Pluggable Transports in Practice

by Michael Pöhn <michael@guardianproject.info>

PGP: 3DBD BA23 810A EE37 7CC8 E9D7 C843 2463 5610 899F

GUARDIAN PROJECT

What’s Pluggable Transports?

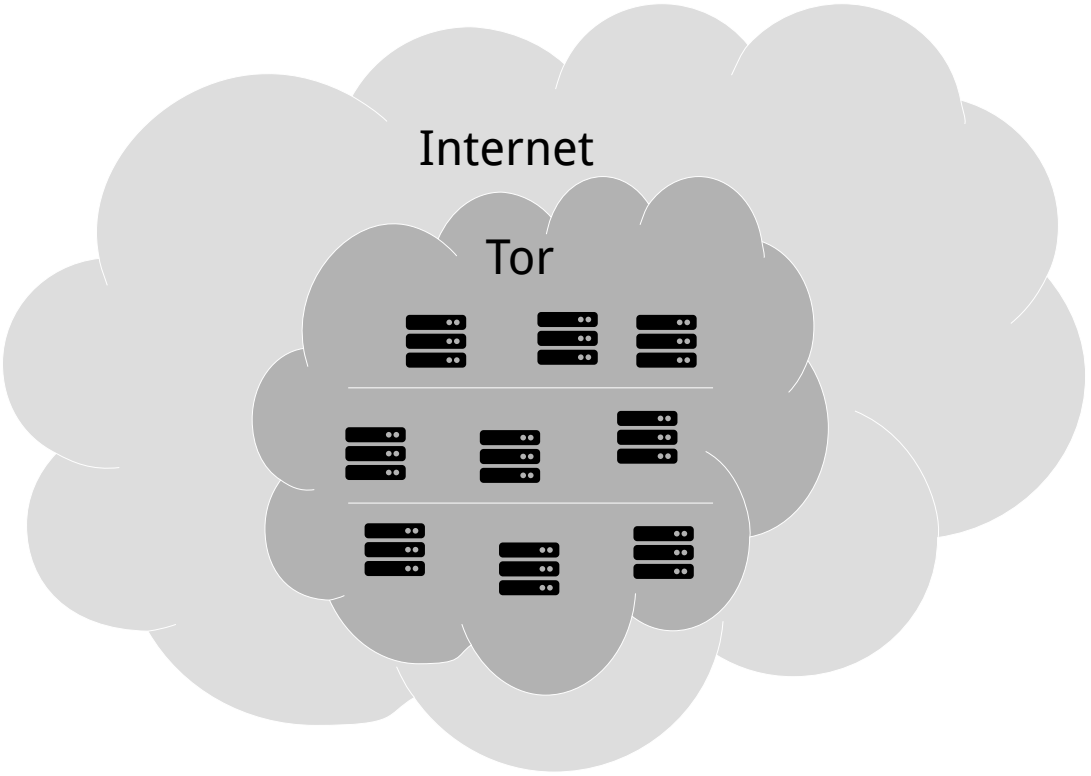
»Pluggable Transports have been created to help developers keep their users connected when censorship occurs. While Transports conform to a single spec, they connect to the network using a variety of different techniques.«

– https://www.pluggabletransports.info/about/

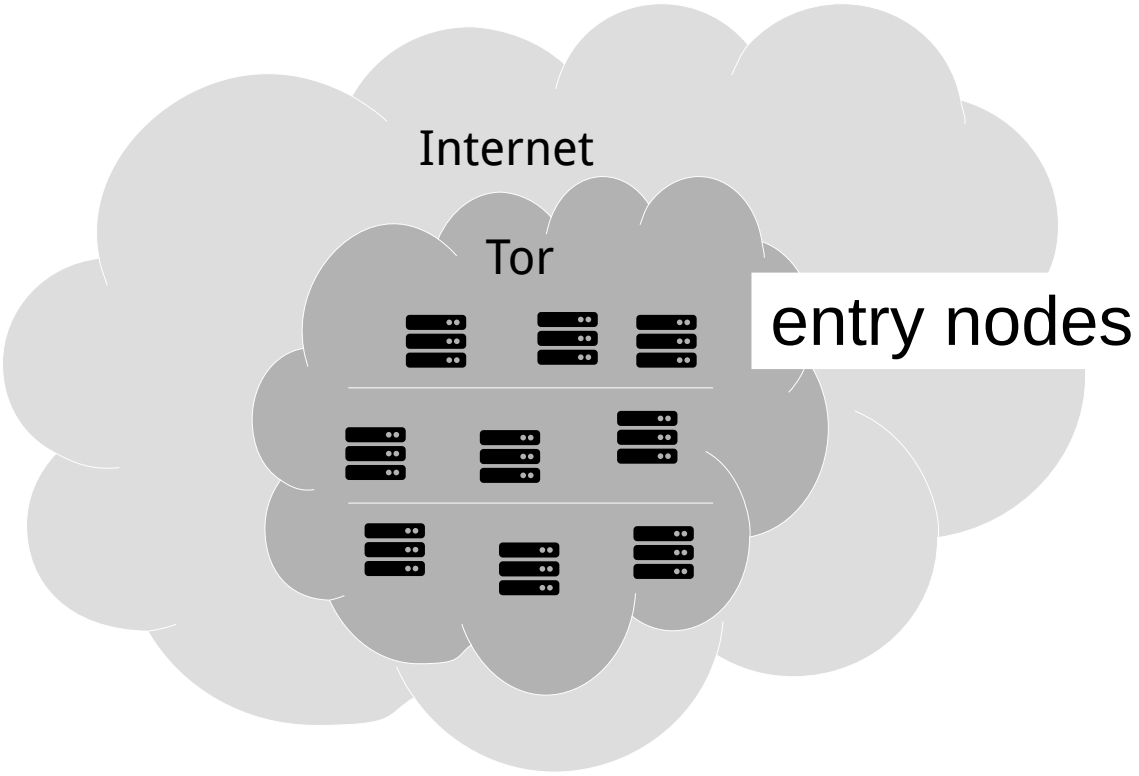
What's Pluggable Transports?

Started out as a censorship circumvention layer of the Tor anonymization network.

