Geneva + Tunnelbear:
Packet Manipulation Evasion at Scale

Kevin Bock

Jules Mazur

UNIVERSITY OF MARYLAND

TunnelBear

This talk

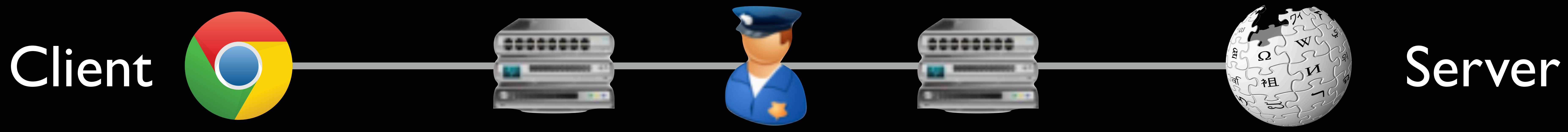
How does Geneva evade censorship?

- Packet manipulation-based censorship evasion
- Leveraging machine learning

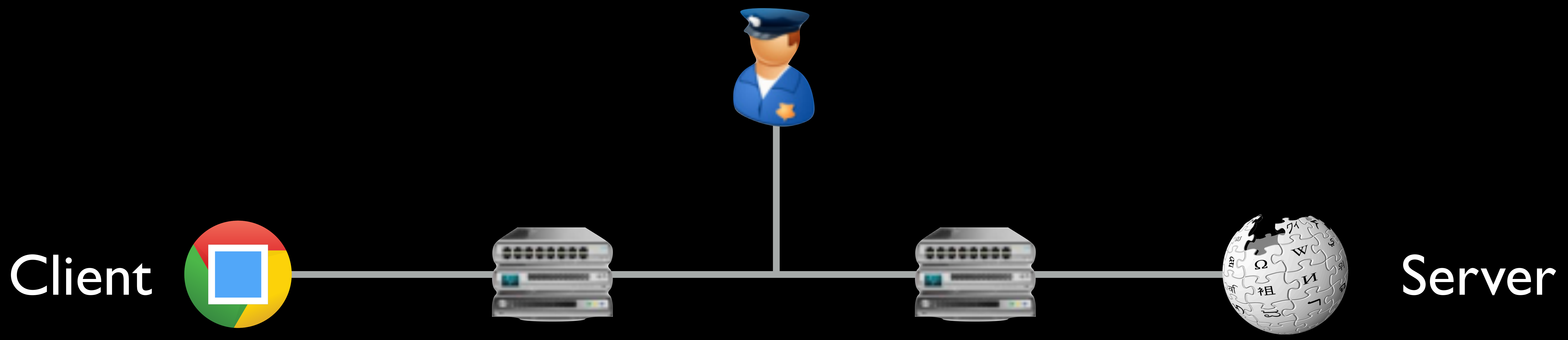
Deploying Geneva’s evasion strategies

- Running many strategies simultaneously
- Deployment despite modern networking complexities

