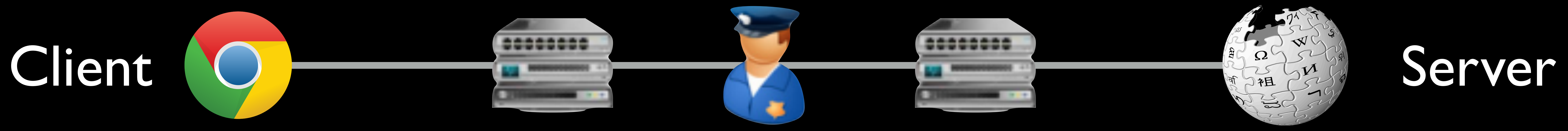
Automating Censorship Evasion

Kevin Bock

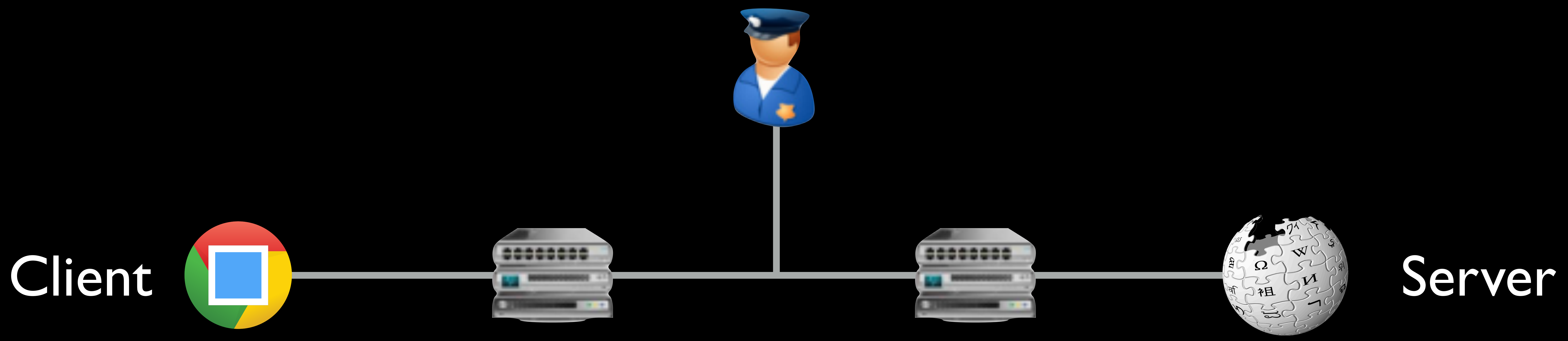
UNIVERSITY OF MARYLAND

TECHNICA

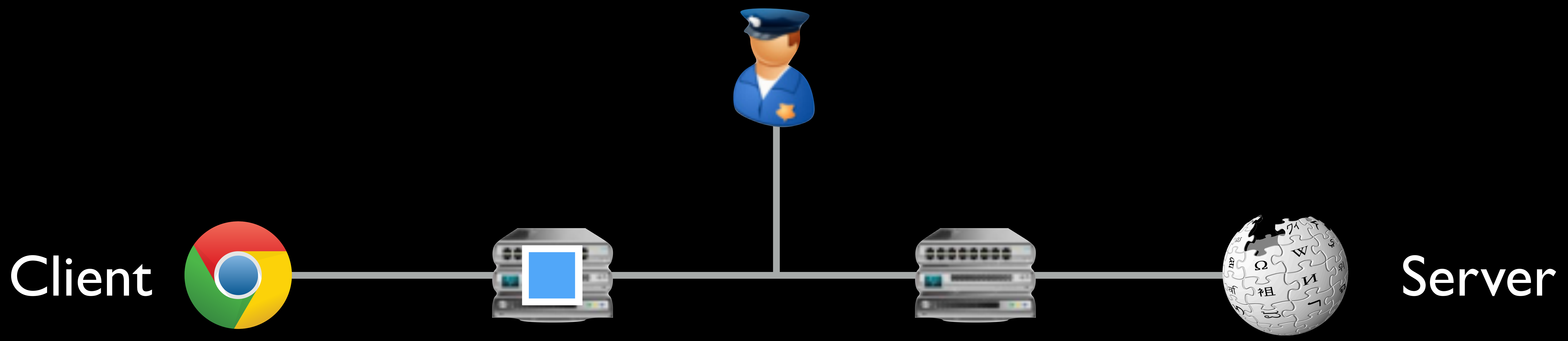
In-network censorship by nation-states



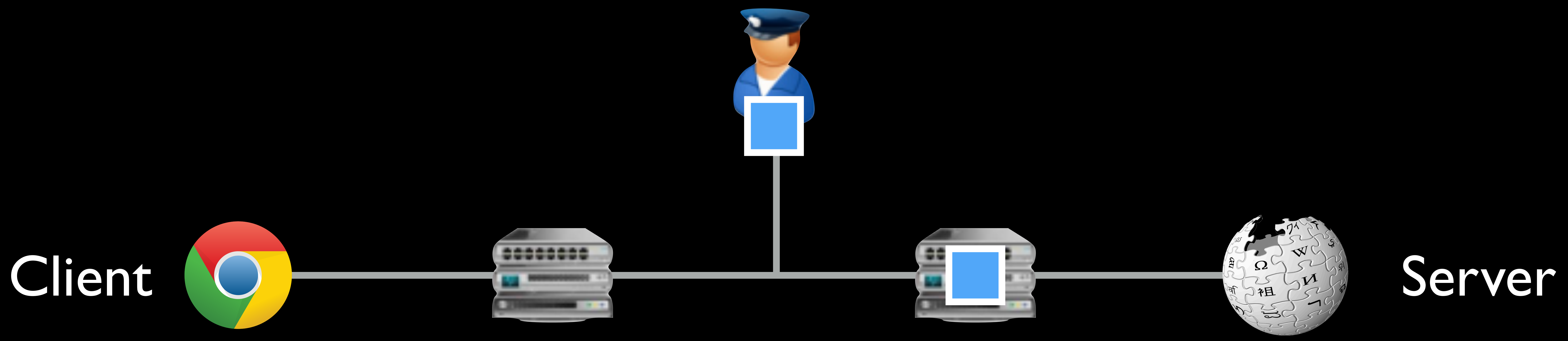
In-network censorship by nation-states



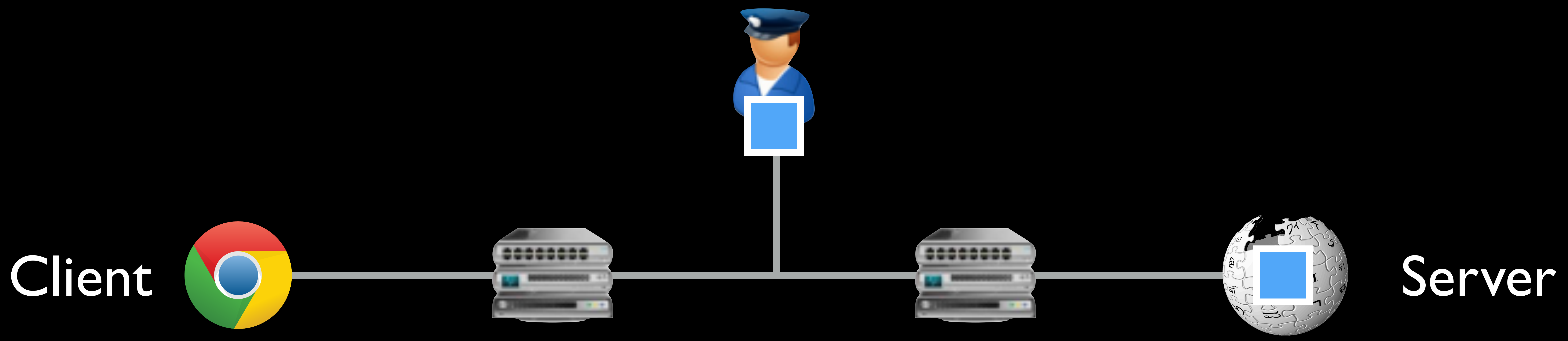
In-network censorship by nation-states



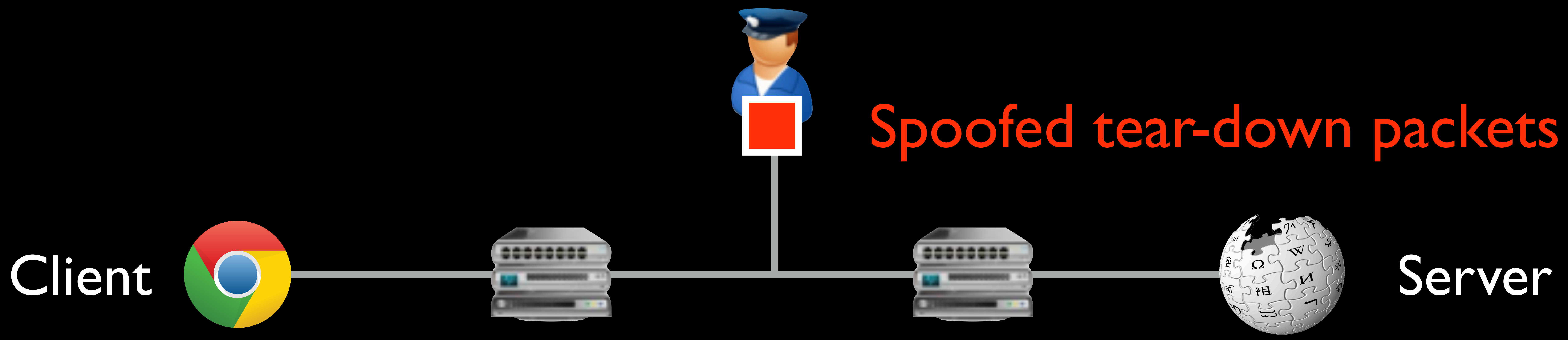
In-network censorship by nation-states



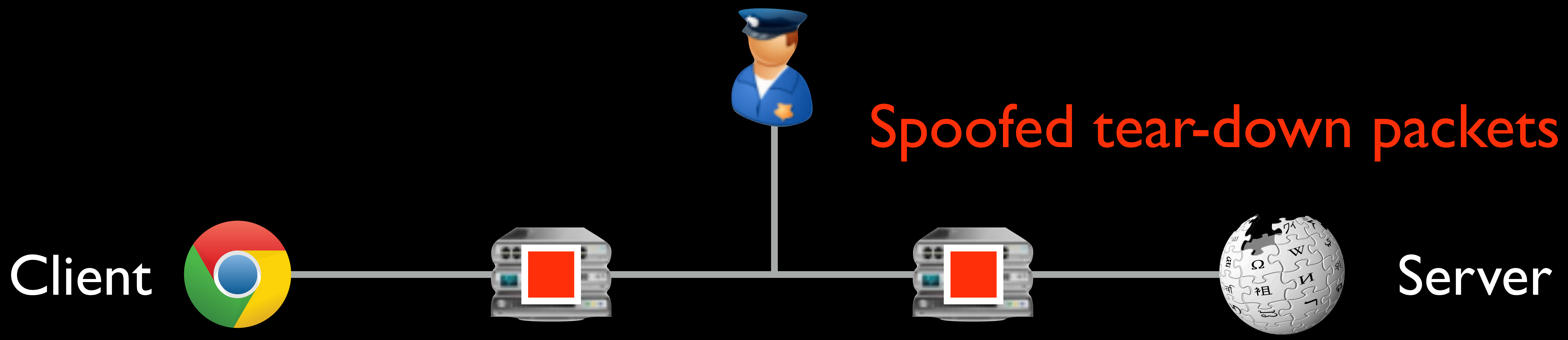
In-network censorship by nation-states



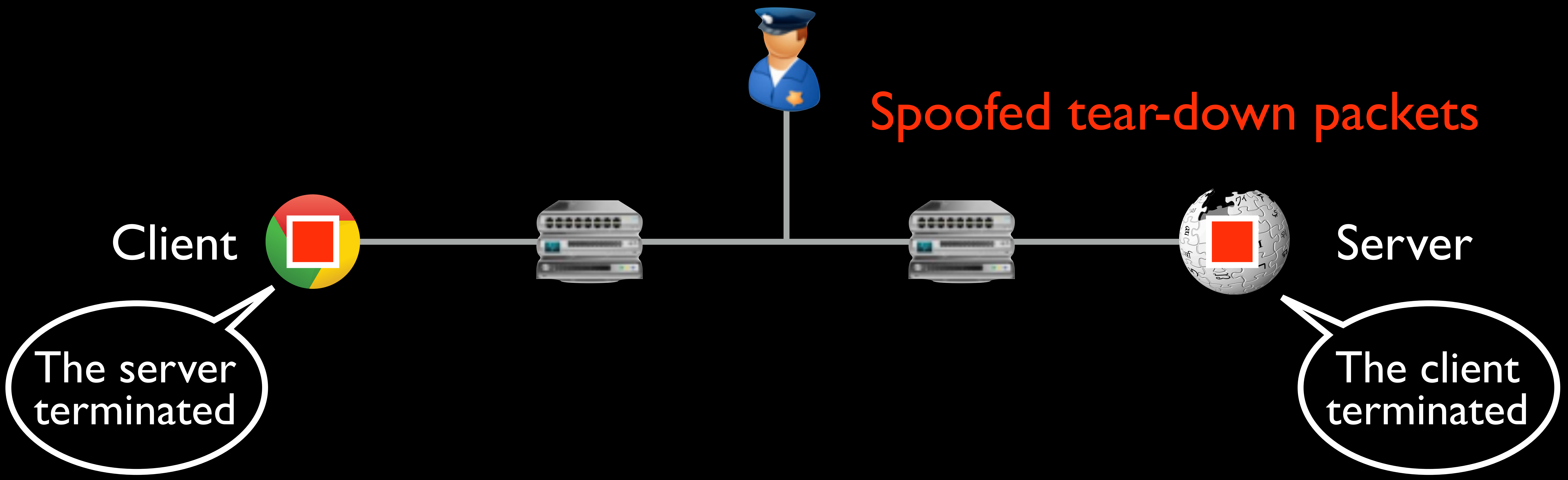
In-network censorship by nation-states



In-network censorship by nation-states



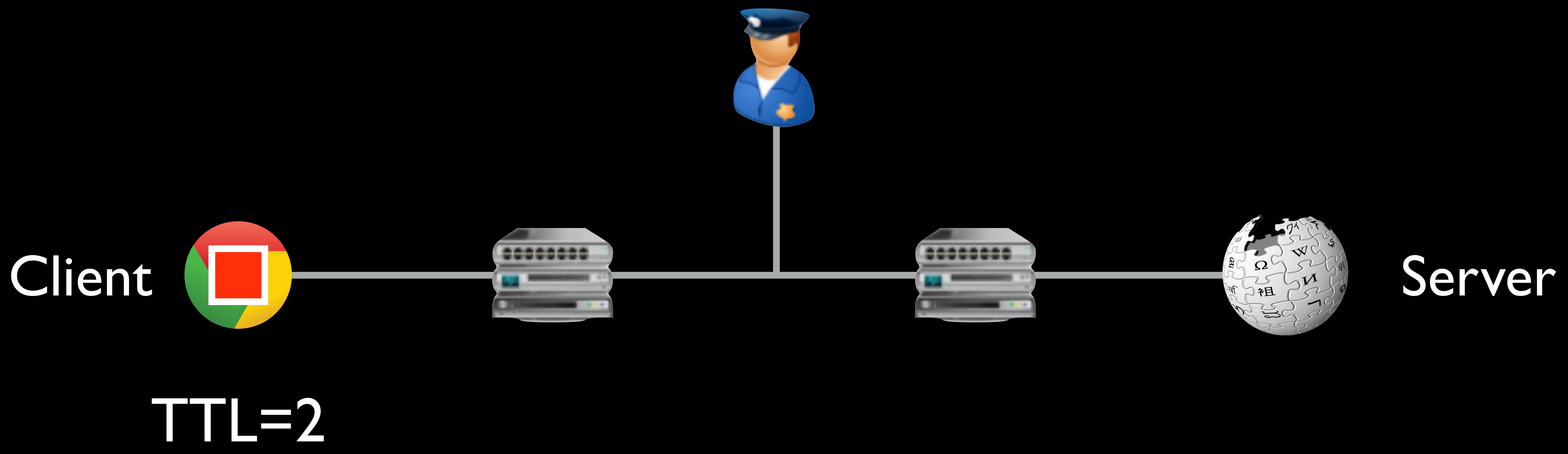
In-network censorship by nation-states



Requires *per-flow state*

Censors fighting *end to end principle*

In-network censorship by nation-states

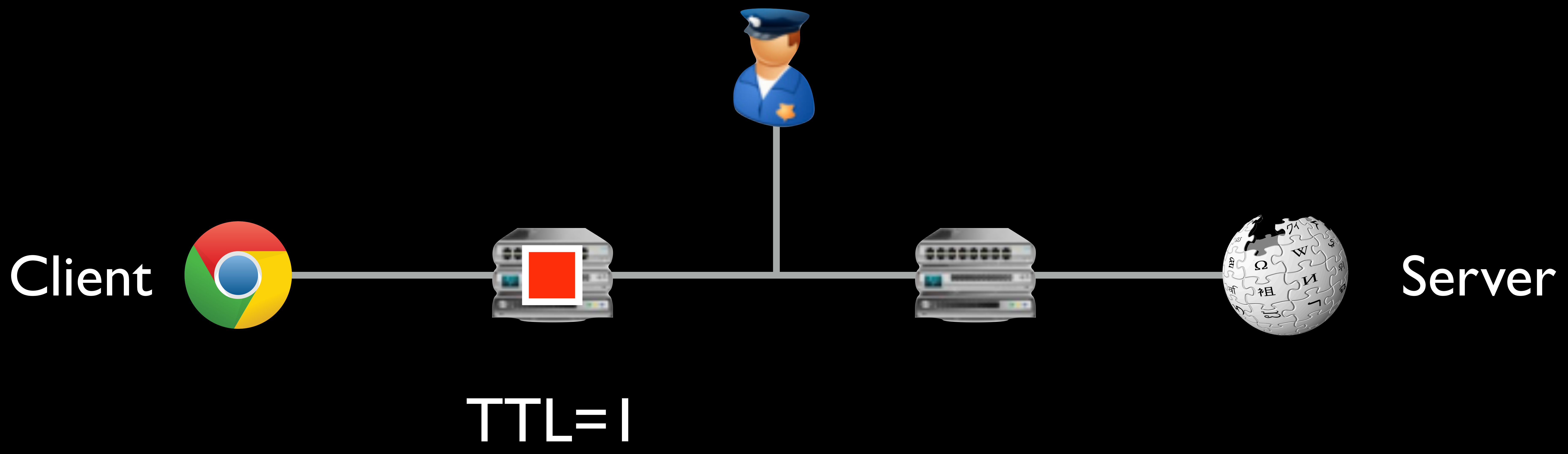


Requires *per-flow state*

Censors fighting *end to end principle*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states

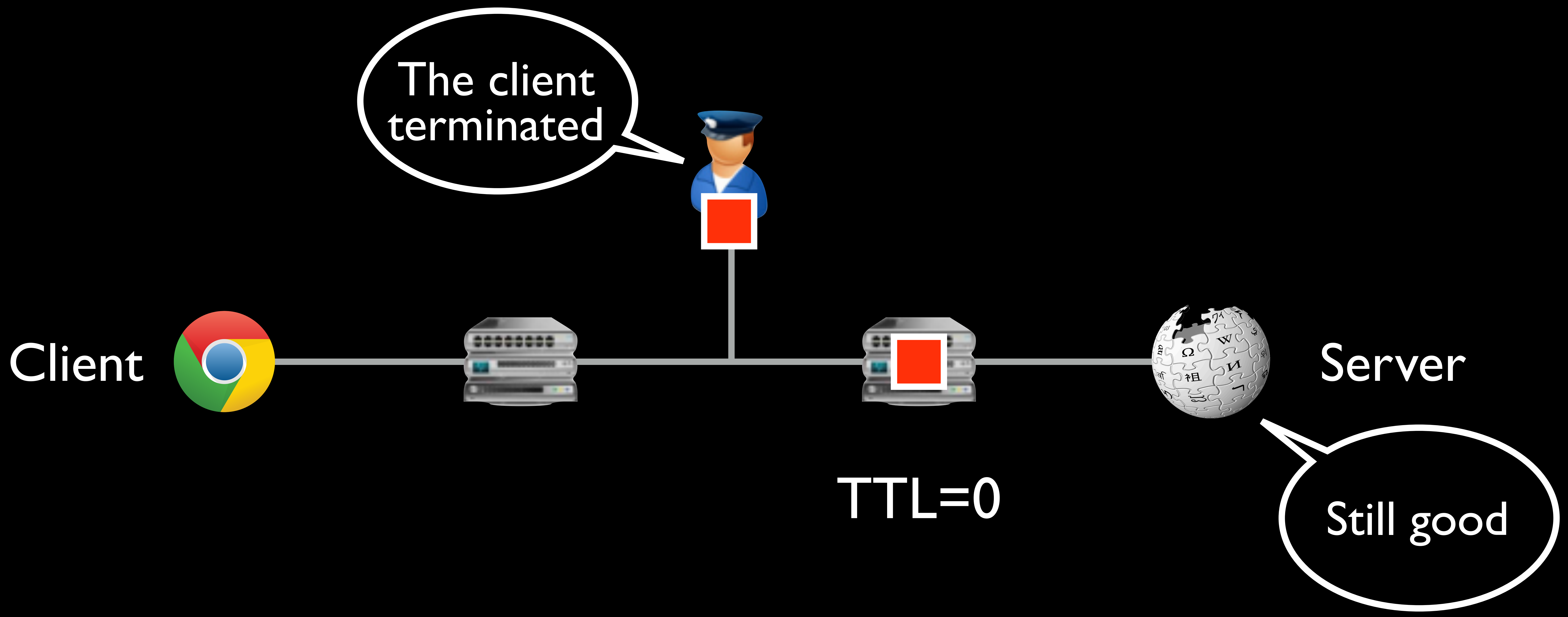


Requires *per-flow state*

Censors fighting *end to end principle*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



Requires *per-flow state*

Censors fighting *end to end principle*

Evasion can take advantage of these shortcuts

Censorship evasion research

1 *Understand* how censors operate

2 *Apply insight* to create evasion strategies

Largely manual efforts give censors the advantage

Our work gives evasion the advantage

Automated censorship evasion research

1. Automate the discovery of new evasion strategies

2. Use the strategies to understand how the censor works

Geneva

Genetic Evasion

1 Automate the discovery of new evasion strategies

2 Use the strategies to understand how the censor works

Geneva

Genetic Evasion

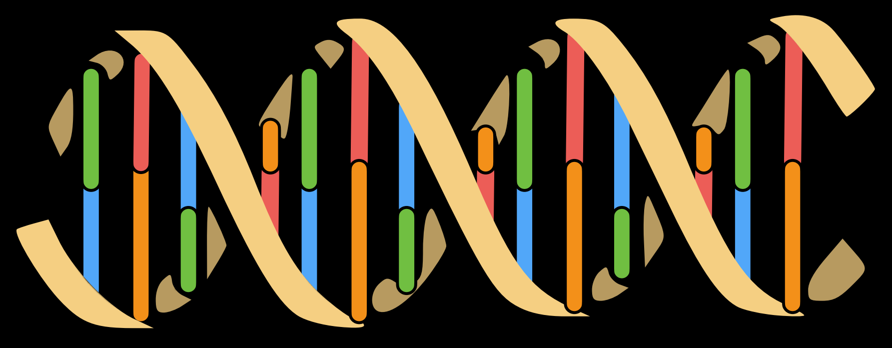
Building Blocks	Composition	Mutation	Fitness