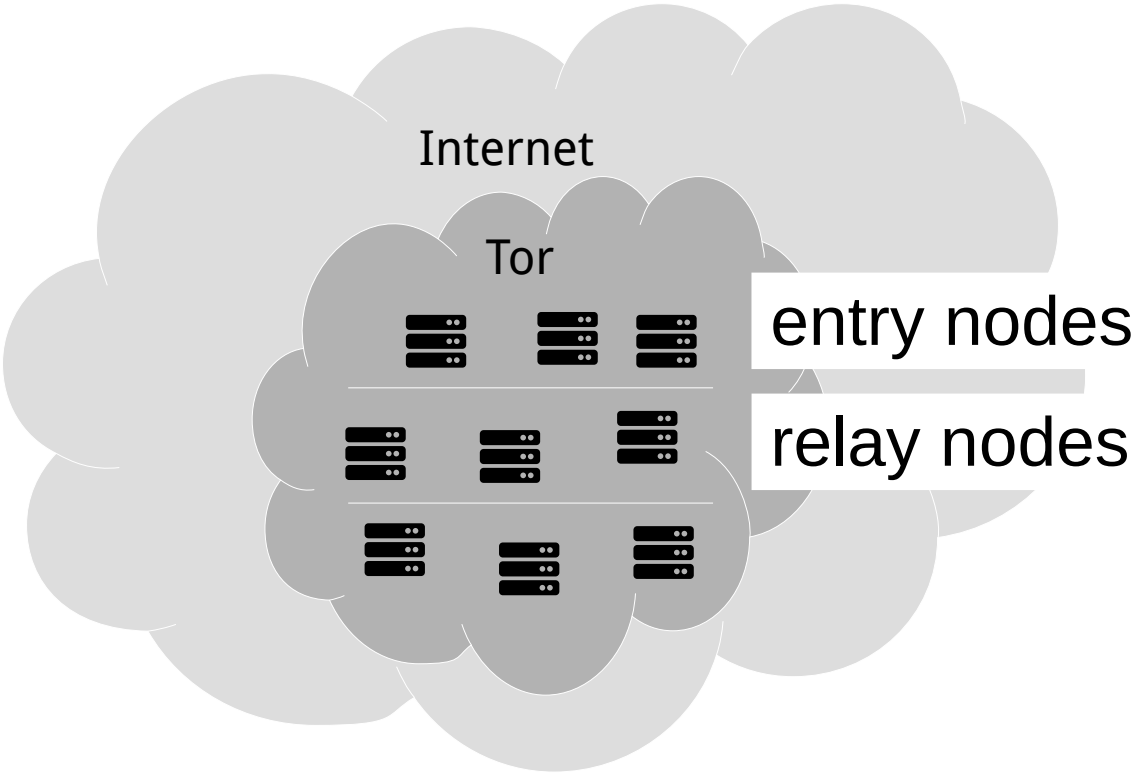
Tor – anonymization



Tor – anonymization

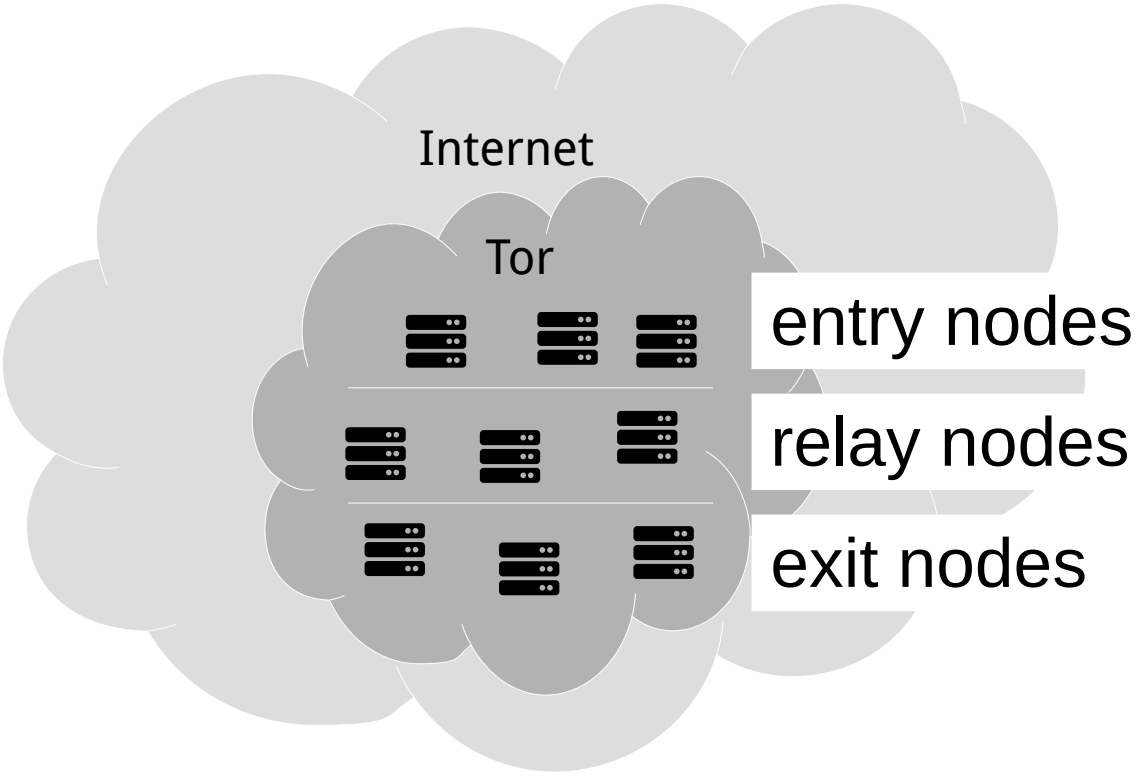


Tor – anonymization



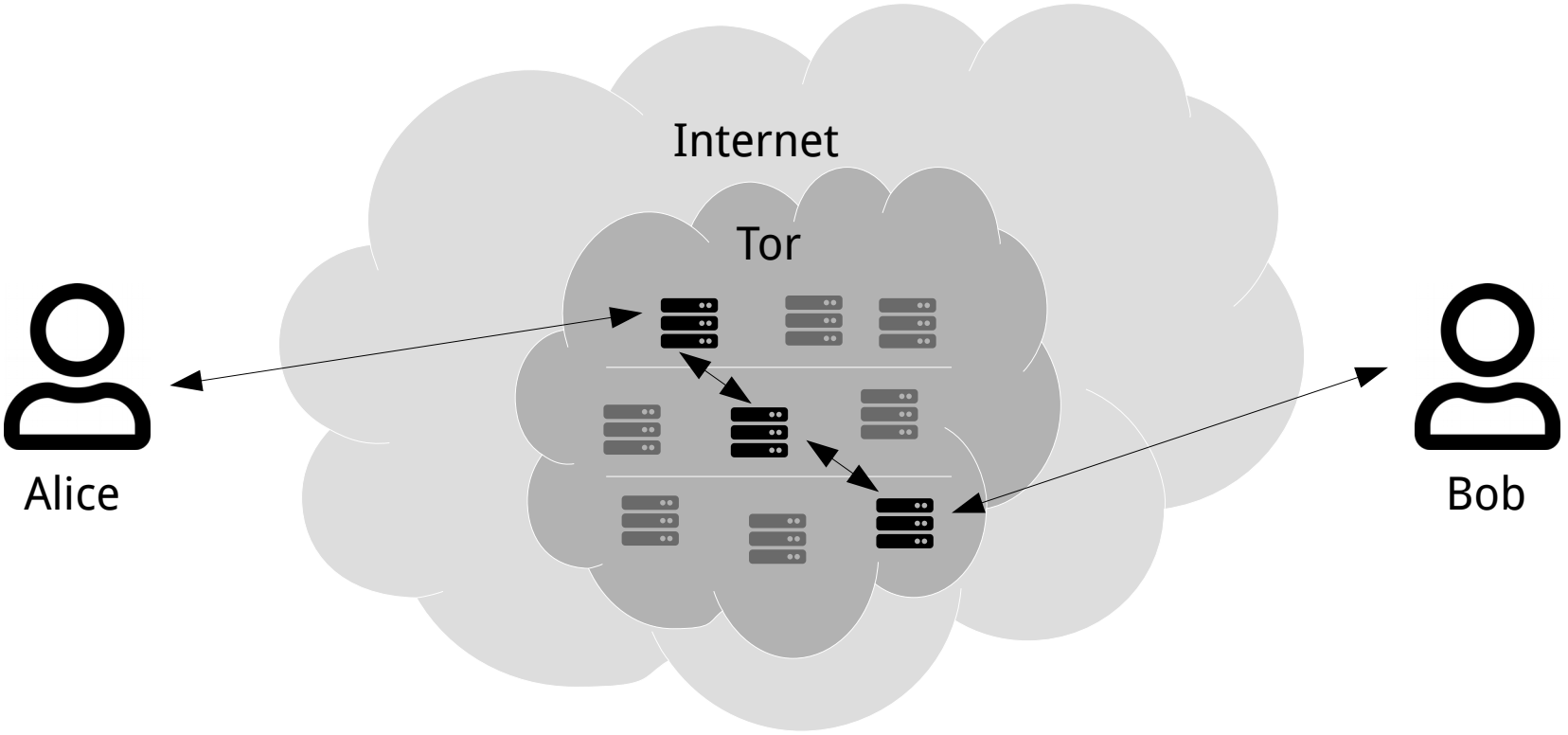
GUARDIAN PROJECT

Tor – anonymization

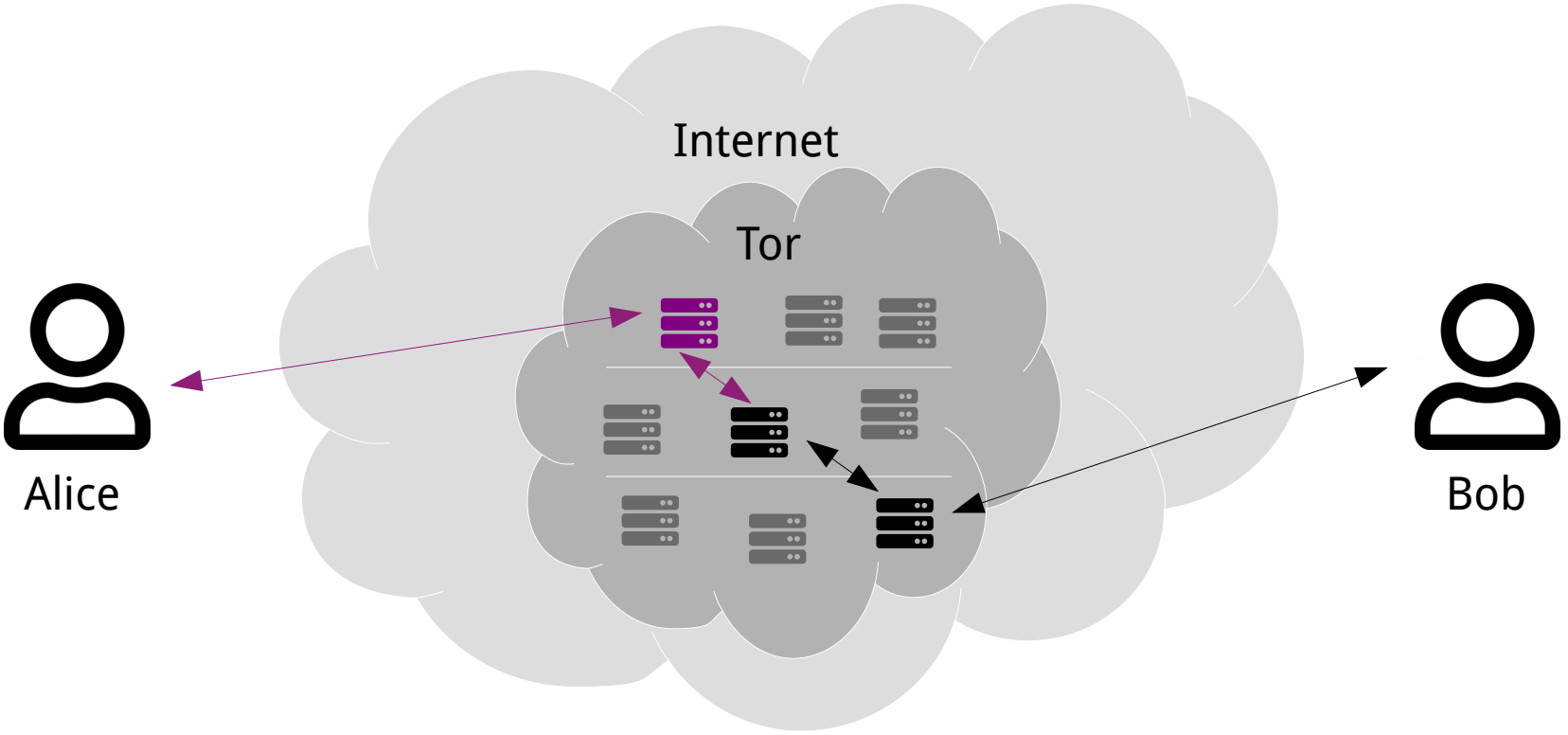


GUARDIAN PROJECT

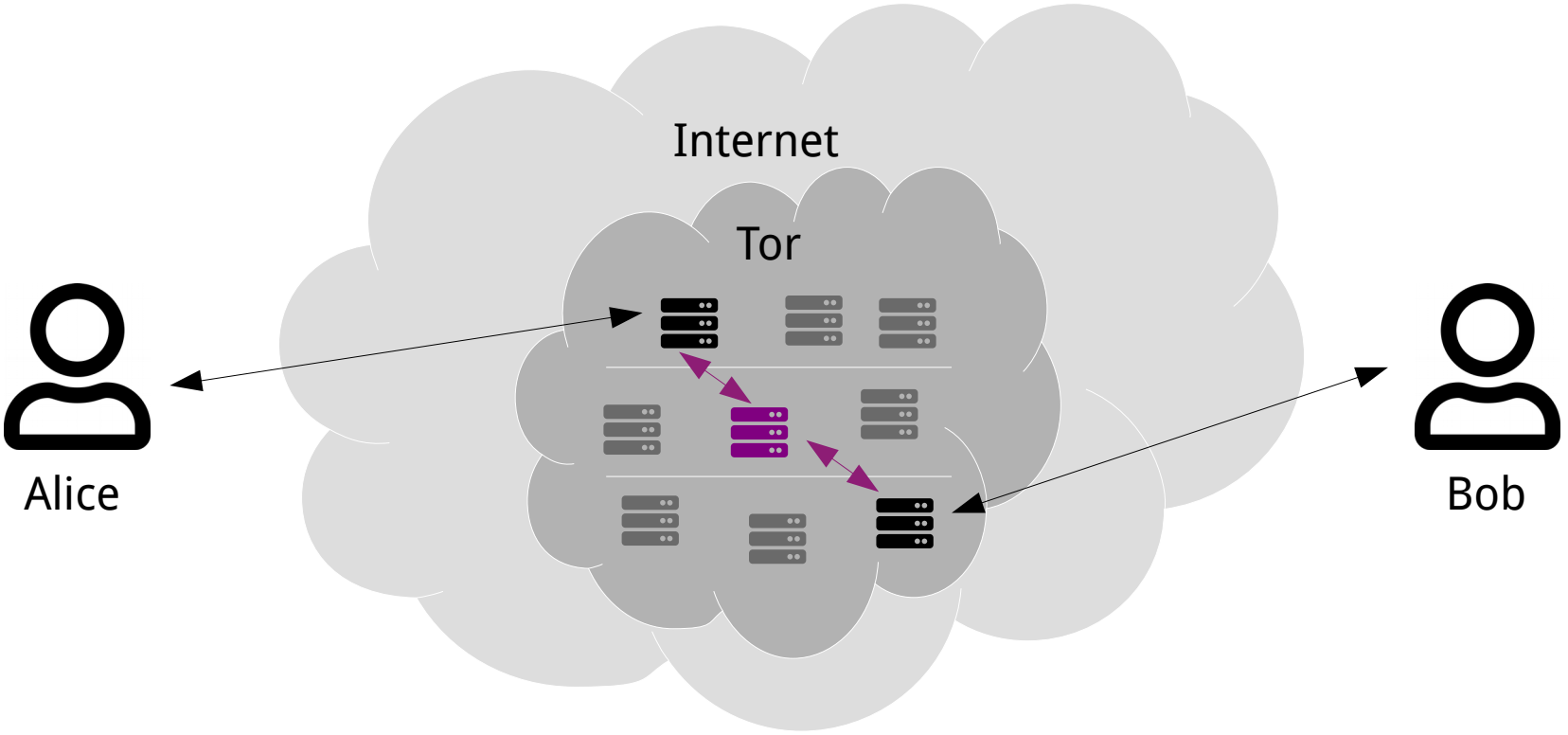
Tor – anonymization



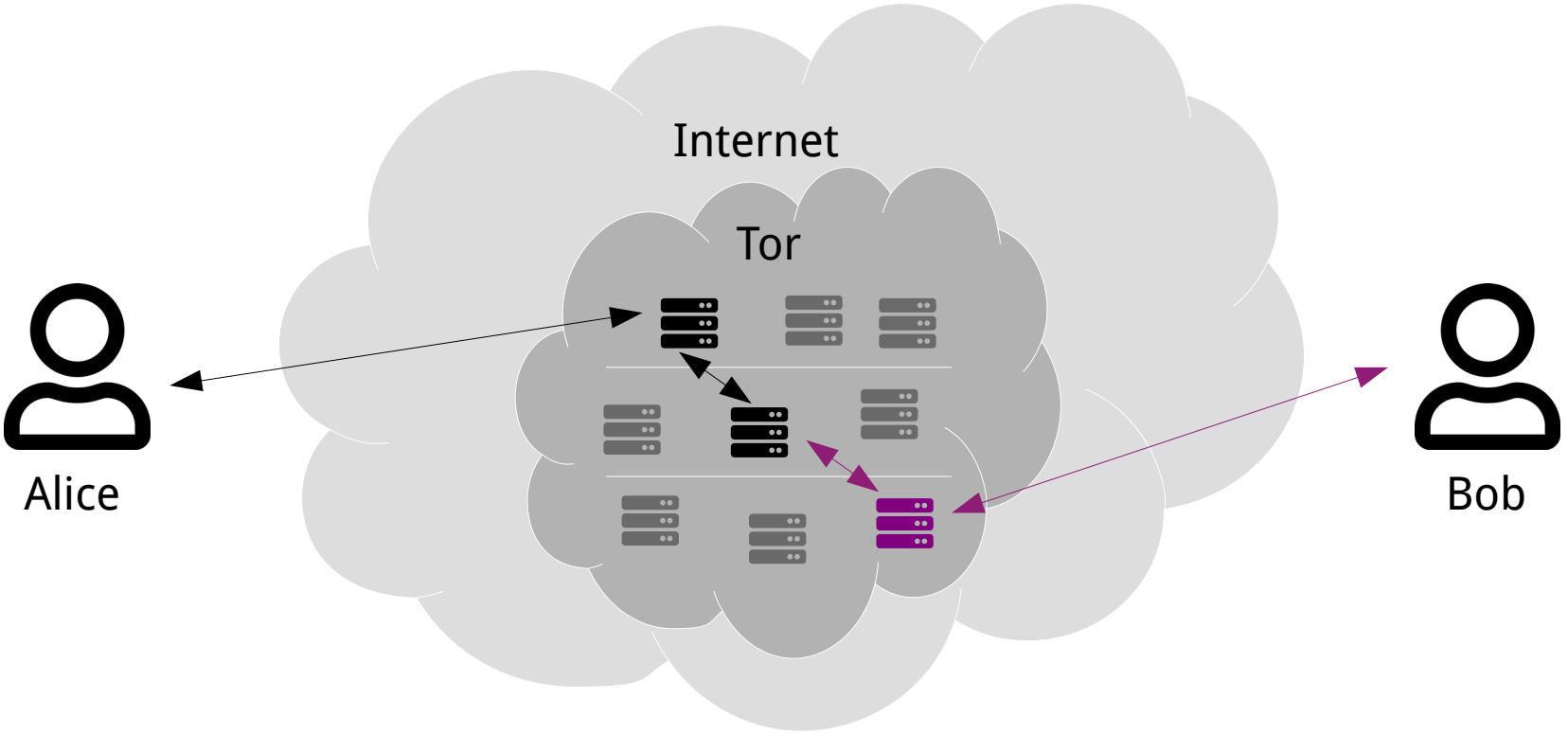
Tor – anonymization



Tor – anonymization



Tor – anonymization



censorship against Tor

- censor Tor software distribution (eg. downloads, store-listings)

censorship against Tor

- censor Tor software distribution (eg. downloads, store-listings)
- censor connections from Tor exit nodes

censorship against Tor

- censor Tor software distribution (eg. downloads, store-listings)
- censor connections from Tor exit nodes
- censor connections to Tor entry nodes

censorship techniques

- DNS blocking

censorship techniques

- DNS blocking