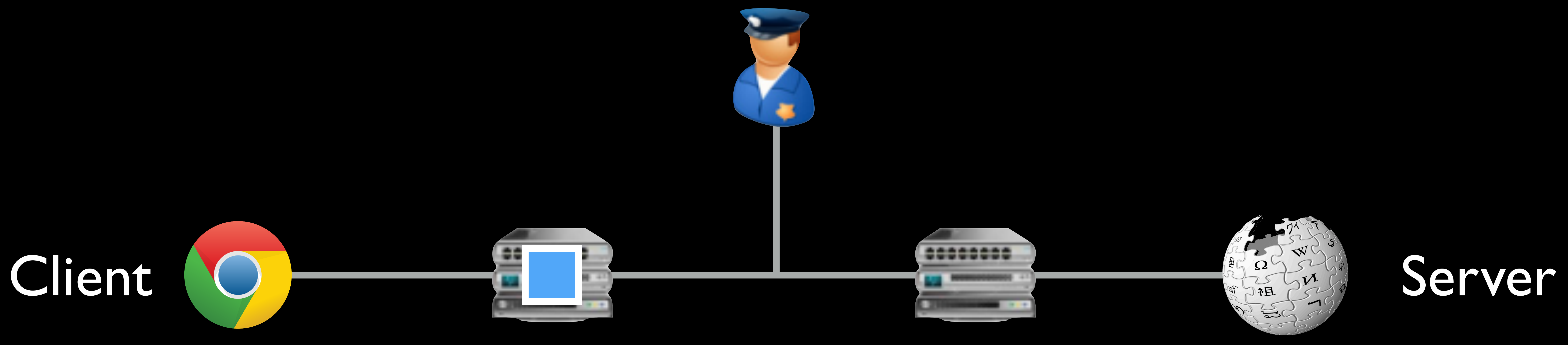
In-network censorship by nation-states



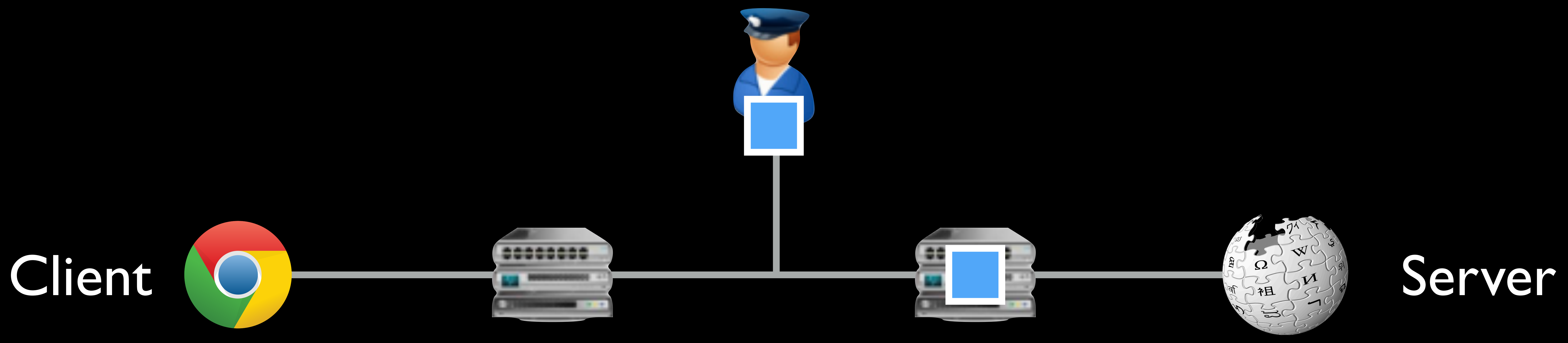
In-network censorship by nation-states



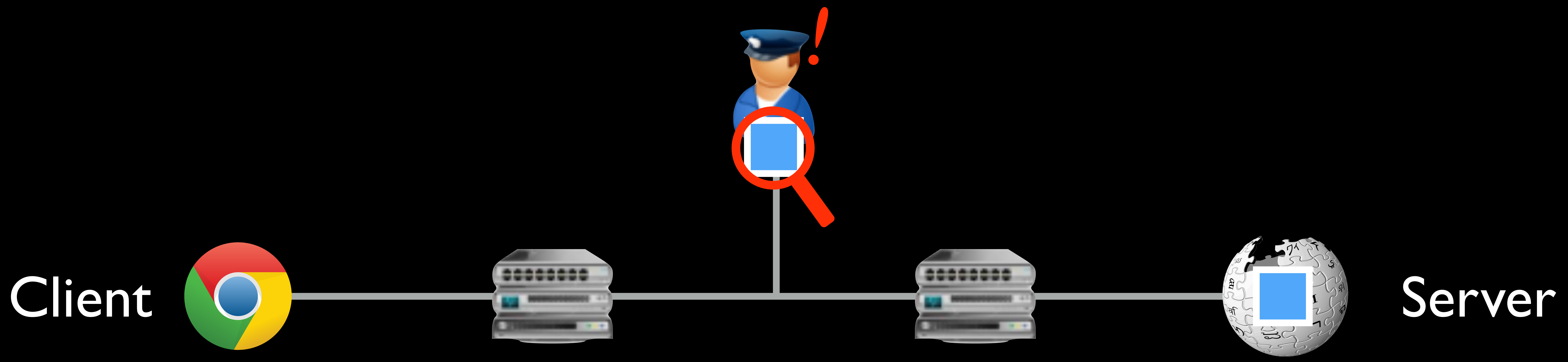
In-network censorship by nation-states



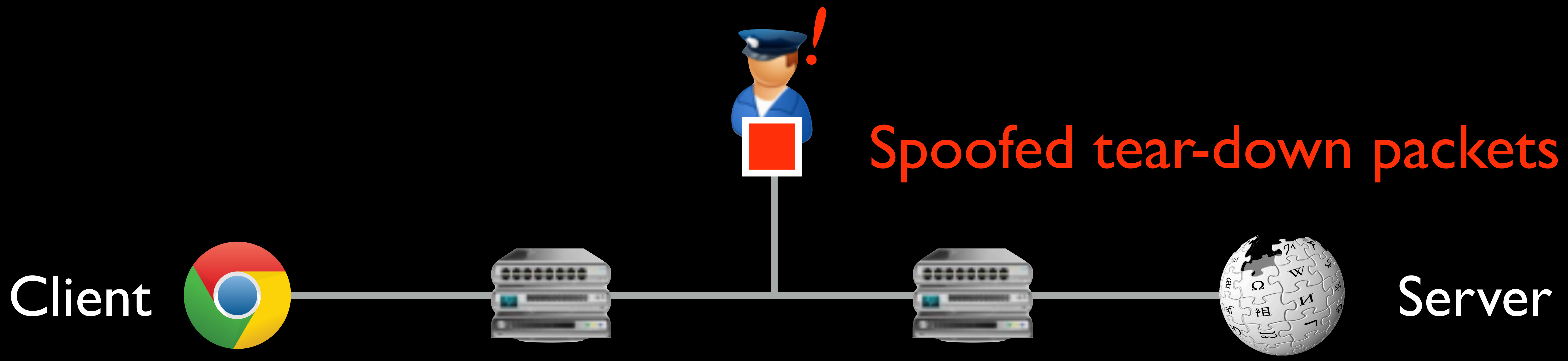
In-network censorship by nation-states



In-network censorship by nation-states

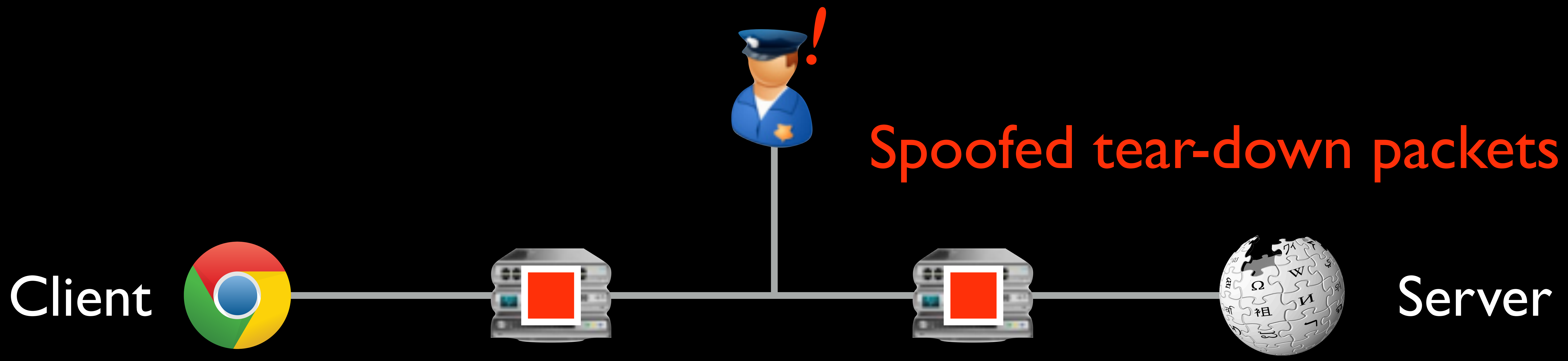


