair Fax* Kyle Hurley* Eric Wustrow† Dave Levin*
*University of Maryland †University of Colorado Boulder

USENIX Security '21

TCP-based reflected amplification

The ugly

They have exploitable bugs and assumptions



Automated tools like Geneva are important in understanding what middleboxes enable

Server-side Evasion

Geneva

Genetic Evasion

Server-side evasion is possible

New insights into censors

Code is open source

Real world deployment

Geneva code and website geneva.cs.umd.edu