

# Passthru: Protocol Omni-multiplexer

CSCI-ECEN 5273 Network Systems, 2022 Fall

Gaukas Wang, Parker Morris, Sharon Moses Jangam

University of Colorado Boulder

# Special Thanks

- Thanks to Bailey
  - for switching presentation timeslot with us!

# Agenda

- Background
- Original Problem – Inefficiency in proxying TLS traffics
- A smarter way – Passthru
  - Design
  - Implementation
  - Application Scenario

# Background

- Reverse Proxy
  - Sits in front of back-end application servers and forward requests
- CDN
  - Geographically distributed networks of (reverse) proxy servers
- International Private Leased Circuit (IPLC)
- International Ethernet Private Line (IEPL)

# Background

- Transport Layer Security (TLS)
  - CA-signed certificate-based cryptographic protocol
  - Integrity, Authenticity and Privacy - MITM attack prevented
  - The foundation of modern HTTP-over-TLS (<https://>)
  - Designed for TCP. UDP adaptations: DTLS(WebRTC), QUIC(HTTP3)

# Traditional ways to proxy TLS

- Reverse Proxy for a TLS service, e.g., HAProxy
  - 2 TCP Connections, 1 TLS Session
  - Port Mapping/Forwarding + optional Load Balancer



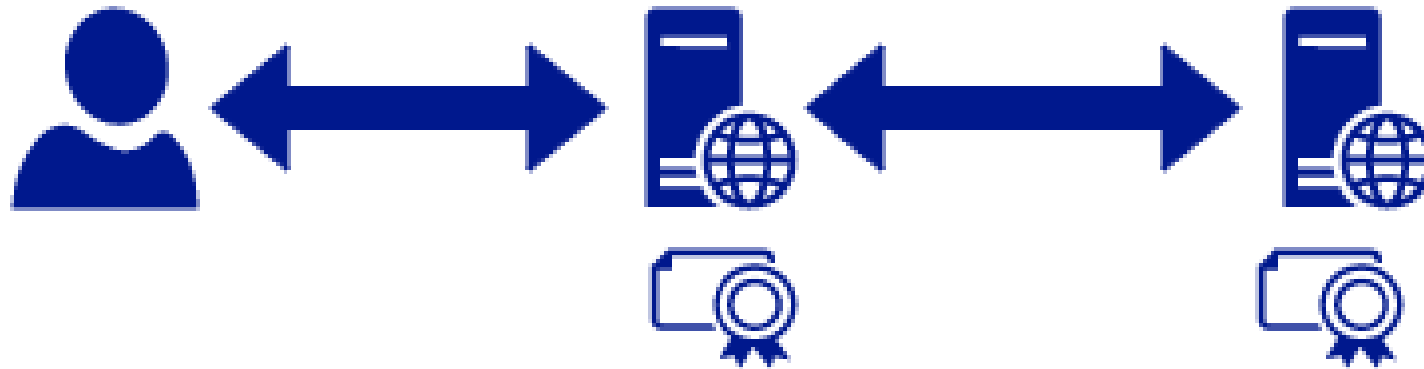
# Traditional ways to proxy TLS

- Reverse Proxy for a TLS service, e.g., HAProxy
  - 2 TCP Connections, 1 TLS Session
  - Port Mapping/Forwarding + optional Load Balancer
  - Basically, 1:1 NAT -- which is expensive



# Traditional ways to proxy TLS

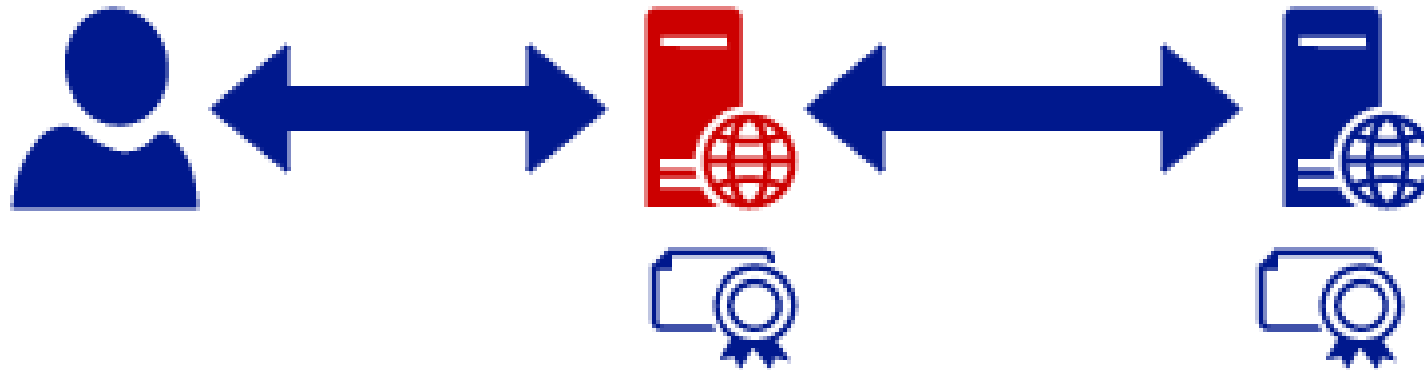
- CDN for a TLS Website (via HTTPS), e.g., Cloudflare
  - 2 TCP Connections, 2 TLS Sessions
  - Caching + Load Balancing + optional Attack Mitigation