In-network censorship by nation-states

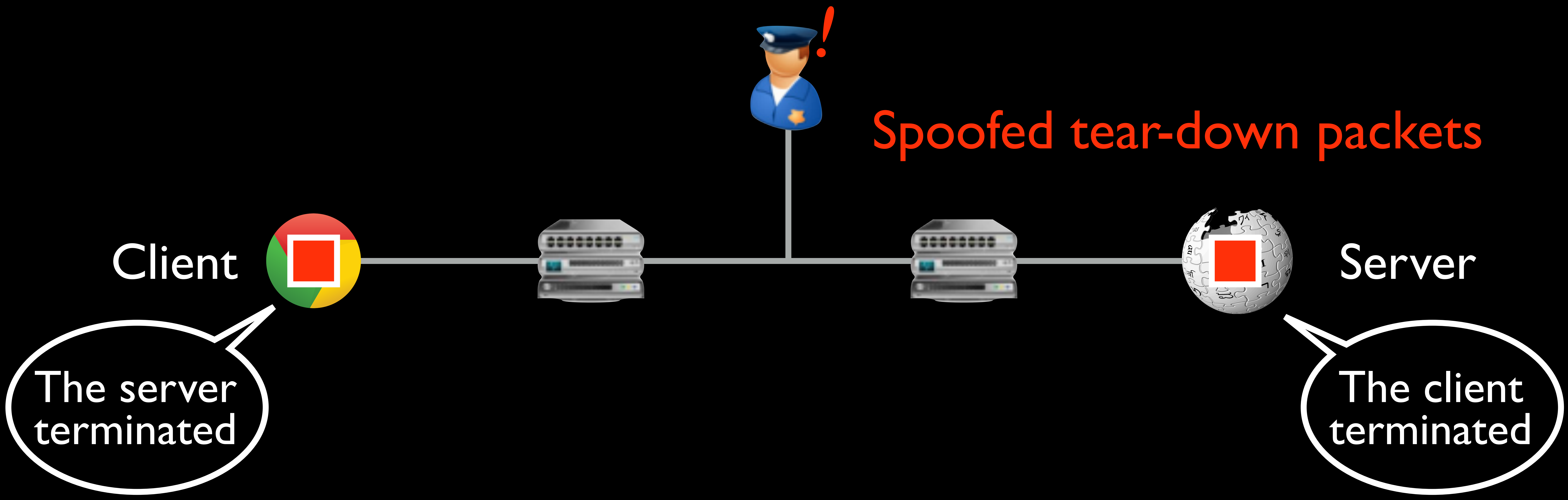


Spoofed tear-down packets

In-network censorship by nation-states

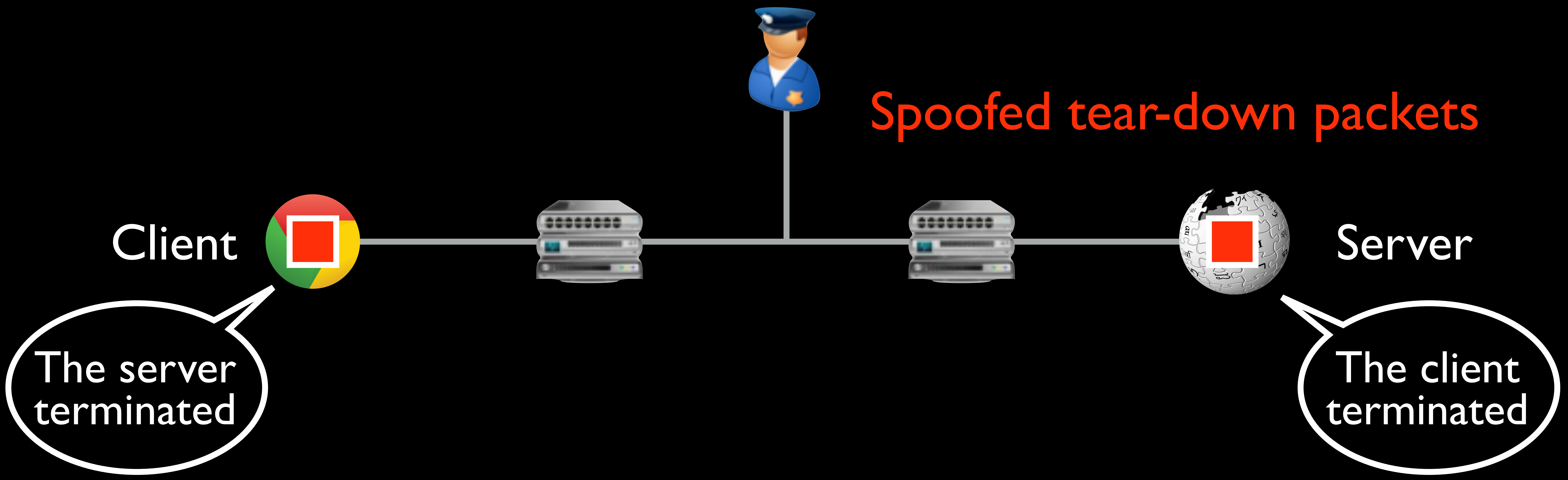


In-network censorship by nation-states



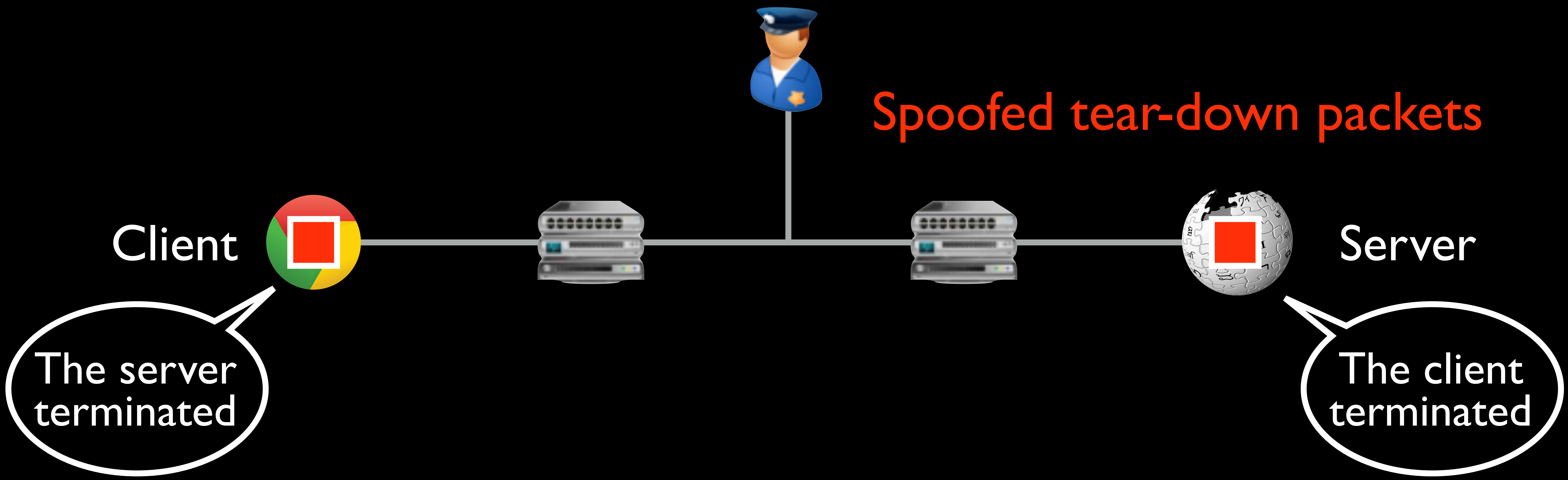
Injecting tear-down packets

In-network censorship by nation-states



Injecting tear-down packets

In-network censorship by nation-states



Spoofed tear-down packets

