HTTP/2 Rapid Reset

IETF 118 – Prague – 2023-11

Lucas Pardue

CVE-2023-44487

Application-layer (layer 7) denial of service attack

Record-breaking levels observed by several cloud operators beginning August 2023

Cloudflare saw over 200 million RPS (3x previous record): https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack/

Attacks abuse a **protocol feature** to cause issues in **some** implementations or deployments

Not all implementations or deployments affected

HTTP Semantics and Syntax

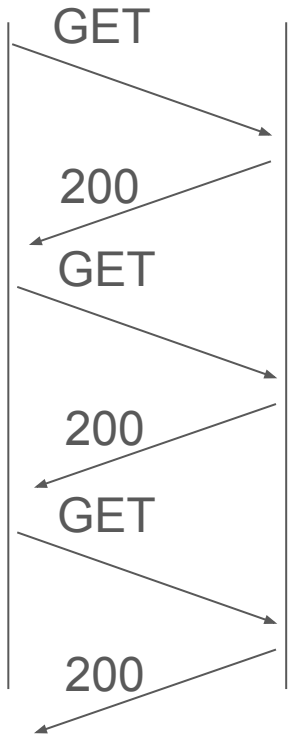
- HTTP is a request/response protocol
- Common **semantics** - RFC 9110, RFC 9111
- Different **wire formats**:
  - HTTP/1.1 - RFC 9112
  - HTTP/2 - RFC 9113
  - HTTP/3 - RFC 9114
- HTTP connections rely on a reliable transport connection underneath
  - TCP & TLS
  - QUIC

Message exchanges

- Clients send requests to servers
  - Method - GET, POST, etc.
  - Path - /index.html, /images/puppy.jpg
  - Host - www.example.org
  - Header fields - User-Agent, etc.
  - Optional Content - data described by other message parts
- Servers send responses to clients
  - Status code - 200, 302, 404, etc.
  - Header fields - Date, Server, etc.
  - Optional Content - data described by other message parts

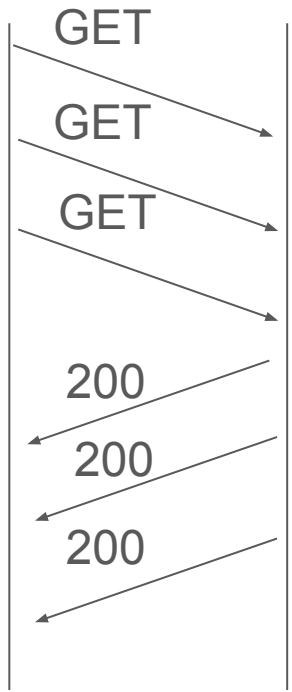
HTTP/1.1

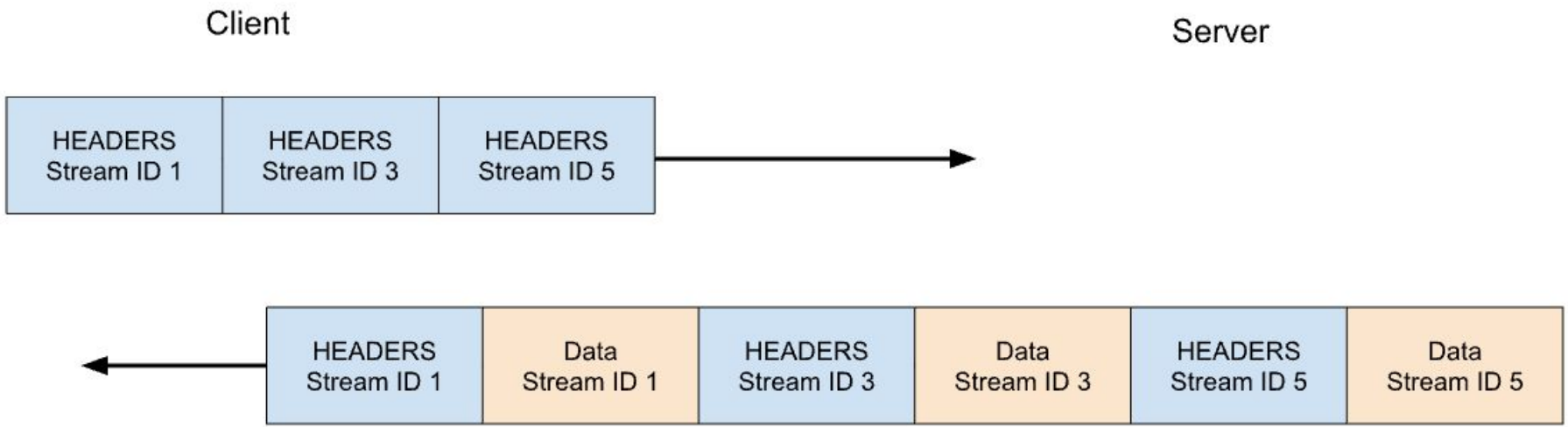
- A **single** HTTP/1.1 connection can be used for **multiple** request/response
- Strictly serial
- Request sent in whole …
- … Response sent in whole
- <repeat>