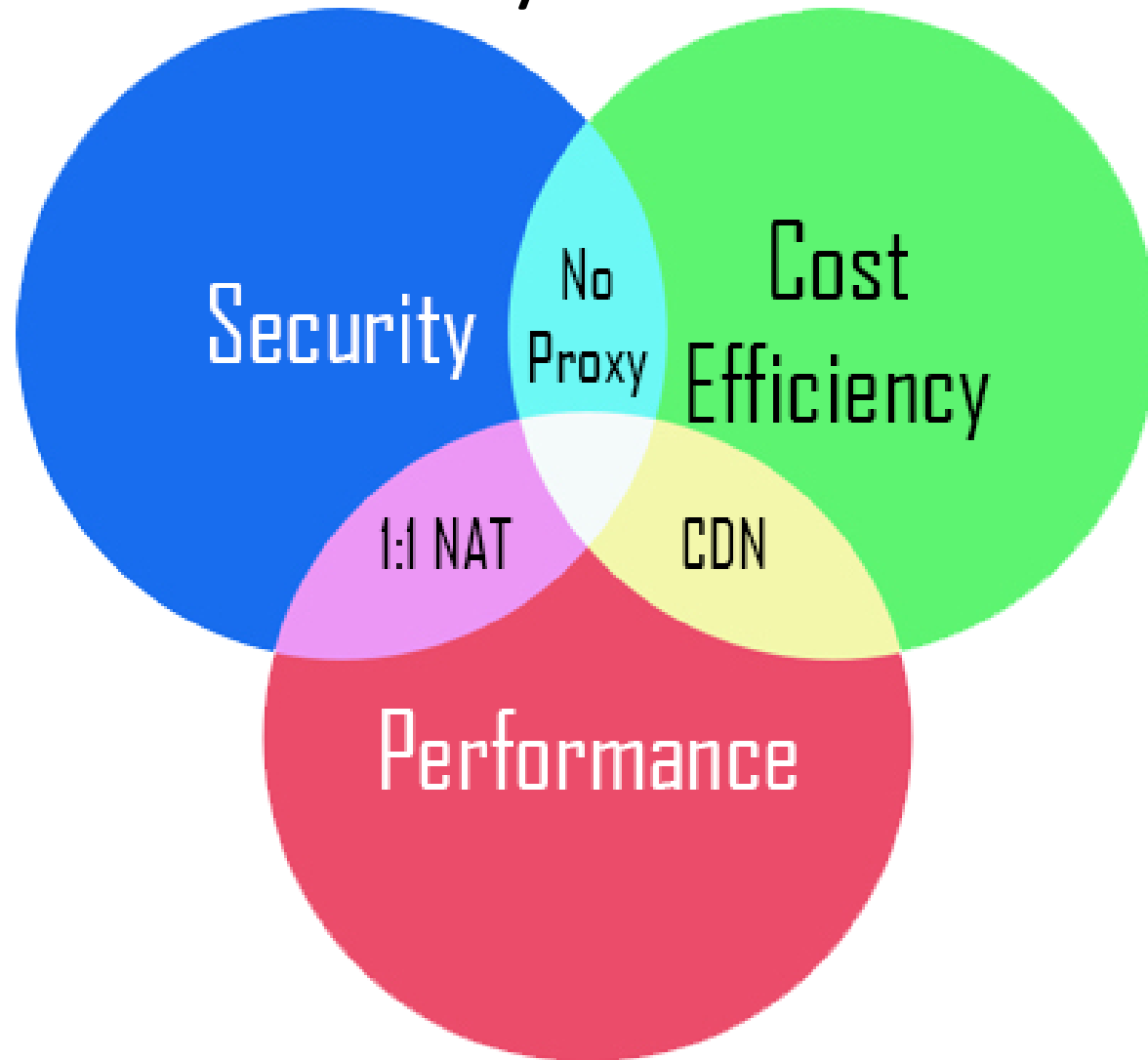


# Traditional ways to proxy TLS

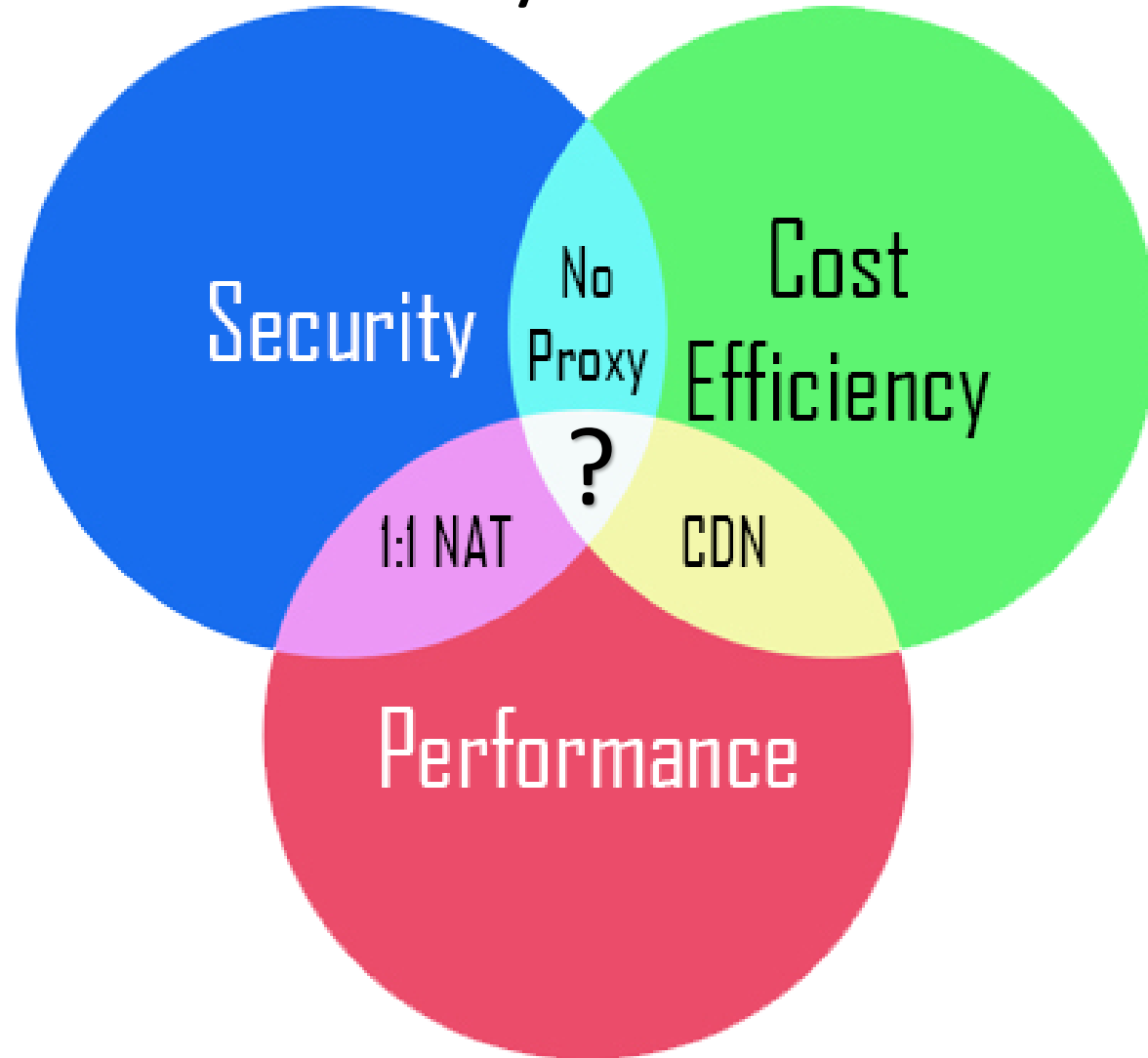
- CDN for a TLS Website (via HTTPS), e.g., Cloudflare
  - 2 TCP Connections, 2 TLS Sessions
  - Caching + Load Balancing + optional Attack Mitigation



# Trilemma: Security vs. Cost vs. Performance



# Trilemma: Security vs. Cost vs. Performance



# A smarter (yet simple) way

- Deep Packet Inspection (DPI)
- Assumptions
  - Protocols MUST be distinguishable before a server responds
  - Protocol Identifications MUST be deterministic and exclusive
  - Protocol identifications SHOULD happen at line speed

# Passthru

- Ad hoc DPI-based Protocol Omni-multiplexer