Client

Server

The server terminated

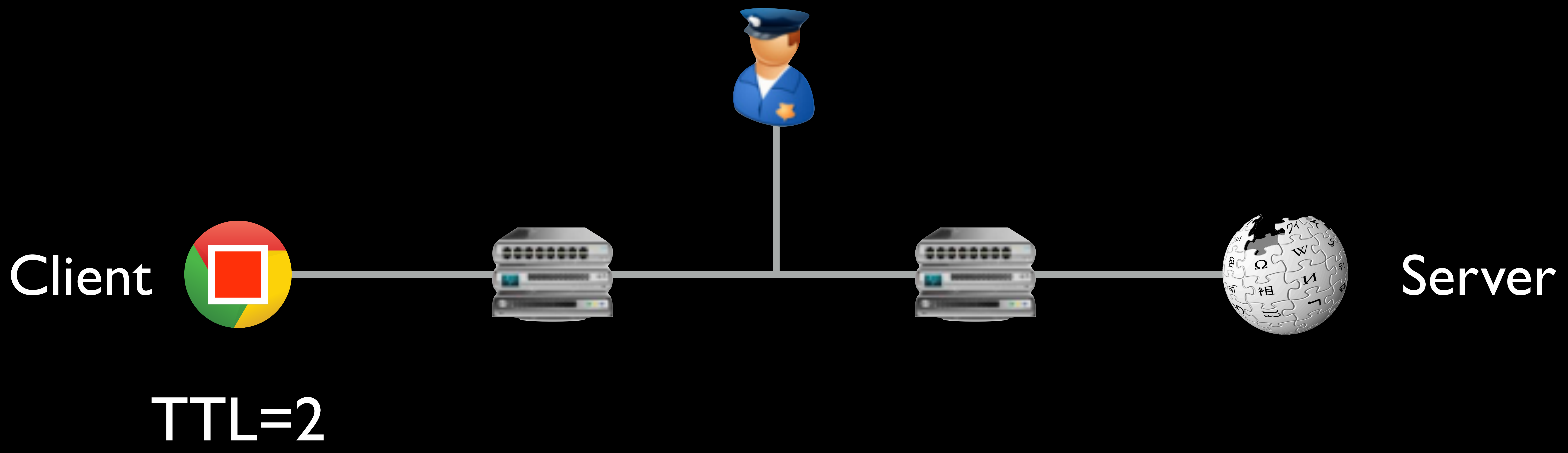
The client terminated

Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

In-network censorship by nation-states



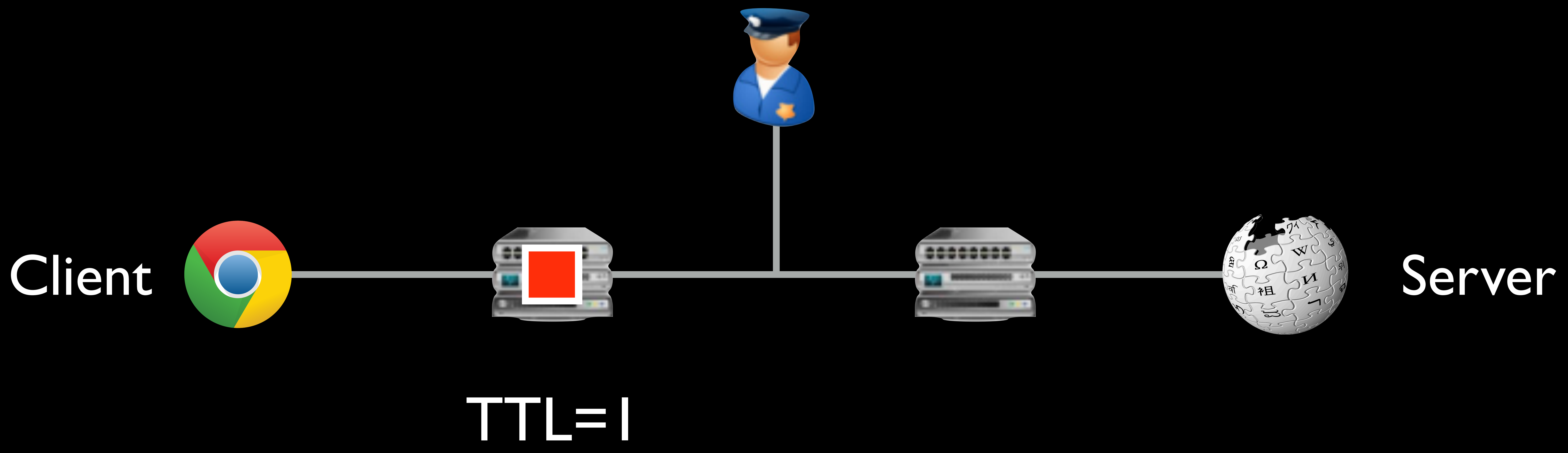
Injecting tear-down packets

Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

In-network censorship by nation-states



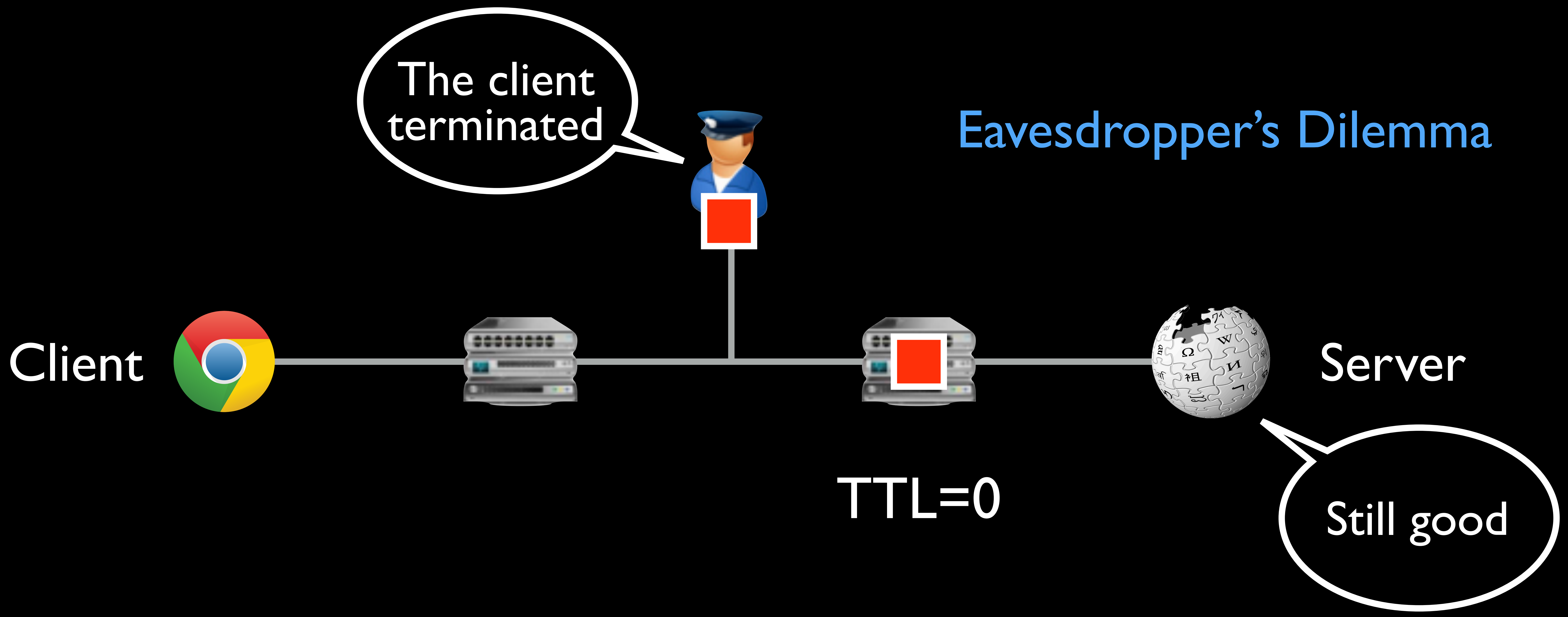
Injecting tear-down packets