- IP filtering

censorship techniques

- DNS blocking
- IP filtering
- Port filtering

censorship techniques

- DNS blocking
- IP filtering
- Port filtering
- DPI filtering

DNS blocking – circumvention

- alternative addresses

DNS blocking – circumvention

- alternative addresses
- uncensored DNS server

DNS blocking – circumvention

- alternative addresses
- uncensored DNS server
- DoH (DNS over HTTPS)

IP filtering – circumvention

- host on alternative / rotating IP addresses

IP filtering – circumvention

- host on alternative / rotating IP addresses
- use proxy / VPN software

Port filtering – circumvention

- host on unfiltered ports

Port filtering – circumvention

- host on unfiltered ports
- use proxy / VPN software

DPI filtering

filter network packets based on IP packet content

DPI filtering

filter network packets based on IP packet content

- high-level protocol headers / content
  eg. OpenVPN package headers

DPI filtering

filter network packets based on IP packet content

- high-level protocol headers / content
  eg. OpenVPN package headers
- content patterns / protocol atypical timing
  eg. characteristic cipher-text preambles

DPI filtering

filter network packets based on IP packet content

- high-level protocol headers / content
  eg. OpenVPN package headers
- content patterns / protocol atypical timing
  eg. characteristic cipher-text preambles
- ...

DPI blocking – against Tor

- since 2012 – China, Iran, Ethiopia
- since 2014 – Syria
- since 2016 – Turkey

DPI blocking – against Tor

- Enterprises all over the world
- since 2012 – China, Iran, Ethiopia
- since 2014 – Syria
- since 2016 – Turkey

DPI blocking – circumvention

- use software / proxy / VPN with circumvention capabilities
  eg. using Psiphon, Lantern, TunnelBear or Pluggable Transports.

Back to Pluggable Transports

What are PT actually doing?