HTTP/2

- A **single** HTTP/2 connection can be used for **multiple** request/response
- Multiplexing and concurrency
- Divide the connection into **streams** with an ID
- Divide messages into **frames**
  - HEADERS for metadata
  - Optional DATA for content
- Frames sent over streams
- Multiplex frames in either direction!



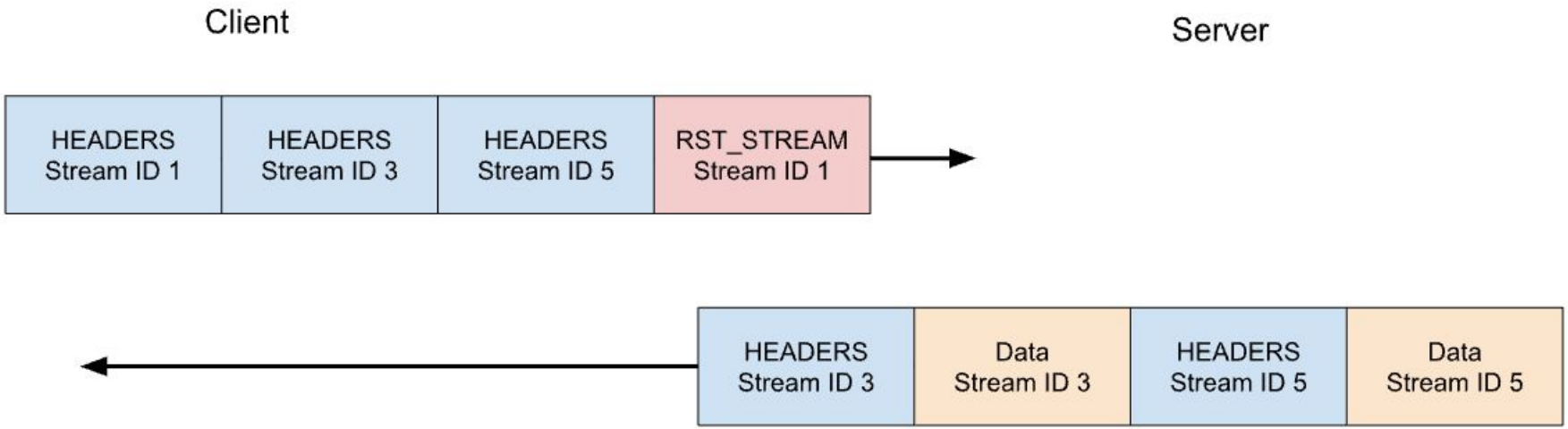

Client
Server
HEADERS
Stream ID 1
HEADERS
Stream ID 3
HEADERS
Stream ID 5
HEADERS
Stream ID 1
Data
Stream ID 1
HEADERS
Stream ID 3
Data
Stream ID 3
HEADERS
Stream ID 5
Data
Stream ID 5