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

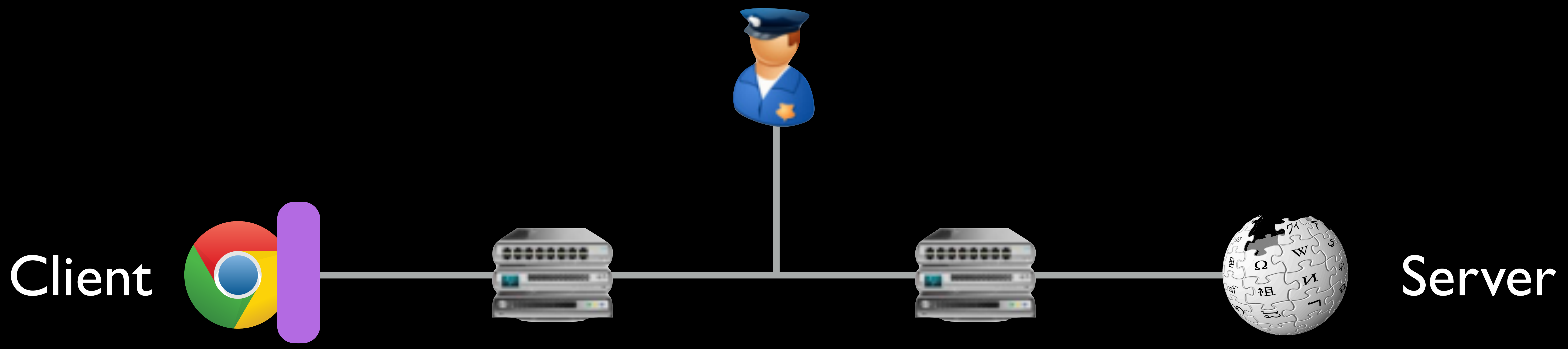
Requires *per-flow state*

Censors necessarily *take shortcuts*

Evasion can take advantage of these shortcuts

Geneva

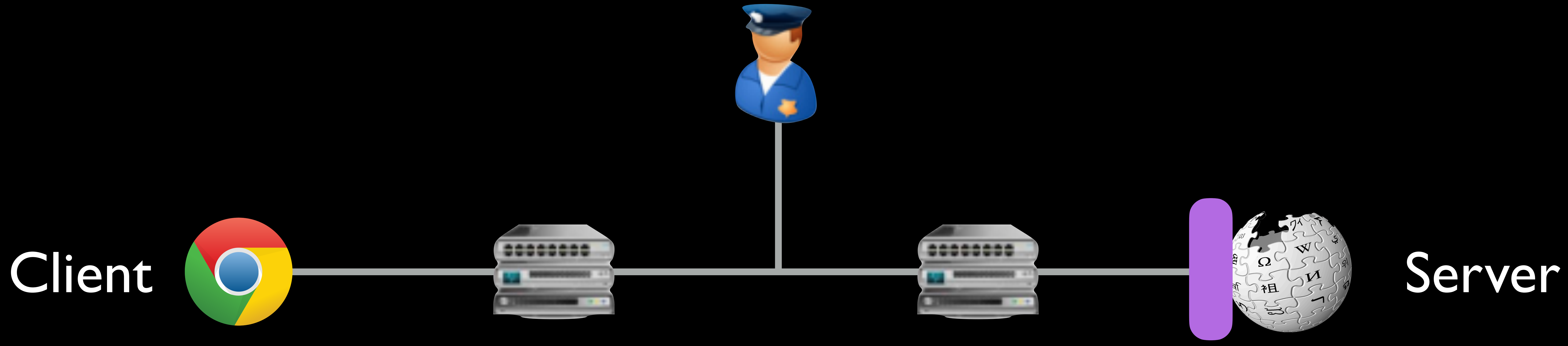
Genetic Evasion



Geneva runs strictly at one side

Geneva

Genetic Evasion



Geneva runs strictly at one side

Manipulates packets as they enter and leave