  - DPI: for Application Data Sniffing
  - Omni: Highly programmable/customizable
  - Protocols are identified at Transport Layer (or lower)
- Prototype
  - Unprivileged user space application at Transport Layer written in Go
  - Detects TCP Protocols only
  - Example: TLS Protocol Detection Module
  - Allows custom pluggable Detection Modules to be added by user

# Definition: Objects

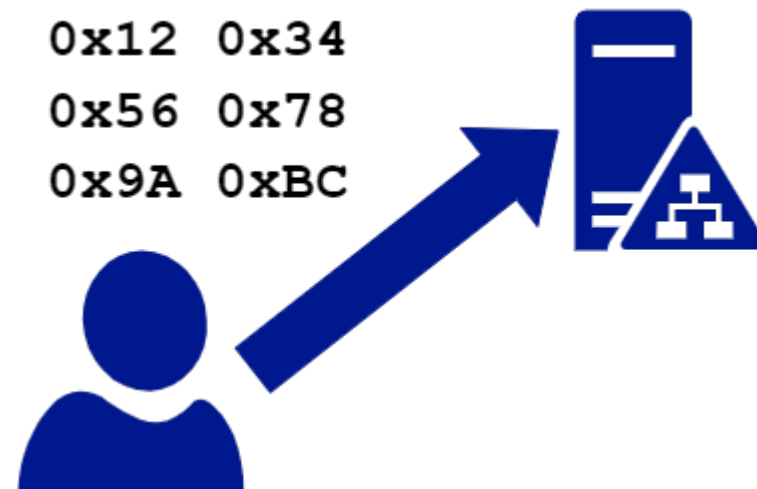
- `Server`
  - Accepts incoming connections
  - Dials outgoing connections
  - Relays data between clients and destination hosts
- `Protocol`
  - Performs DPI on incoming bytes
  - Tells `Server` if matching