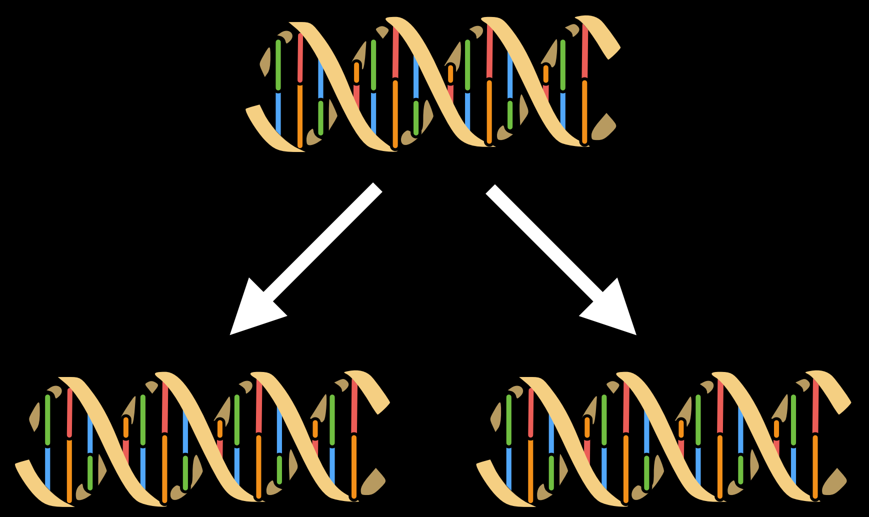
A

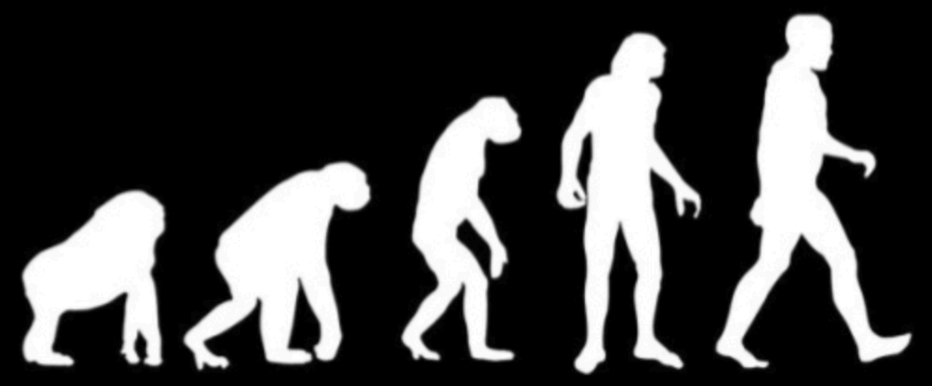
T

C

G



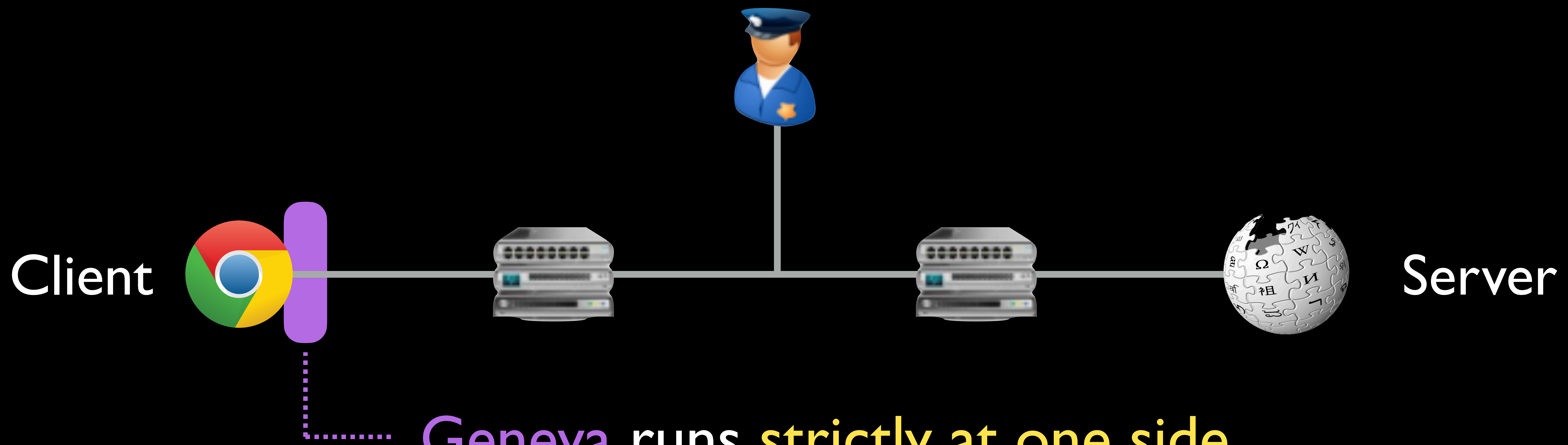




Geneva

Genetic Evasion

Building Blocks



Geneva runs strictly at one side

Manipulates packets to and from the client

Geneva

Genetic Evasion

Building Blocks

Manipulates packets to and from the client

Bit manipulation

Versatile but inefficient

Known strategies

Efficient but limited

Geneva

Genetic Evasion

Building Blocks

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

Geneva

Genetic Evasion

Building Blocks

Actions manipulate individual packets

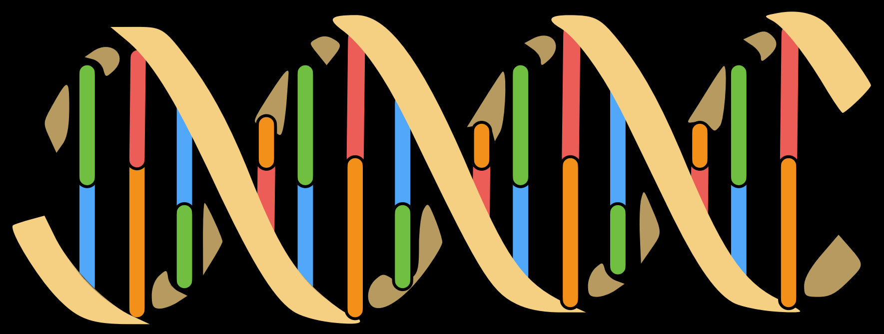
Duplicate

Tamper

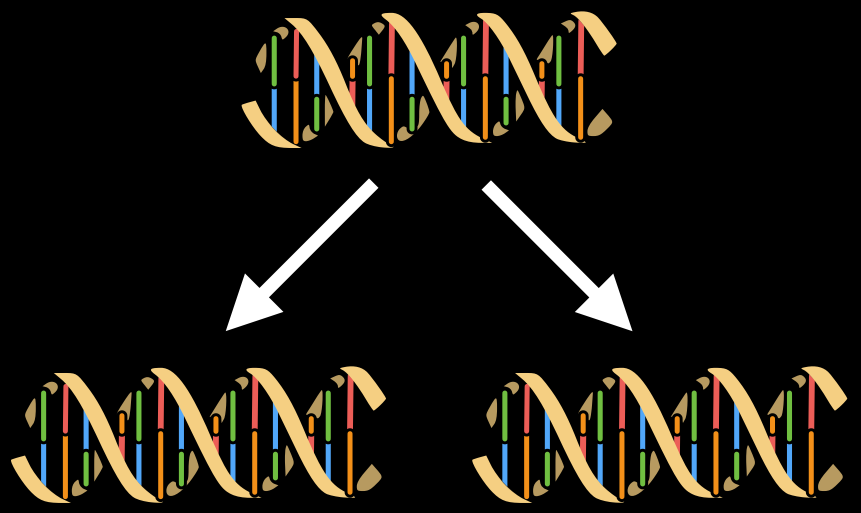
Fragment

Drop

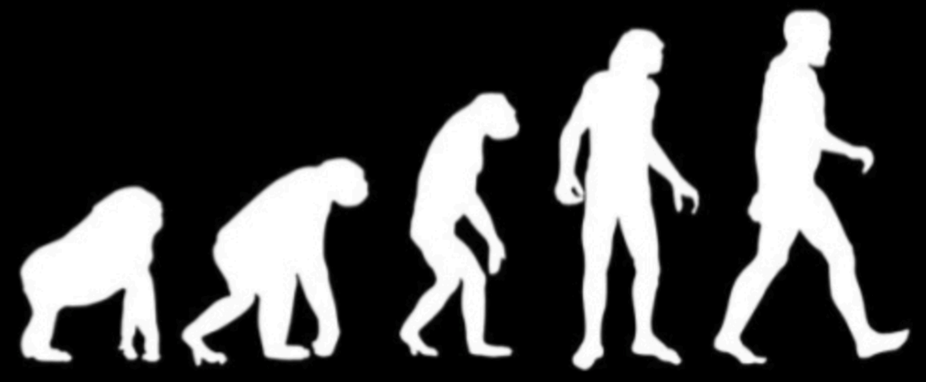
Composition



Mutation



Fitness



Geneva

Genetic Evasion

Building Blocks

Actions manipulate individual packets

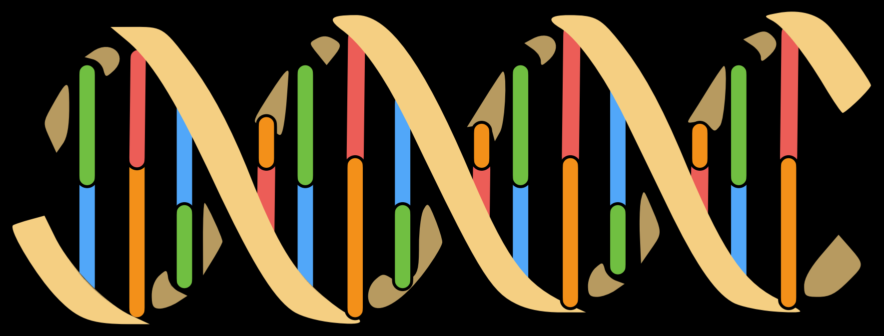
Duplicate

Tamper

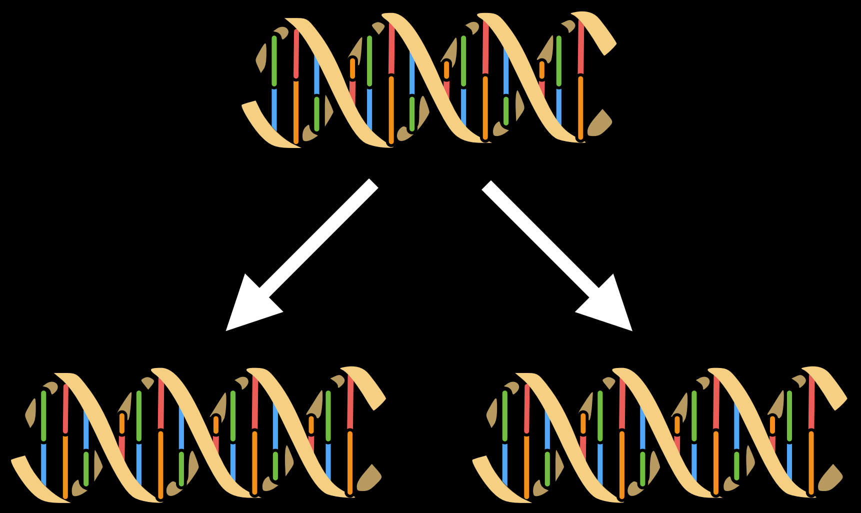
Fragment

Drop

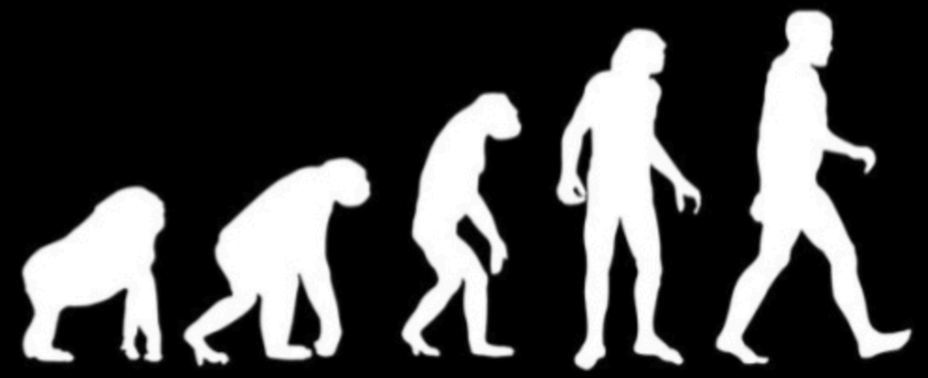
Composition



Mutation



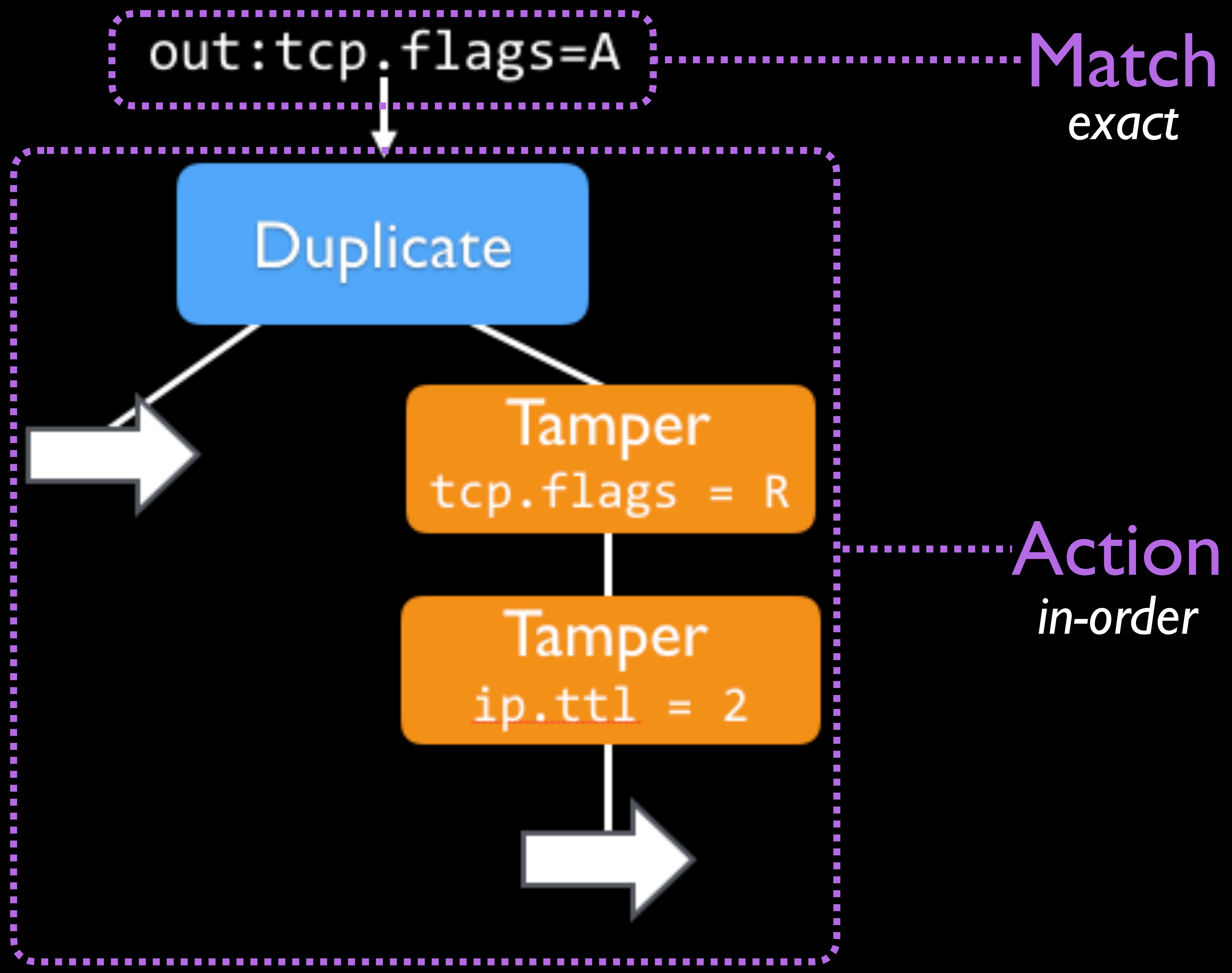
Fitness



Geneva

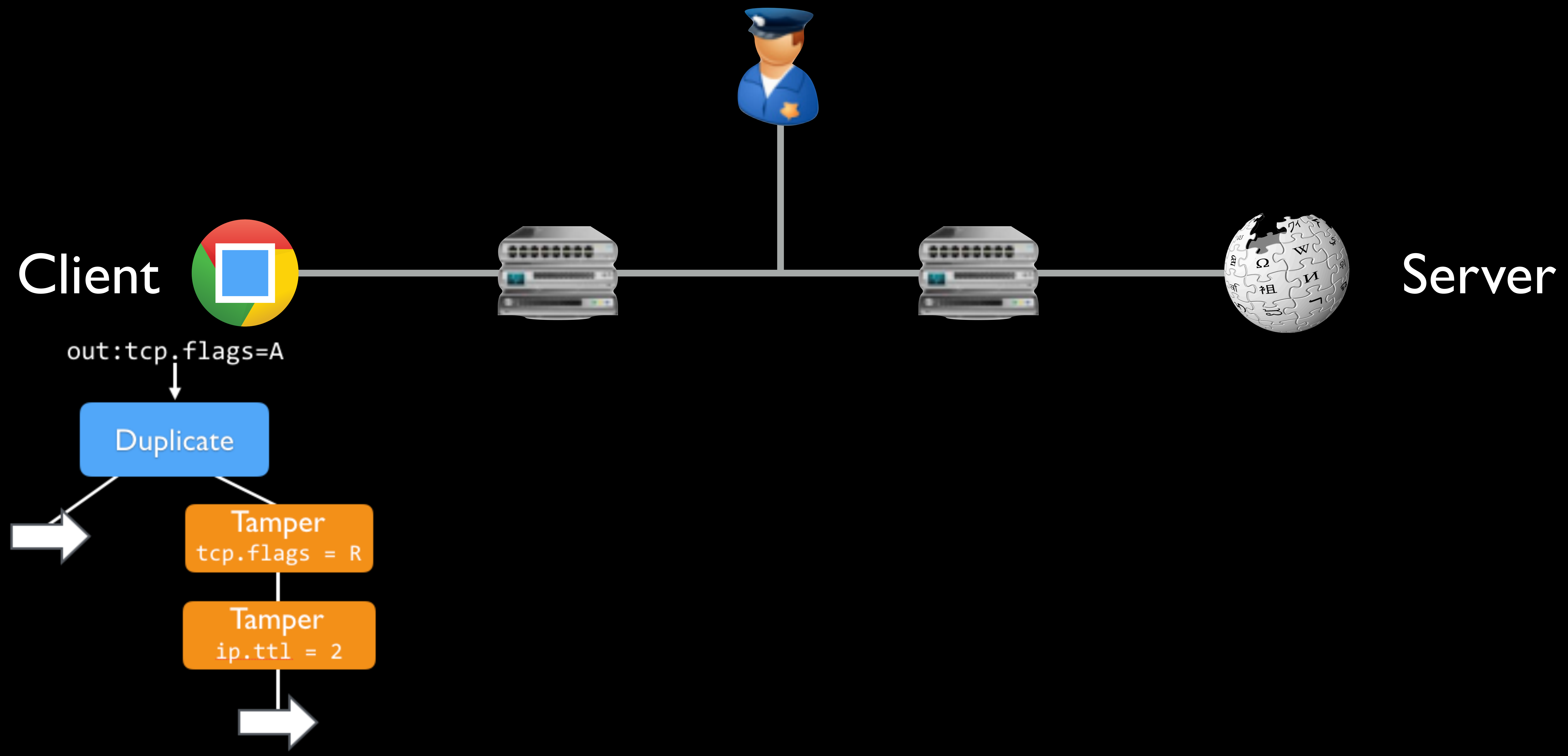
Genetic Evasion

Composition

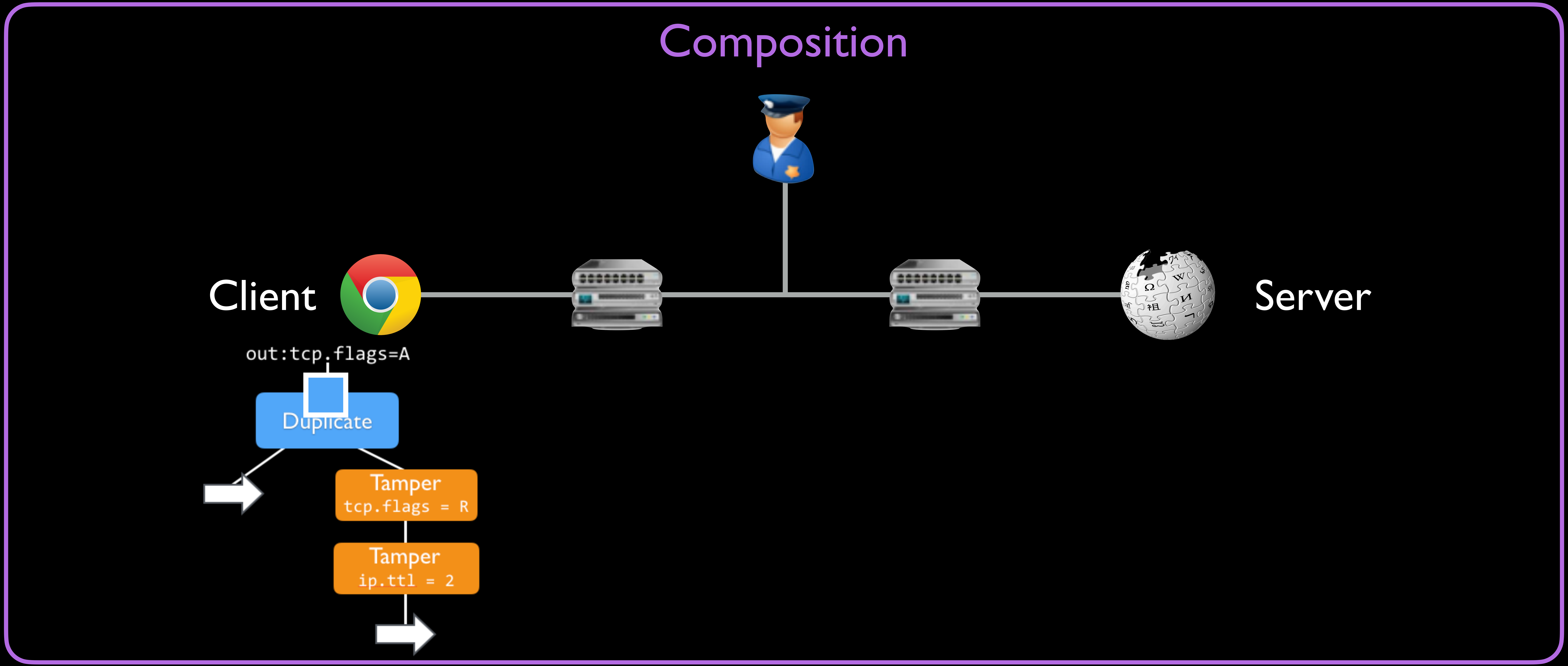


Running a Strategy

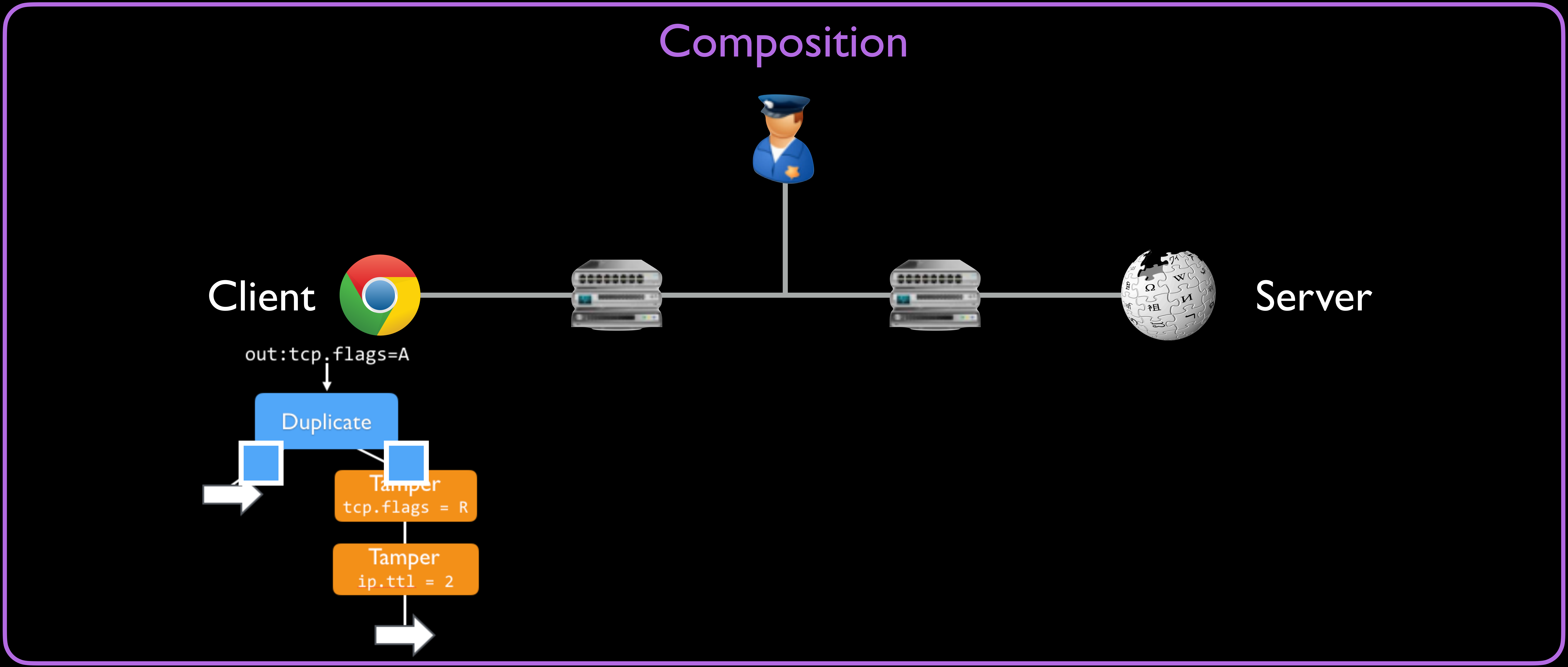
Composition



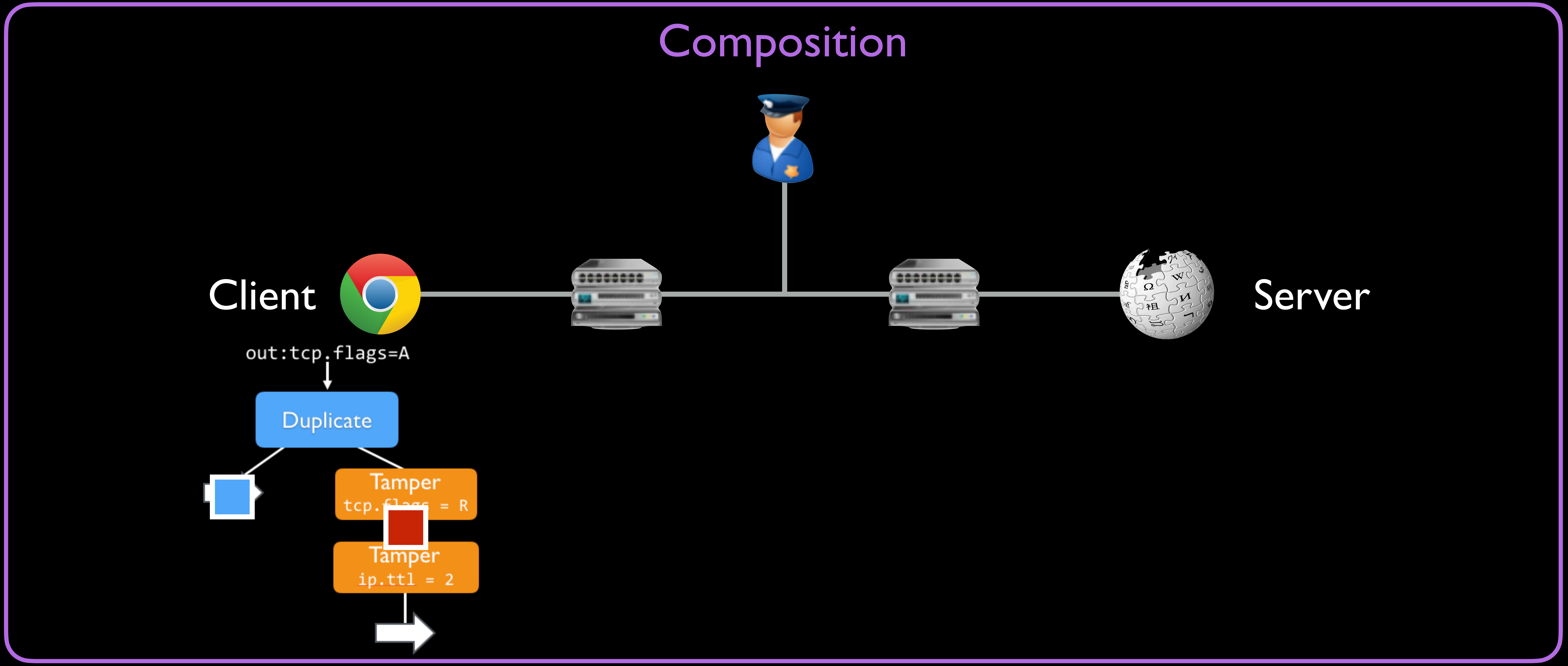
Running a Strategy



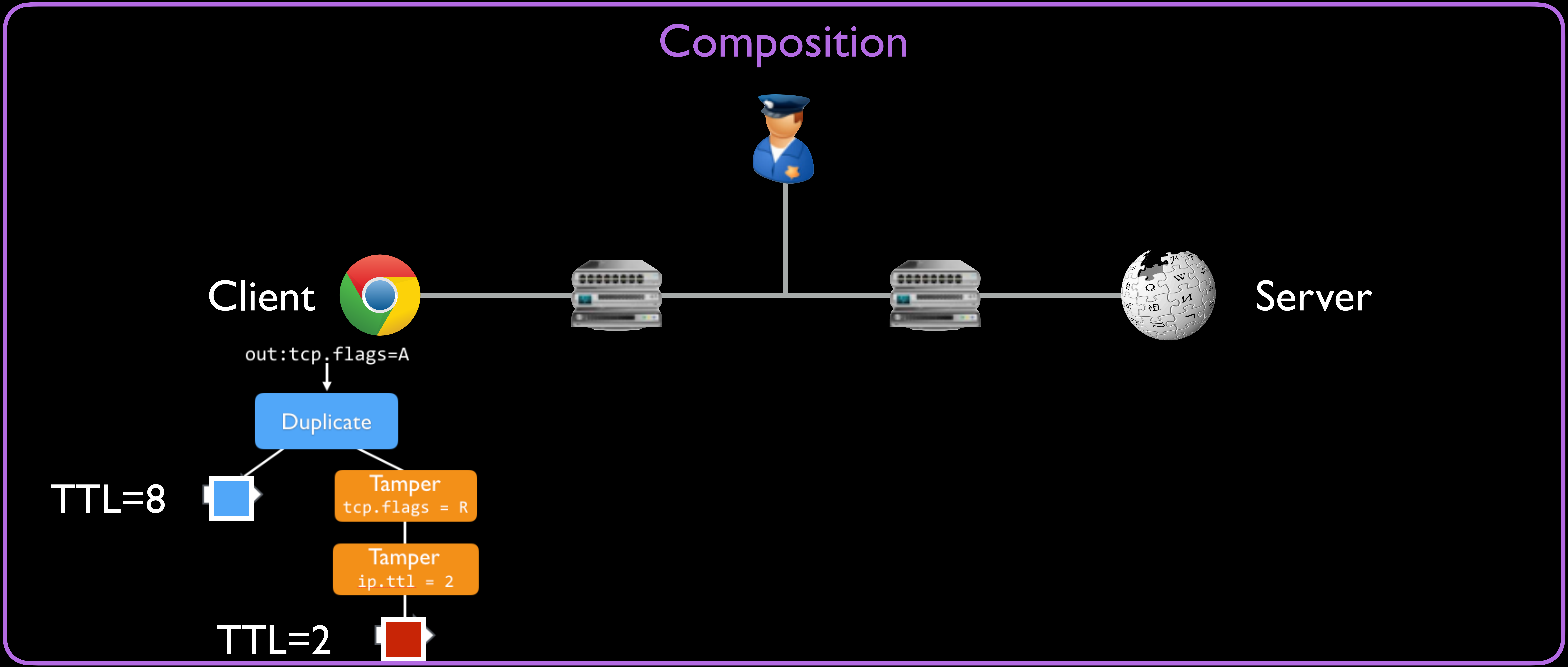
Running a Strategy



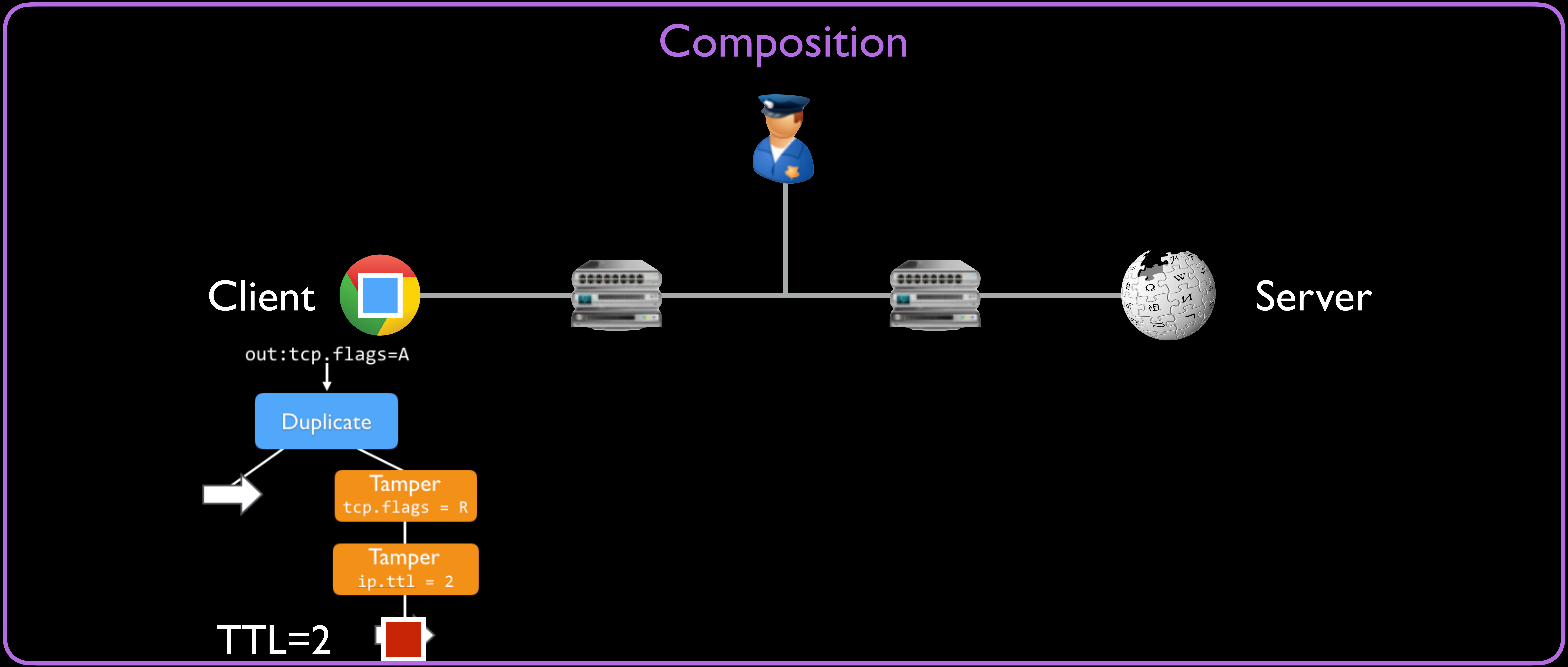
Running a Strategy



Running a Strategy

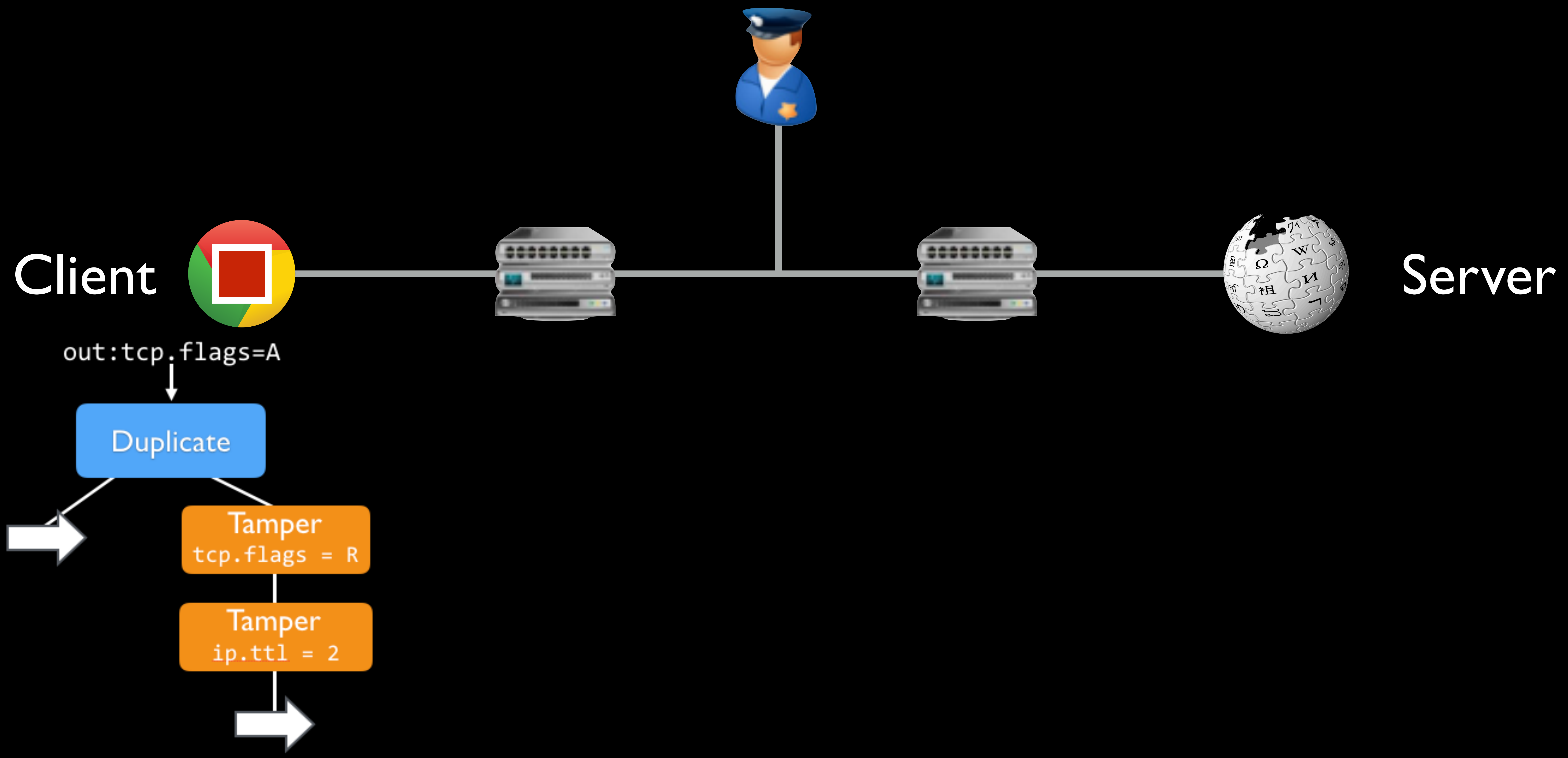


Running a Strategy



Running a Strategy

Composition



Geneva

Genetic Evasion

Building Blocks

Actions manipulate individual packets

Duplicate

Tamper

Fragment

Drop

Composition

Actions compose to form trees

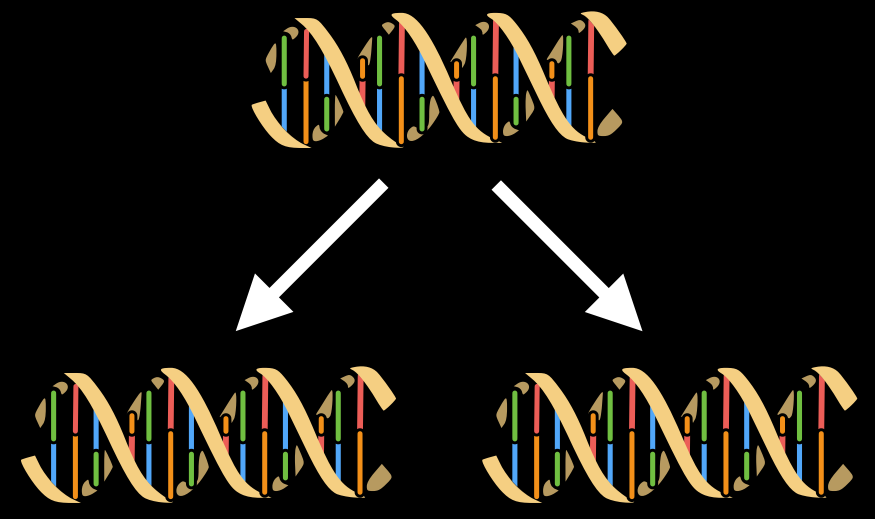
`out:tcp.flags=A`

Duplicate

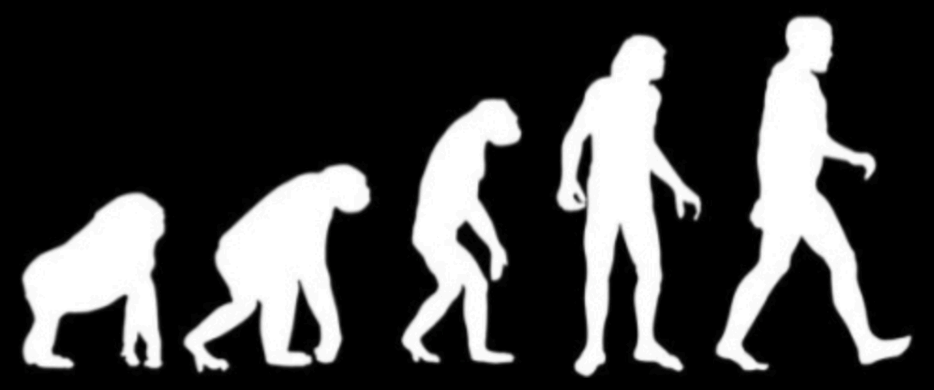
Tamper
`tcp.flags = R`

Tamper
`ip.ttl = 2`

Mutation



Fitness



Geneva

Genetic Evasion

Building Blocks

Actions manipulate individual packets

Duplicate

Tamper

Fragment

Drop

Composition

Actions compose to form trees

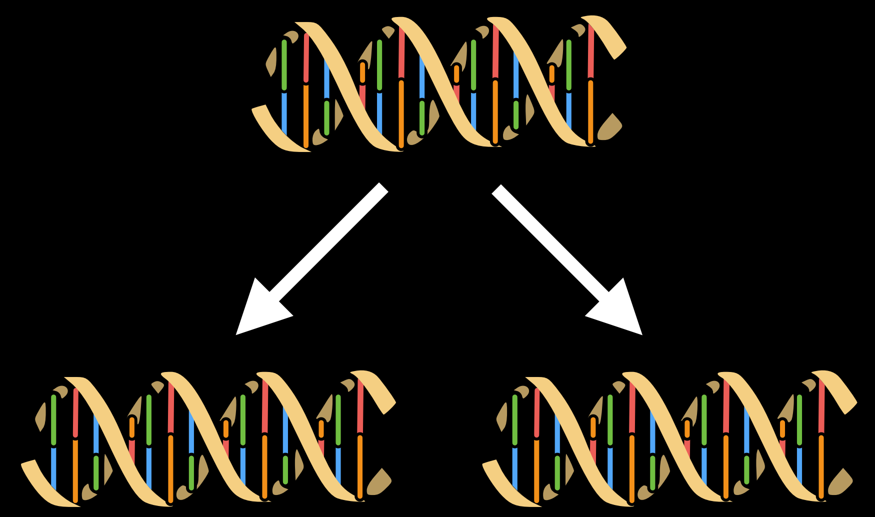
`out:tcp.flags=A`

Duplicate

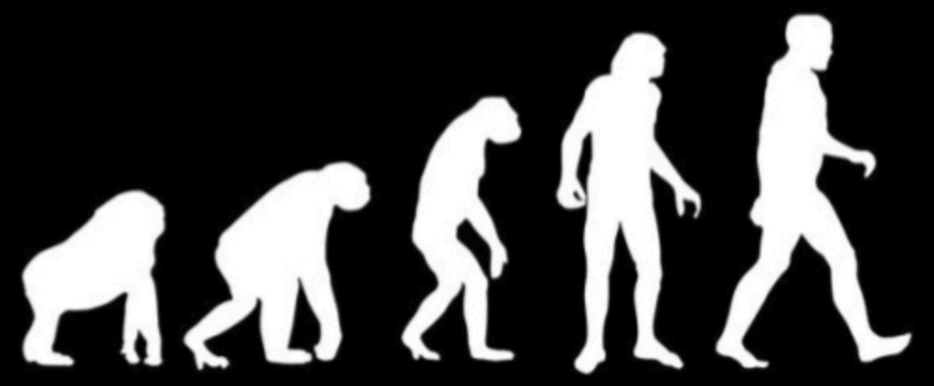
Tamper
`tcp.flags = R`

Tamper
`ip.ttl = 2`

Mutation



Fitness



Geneva

Genetic Evasion

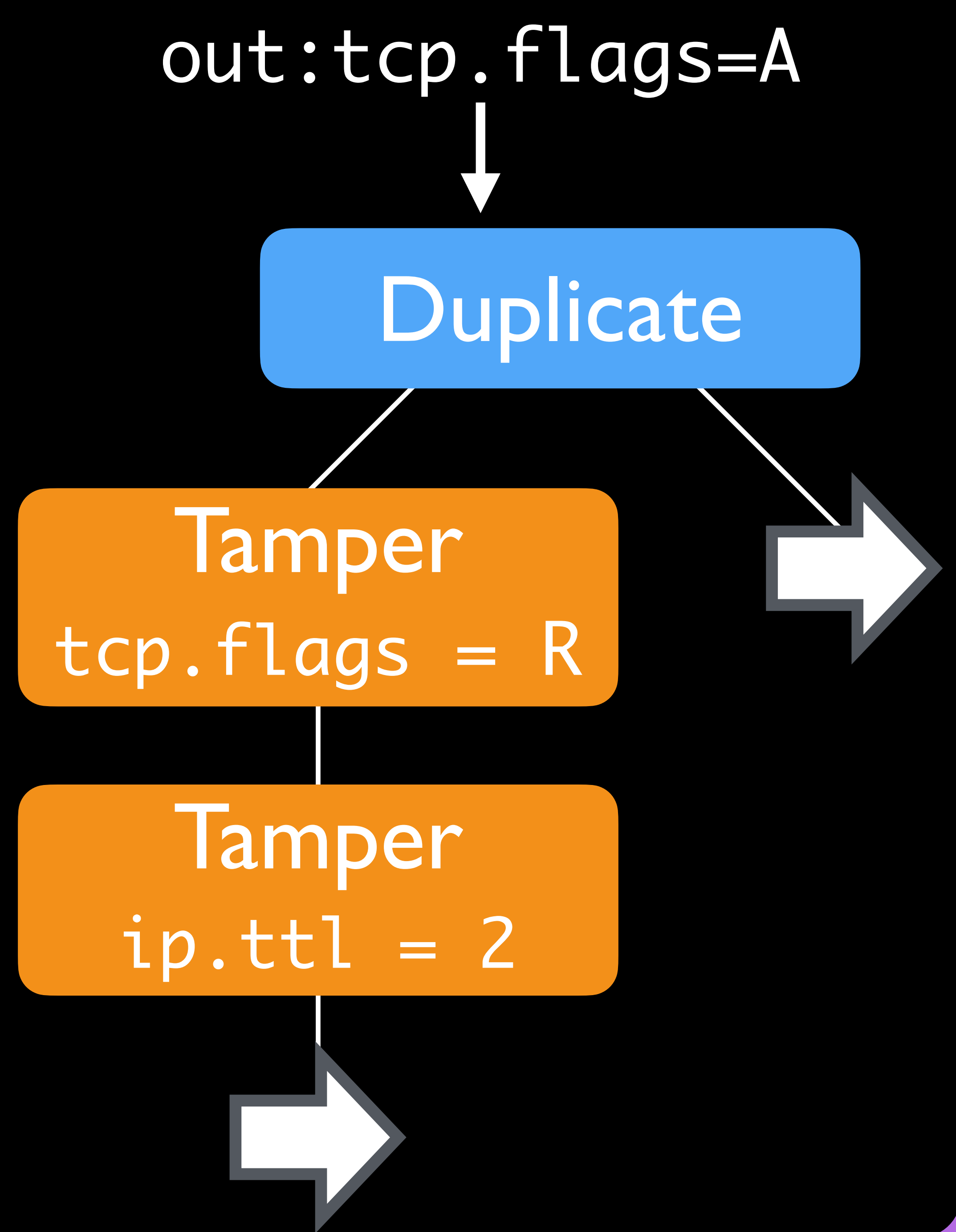
Building Blocks

Actions manipulate individual packets

- Duplicate
- Tamper
- Fragment
- Drop

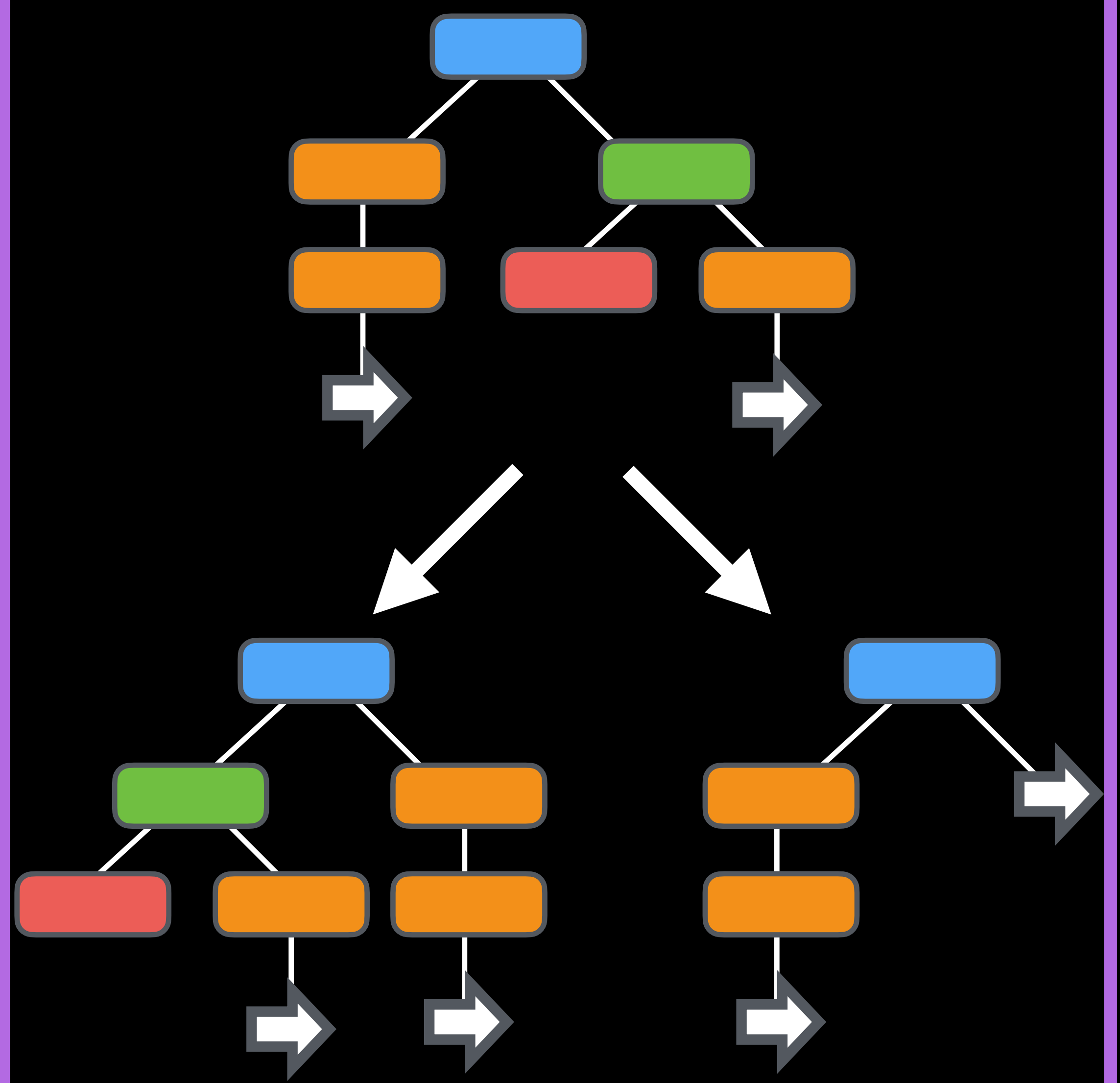
Composition

Actions compose to form trees

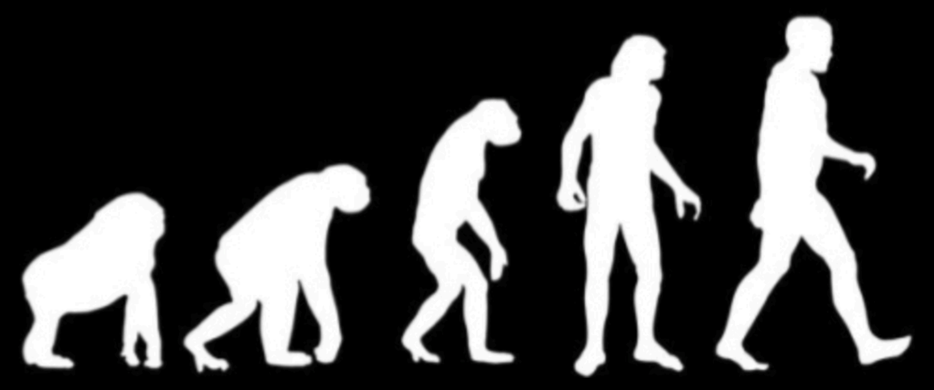


Mutation

Randomly alter types, values, and trees



Fitness



Geneva

Genetic Evasion

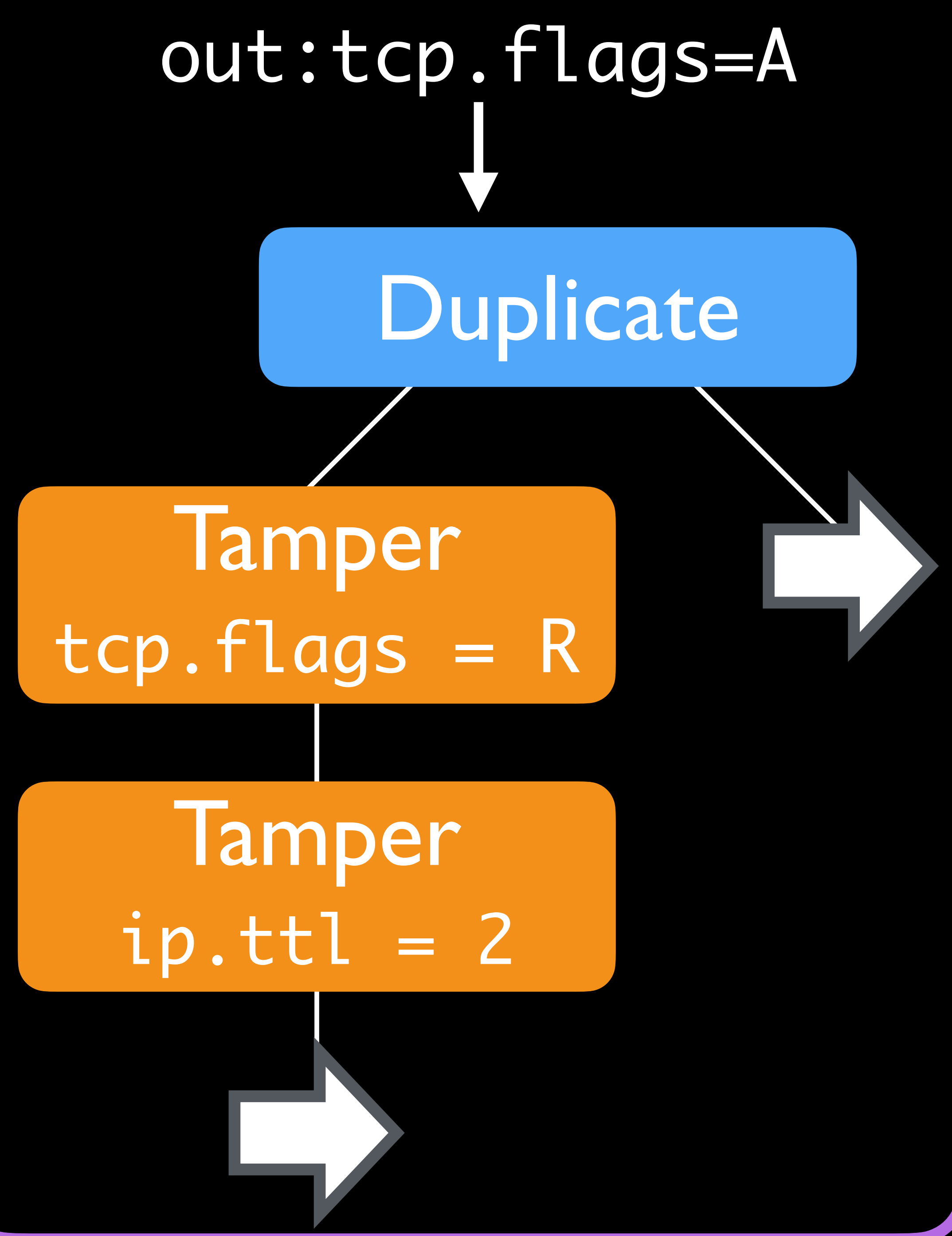
Building Blocks