- provide network traffic obfuscation implementations
- provide standardized interface for tunneling a network connections

available PT implementations

available PT implementations

fronting:

- meek
- SnowFlake
- ...

available PT implementations

fronting:

- meek
- SnowFlake
- ...

scrambling:

- Obfs4
- ScrambleSuit
- ...

available PT implementations

fronting:

- meek
- SnowFlake
- ...

scrambling:

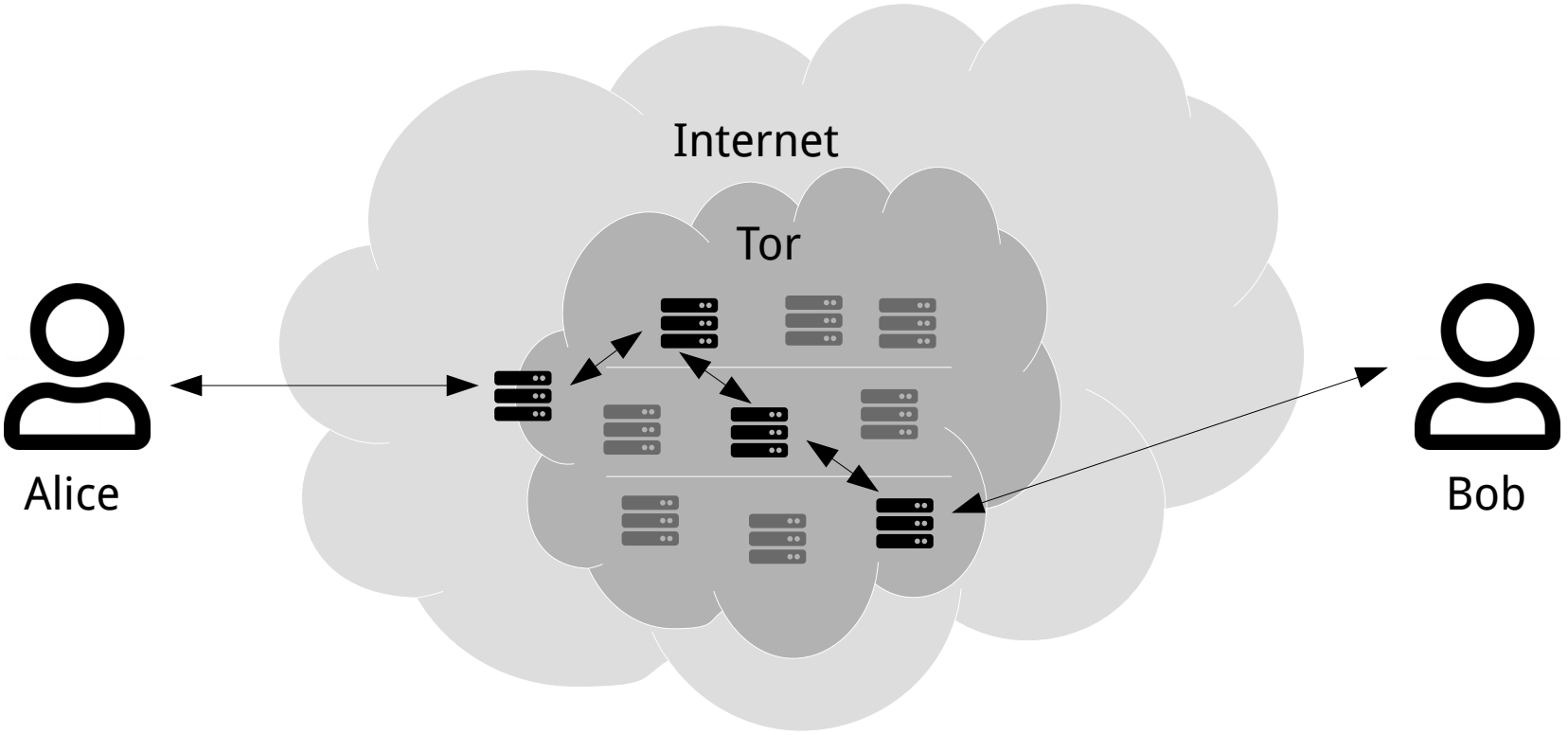
- Obfs4
- ScrambleSuit
- ...

shape-shifting:

- Dust2
- FTEProxy
- ...

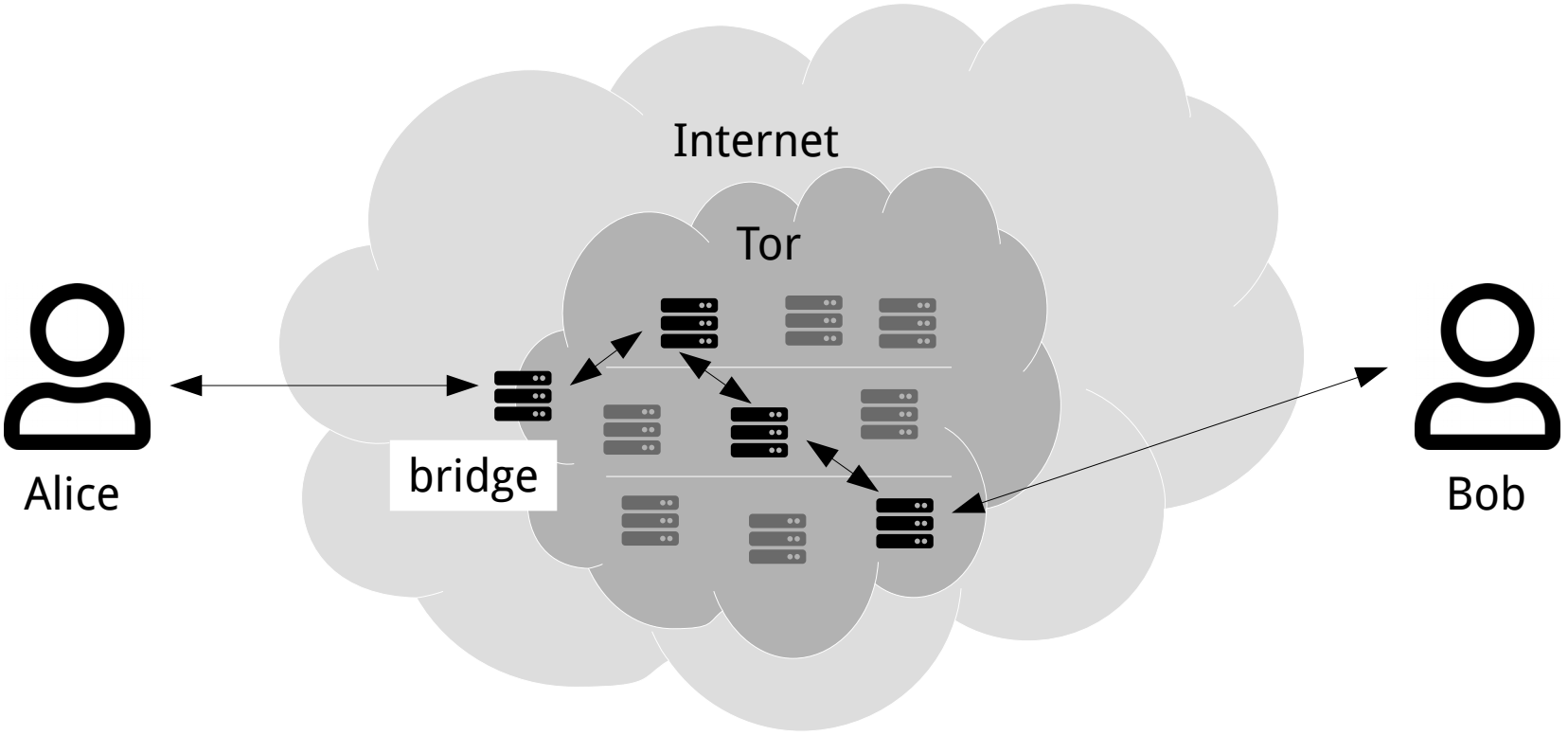
GUARDIAN PROJECT

Tor with Pluggable Transports



GUARDIAN PROJECT

Tor with Pluggable Transports



PT APIs

PT APIs

Dispatcher:

- process for establishing PT tunneling socket

PT APIs

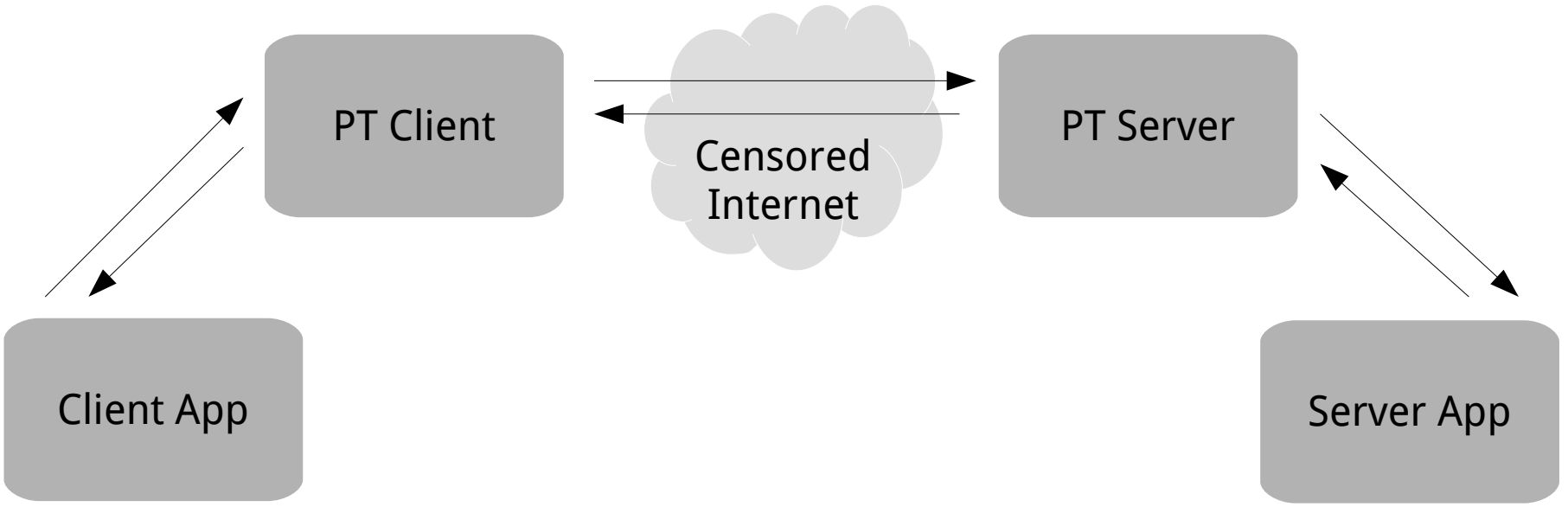
Dispatcher:

- process for establishing PT tunneling socket

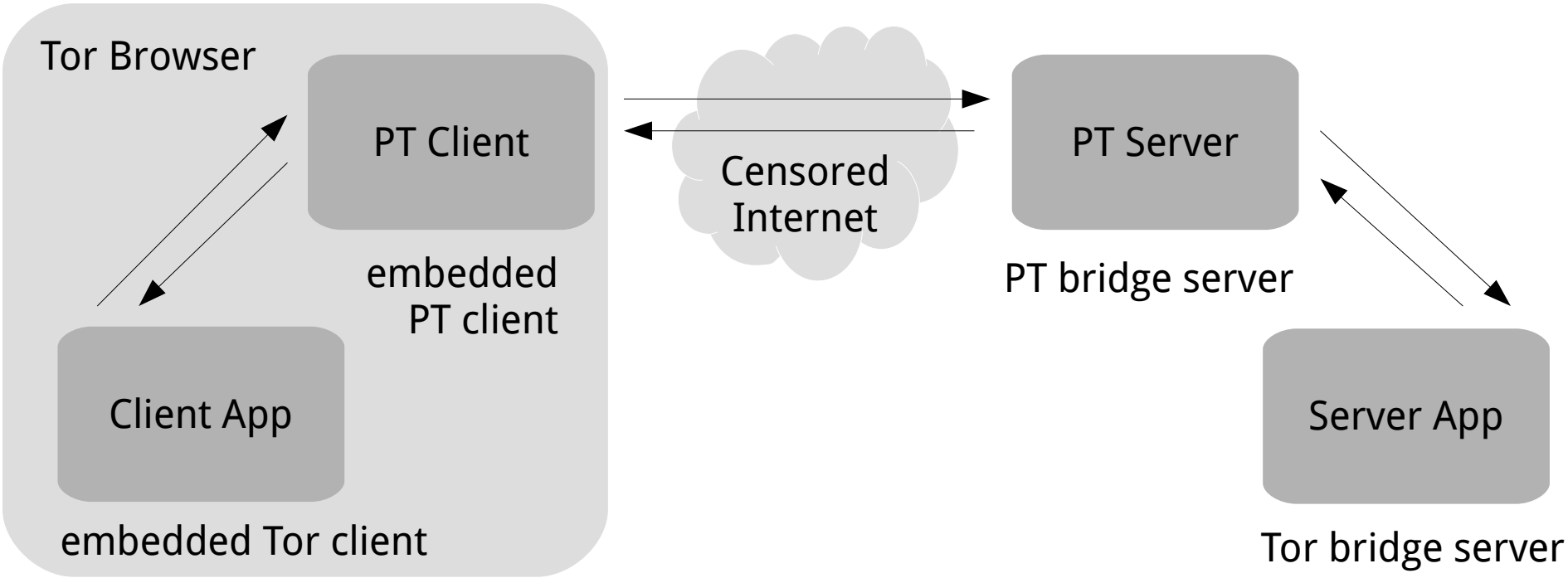
Language/Platform specific APIs:

- Go API, Swift API (iOS), AndroidPluggableTransports, ...

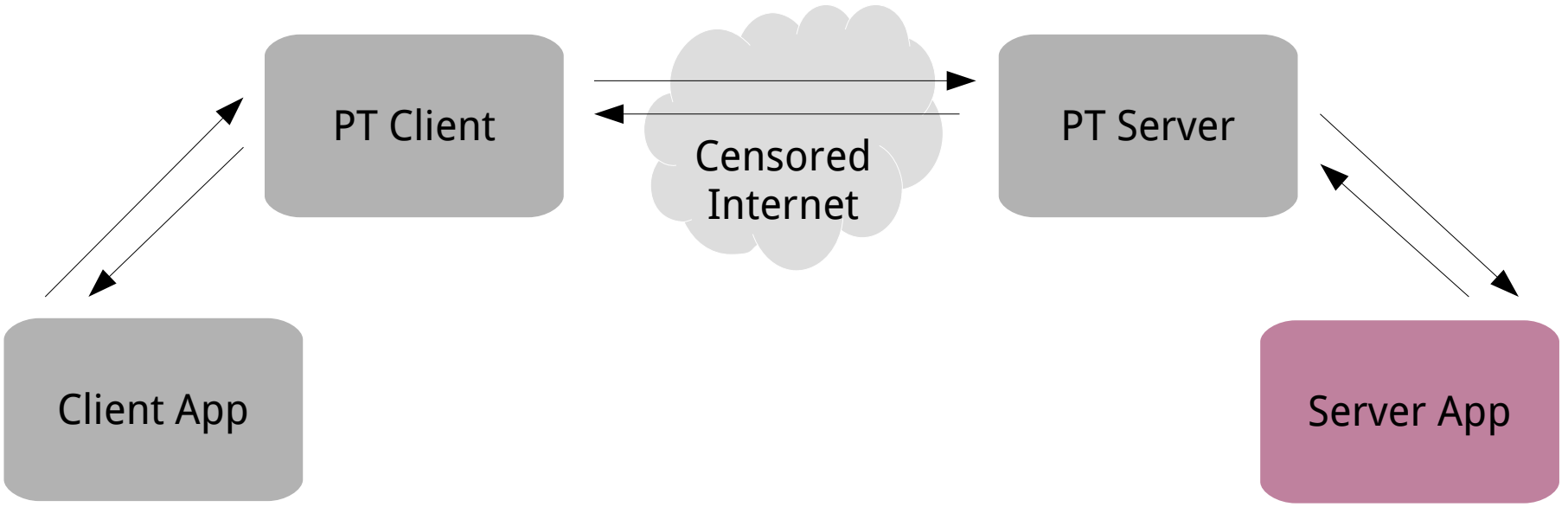
PT Dispatcher Connection



PT Connection – Tor example



PT Dispatcher: Server App



HTTP over Obfs4 – start Server App

let’s start a simple HTTP server, eg:

```
$ cat "<html><h1>Hello proxied World.</h1></html>" \
      > index.html

$ python3 -m http.server
```