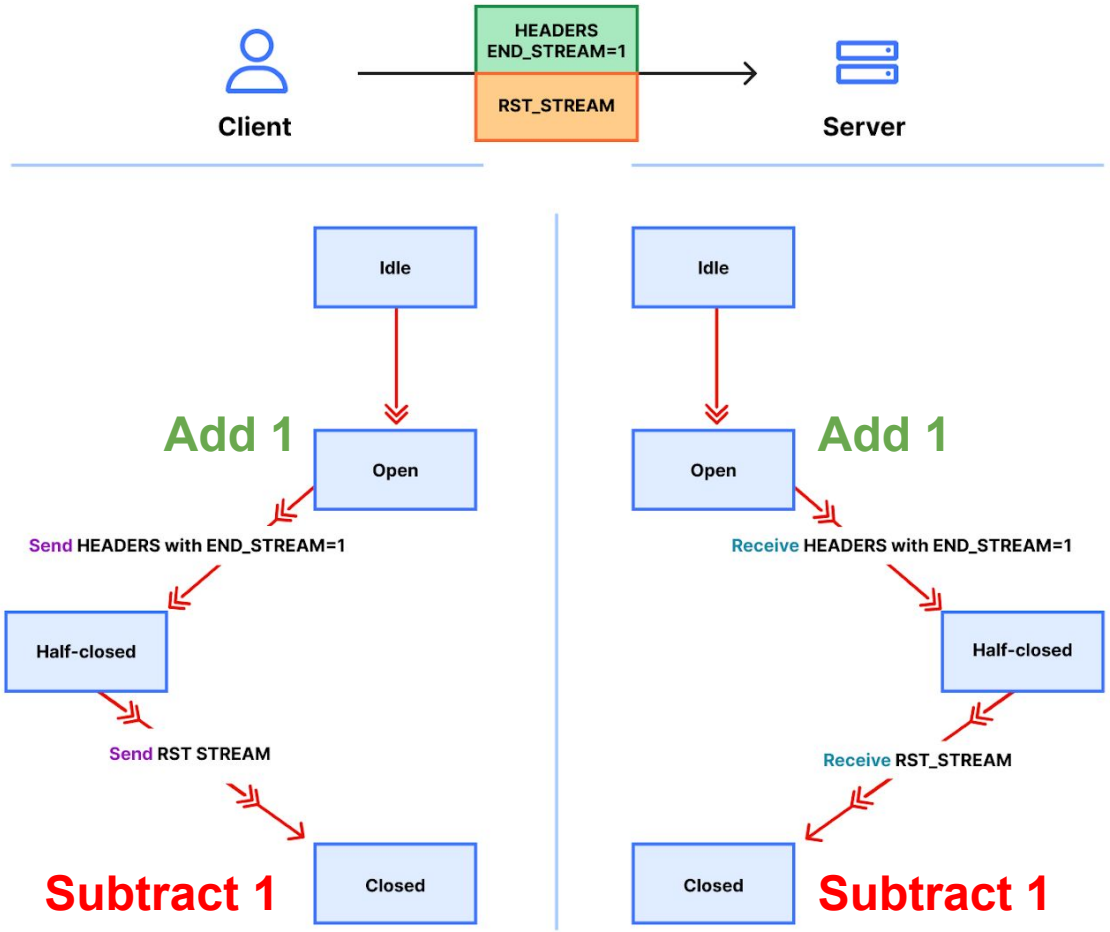
Concurrency and parallelisation

- HTTP requests cause a server to do work and allocate resources
- HTTP/2 allows a single connection to cheaply make multiple requests
  - Pros: performance
  - Cons: easy to generate more work and allocate more resources
- MAX_CONCURRENT_STREAMS setting
  - Limits the number of active requests
  - Streams are **opened** and count towards the limit
  - Streams are **closed** and **do not** count towards the limit