Actions manipulate individual packets

- Duplicate
- Tamper
- Fragment
- Drop

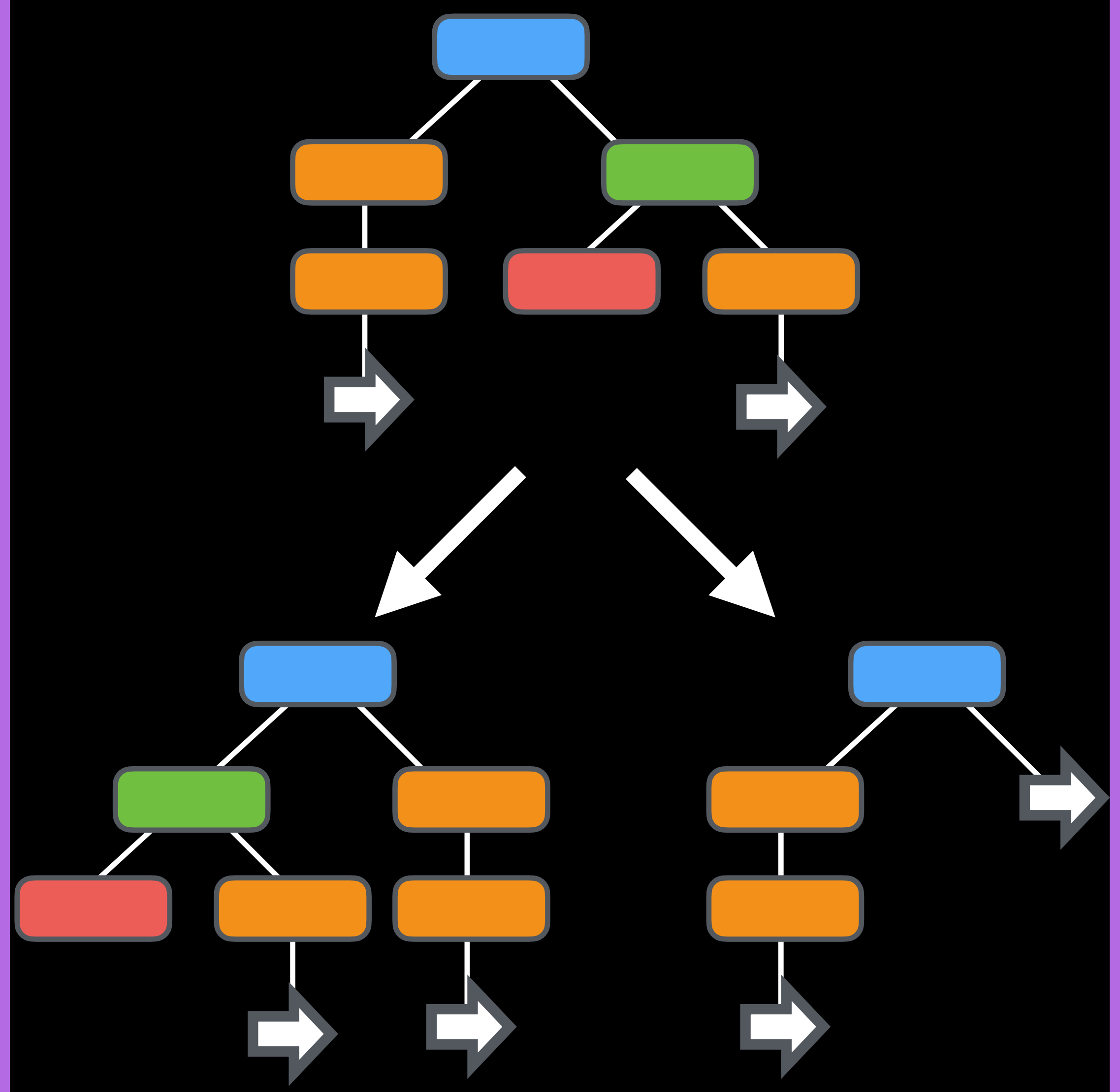
Composition

Actions compose to form trees

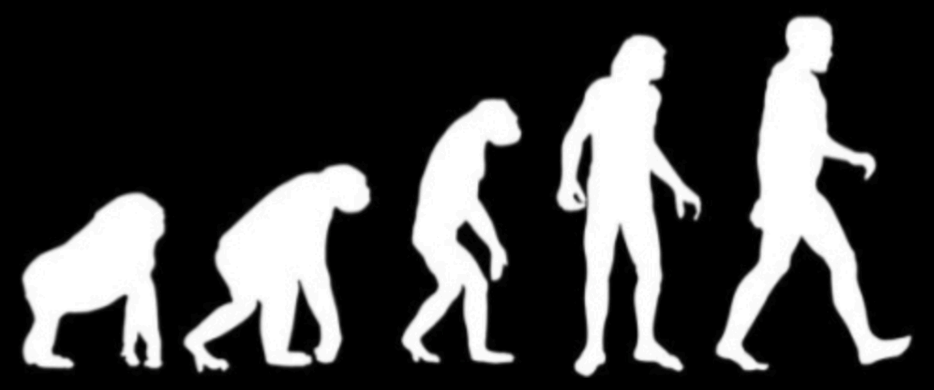


Mutation

Randomly alter types, values, and trees



Fitness

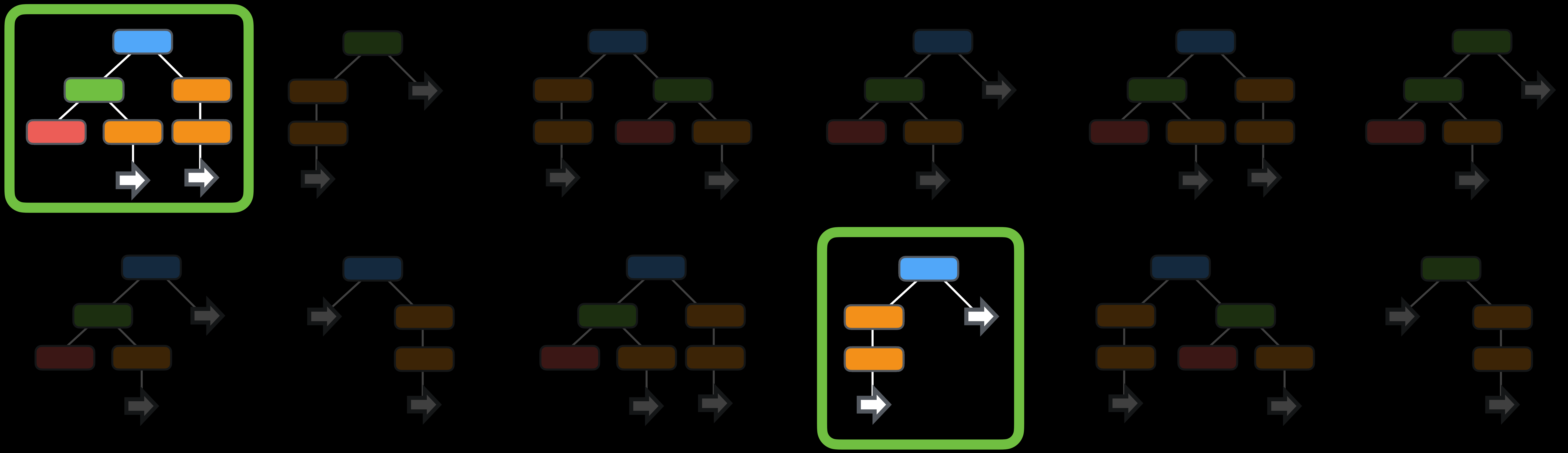


Geneva

Genetic Evasion

Fitness

Which individuals should survive to the next generation?

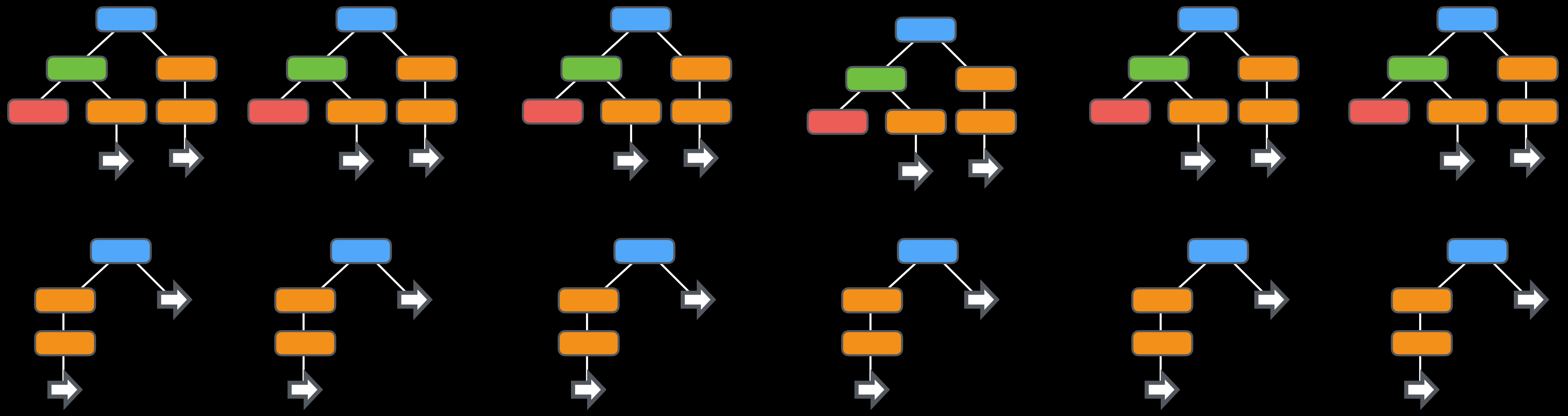


Geneva

Genetic Evasion

Fitness

Which individuals should survive to the next generation?



Geneva

Genetic Evasion

Fitness

Which individuals should survive to the next generation?

- Not triggering on any packets
- Breaking the TCP connection

+ Successfully obtaining forbidden content
+ Conciseness

Geneva

Genetic Evasion

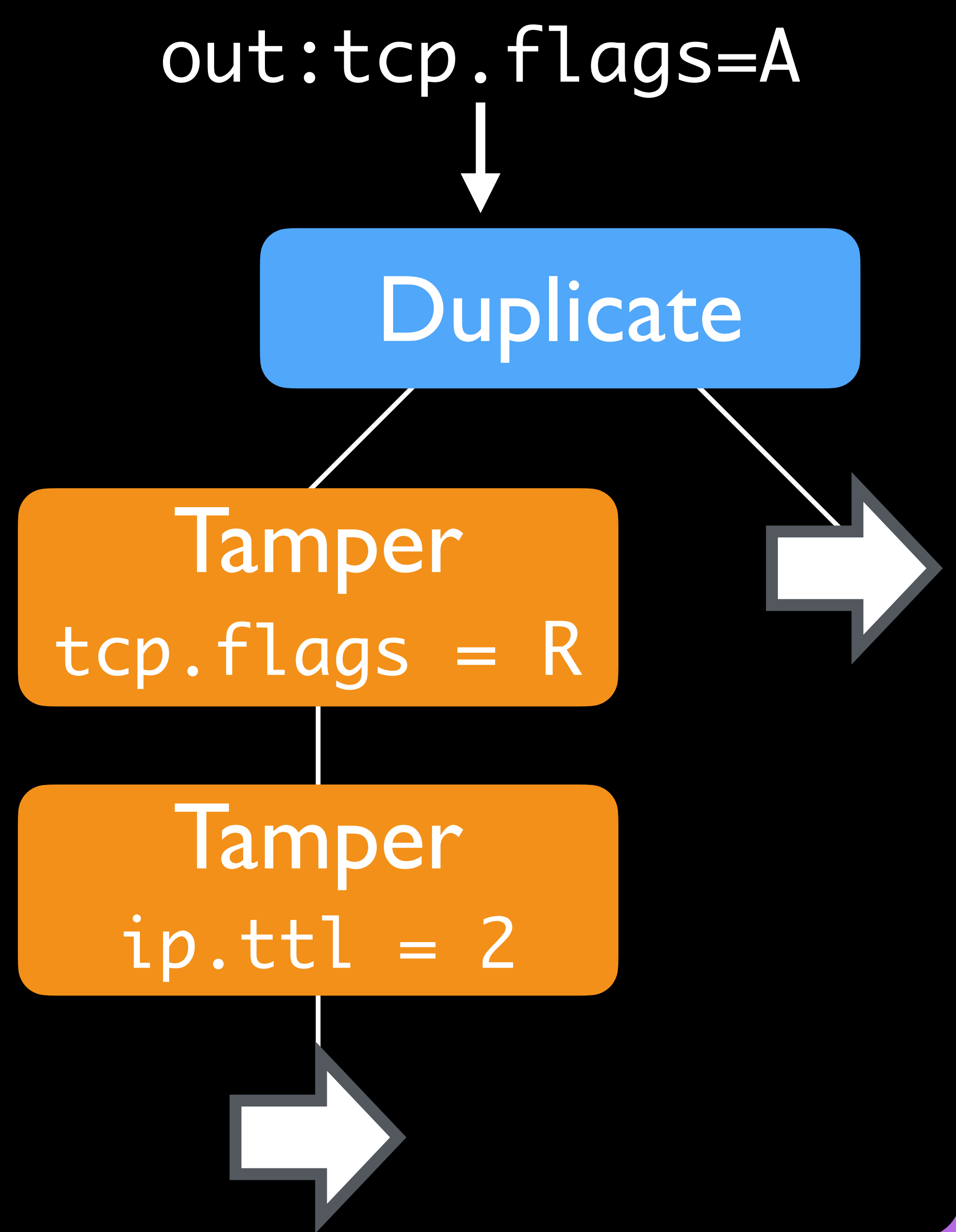
Building Blocks

Actions manipulate individual packets

- Duplicate
- Tamper
- Fragment
- Drop

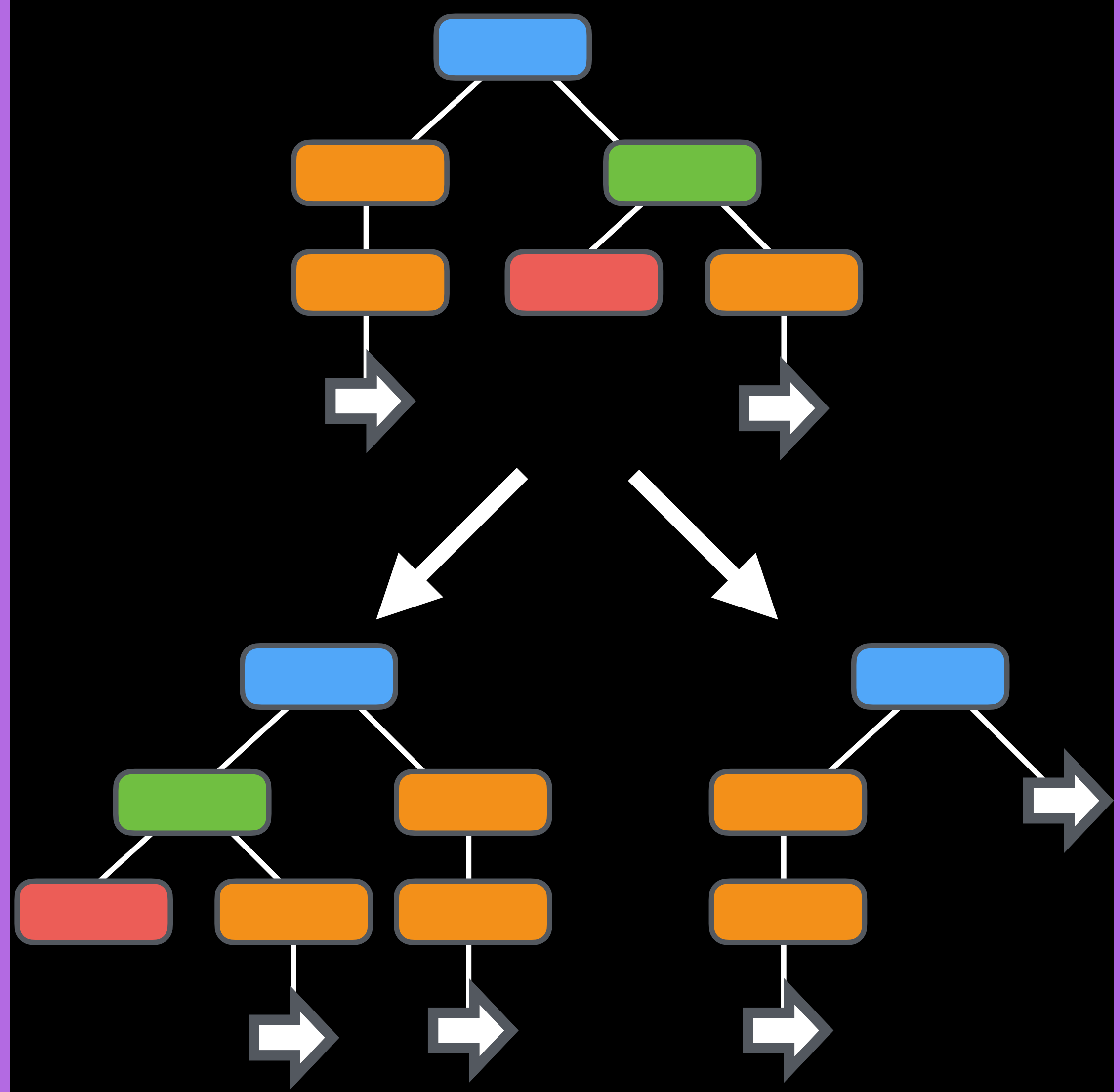
Composition

Actions compose to form trees



Mutation

Randomly alter types, values, and trees



Fitness

Goal: Fewest actions needed to succeed

- No trigger
- Break TCP
- Successful
- Concise

Client-side results

In-lab experiments
Against mock censors

Found virtually all of the previously known strategy species

Failed to find the strategies we did not give building blocks for

		Found?			
Species	**Strategy**	[21]	[33]	[41]	Geneva
TCB Creation	w/ low TTL	✓		✓	✓
	w/ corrupt checksum			✓	✓
	(Improved) and Resync/Desync			✓	✓
TCB Teardown	w/ RST and low TTL	✓	✓	✓	✓
	w/ RST and corrupt checksum		✓	✓	✓
	w/ RST and invalid timestamp			✓	✓
	w/ RST and invalid MD5 Header			✓	✓
	w/ RST/ACK and corrupt checksum			✓	✓
	w/ RST/ACK and low TTL	✓	✓	✓	✓
	w/ RST/ACK and invalid timestamp			✓	✓
	w/ RST/ACK and invalid MD5 Header			✓	✓
	w/ FIN and low TTL	✓		✓	✓
	w/ FIN and corrupt checksum			✓	✓
	(Improved)			✓	✓
	and TCB Reversal			✓	✓
Reassembly	TCP Segmentation reassembly out of order data		✓	✓	✓
	Overlapping fragments	✓		✓	✓
	Overlapping segments	✓		✓	✓
	In-order data w/ low TTL			✓	✓
	In-order data w/ corrupt ACK	✓		✓	✓
	In-order data w/ corrupt checksum			✓	✓
	In-order data w/ no TCP flags			✓	✓
	Out-of-order data w/ IP fragments			✓	✓
	Out-of-order data w/ TCP segments			✓	✓
	(Improved) In-order data overlapping			✓	✓
	Payload splitting		✓		✓
	Payload reordering		✓		✓
Traffic Misclassification	Inert Packet Insertion w/ corrupt checksum		✓		✓
	Inert Packet Insertion w/o ACK flag		✓		✓
State Exhaustion	Send > 1KB of traffic	✓			
	Classification Flushing – Delay	✓	✓		
HTTP Incompleteness	GET w/ > 1 space between method and URI	✓			
	GET w/ keyword at location > 2048	✓			
	GET w/ keyword in 2nd or higher of multiple requests in one segment	✓			
	GET w/ URL encoded (except %-encoding)	✓			