# Design



# Design





# Design



0x12 0x34  
0x56 0x78  
0x9A 0xBC

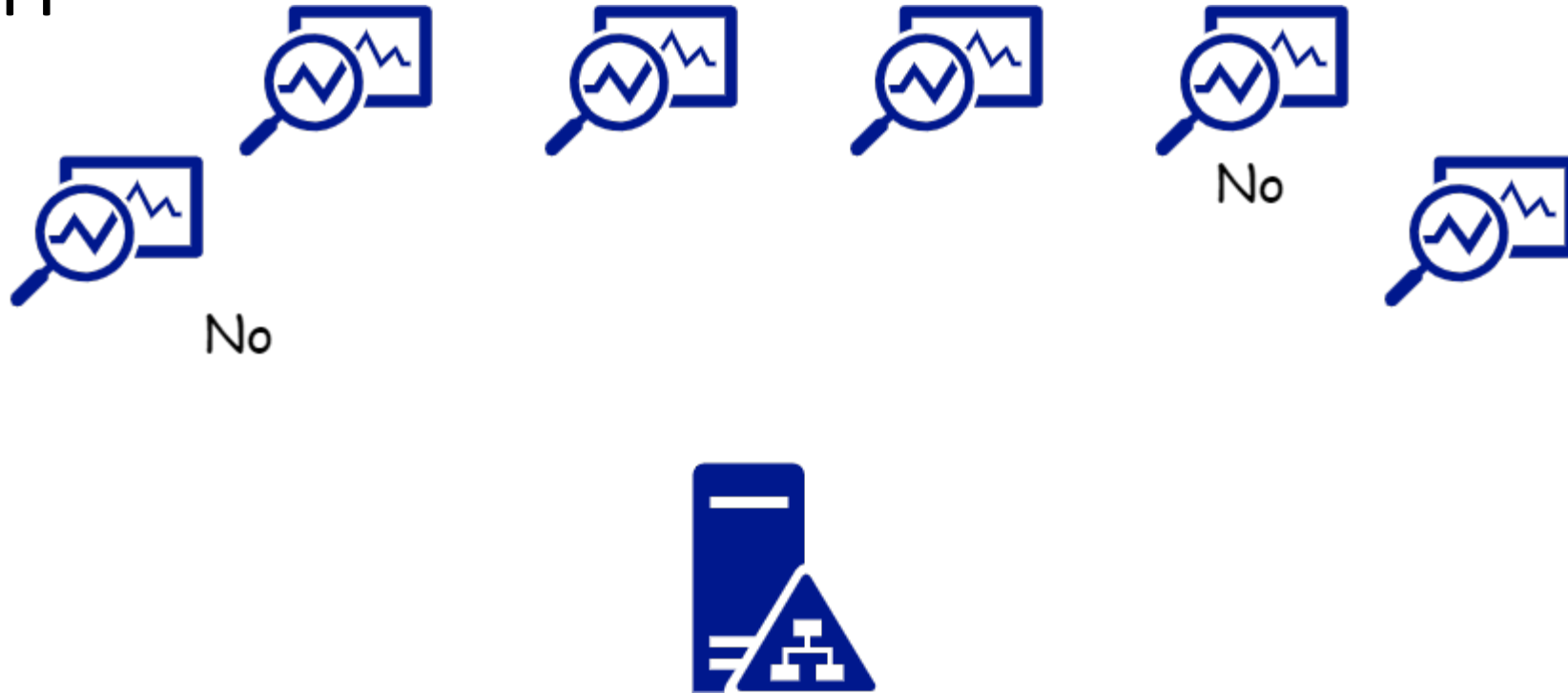


Anyone recognize this?  
If so, tell me: what to do?