HTTP over Obfs4 – start Server App

let’s start a simple HTTP server, eg:

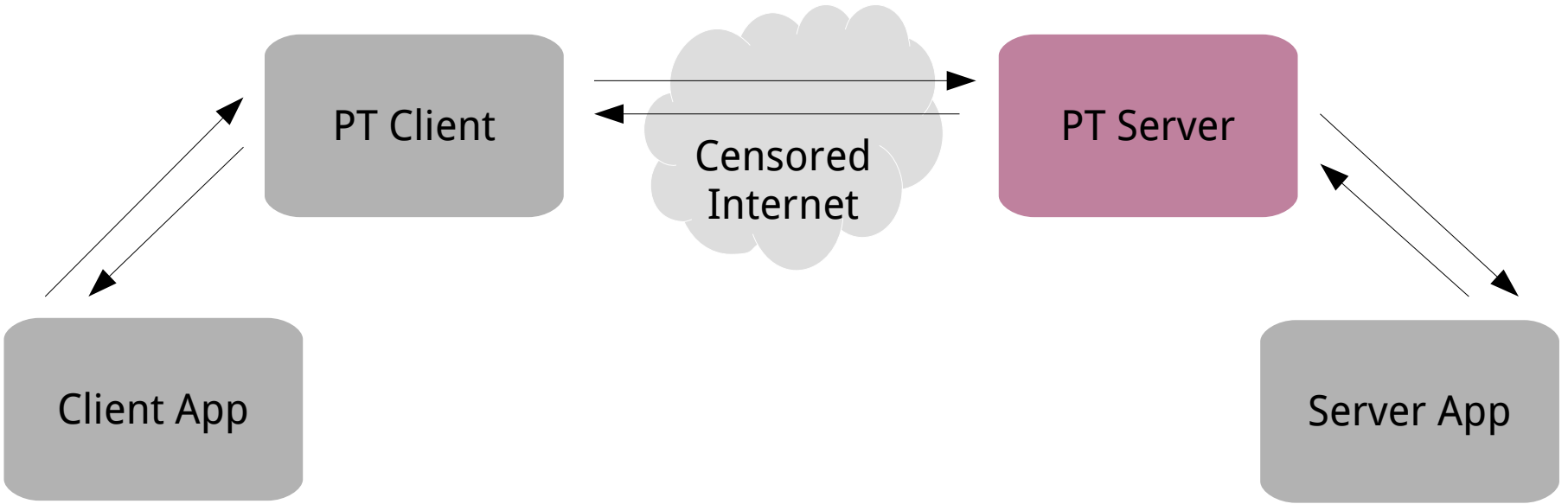
```
$ cat "<html><h1>Hello proxied World.</h1></html>" \
      > index.html

$ python3 -m http.server
```

HTTP over Obfs4 – start Server App

let’s start a simple HTTP server, eg:

```
$ cat "<html><h1>Hello proxied World.</h1></html>" \
      > index.html

$ python3 -m http.server
```

PT Dispatcher: PT Server



HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

GUARDIAN PROJECT

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

GUARDIAN PROJECT

HTTP over Obfs4 – start PT Server

now start `obfs4proxy` server, eg:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_server_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_SERVER_TRANSPORTS=obfs4
$ export TOR_PT_SERVER_BINDADDR=obfs4-0.0.0.0:9876
$ export TOR_PT_ORPORT=127.0.0.1:8000

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Server

check output of PT Server `obfs4proxy`:

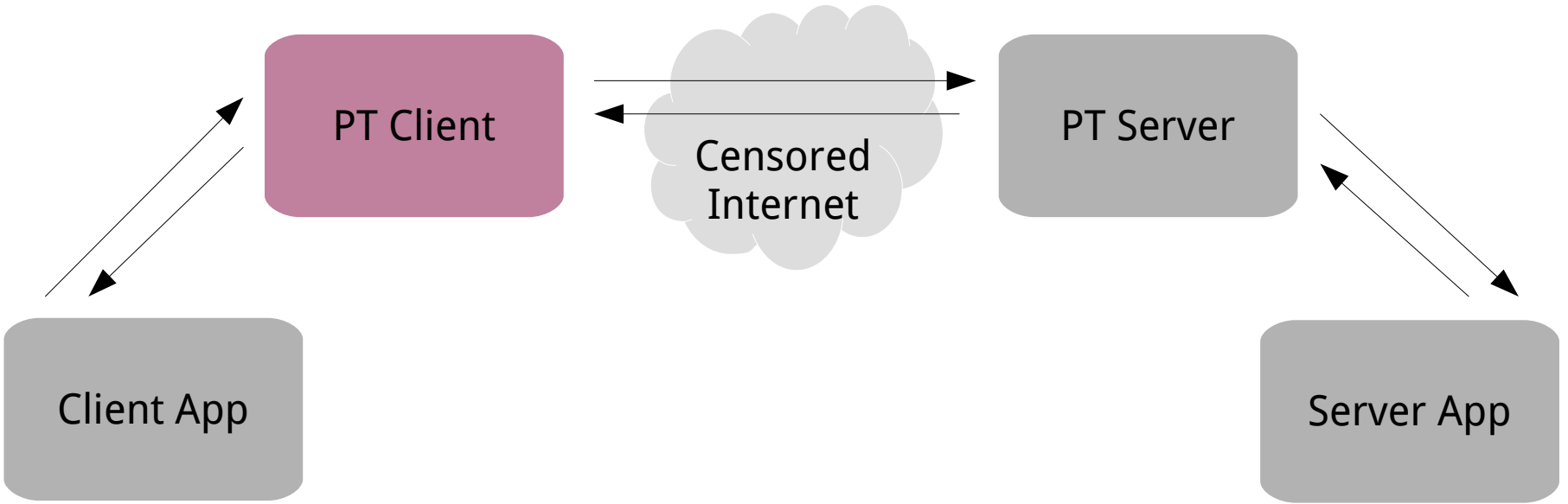
```
$ obfs4proxy -enableLogging -logLevel=DEBUG

VERSION 1
SMETHOD obfs4 [::]:9876
ARGS:cert=g1zoXZNOcxgZZcKMtk9ReIZVugDYiWDbAGRItNRPGC4ETKW2
Zpy9kLmySrR/TuRwSmb3DQ,iat-mode=0
SMETHODS DONE
```

HTTP over Obfs4 – start PT Server

check output of PT Server `obfs4proxy`:

```
$ obfs4proxy -enableLogging -logLevel=DEBUG

VERSION 1
SMETHOD obfs4 [::]:9876
ARGS:cert=g1zoXZNOcxgZZcKMtk9ReIZVugDYiWDbAGRItNRPGC4ETKW2
Zpy9kLmySrR/TuRwSmb3DQ,iat-mode=0
SMETHODS DONE
```

PT Dispatcher: PT Client



HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

GUARDIAN PROJECT

HTTP over Obfs4 – start PT Client

next to do is staring PT Client `obfs4proxy`:

```
$ export TOR_PT_MANAGED_TRANSPORT_VER=1
$ export TOR_PT_STATE_LOCATION=/vagrant/pt_client_state
$ export TOR_PT_EXIT_ON_STDIN_CLOSE=0
$ export TOR_PT_CLIENT_TRANSPORTS=obfs4

$ obfs4proxy -enableLogging -logLevel=DEBUG
```