Client-side results – Real censor experiments

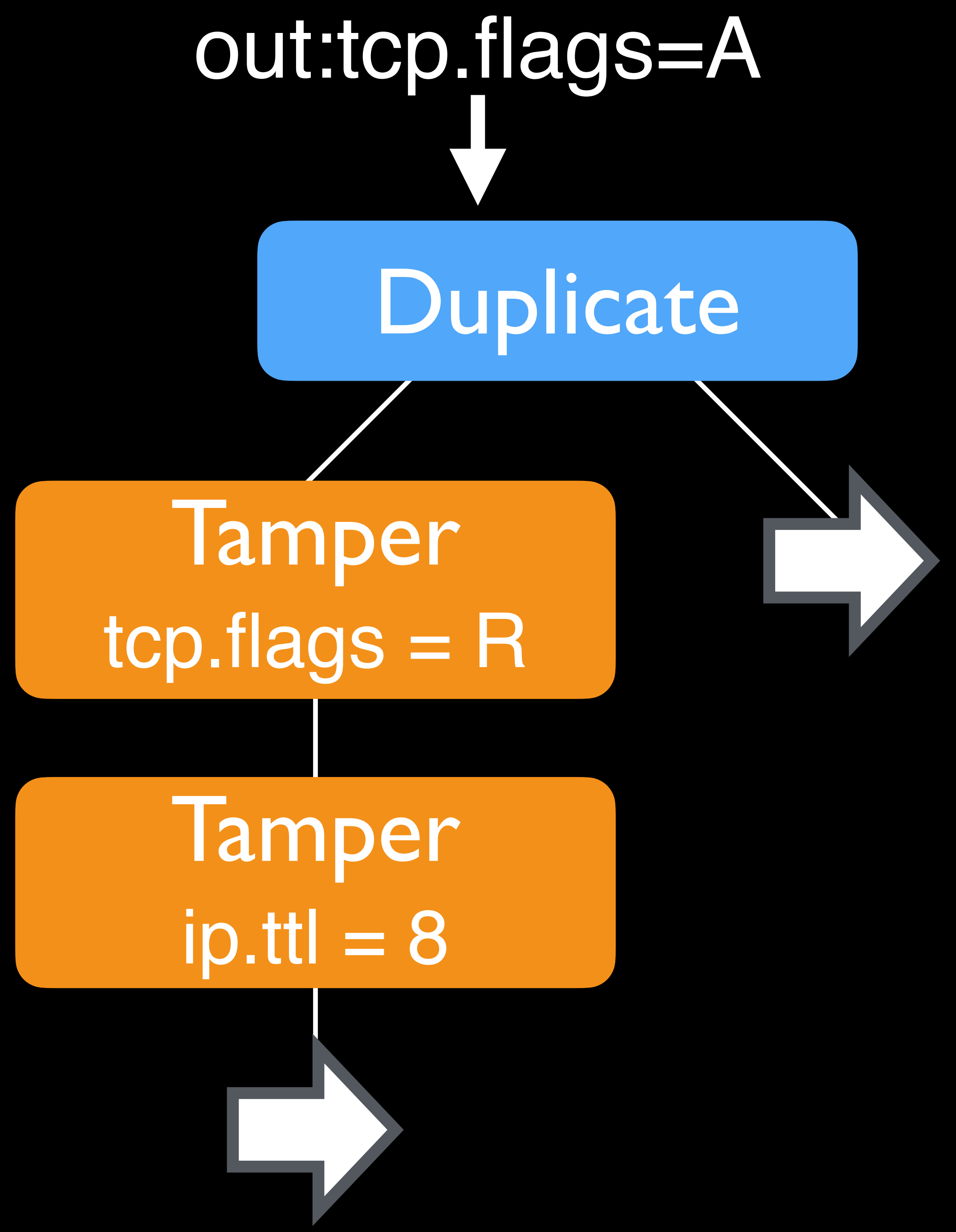
20+ Species — The underlying bug

30+ Sub-species — How Geneva exploits it

80+ Variants — Functionally distinct

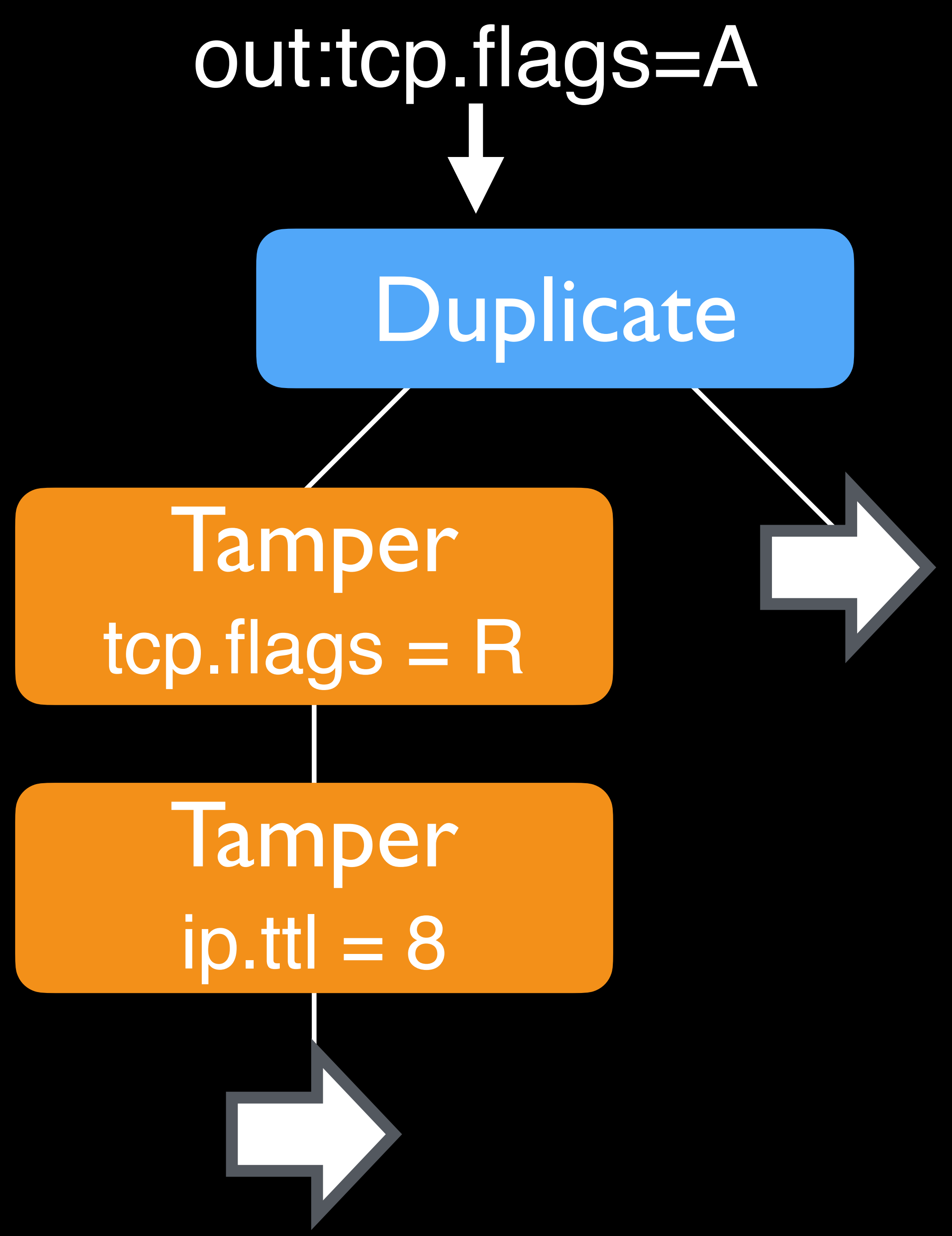
45+	15+	13+	24+
			
China	India	Iran	Kazakhstan

Teardown species



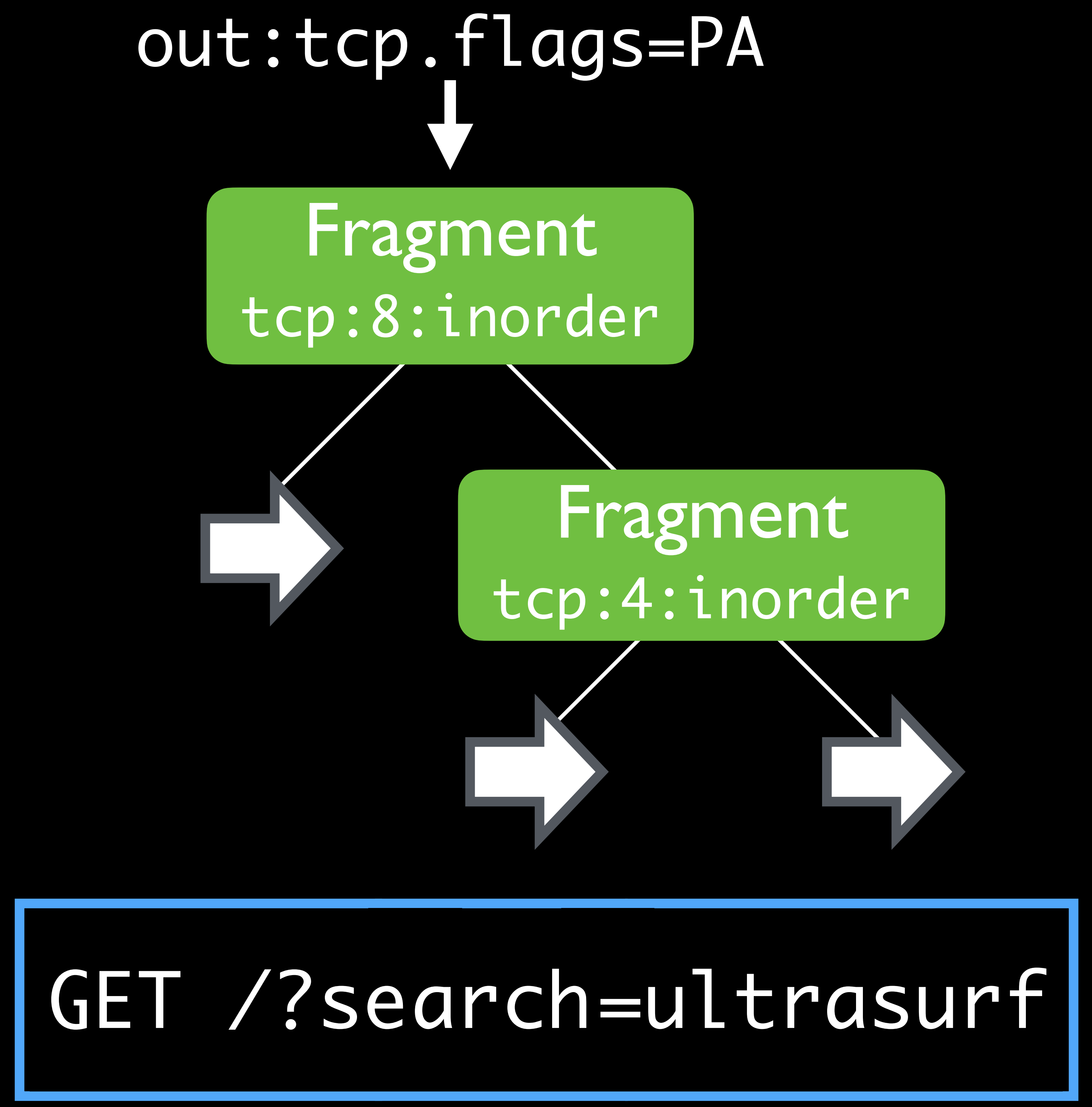
During the TCP handshake, insert a TTL-limited RST

Teardown species



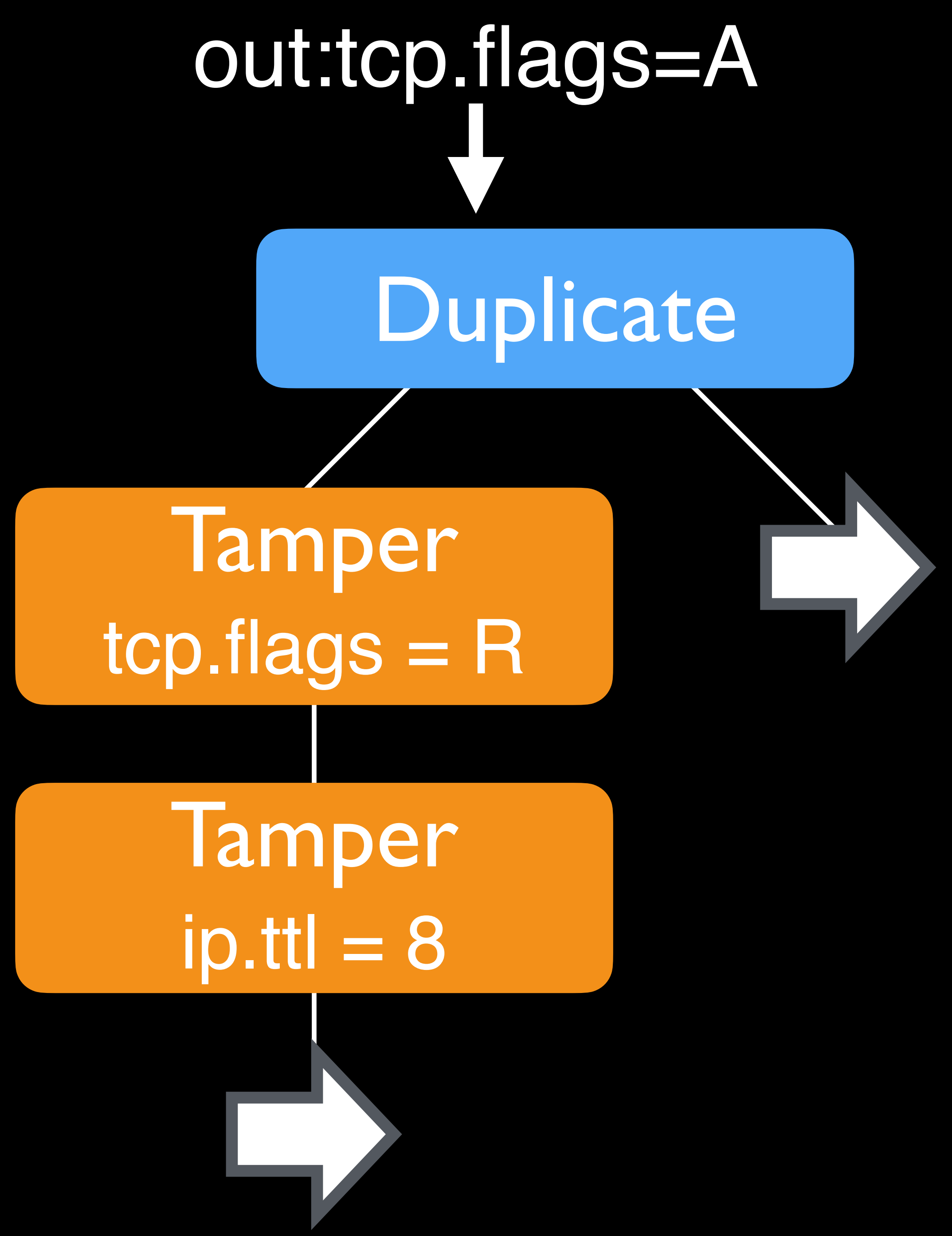
During the TCP handshake, insert a TTL-limited RST

Segmentation species



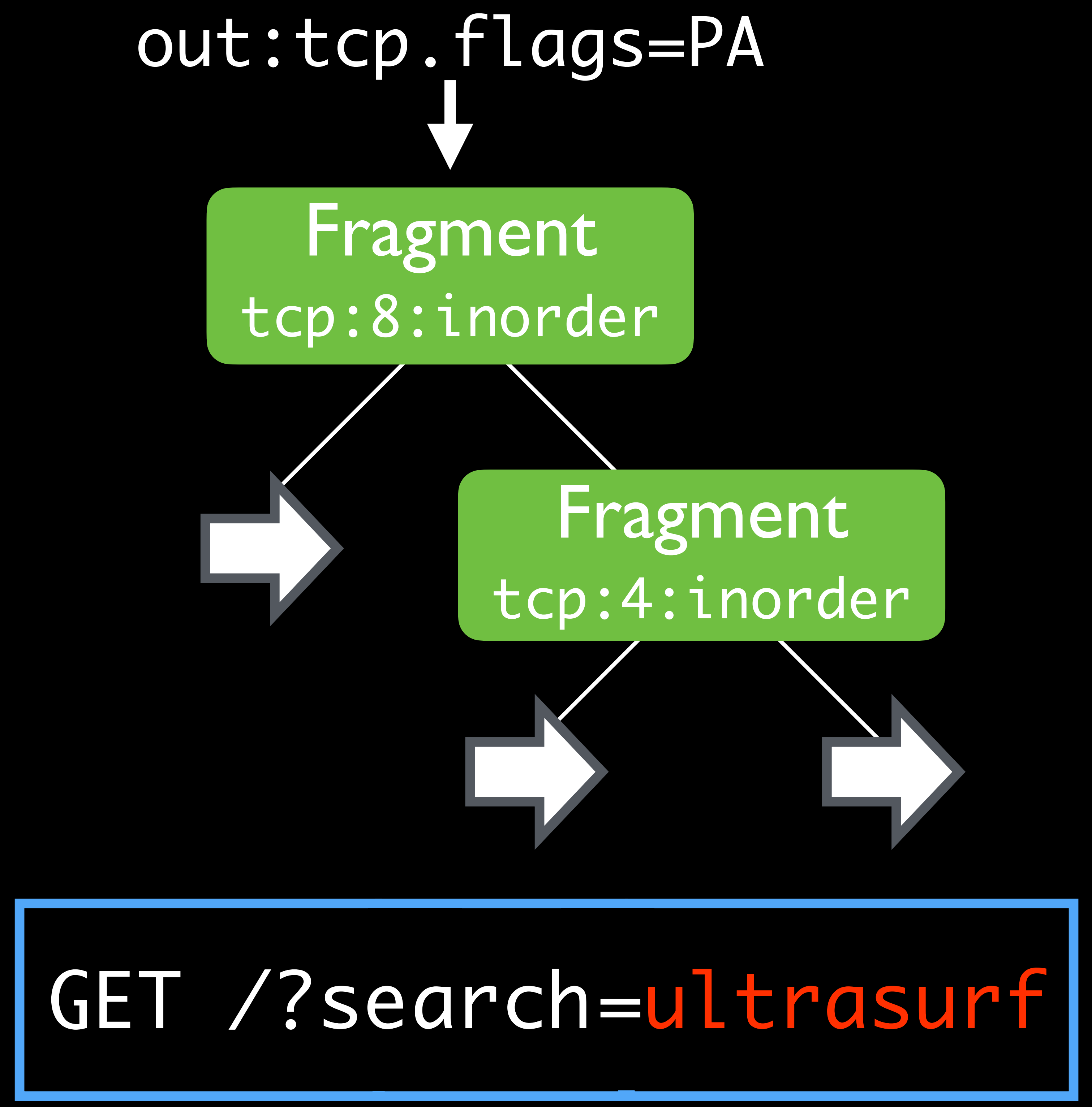
Segment the request

Teardown species



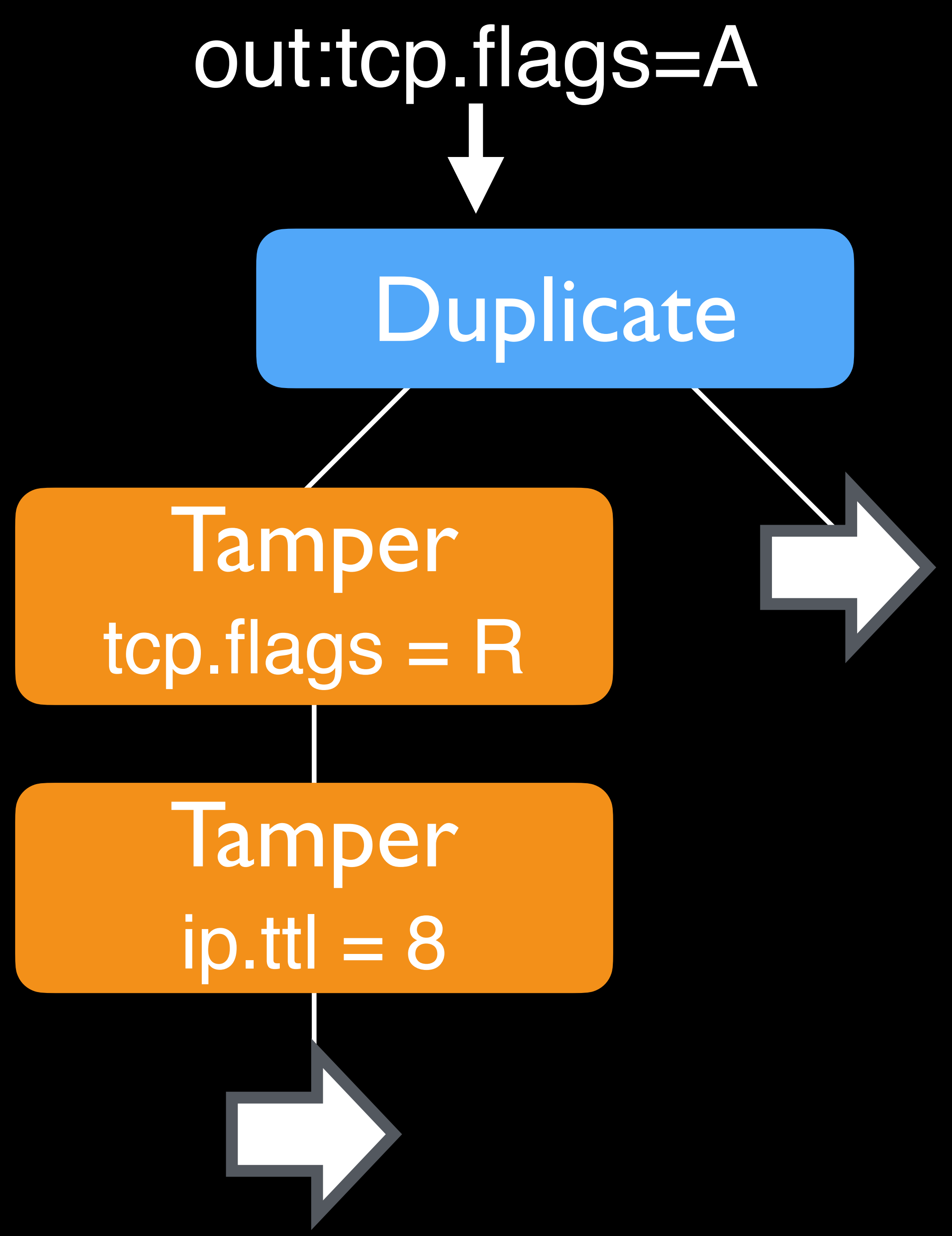
During the TCP handshake, insert a TTL-limited RST

Segmentation species



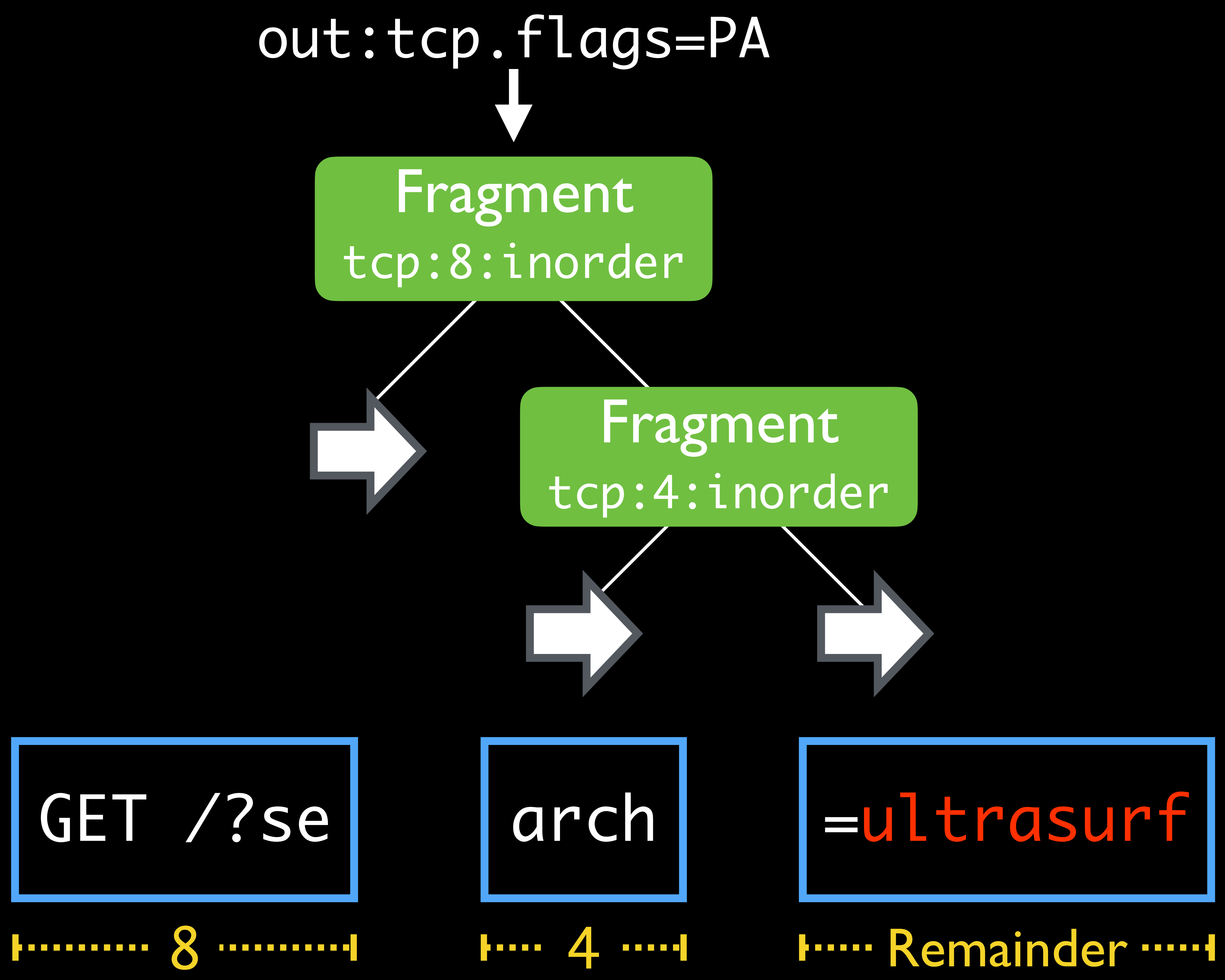
Segment the request

Teardown species



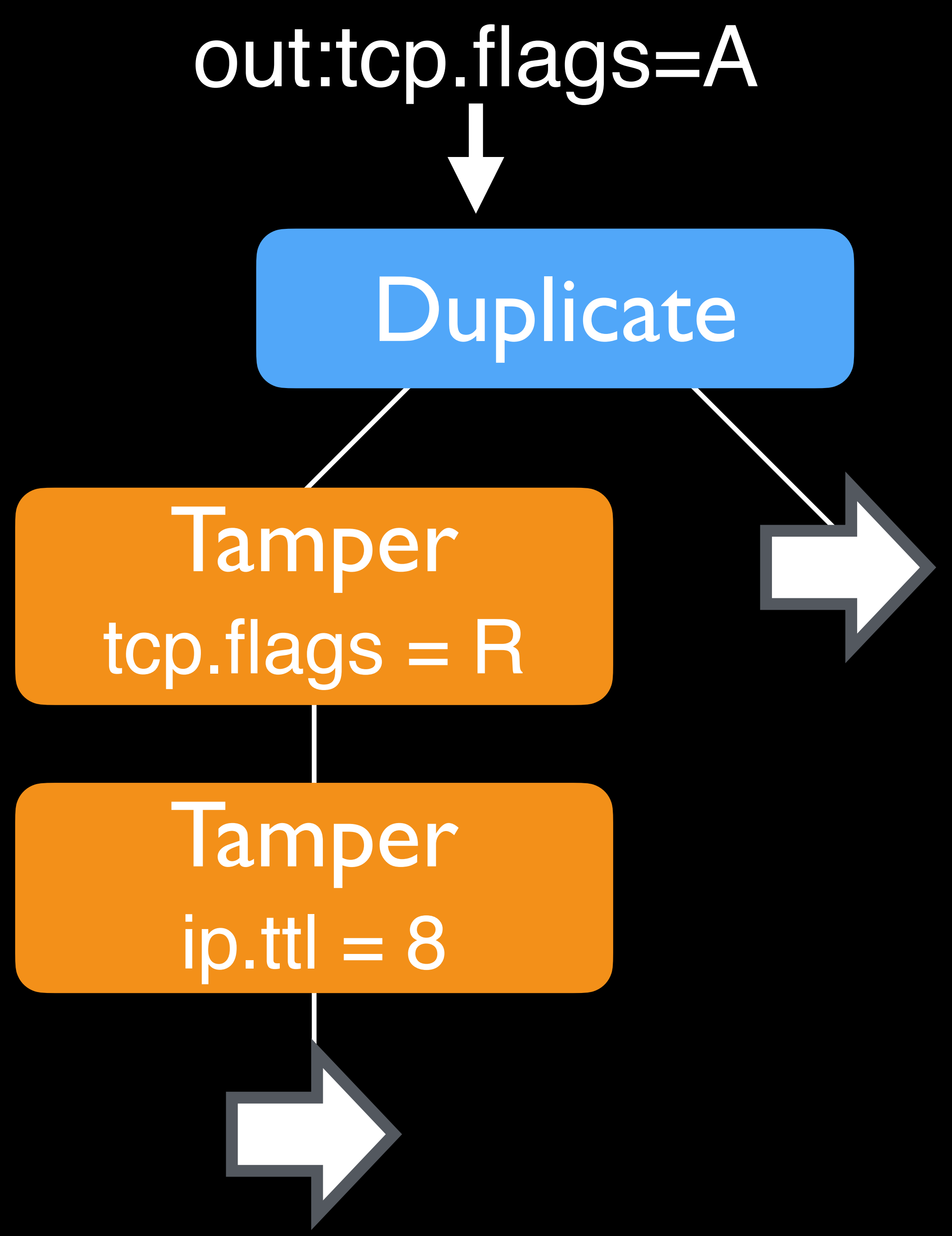
During the TCP handshake, insert a TTL-limited RST

Segmentation species



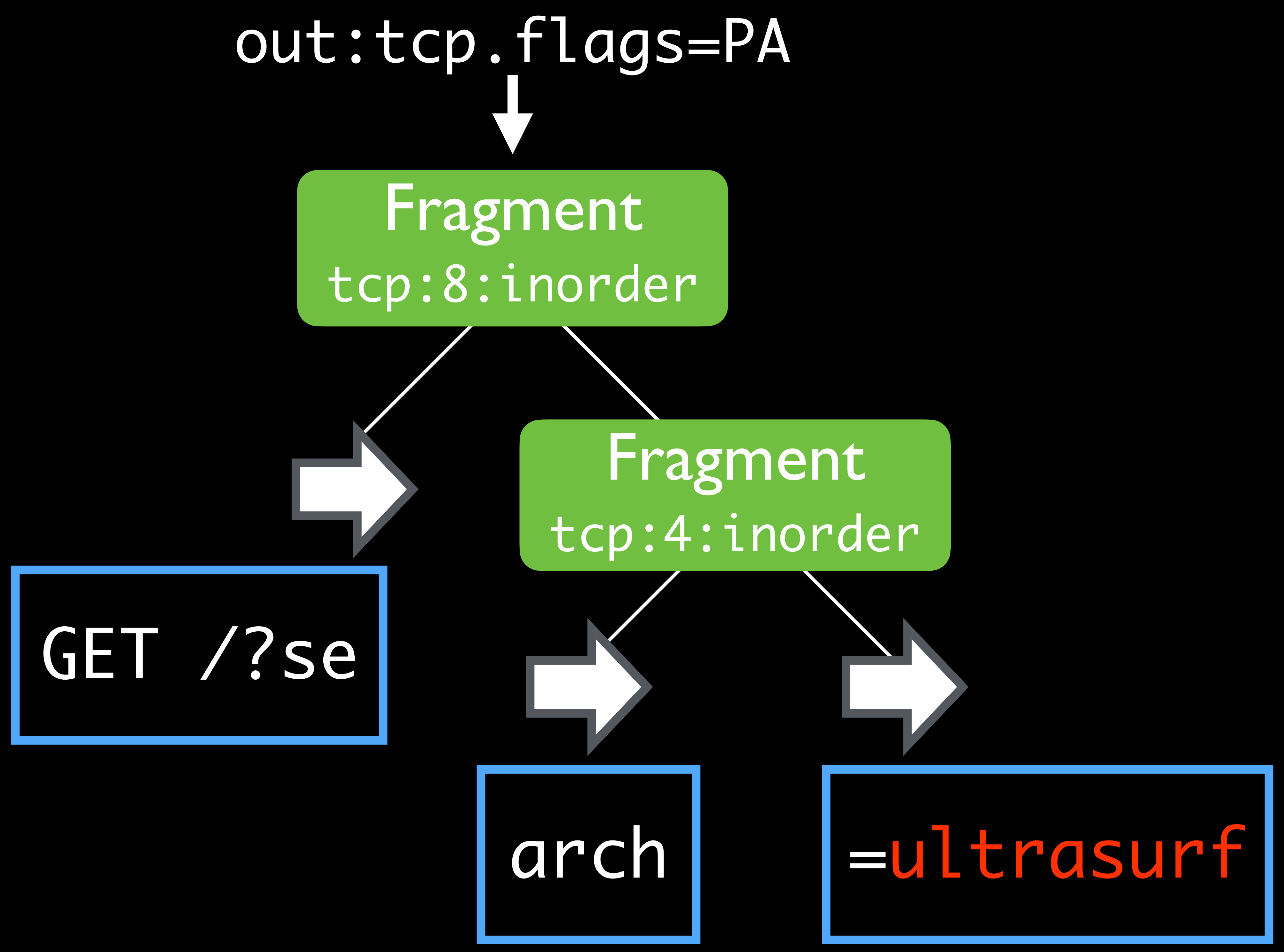
Segment the request, but *not the keyword*