# Design



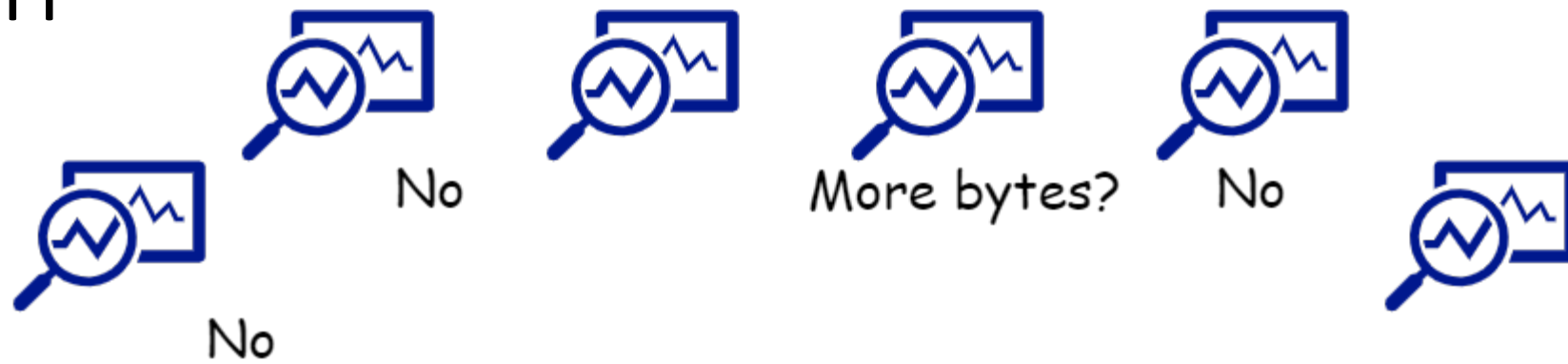
# Design



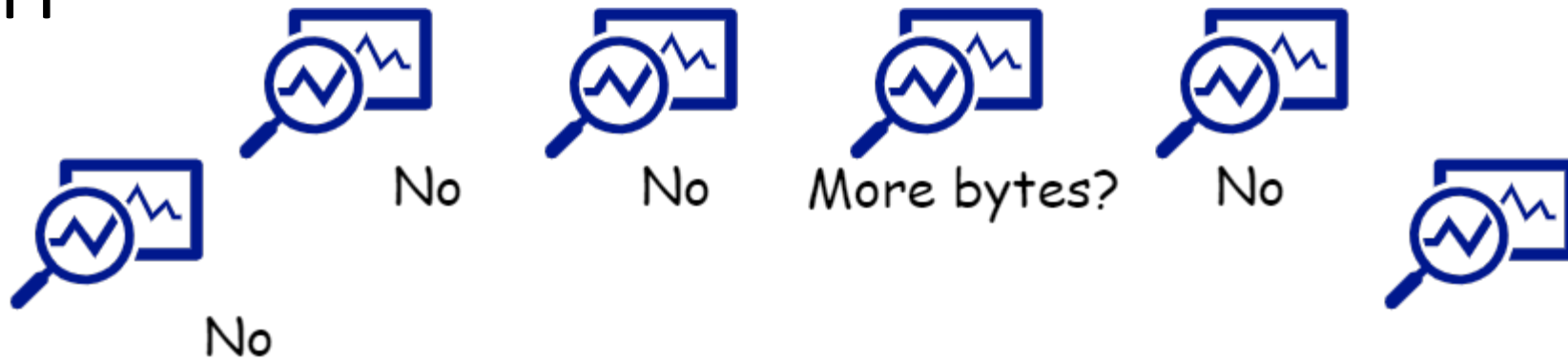
# Design



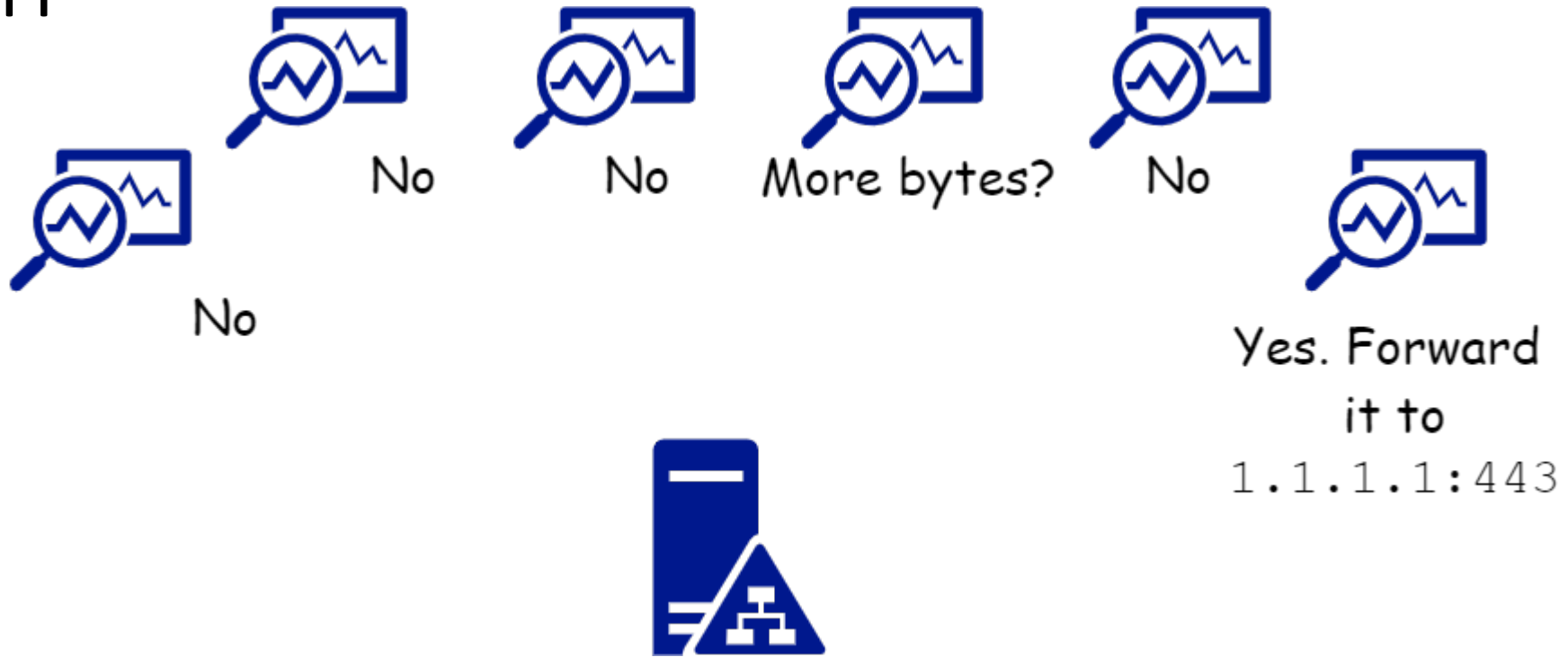