Cancelling requests

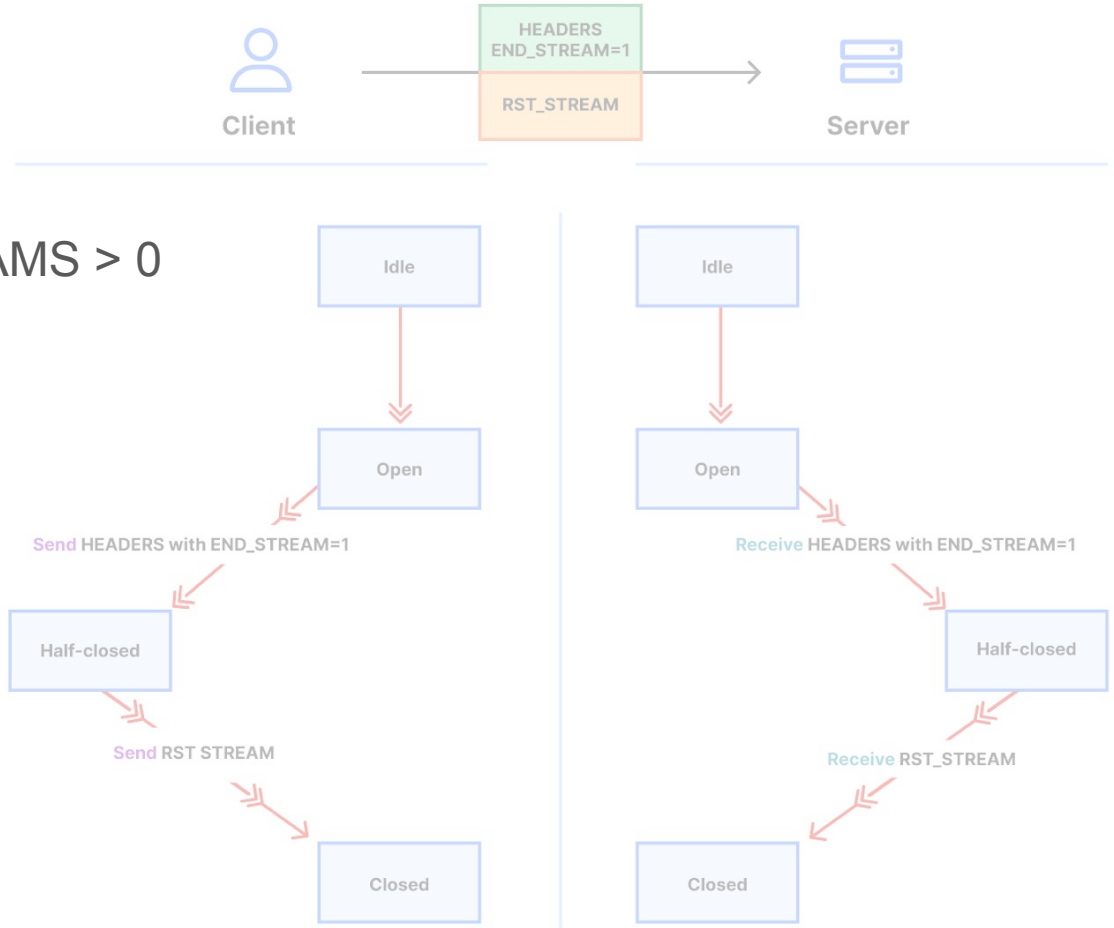
- Make a request for a large file
- No longer need it? Cancel it
- HTTP/1.1 requests are serial
  - So just terminate the HTTP connection
- HTTP/2 requests are parallel
  - Terminating the entire connection will affect all streams
  - RST_STREAM frame allows cancelling just one


Client
Server
HEADERS
Stream ID 1
HEADERS
Stream ID 3
HEADERS
Stream ID 5
RST_STREAM
Stream ID 1
HEADERS
Stream ID 3
Data
Stream ID 3
HEADERS
Stream ID 5
Data
Stream ID 5


Client
HEADERS
END_STREAM=1
RST_STREAM
Server
Idle
Add 1
Open
Send HEADERS with END_STREAM=1
Half-closed
Send RST STREAM
Subtract 1
Closed
Idle
Add 1
Open
Receive HEADERS with END_STREAM=1
Half-closed
Receive RST_STREAM
Closed
Subtract 1