Teardown species



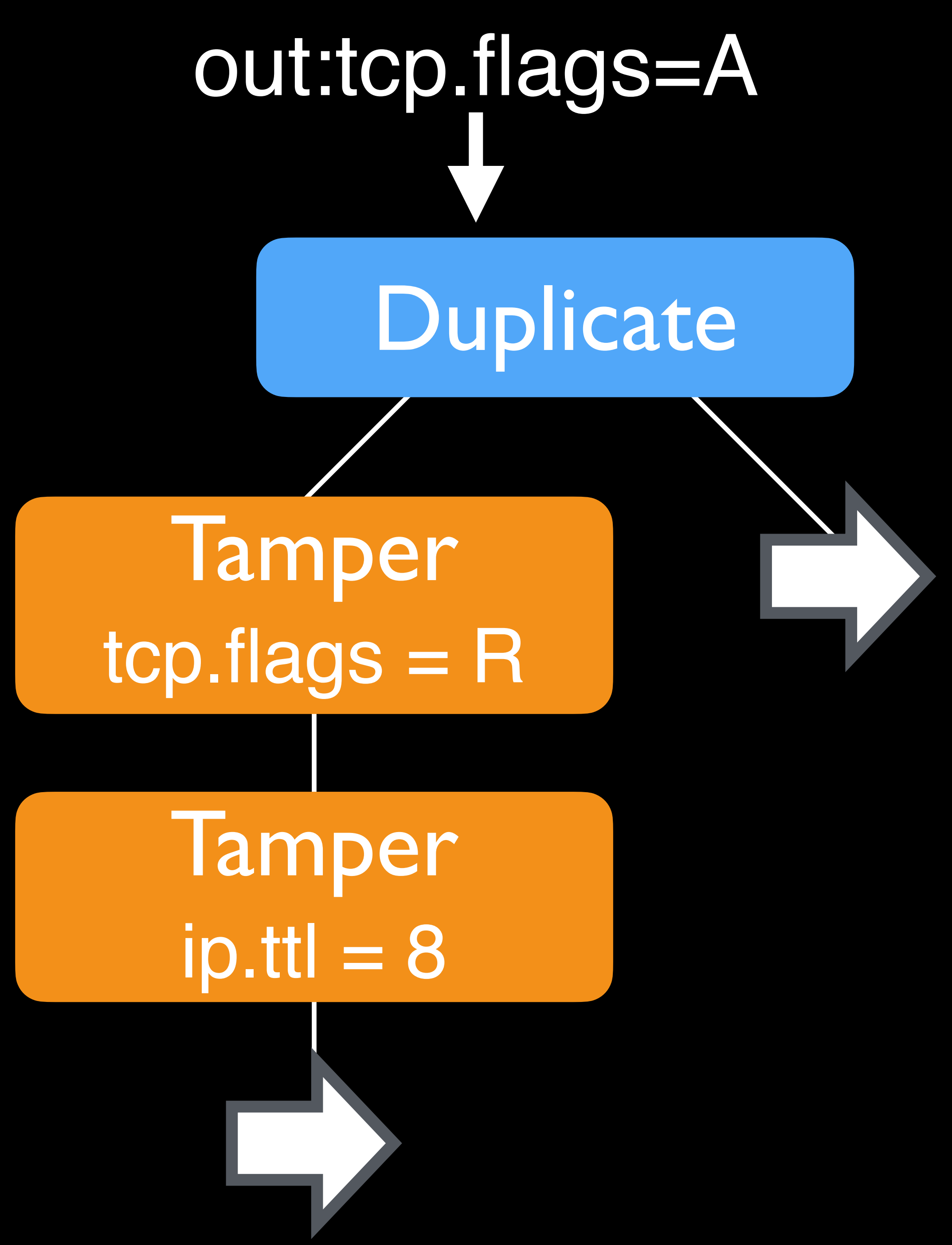
During the TCP handshake, insert a TTL-limited RST

Segmentation species



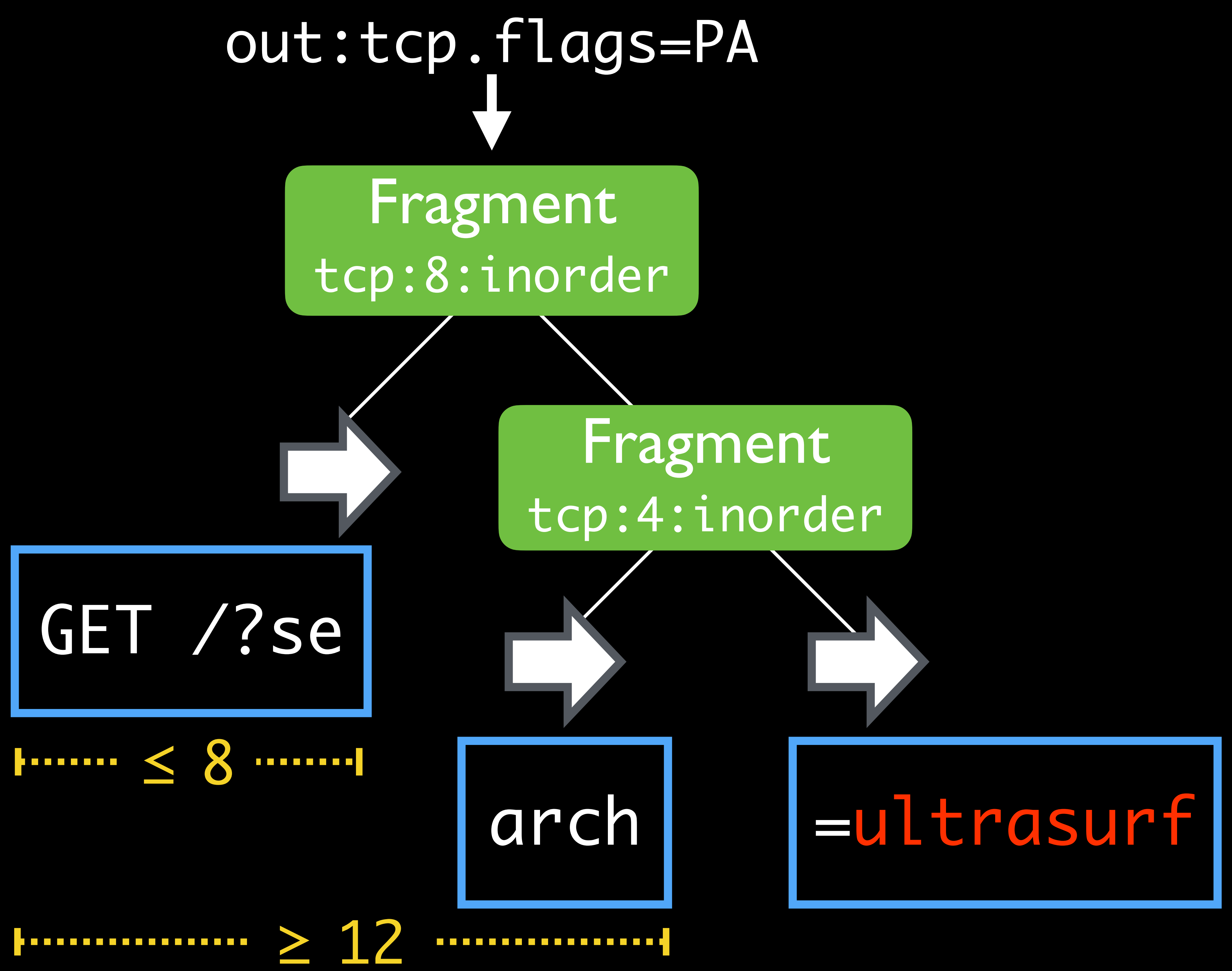
Segment the request, but *not the keyword*

Teardown species



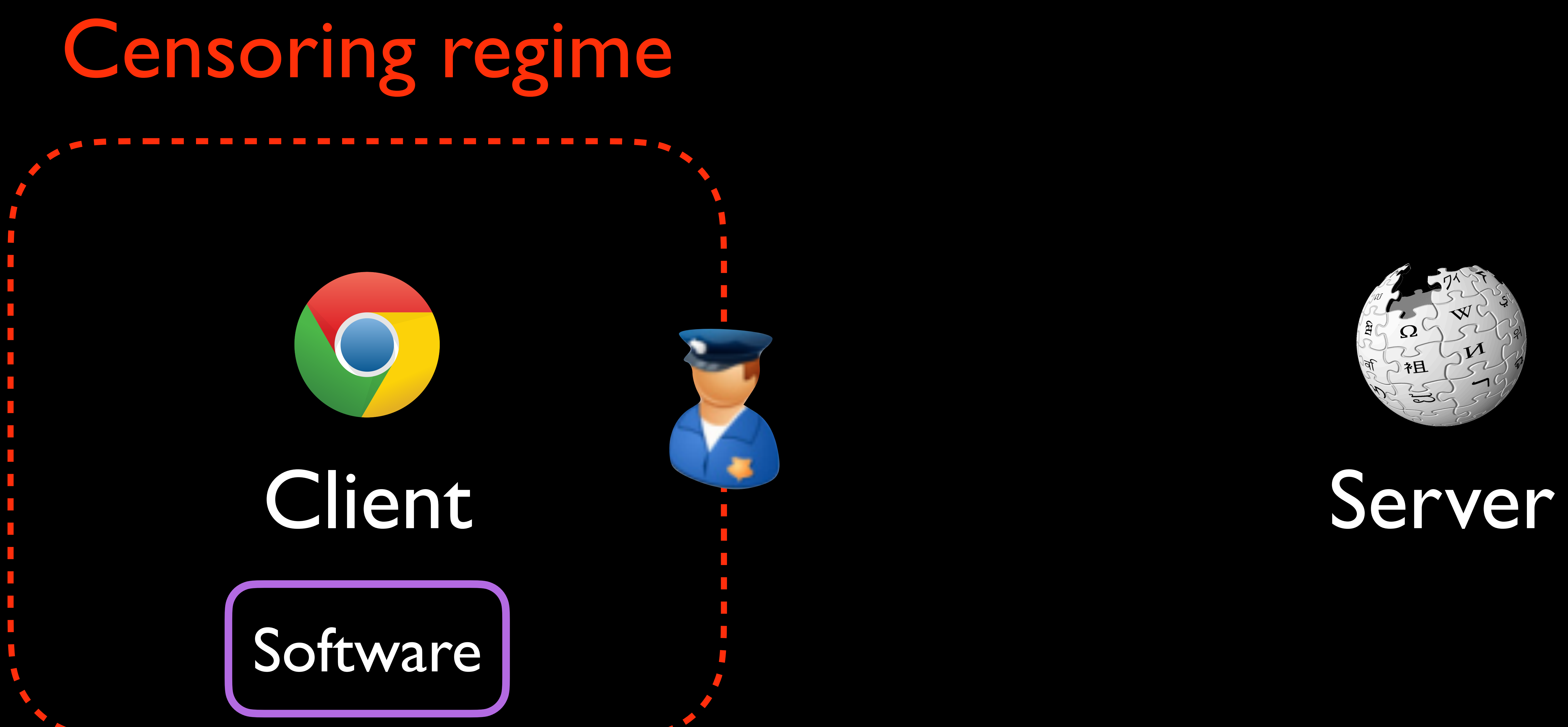
During the TCP handshake, insert a TTL-limited RST

Segmentation species



Segment the request, but *not the keyword*

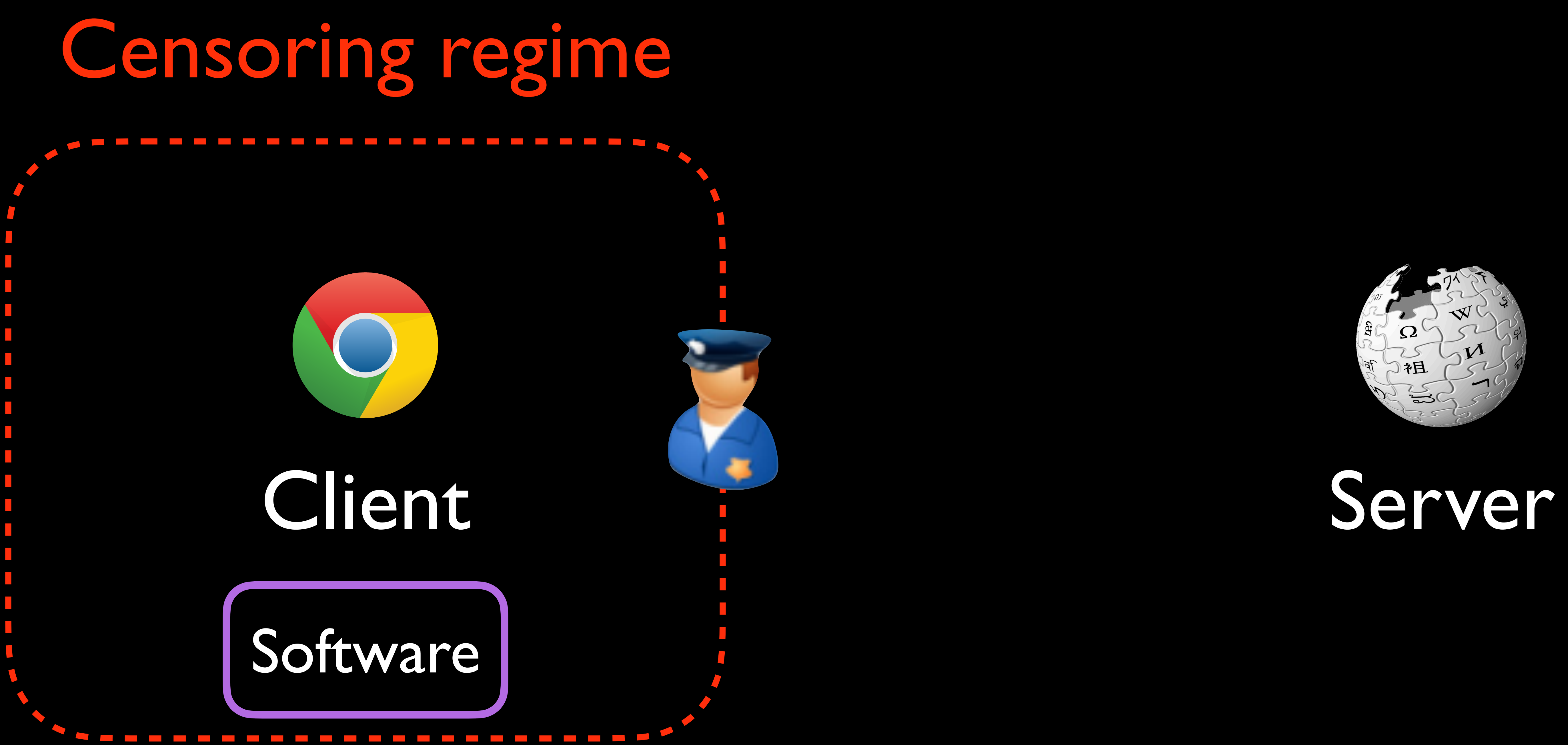
Censorship evasion has always involved the client