# Design



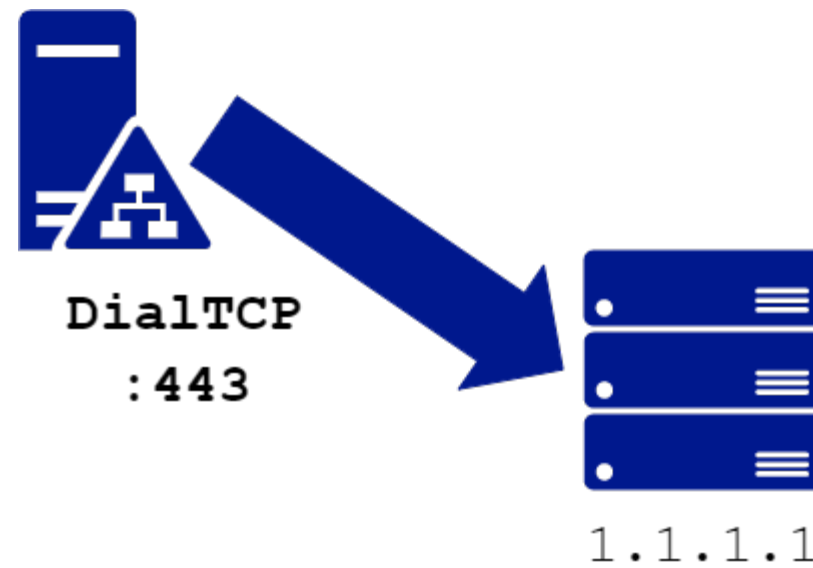
# Design



# Design

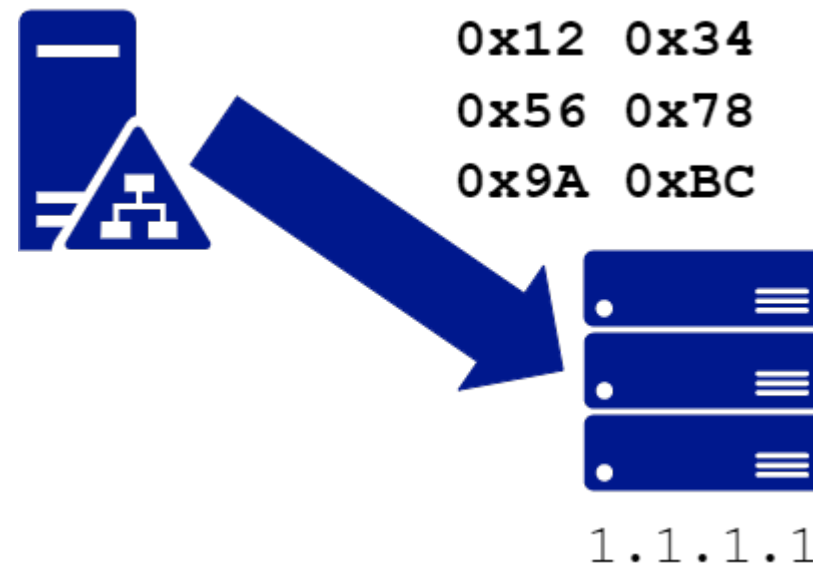