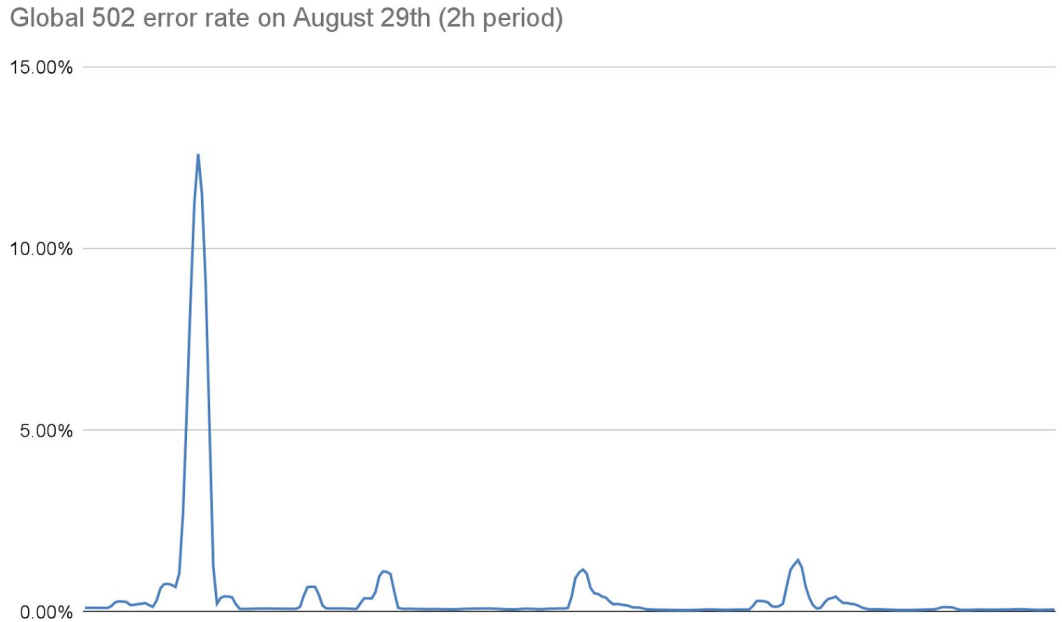
As long as MAX_CONCURENT_STREAMS > 0

Clients that immediately cancel a stream are never affected by concurrency limits



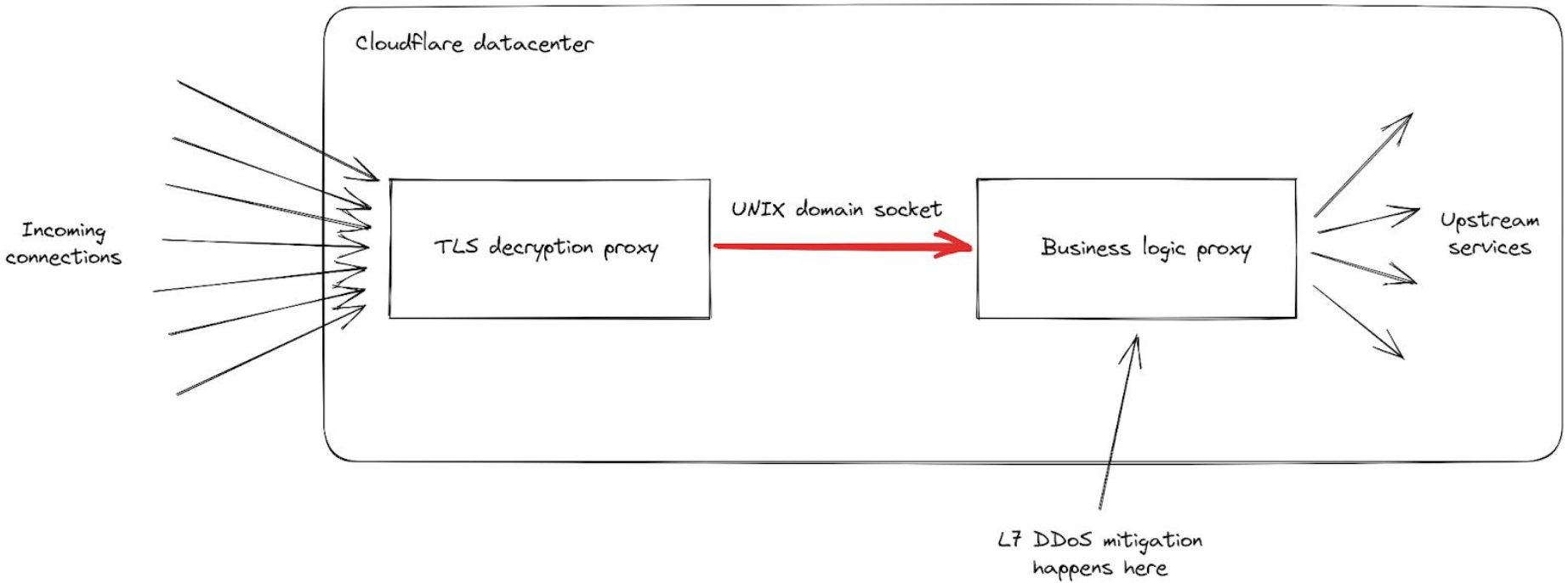
Rapid reset itself is not a problem

- Processing HEADERS to create request state and processing RST_STREAM to destroy state is not hard
- Yet, services saw overcommitment of state leading to denial of service



https://blog.cloudflare.com/technical-breakdown-http2-rapid-reset-ddos-attack/

System design and threat modelling



High-level Summary

- A protocol feature is working **as designed**
  - Originally published **2015**
- It can be abused to exploit characteristics of **some** implementations, or the deployments of those implementations
  - Mass-scale attacks **2023**
- The potential for this was discussed during standardization
- Discuss: Do formal methods provide an opportunity to do better?