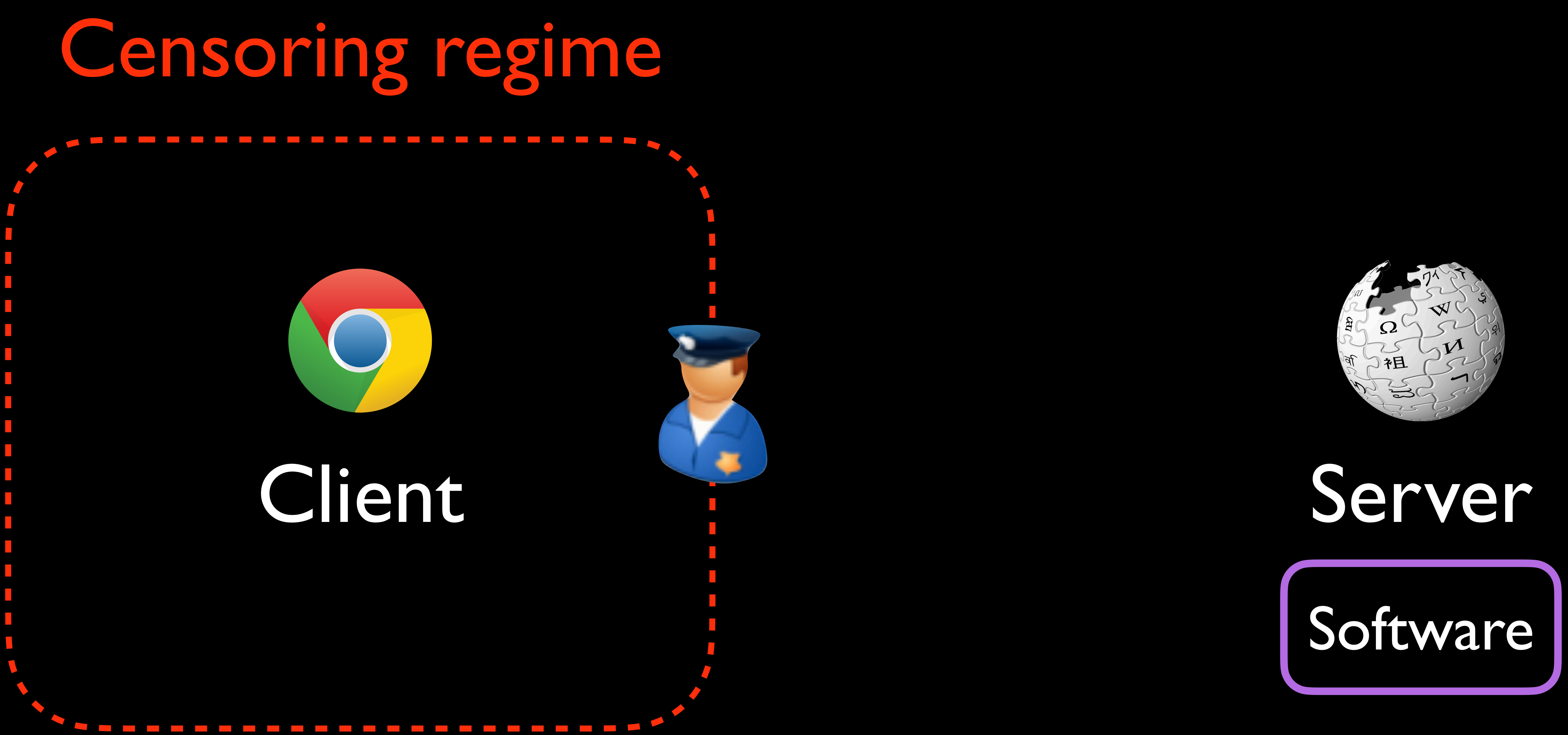
Poses risks to users

Cannot help those who
do not know they are censored

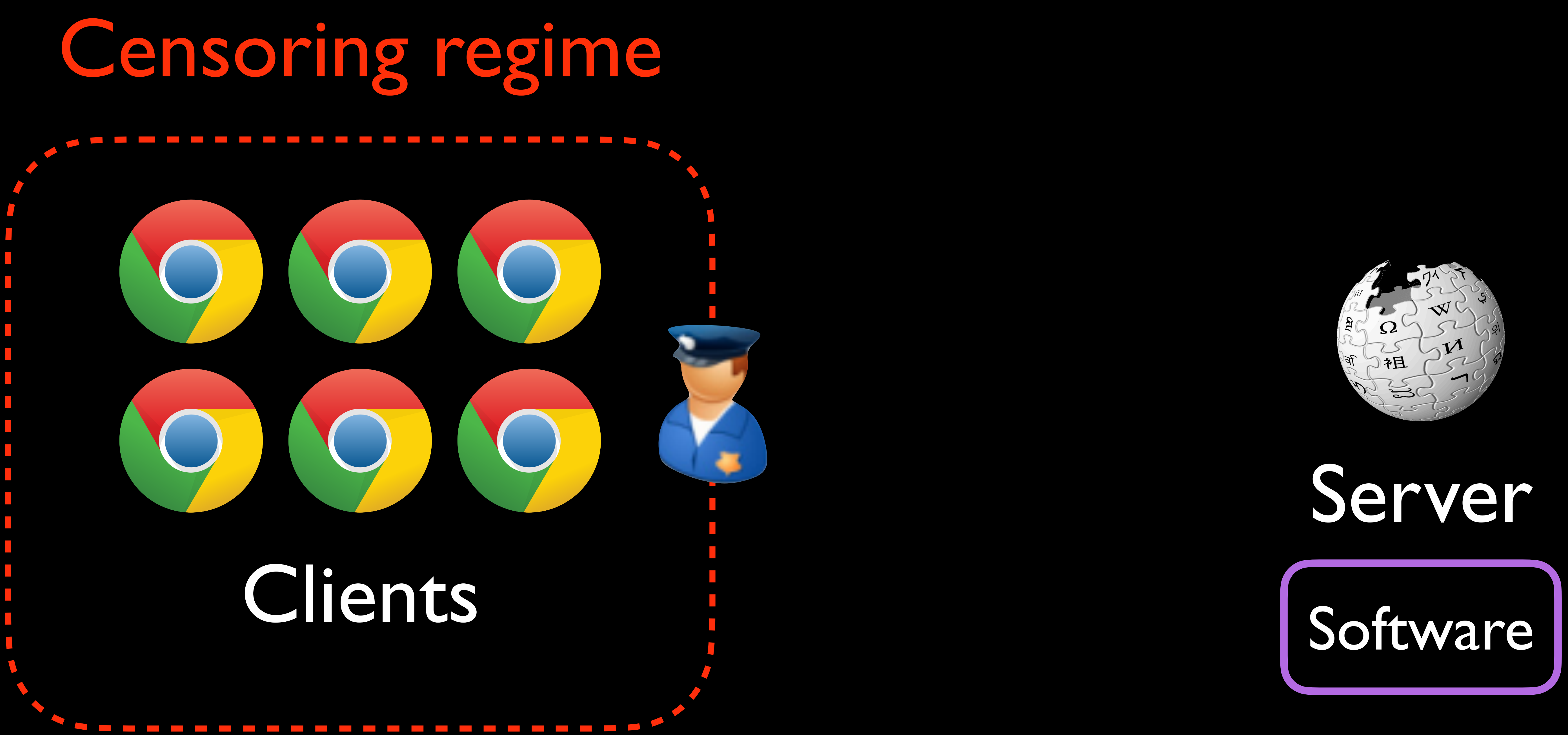
Server-side evasion



Server-side evasion

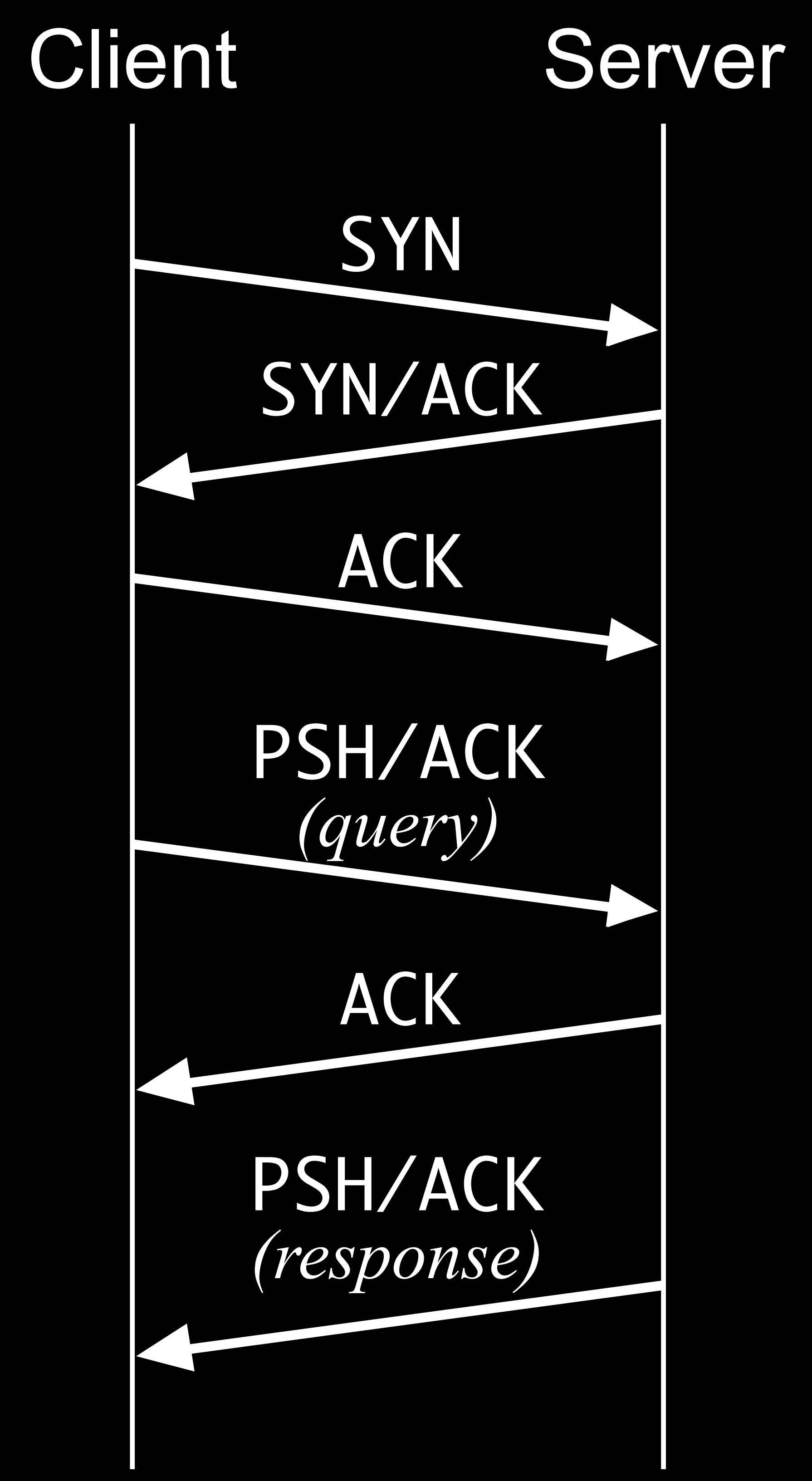


Server-side evasion

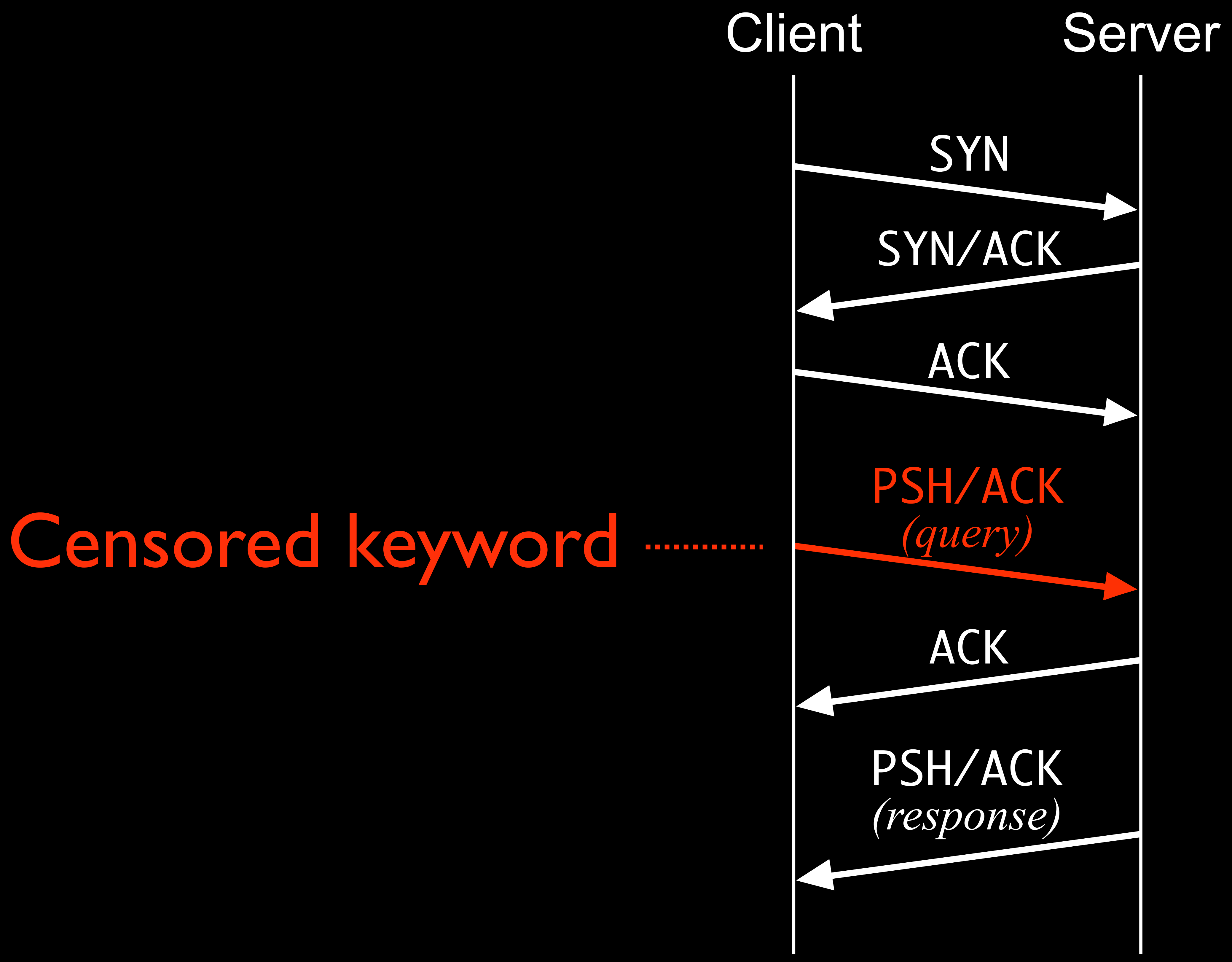


Potentially broadens reachability
without *any* client-side deployment

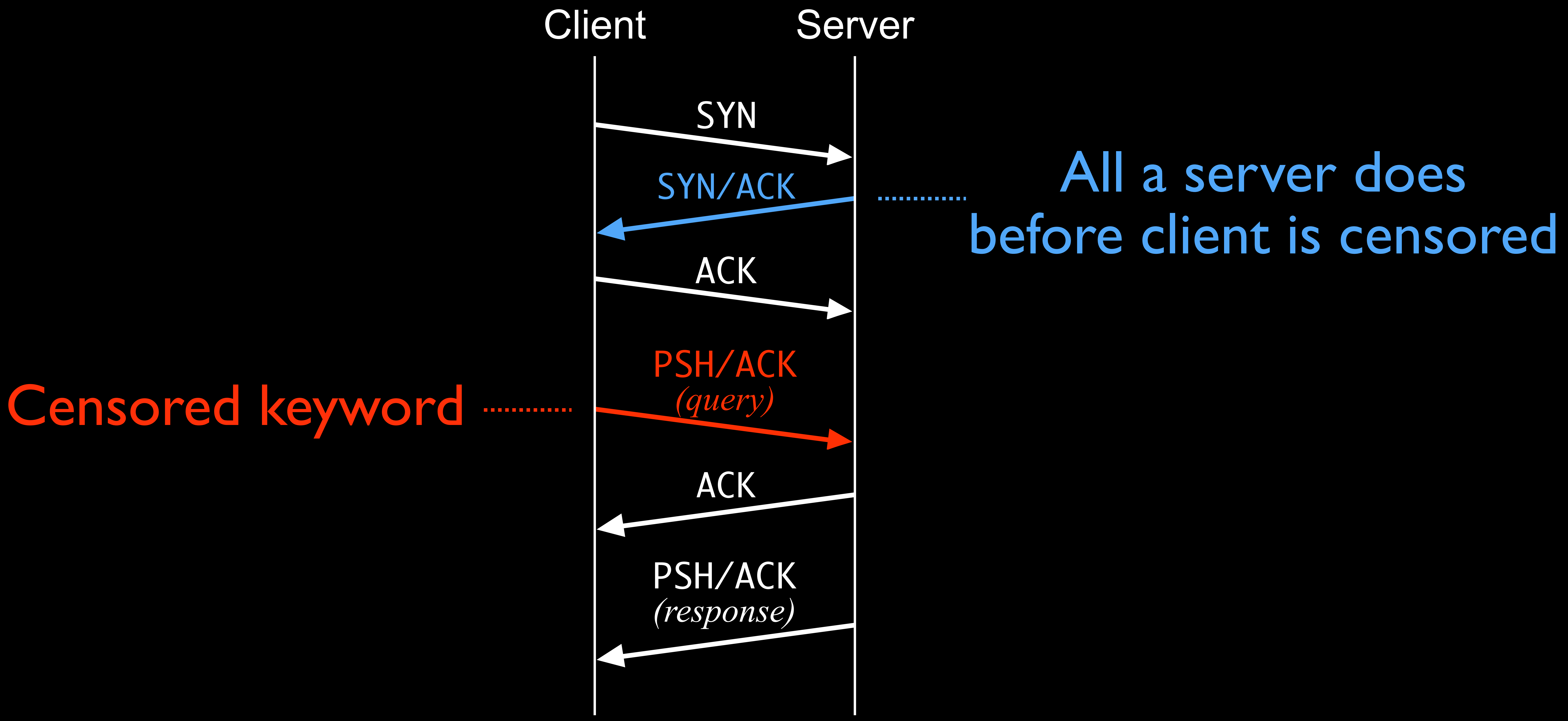
Server-side evasion “shouldn’t” work



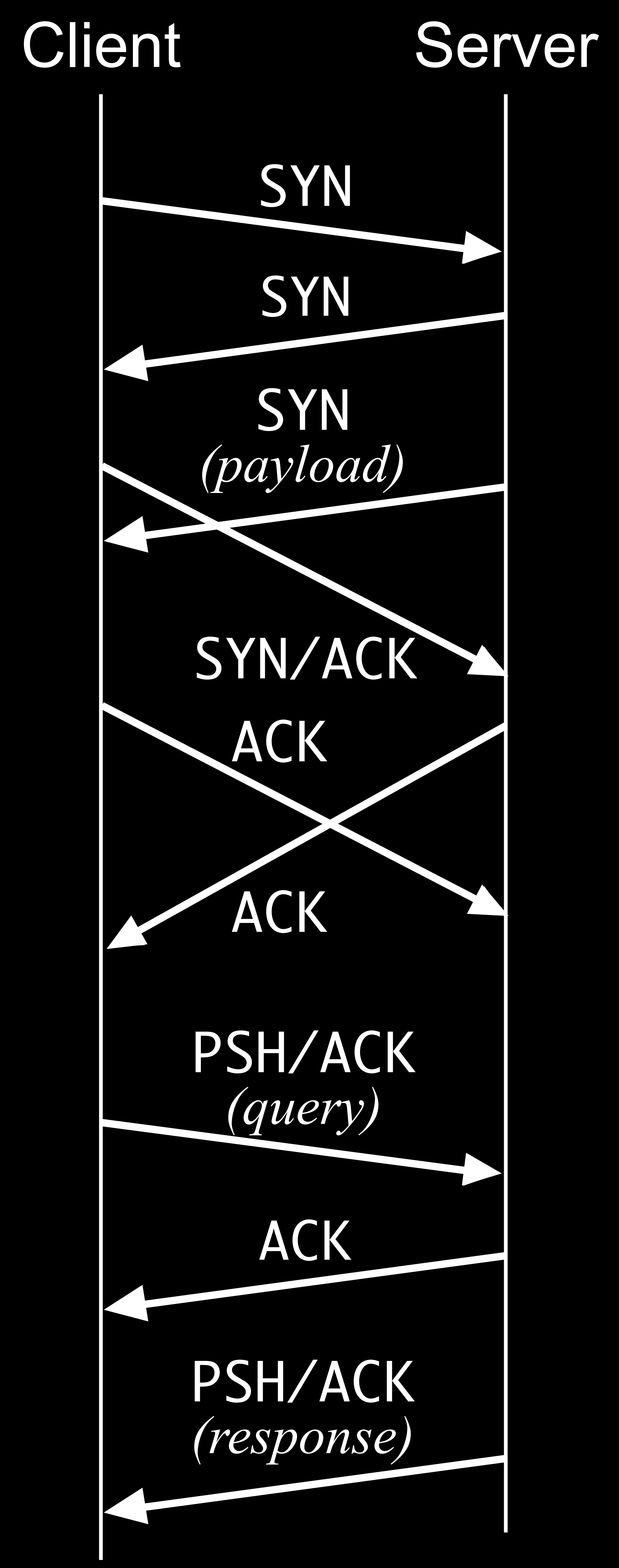
Server-side evasion “shouldn’t” work



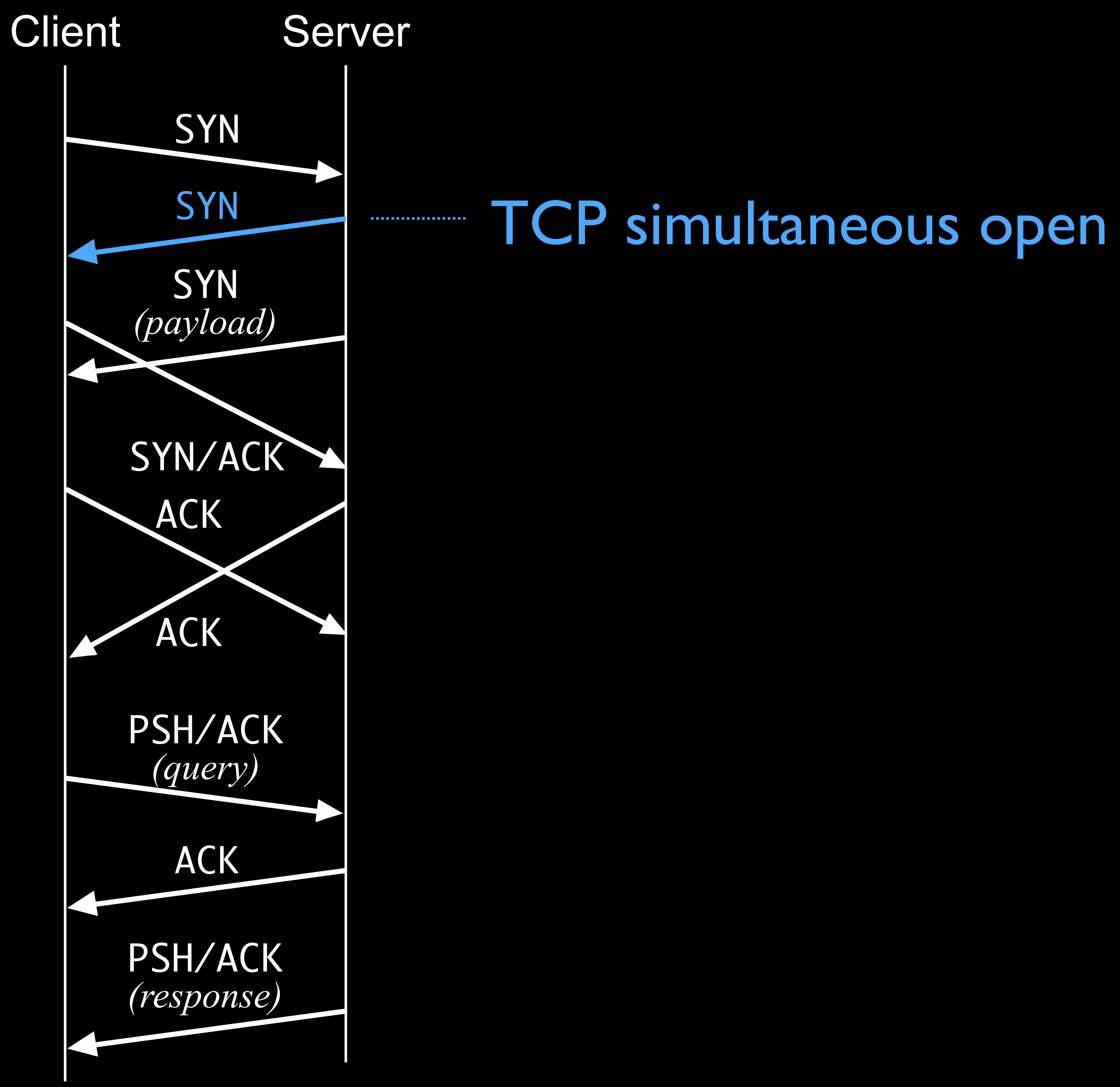
Server-side evasion "shouldn't" work



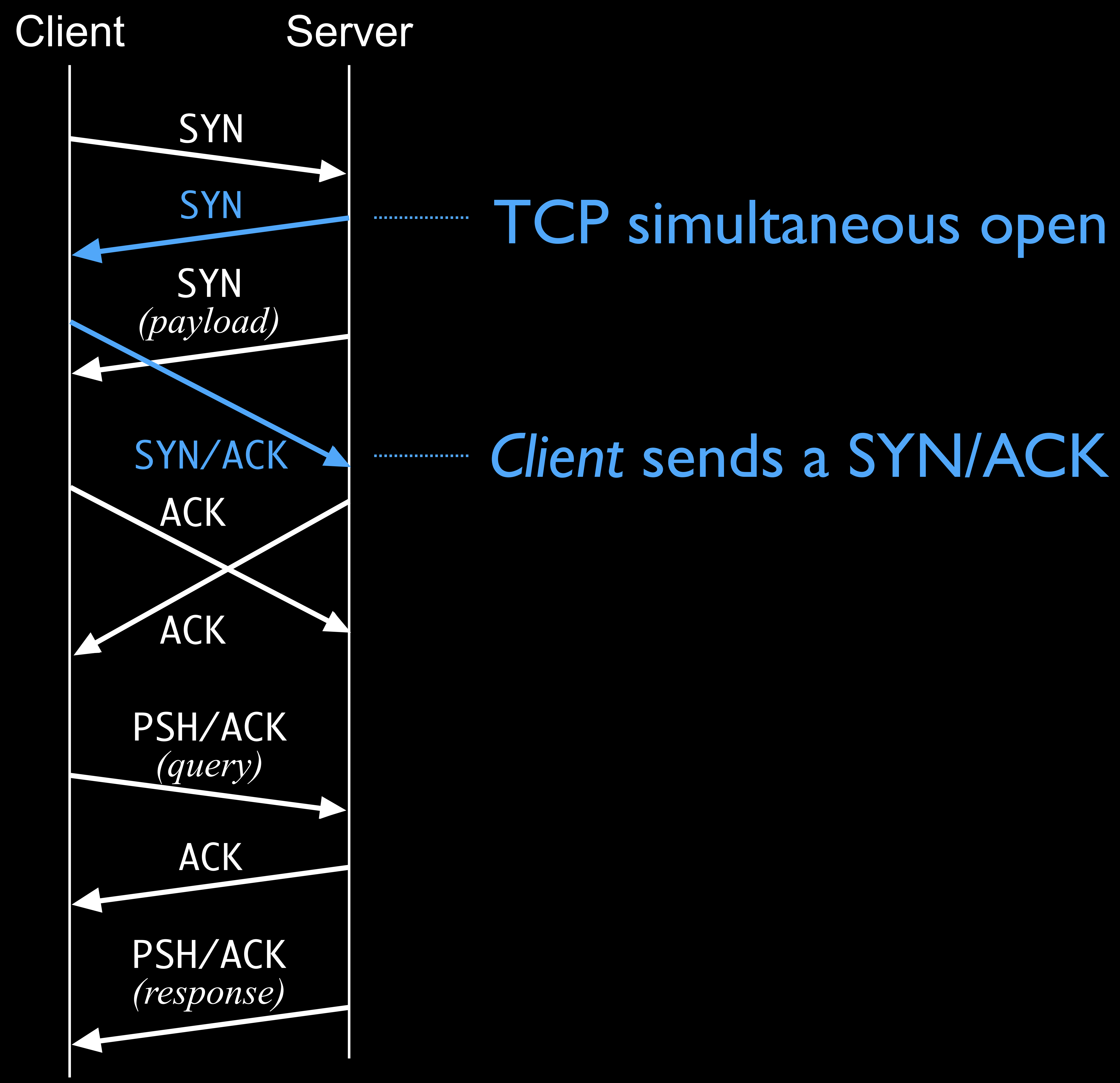
A successful server-side evasion strategy



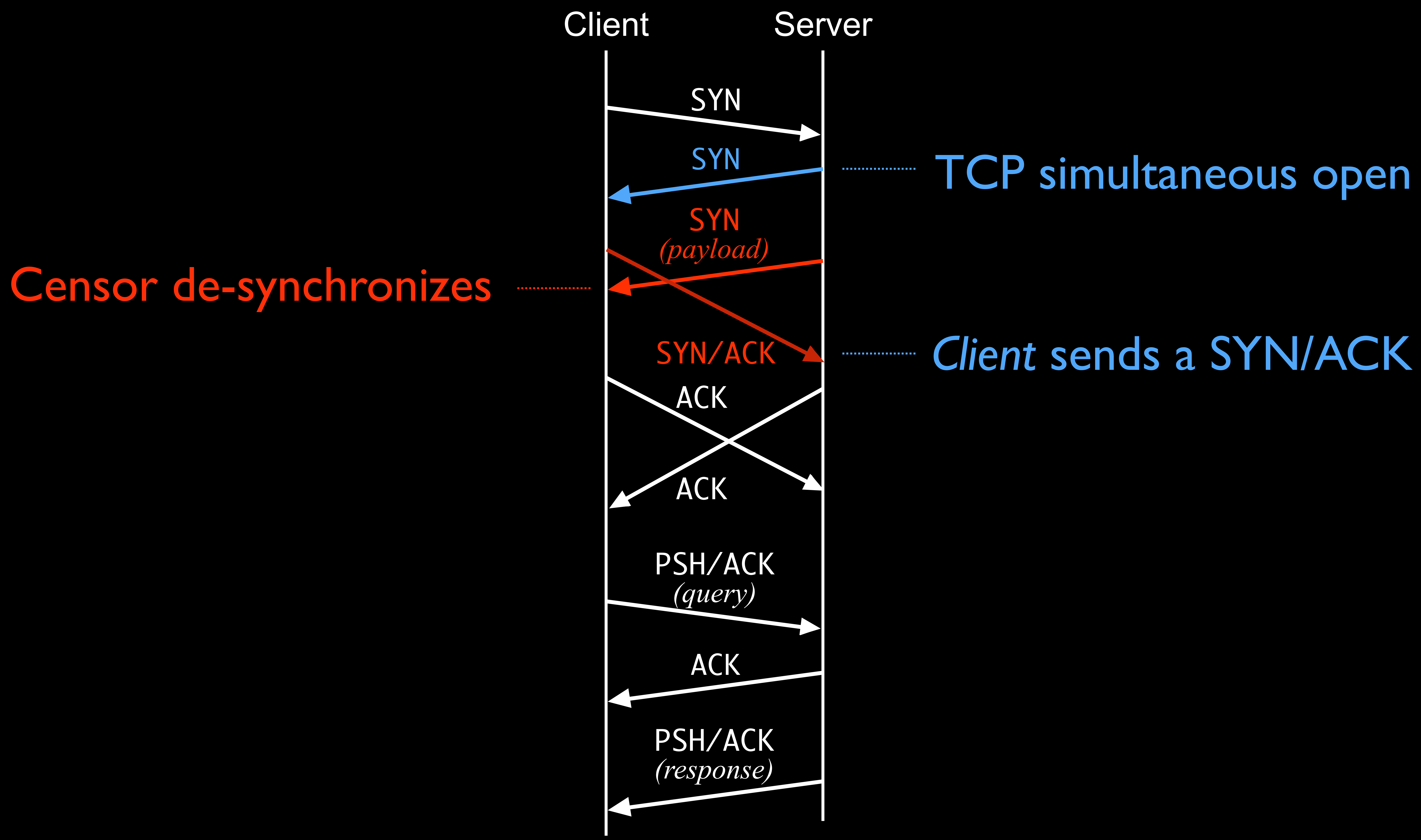
A successful server-side evasion strategy



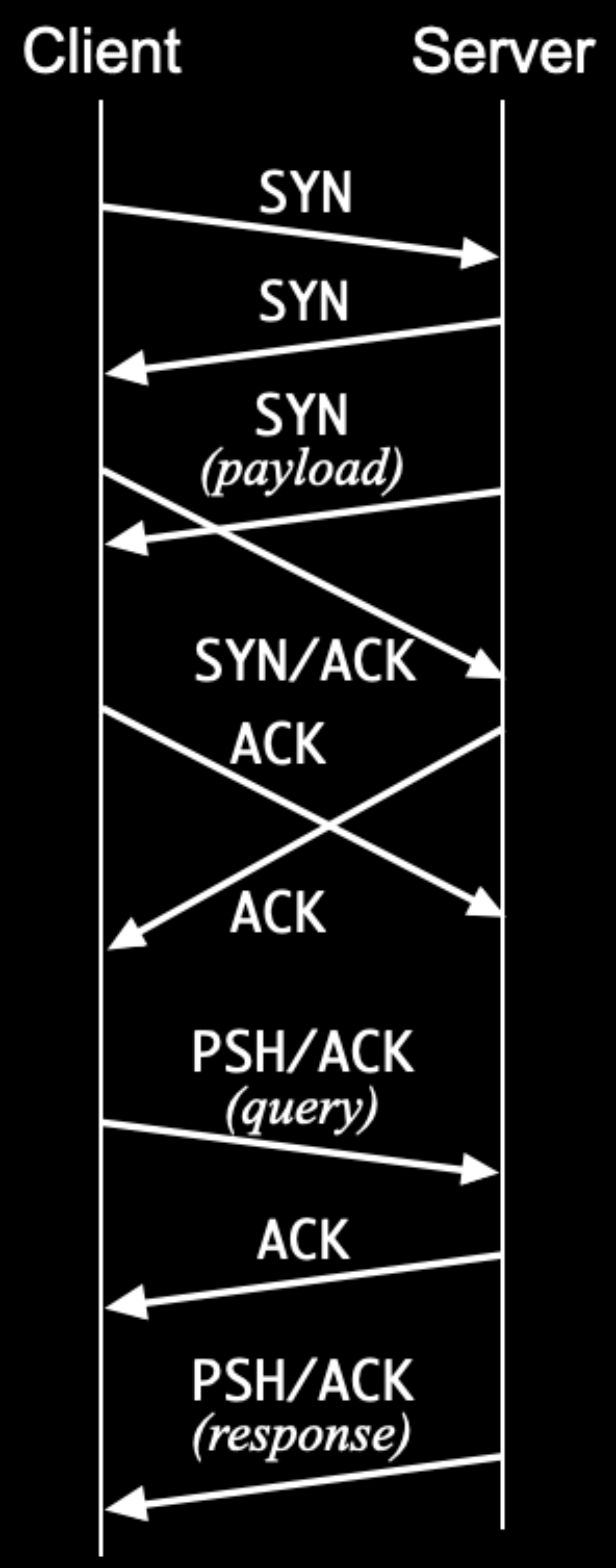
A successful server-side evasion strategy



A successful server-side evasion strategy



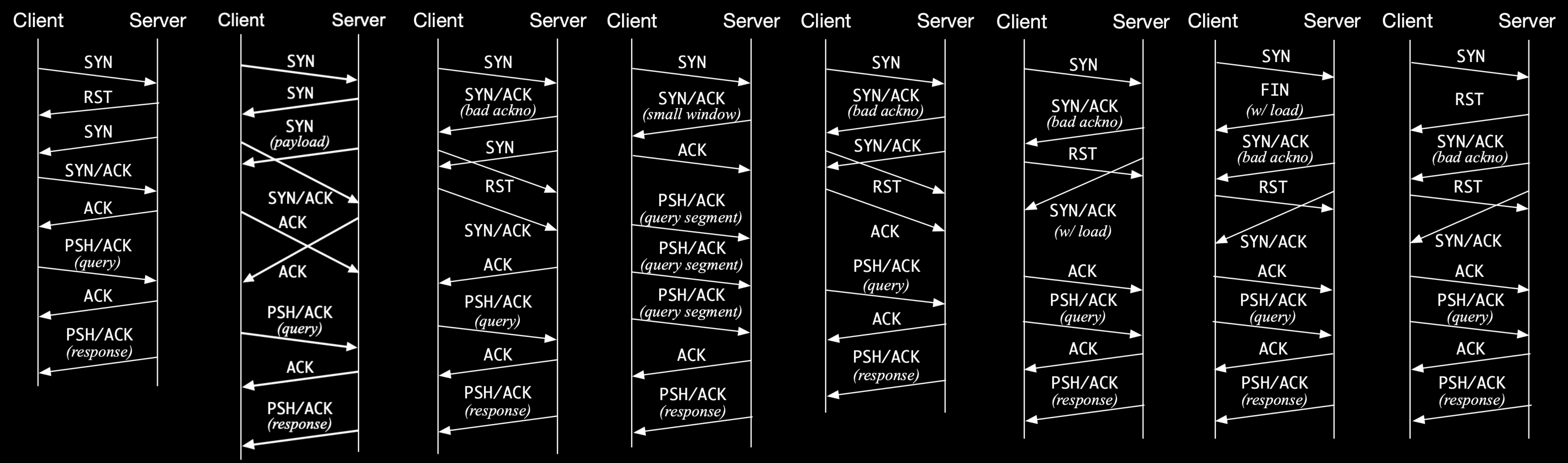
A successful server-side evasion strategy



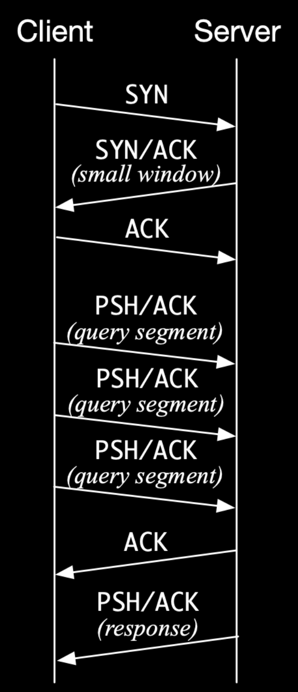
Success rates

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

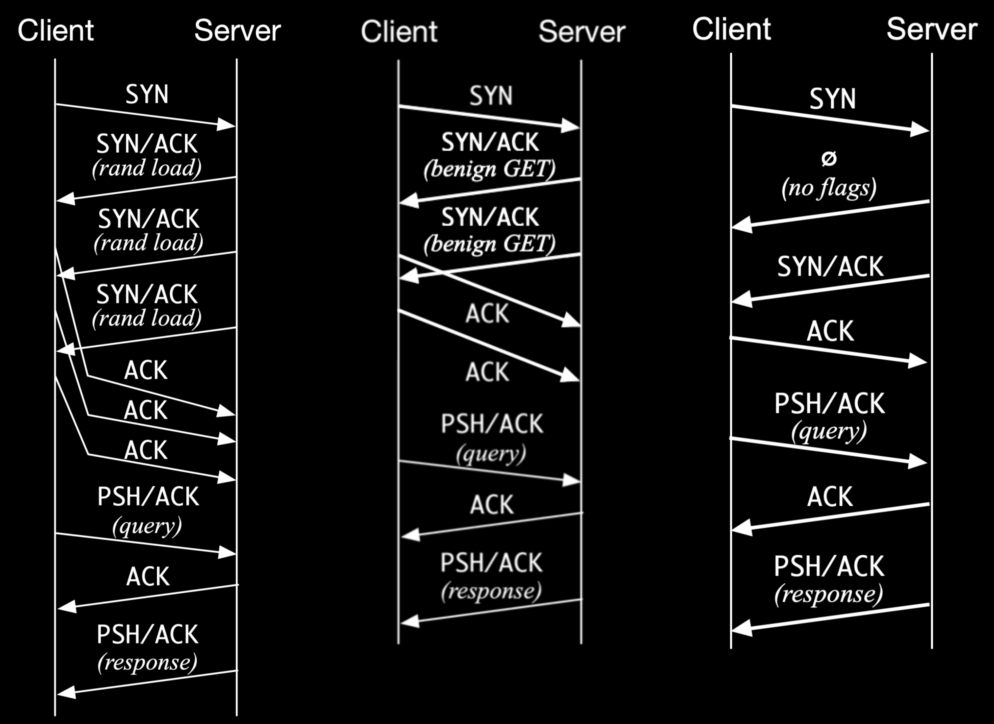
Server-side evasion strategies



China
8 strategies



Iran/India
1 strategy

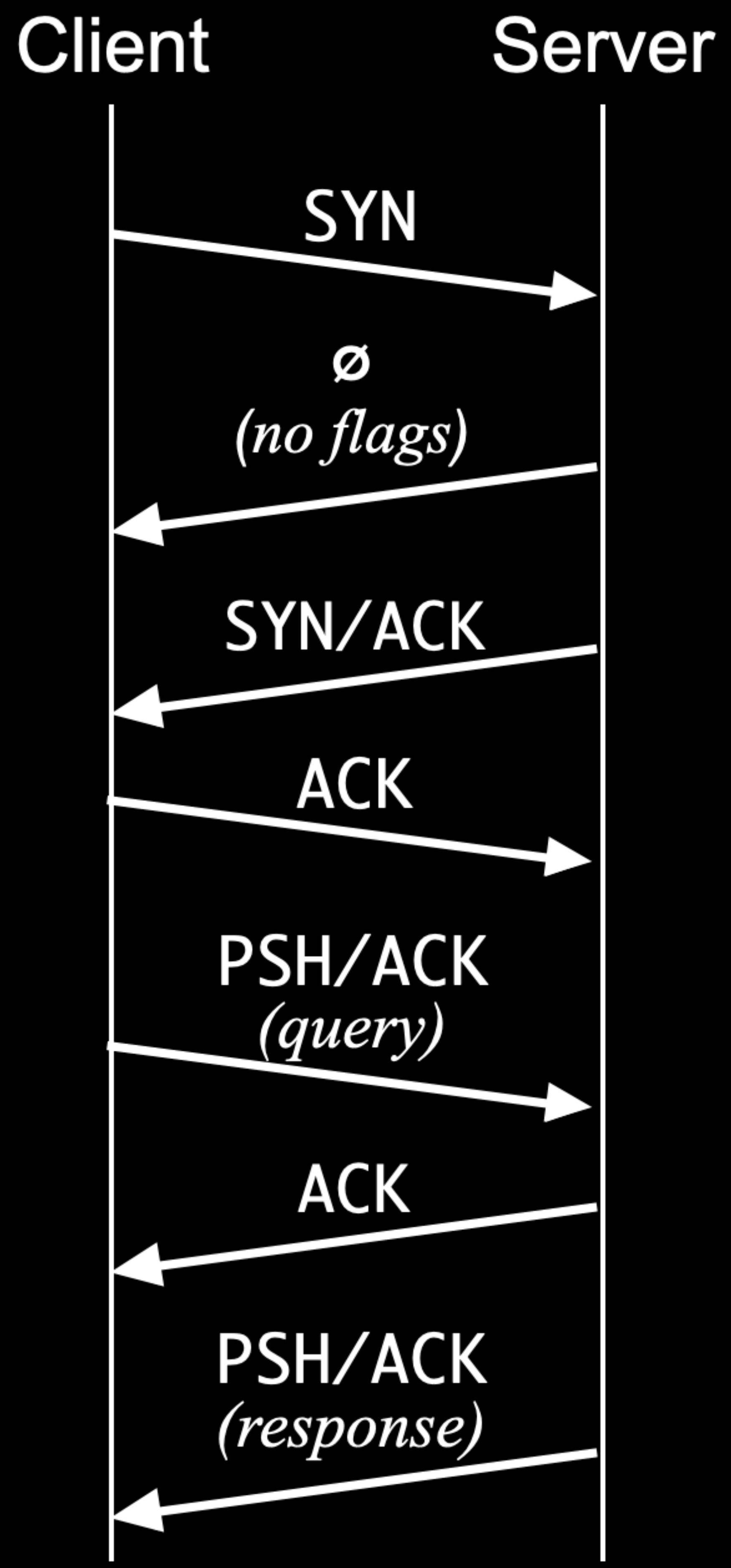


Kazakhstan
3 strategies

None of these require *any* client-side deployment

Server-side evasion results

NULL TCP Flags

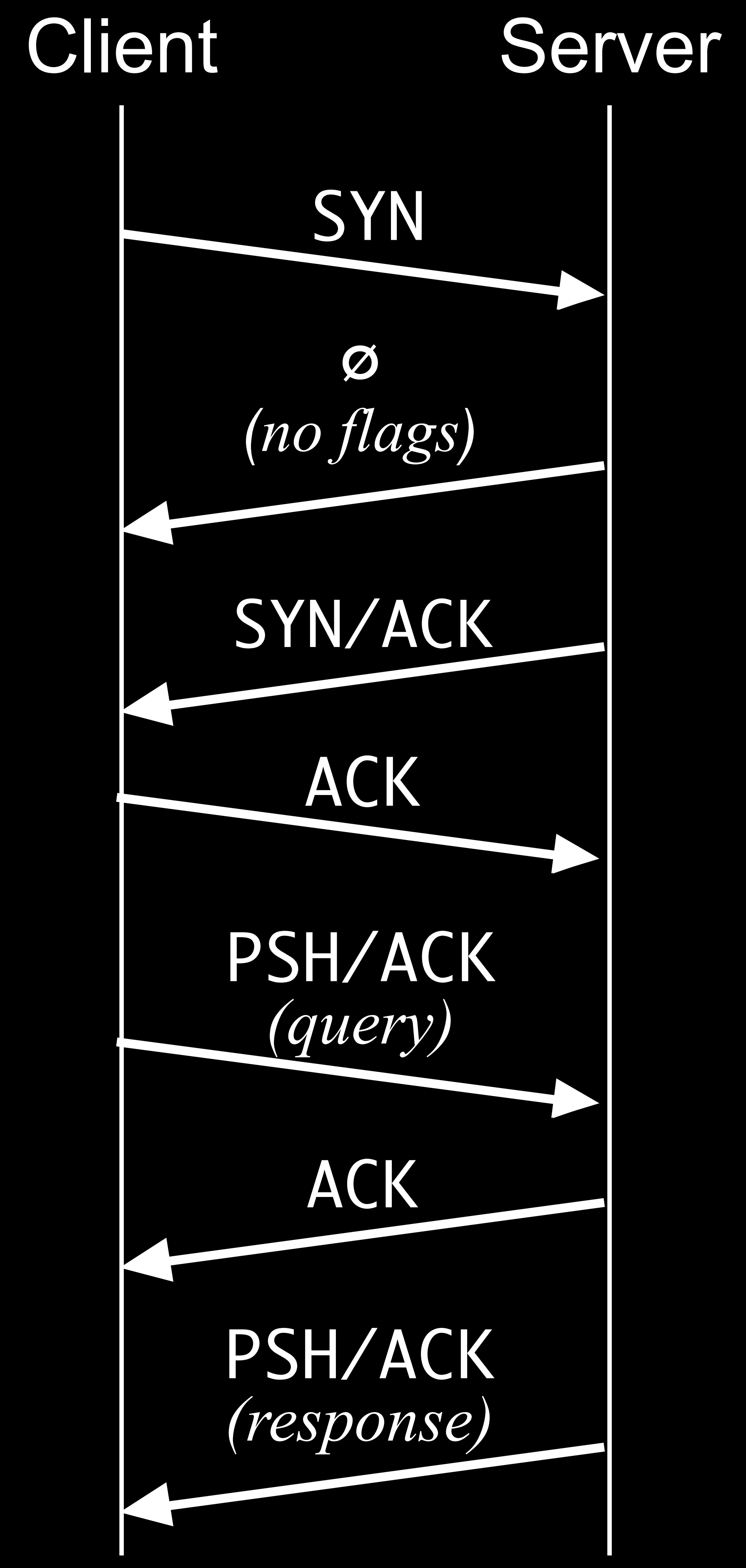


Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags

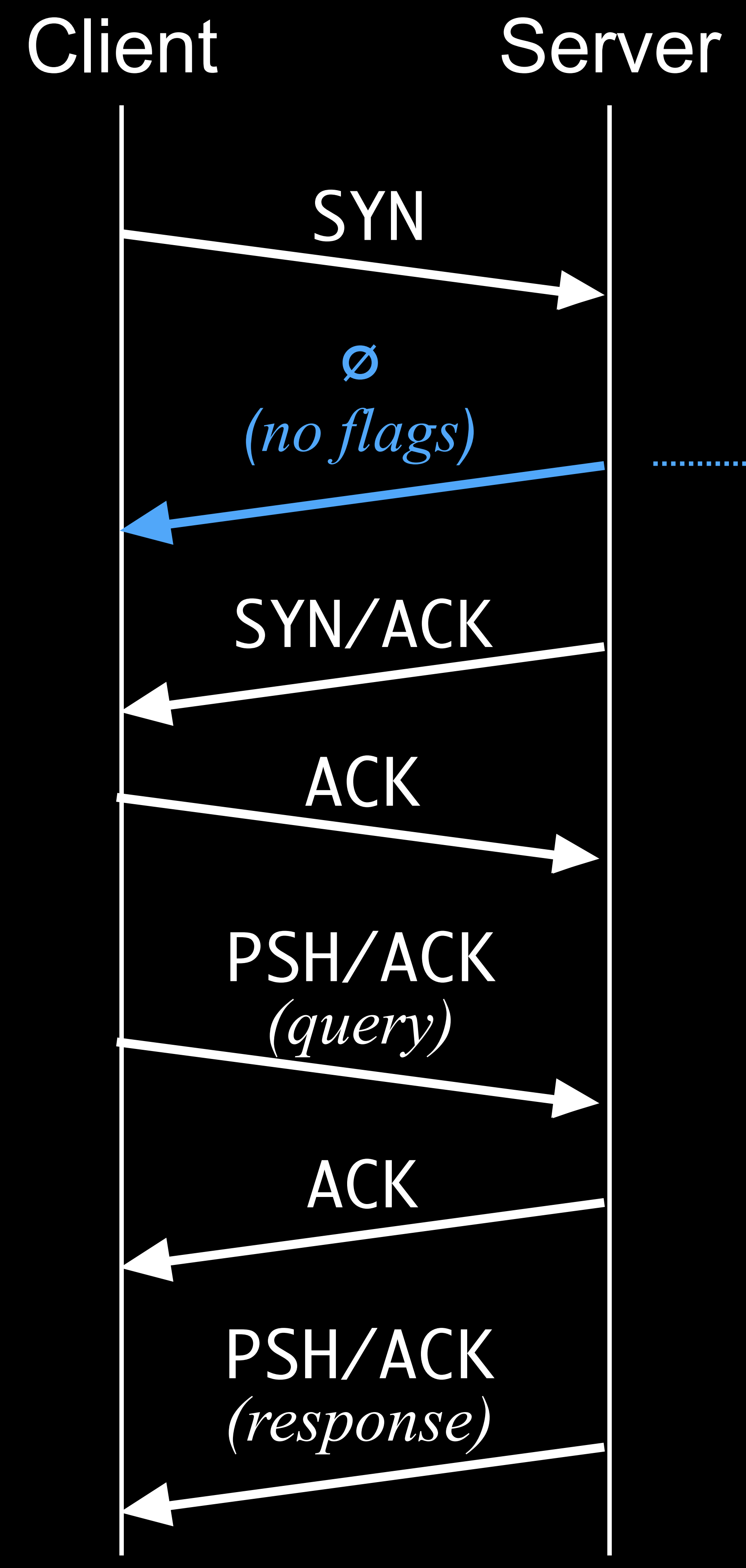


Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags



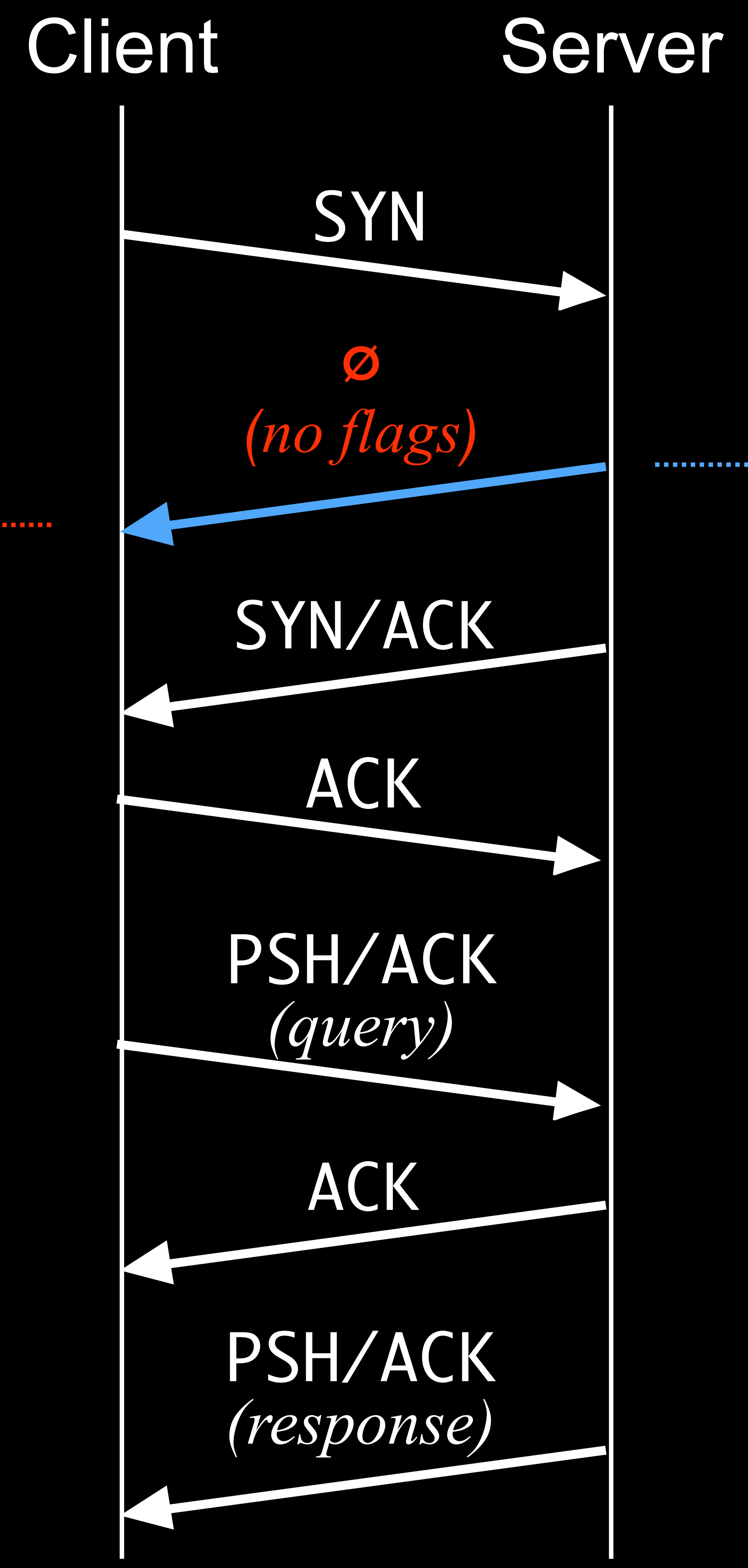
Server sends a packet with no TCP flags set