# Design

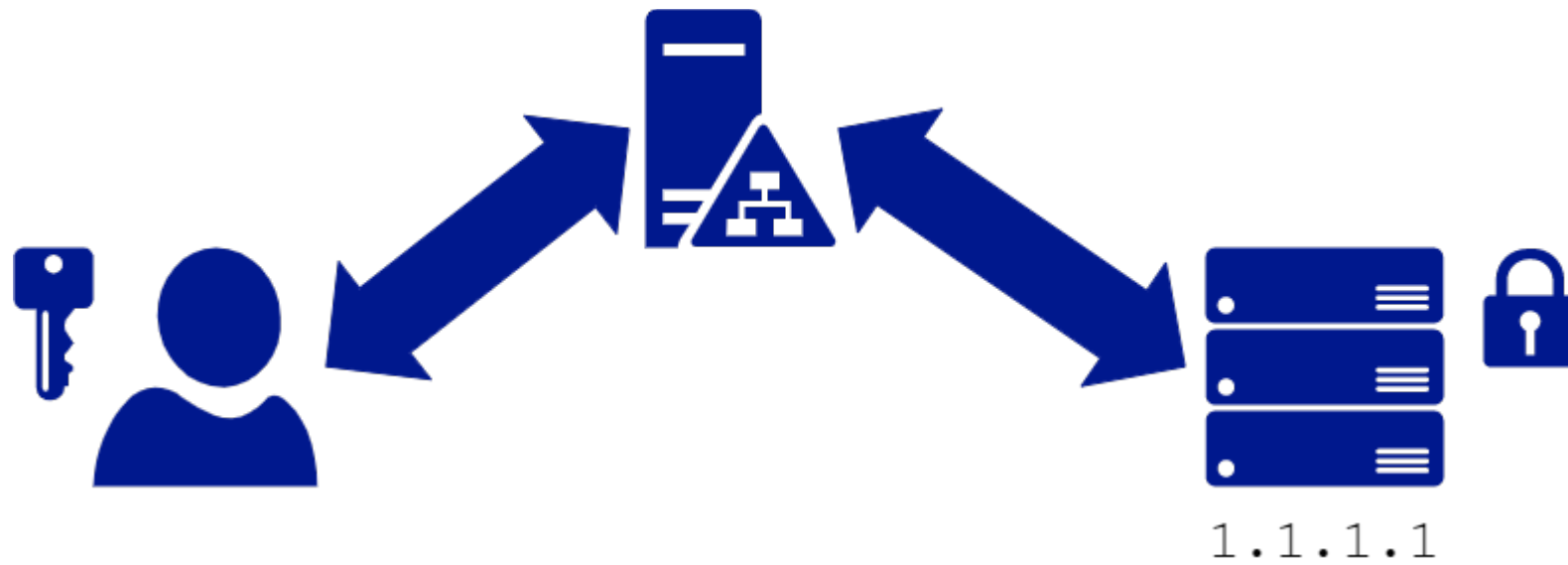




# Design

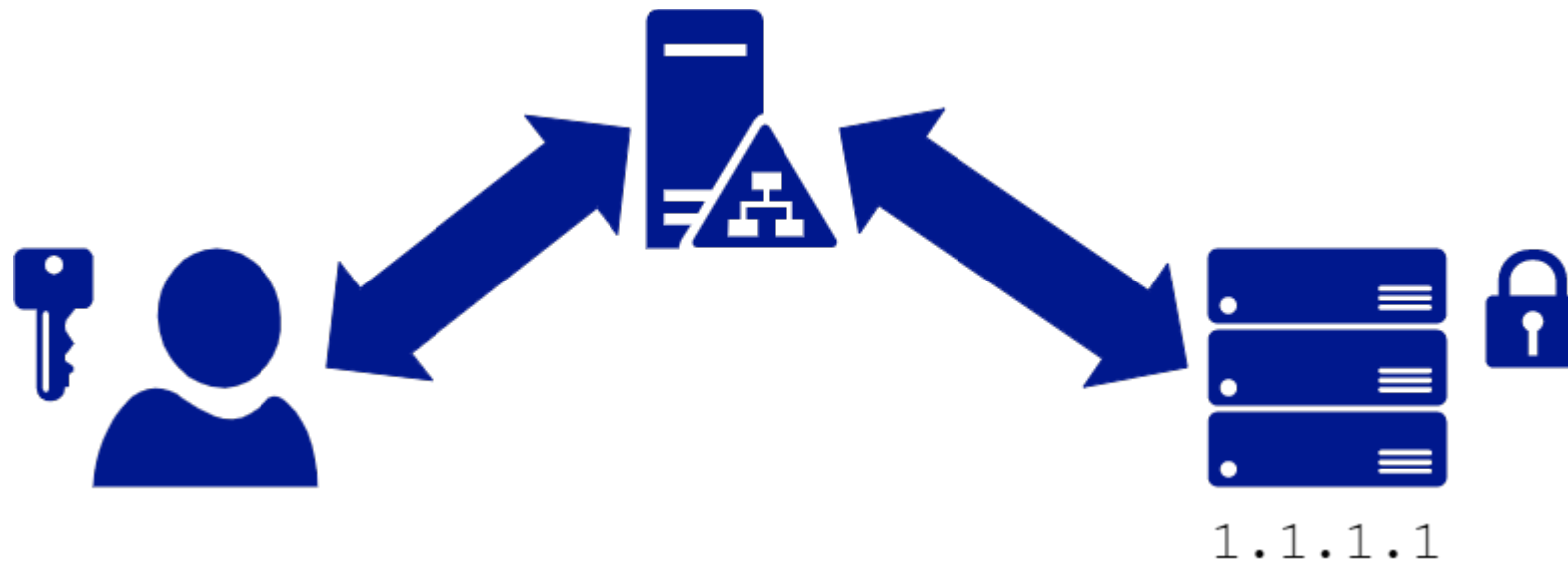


# Design



# Design

End-to-End Encryption Available!