Geneva code and website censorship.ai

This talk

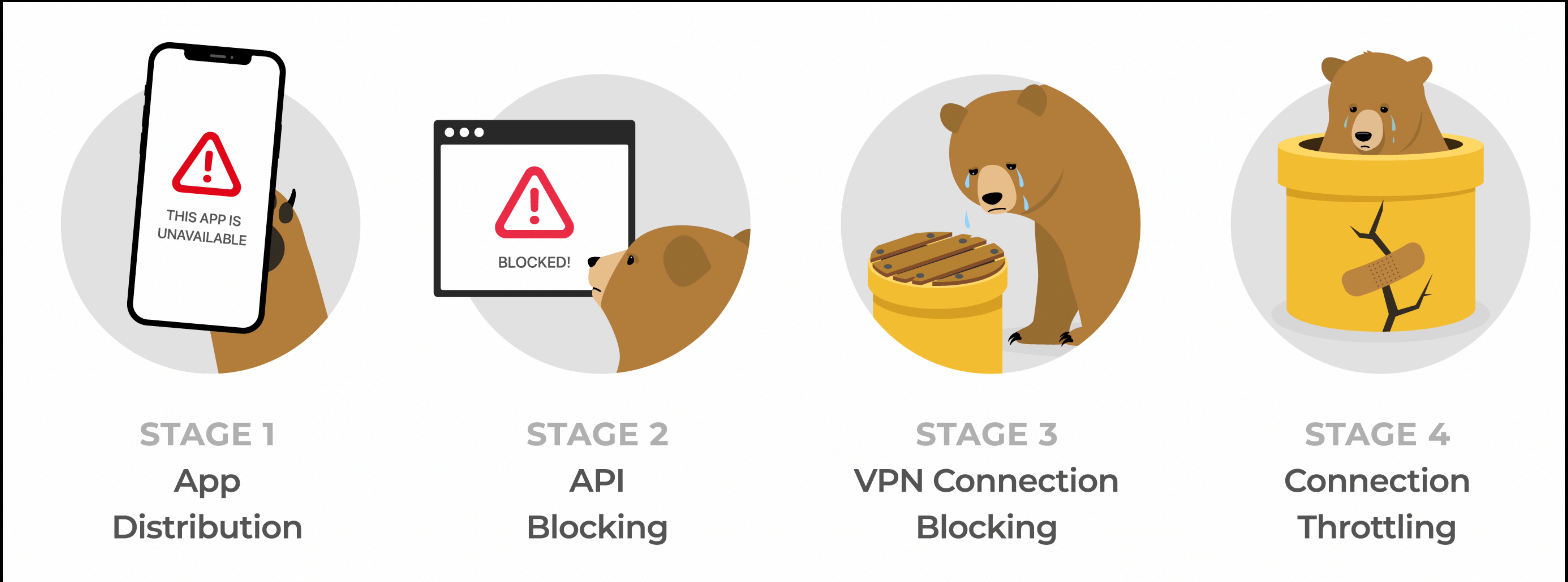
How does Geneva evade censorship?

- Packet manipulation-based censorship evasion
- Leveraging machine learning

Deploying Geneva's evasion strategies

- Running many strategies simultaneously
- Deployment despite modern networking complexities

Geneva at Tunnelbear



STAGE 1
App Distribution

STAGE 2
API Blocking

STAGE 3
VPN Connection Blocking

STAGE 4
Connection Throttling

Deploying Geneva and its challenges

- Work started in fall 2020 with a refactor of the Python project
- Most of the work has been around deployment strategies
  - Terraform
  - Dockerization
  - Deployment tests
- We hit many edge and corner cases in AWS and ECS along the way

What's next for Geneva at Tunnelbear

- Performance measurements and iteration from there
- Suspected hot sections of code have been instrumented
- Bottlenecks will dictate if and how parts of it should be rewritten
- Options being considered:
    - Cython
    - `iptables` extension
    - Kernel module

Packet Manipulation Evasion at Scale

Geneva

Genetic Evasion

+

TunnelBear

Large scale deployment at server-side

Rapid deployment of new strategies

Protects and enables bootstrapping

Geneva code and website censorship.ai