Success rates

HTTP 100%

Server-side evasion results

NULL TCP Flags



Censor can't handle unexpected flags

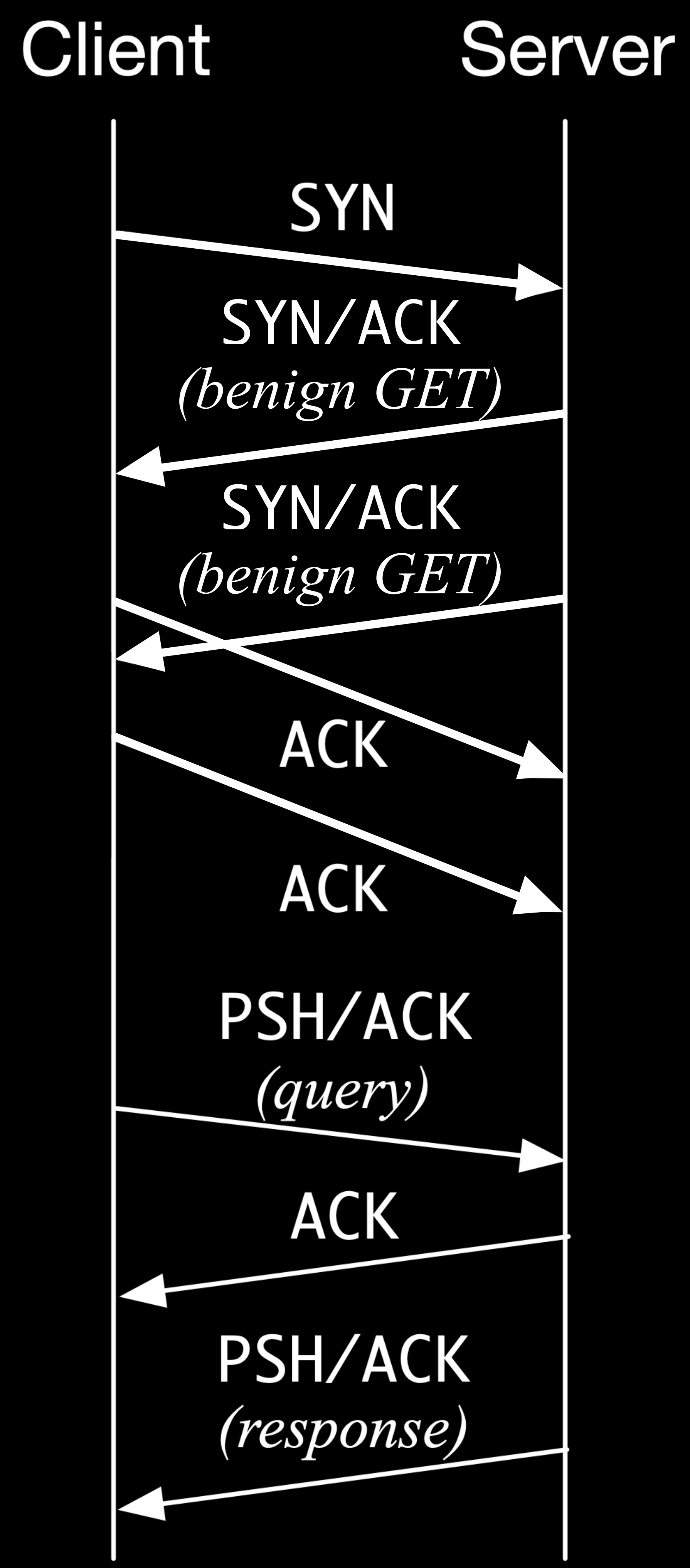
Server sends a packet with no TCP flags set

Success rates

HTTP 100%

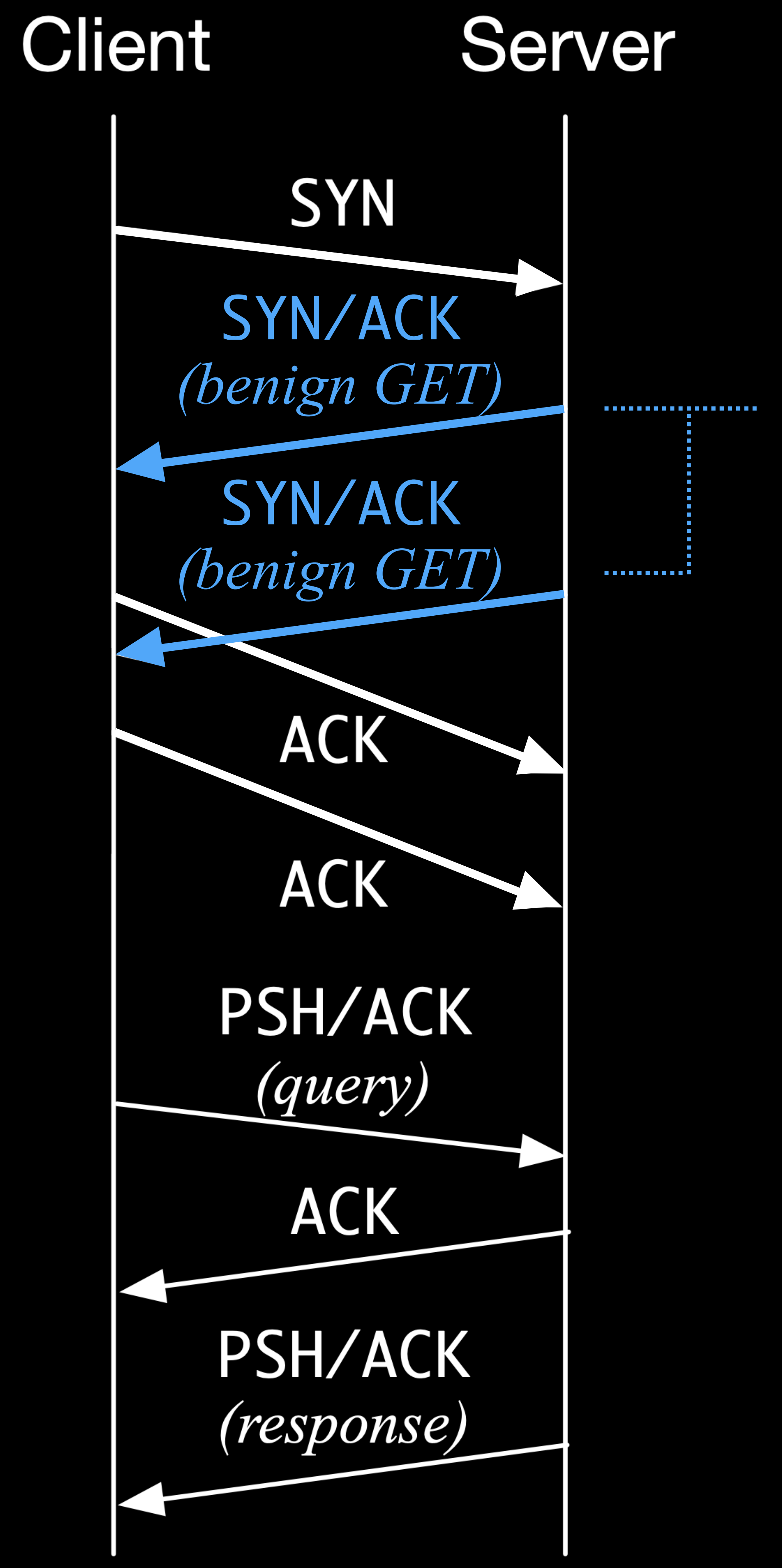
Server-side evasion results

Double benign-GETs



Server-side evasion results

Double benign-GETs

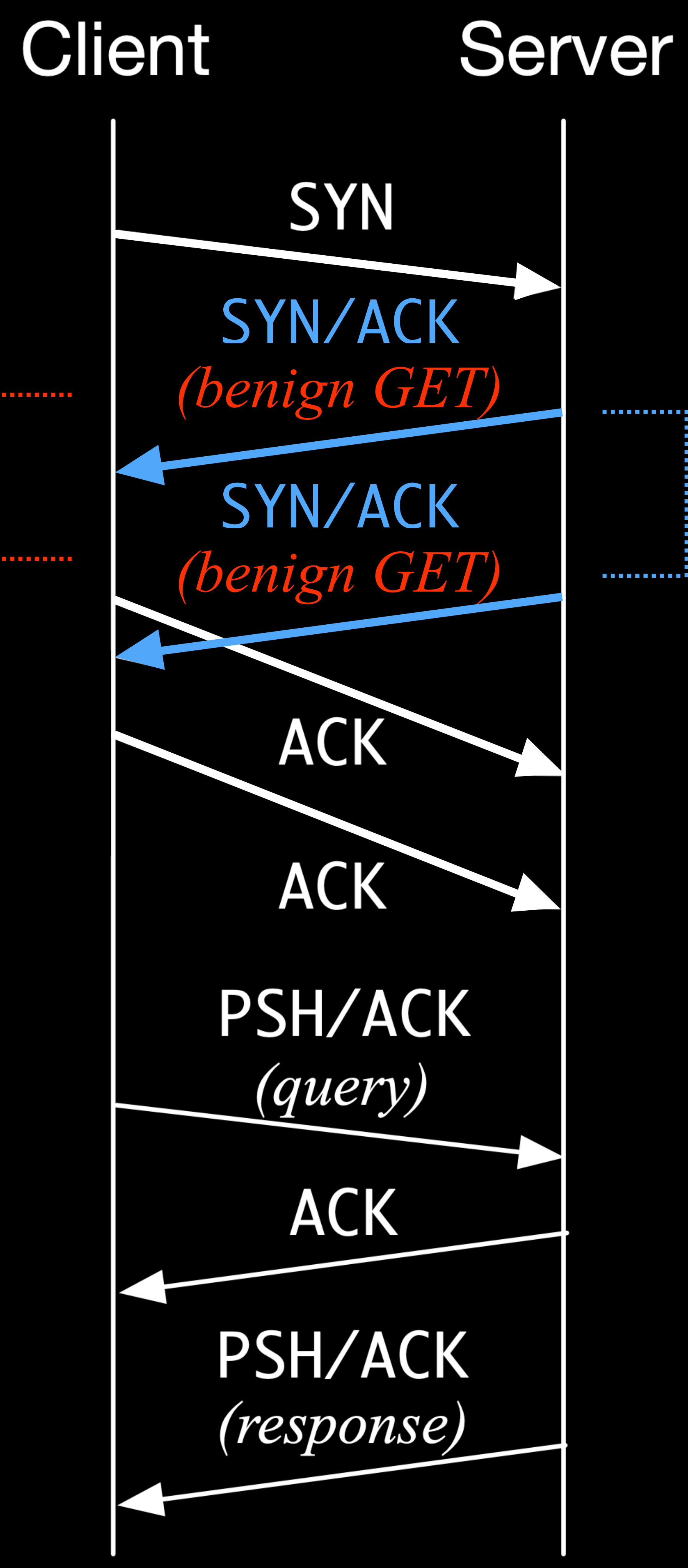


Server sends uncensored GETs inside *two* SYN/ACKs

Server-side evasion results

Double benign-GETs

Censor confuses connection direction



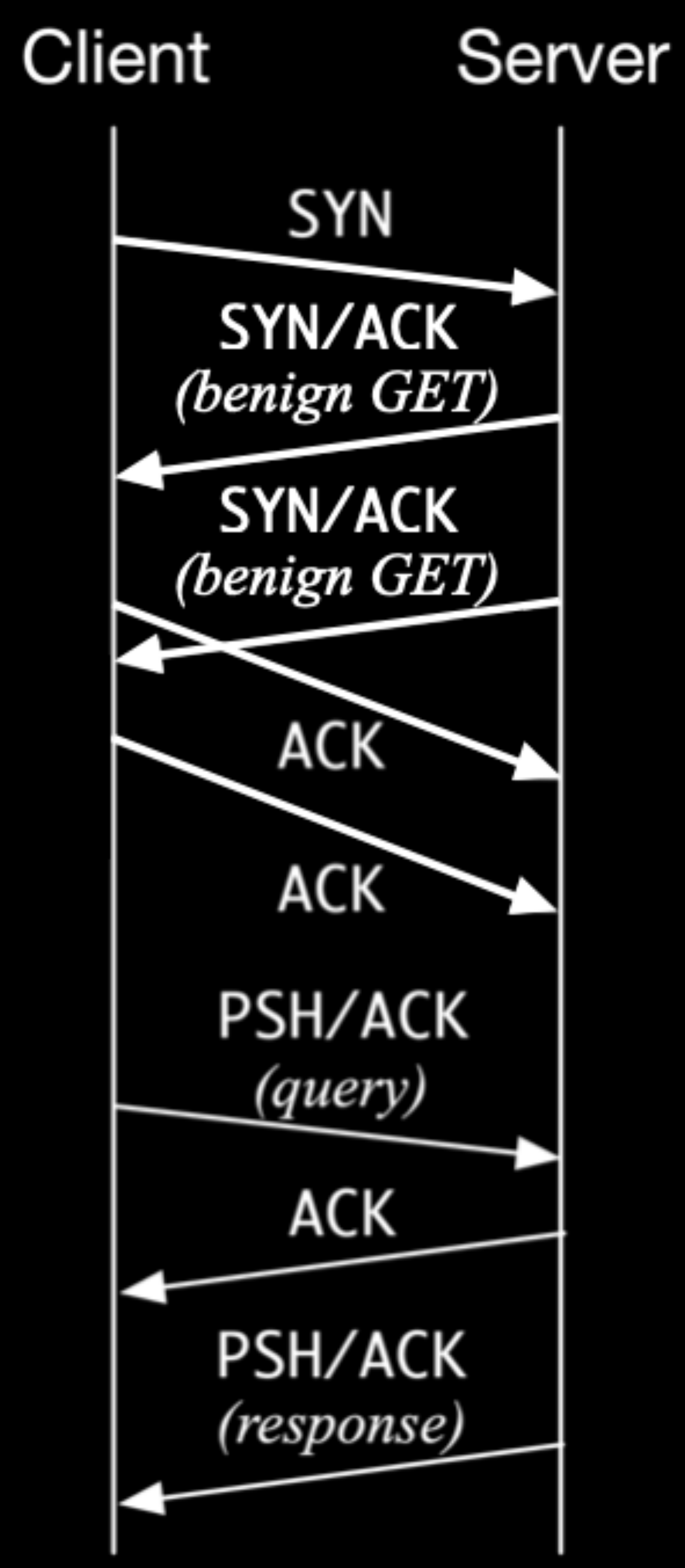
Server sends uncensored GETs inside *two* SYN/ACKs

Success rates

HTTP 100%

Server-side evasion results

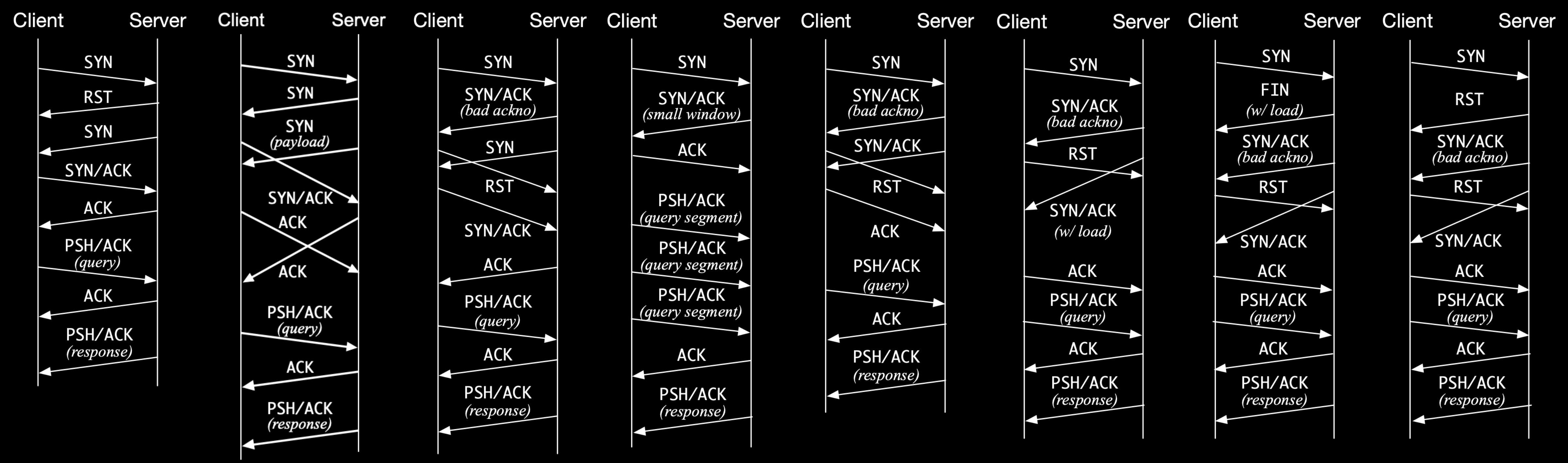
Double benign-GETs



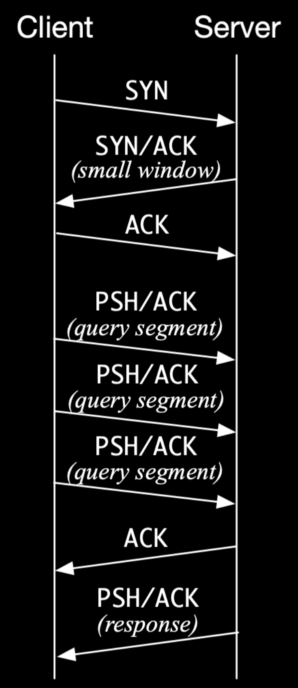
Success rates

HTTP	100%

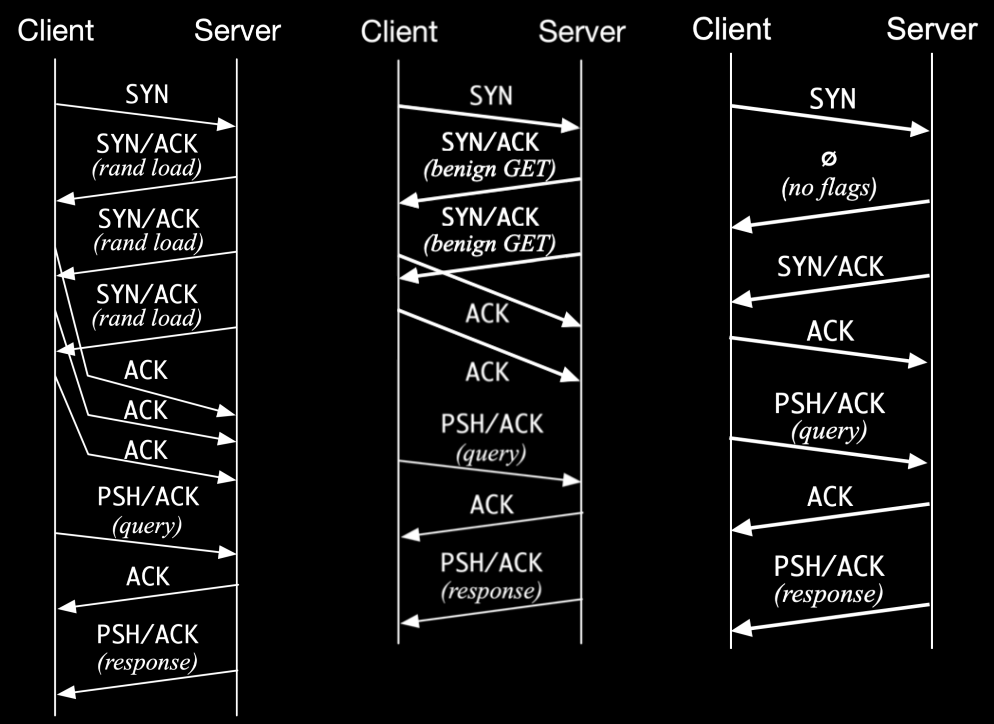
Server-side evasion strategies



China
8 strategies



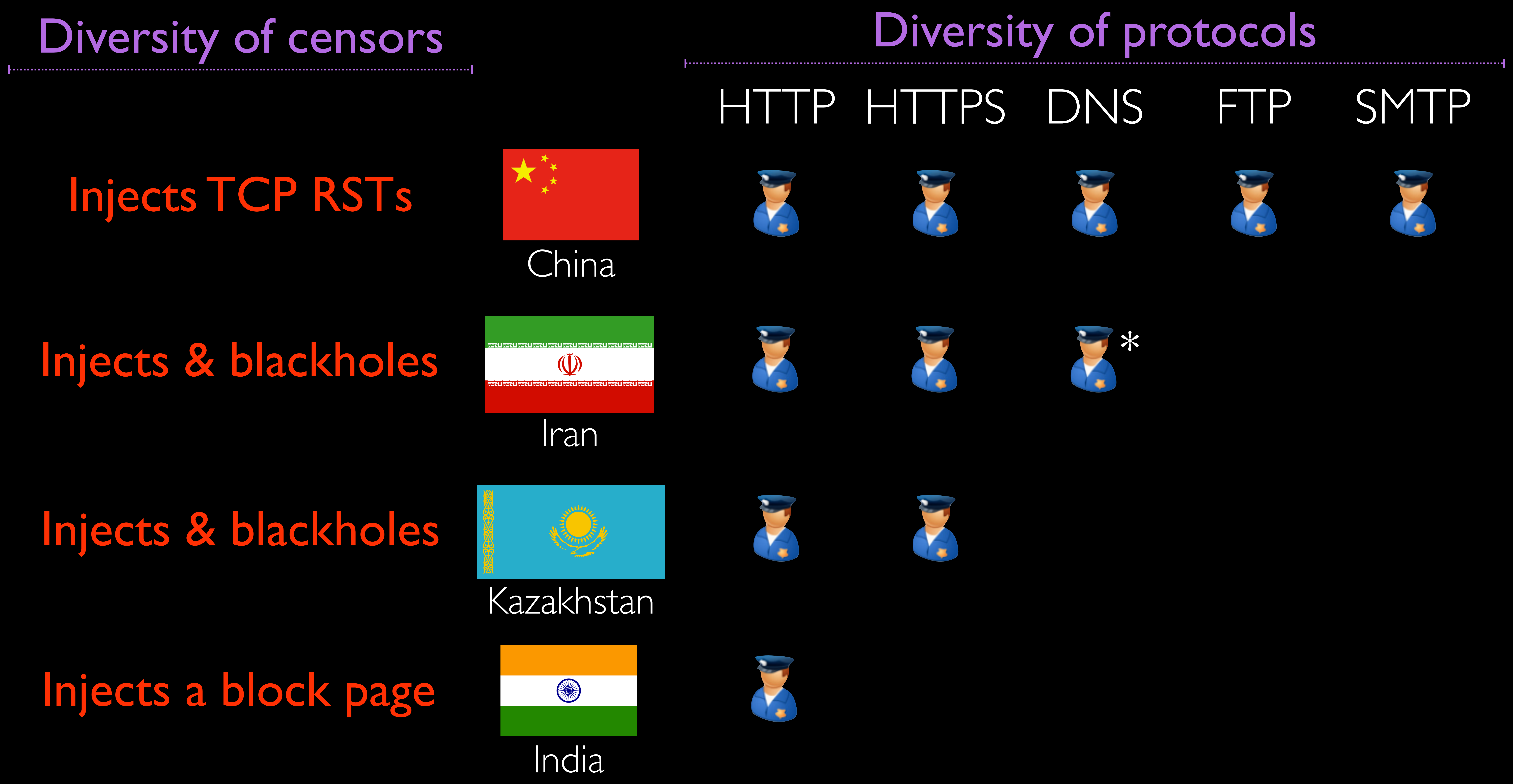
Iran/India
1 strategy



Kazakhstan
3 strategies

None of these require *any* client-side deployment

Server-side results – Real censor experiments



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

iOS 13.3

Android 10

Centos 6
Centos 7

Ubuntu 12.04
Ubuntu 14.04
Ubuntu 16.04
Ubuntu 18.04

What's next?

New insight into how censors work

- Success rate changes by protocol
- "Multi-box theory"

Rapid response to new censorship

- Iran's new protocol filter (Feb 2020)
- China's new ESNI filter (July 2020)

New insights into how censors work

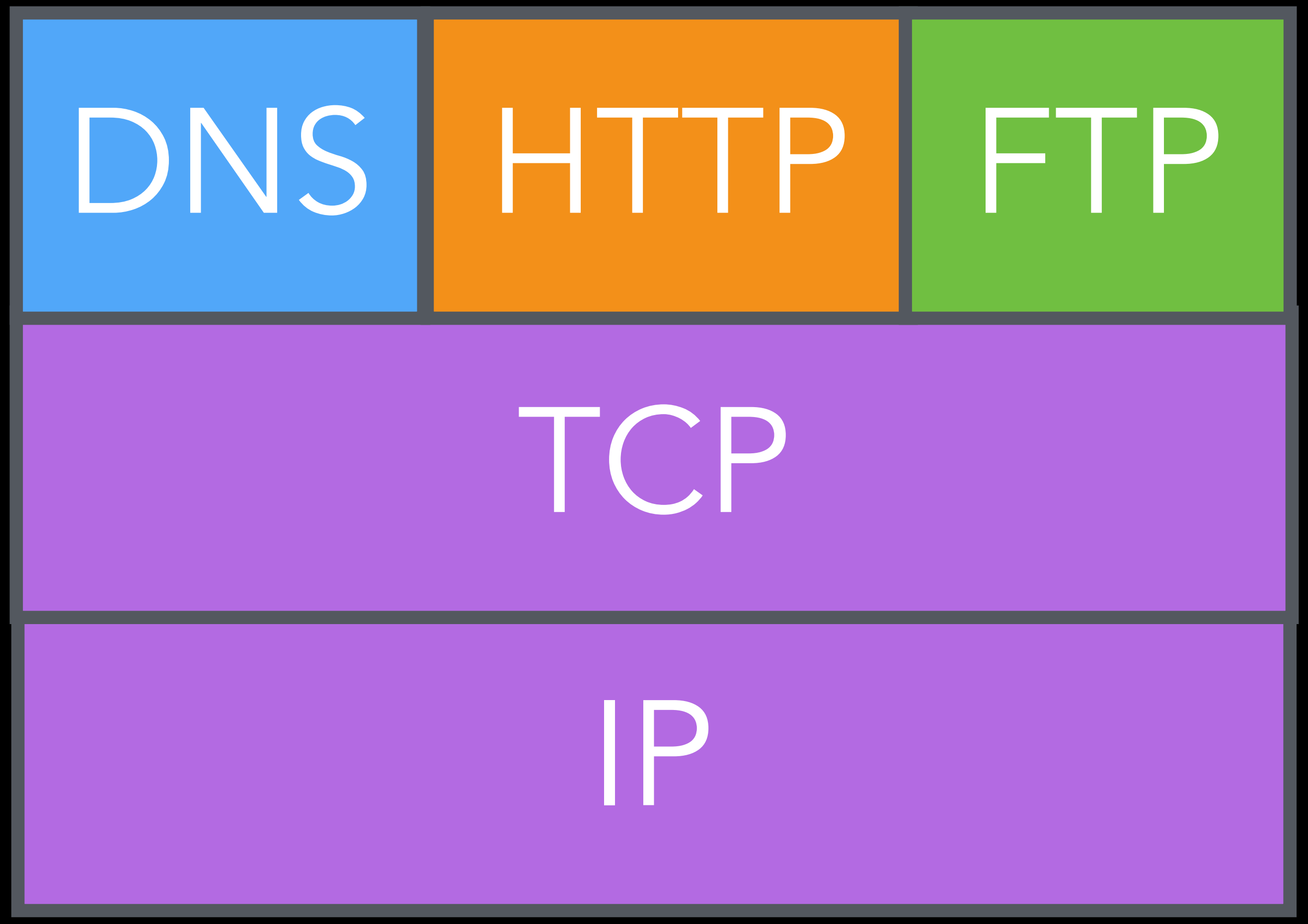
Strategy #	Description	DNS	FTP	HTTP	HTTPS	SMTP
		Success Rates				
China						
–	No evasion	2%	3%	3%	3%	26%
1	Sim. Open, Injected RST	89%	52%	54%	14%	70%
2	Sim. Open, Injected Load	83%	36%	54%	55%	59%
3	Corrupt ACK, Sim. Open	26%	65%	4%	4%	23%
4	Corrupt ACK Alone	7%	33%	5%	5%	22%
5	Corrupt ACK, Injected 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

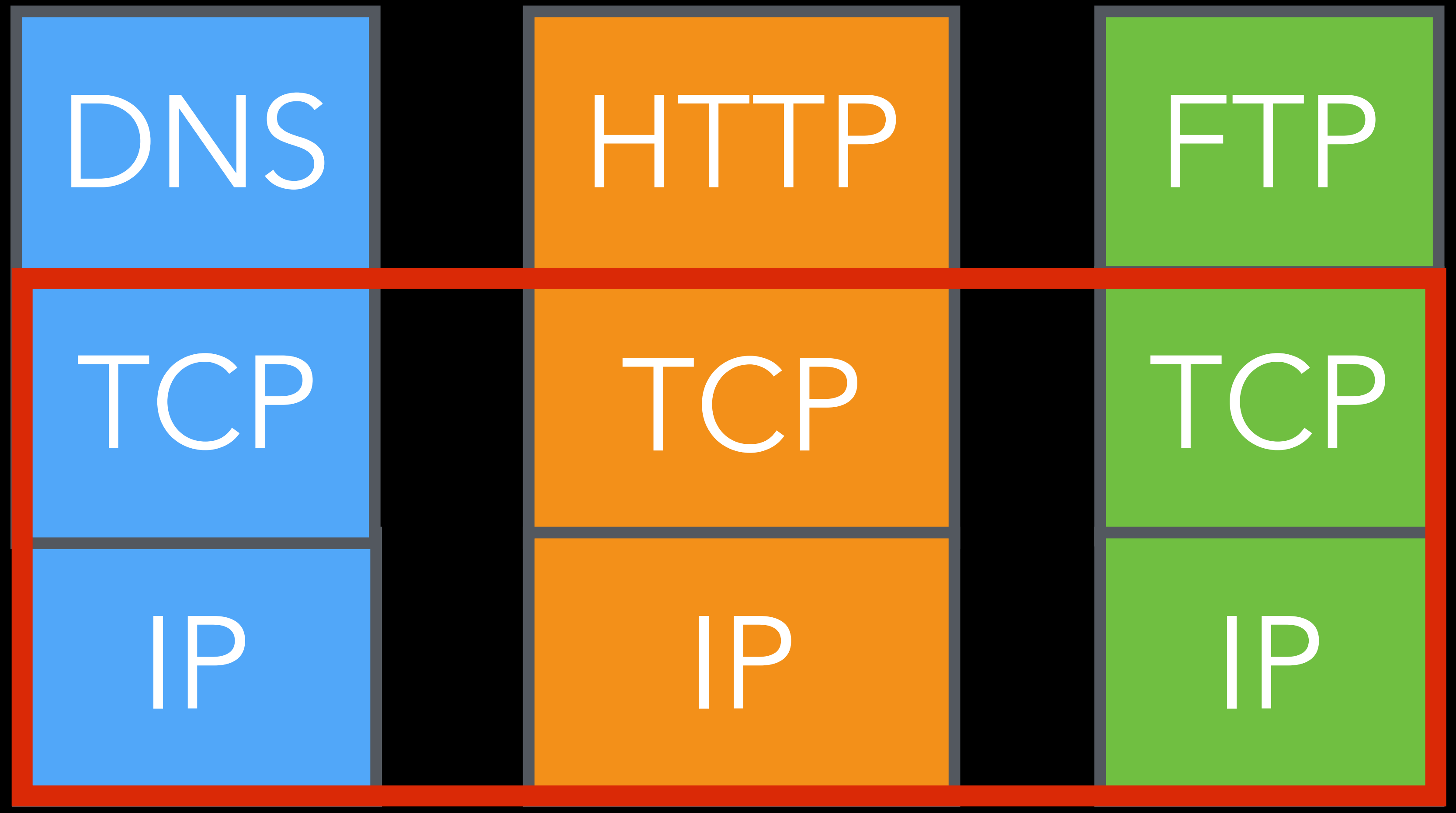
So why are different applications affected differently in China?