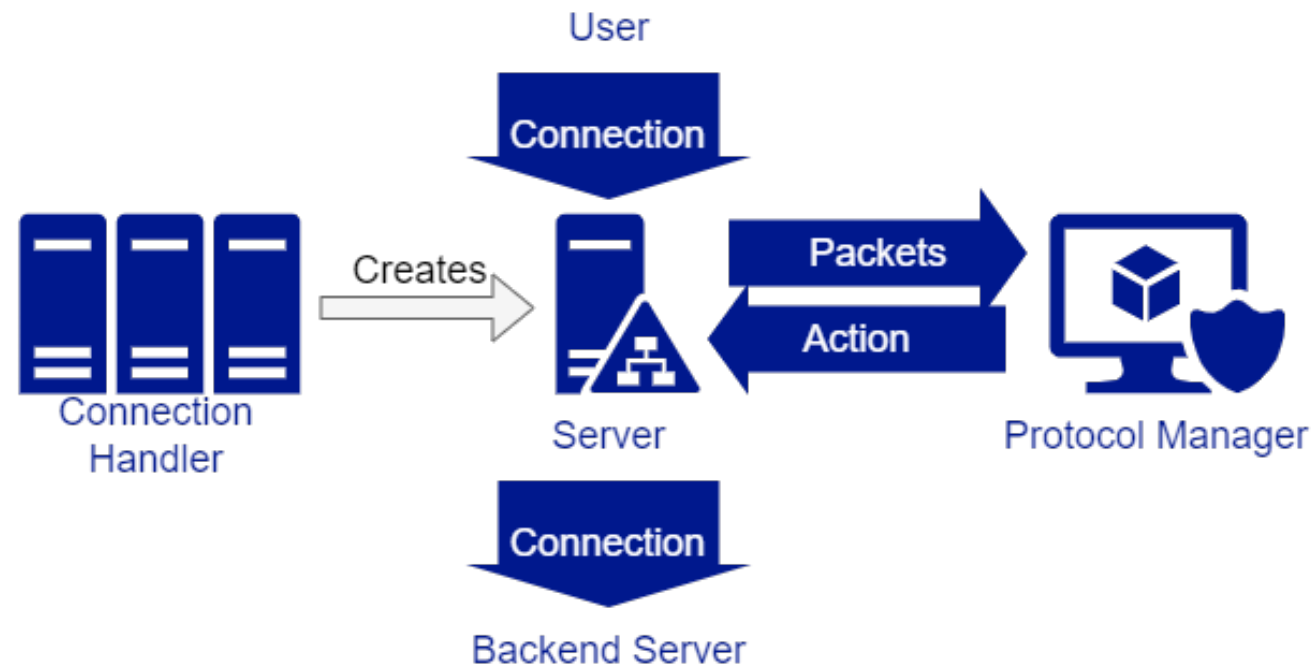
# Implementation

- Configuration File
  - Standard JSON Format
- Config File Structure
  - Version
  - Server Group
    - Server
      - Protocol
        - Rules
          - Action

```
1  {
2      "version": "v0.2.0",
3      "servers": {
4          "127.0.0.1:443": {
5              "TLS": {
6                  "SNI_gaukas.wang": {
7                      "action": "FORWARD",
8                      "to_addr": "185.199.111.153:443"
9                  },
10                 "APLN_h2": {
11                     "action": "FORWARD",
12                     "to_addr": "1.1.1.1:443"
13                 },
14                 "CATCHALL": {
15                     "action": "REJECT"
16                 }
17             },
18             "CATCHALL": {
19                 "CATCHALL": {
20                     "action": "FORWARD",
21                     "to_addr": "neverssl.com:443"
22                 }
23             }
24         }
25     }
26 }
```

# Implementation

- Connection Handler
  - Creates `Server`, and...
  - Protocol Manager **per** `Server`



# Implementation

- Protocol Manager
  - Knows mapping from Rules to Actions
  - Protocols are configured with their Rules



# Application Scenario

- (Virtual) Gateway Integration
  - Enables IP-layer forwarding to Passthru to preserve Client IP
- Flexible Reverse Proxy/IPLC/IEPL Integration
  - Better utilizations of shared infrastructures
- Better Security
  - Hides services from probes and consequential attacks

# Recap

- Passthru: a protocol omni-multiplexor
- Prototype
  - Go based, unprivileged user space application
  - Open-source under **GPL 3.0** on **GitHub** ([github.com/gaukas/passthru](https://github.com/gaukas/passthru))
- Application



# Recap

- Passthru: a protocol omni-multiplexor
- Prototype
  - Go based, unprivileged user space application
  - Open-source under **GPL 3.0** on **GitHub** ([github.com/gaukas/passthru](https://github.com/gaukas/passthru))
- Application
- Questions?