New Model for Chinese Censorship

Sane

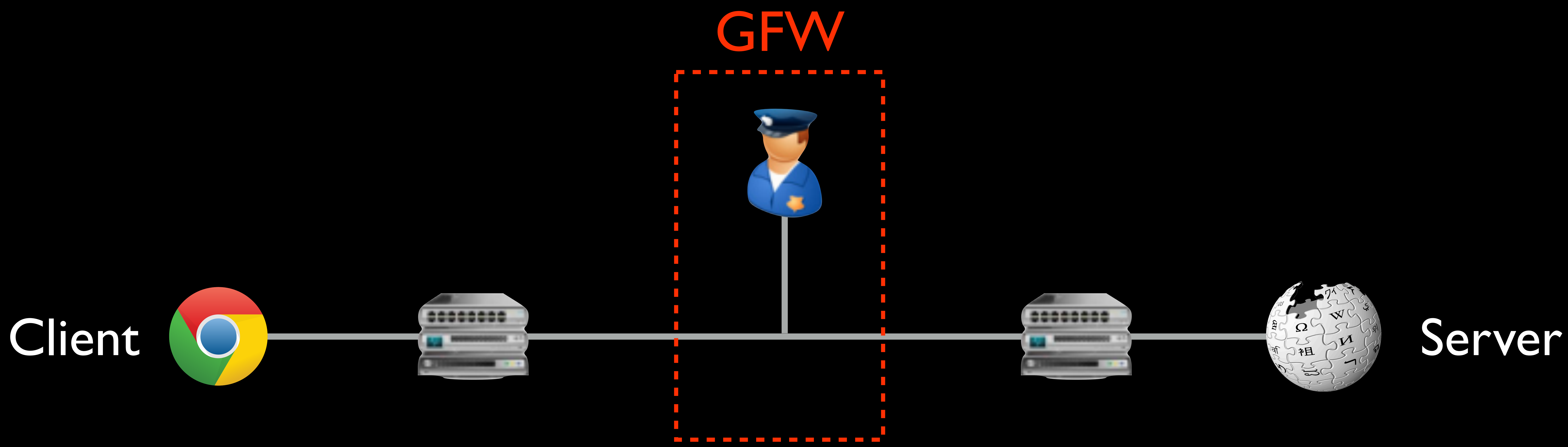


Apparently what's happening

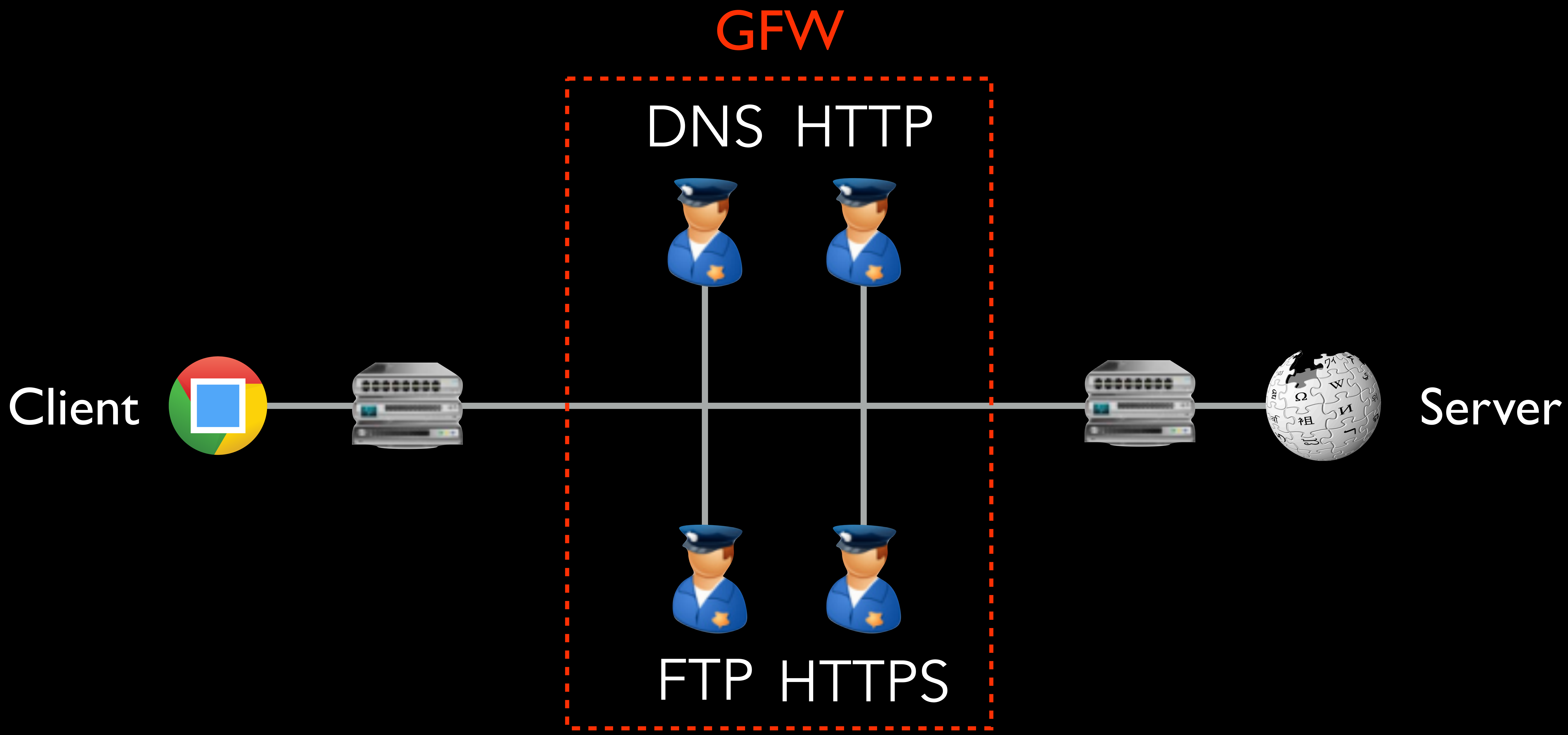


Results suggest GFW is running
multiple censoring middleboxes in parallel

Multi-box theory

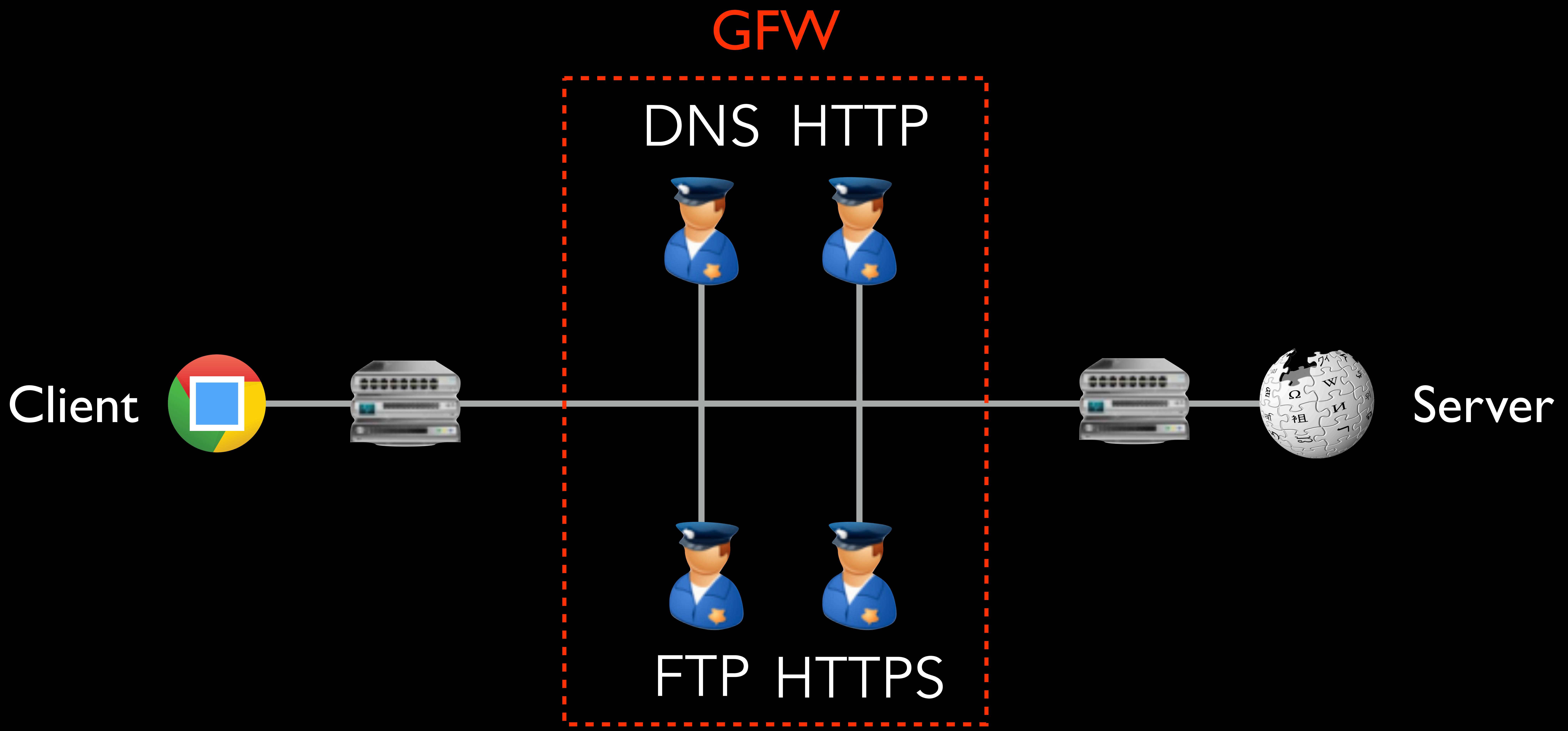


Multi-box theory



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

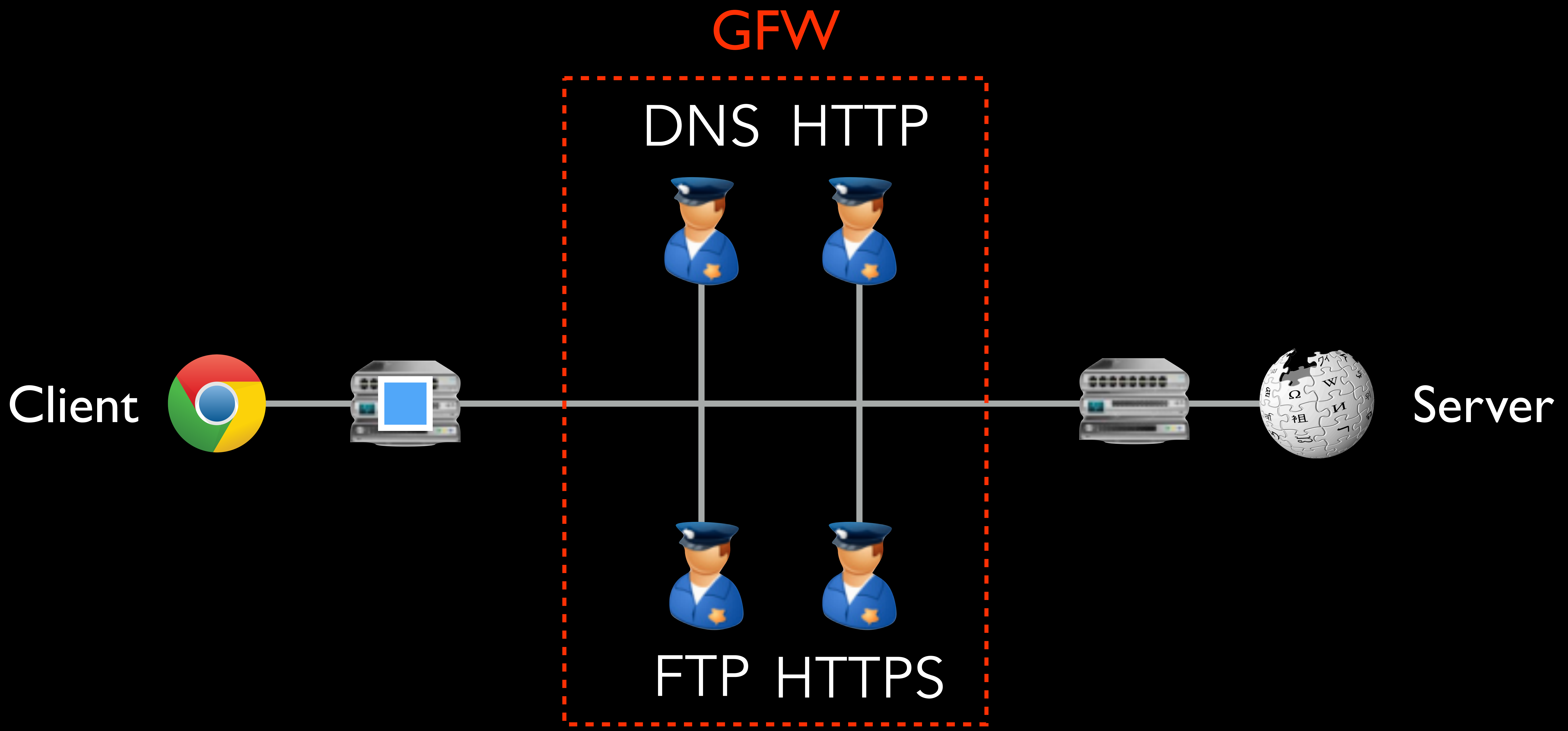
Multi-box theory



Not port number

Censors effectively on any port

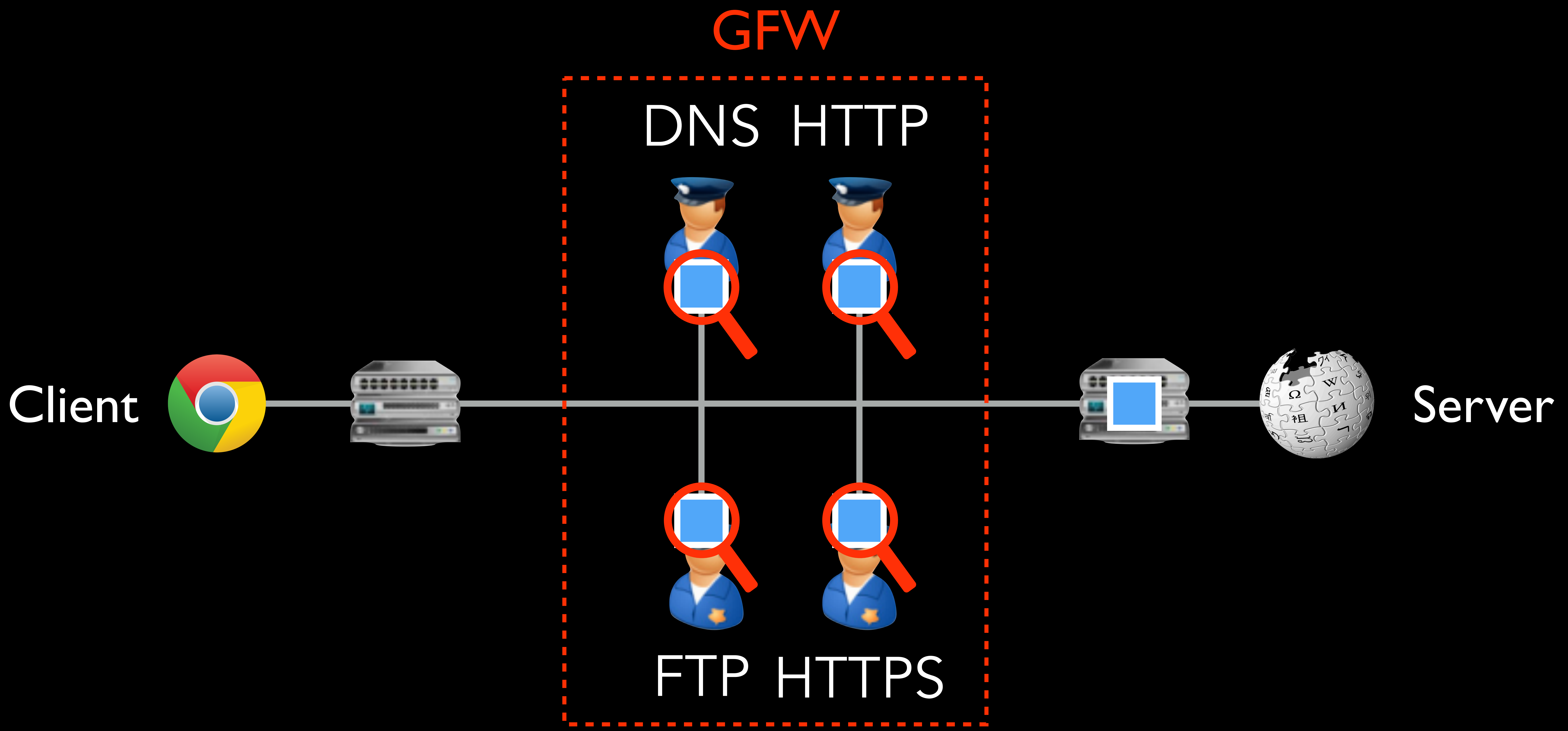
Multi-box theory



Not port number

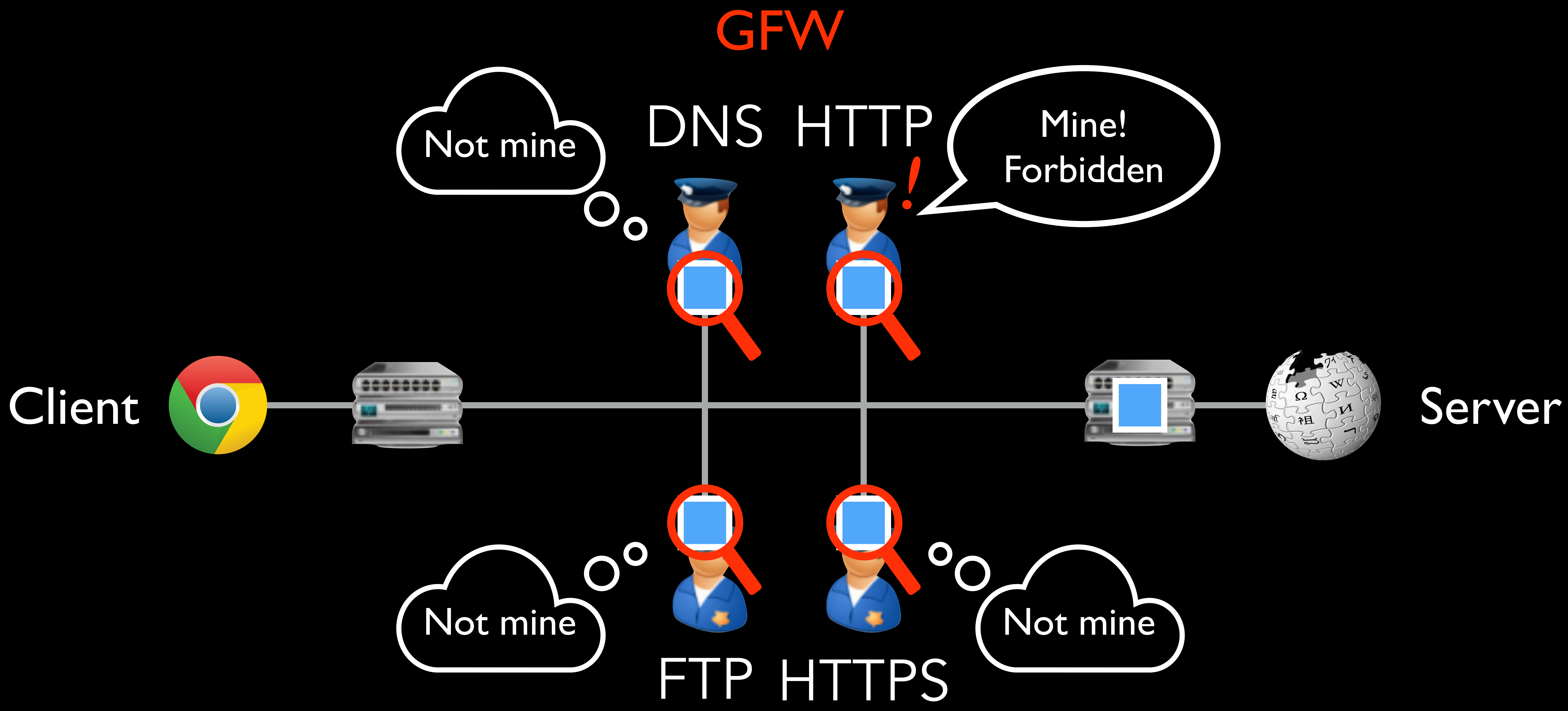
Censors effectively on any port

Multi-box theory



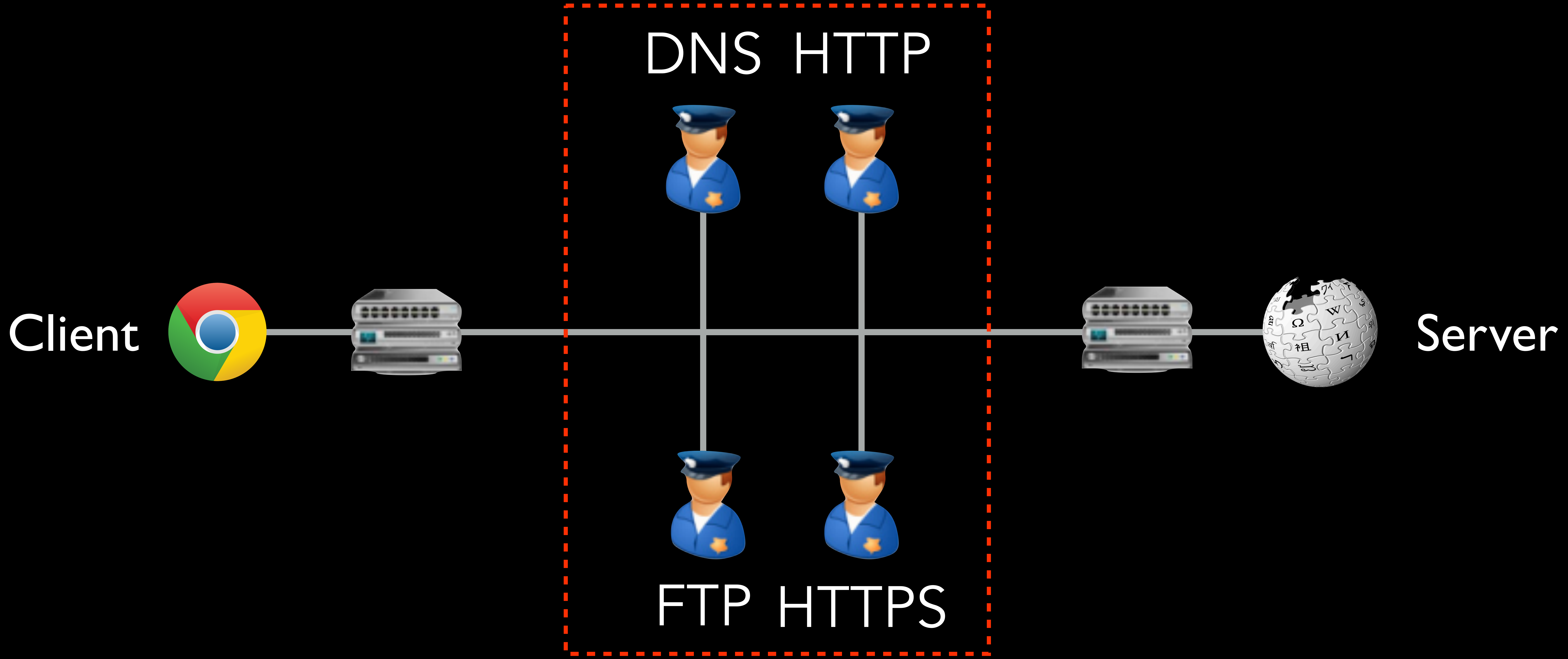
Applies protocol fingerprinting

Multi-box theory



Applies protocol fingerprinting

Where are these middleboxes?



Used TTL-limited probes

Co-located at the network level

Geneva

Genetic Evasion

1 Automate the discovery of new evasion strategies

2 Use the strategies to understand how the censor works

What's next?

New insight into how censors work

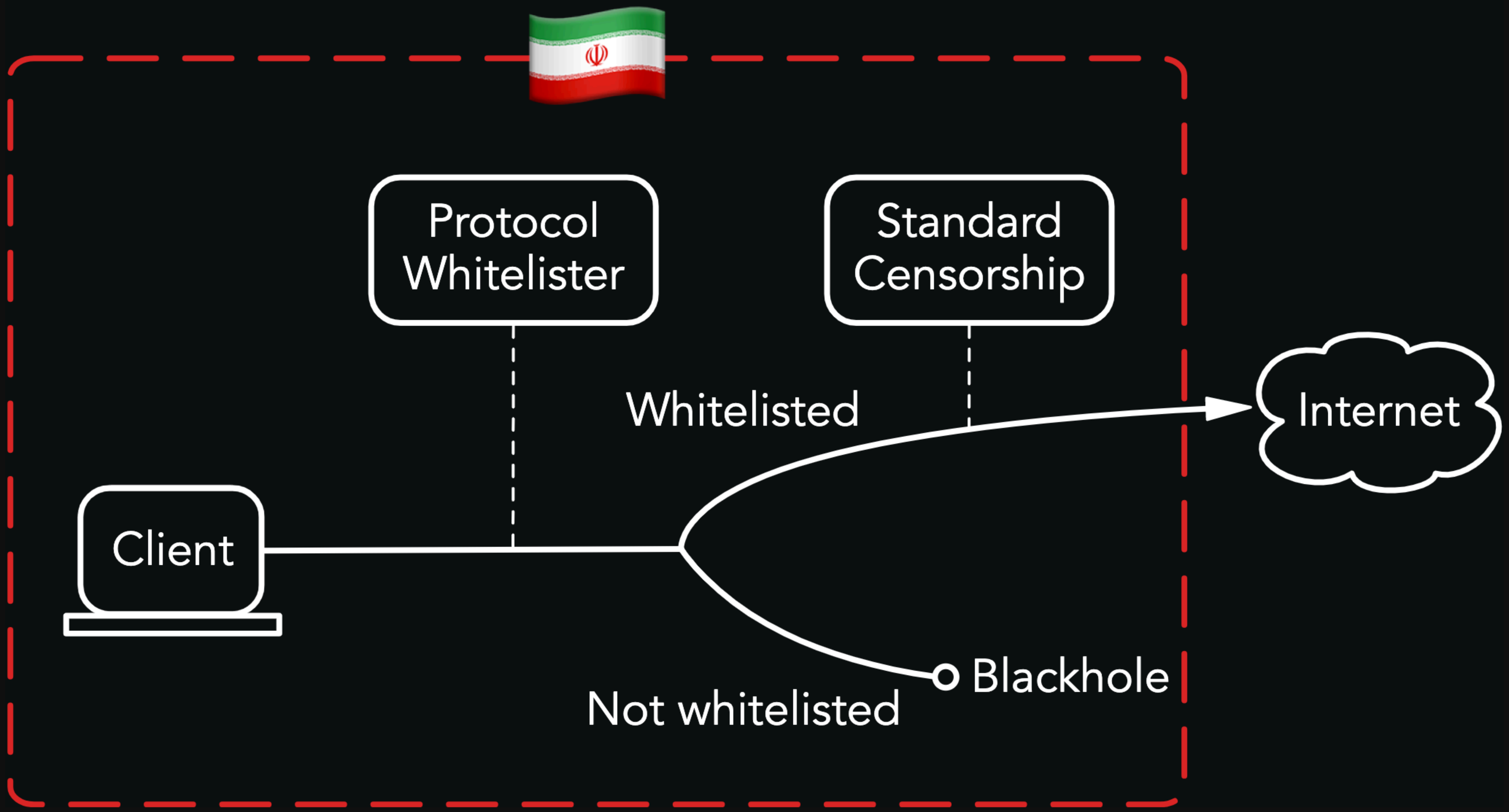
- Success rate changes by protocol
- "Multi-box theory"

Rapid response to new censorship

- Iran's new protocol filter (Feb 2020)
- China's new ESNI filter (July 2020)

Responsive to new censorship events

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

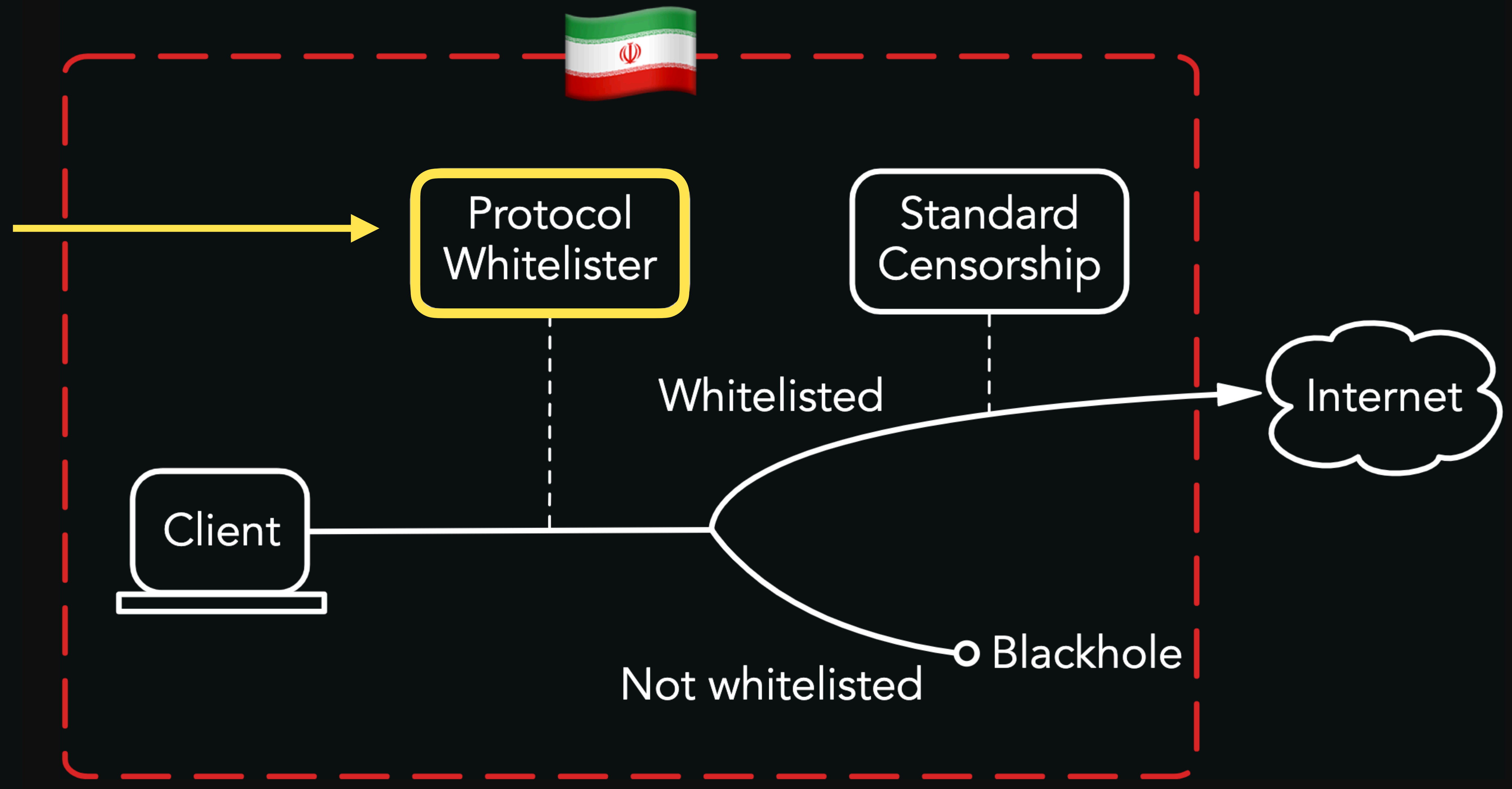


Responsive to new censorship events

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

Censors connections that do not match protocol fingerprints

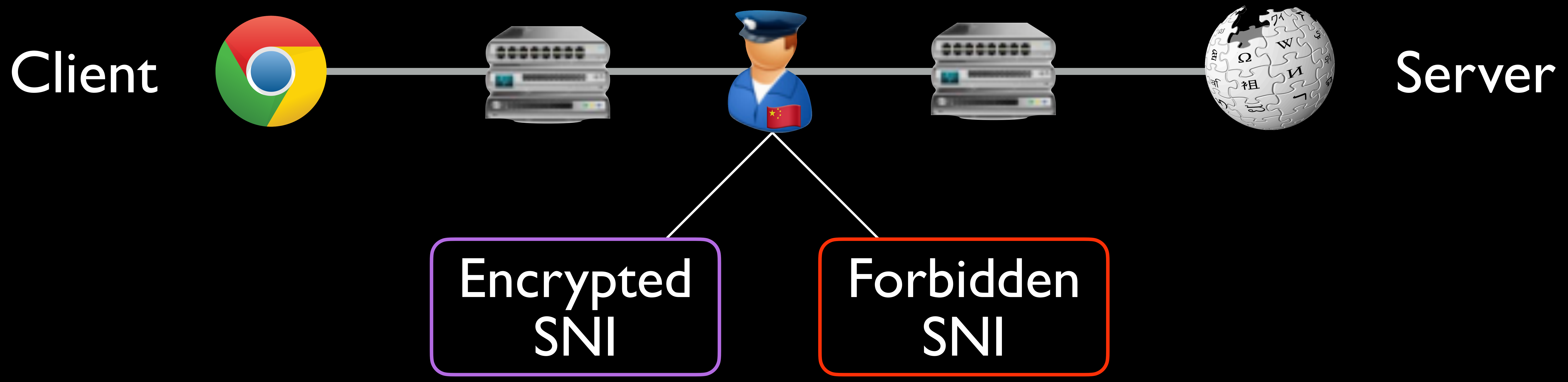
Those that do match are then subjected to standard censorship



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



Geneva discovered 6 strategies to evade ESNI censorship

Automating the arms race

AI has the potential to fast-forward the arms race *for both sides*

Bugs in implementation	Easy for censors to fix the low-hanging fruit
Gaps in logic	Harder for censors to fix systemic issues

What is the *logical conclusion* of the arms race?

Automating Censorship Evasion

Geneva

Genetic Evasion

Discovers strategies quickly

New insights into GFW

Server-side evasion is possible

Code is open source

Geneva code and website geneva.cs.umd.edu