HTTP over Obfs4 – start PT Client

check output of PT Client `obfs4proxy`:

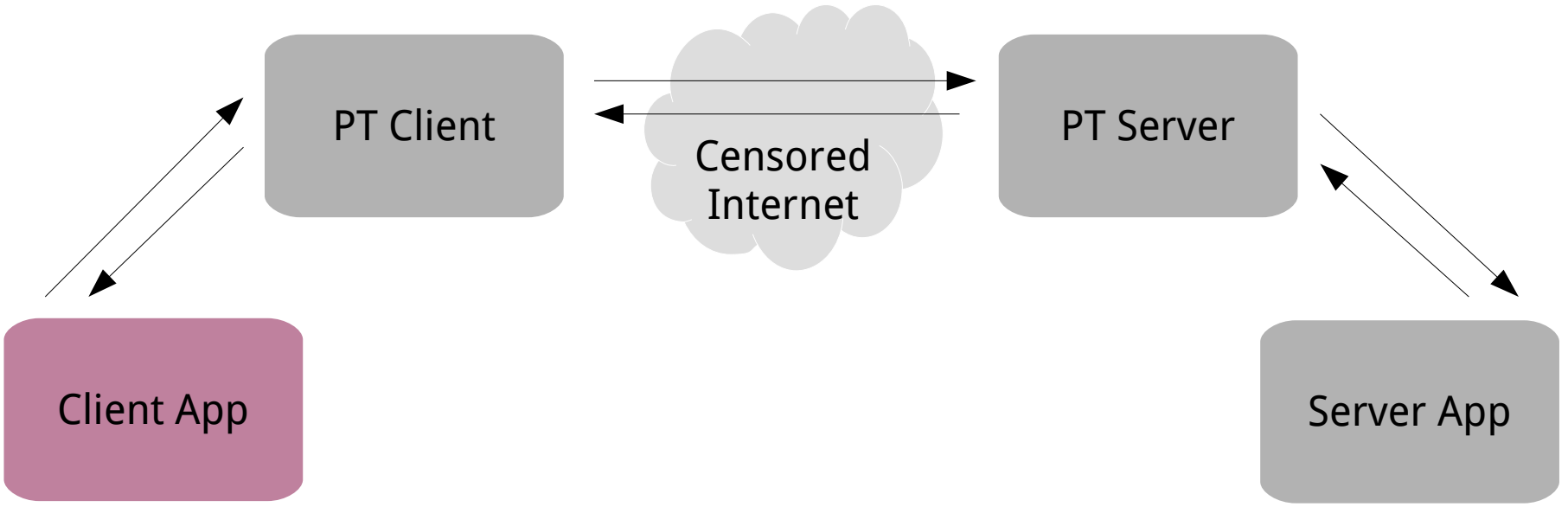
```
$ obfs4proxy -enableLogging -logLevel=DEBUG

VERSION 1
CMETHOD obfs4 socks5 127.0.0.1:33377
CMETHODS DONE
```

HTTP over Obfs4 – start PT Client

check output of PT Client `obfs4proxy`:

```
$ obfs4proxy -enableLogging -logLevel=DEBUG

VERSION 1
CMETHOD obfs4 socks5 127.0.0.1:33377
CMETHODS DONE
```

PT Dispatcher: Client App



HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

HTTP over Obfs4 – Client App

run HTTP request through Obfs4 Tunnel:

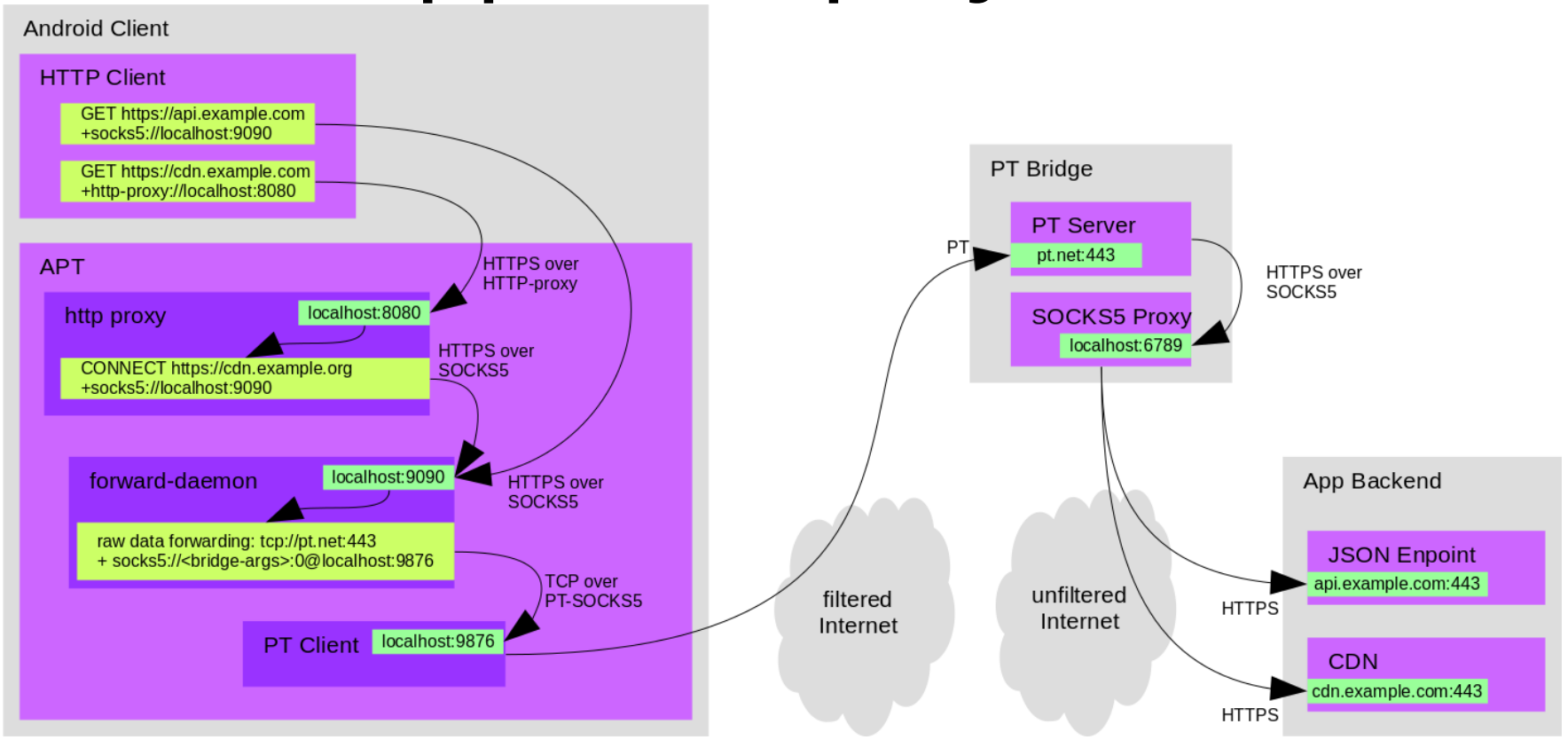
```
$ SOCKS_CON="127.0.0.1:33377"
$ SOCKS_AUTH="socks5://cert=g1zoXZNO...mb3DQ;iat-mode=0:0"
$ BRIDGE="127.0.0.1:9876"

$ curl -proxy "${SOCKS_AUTH}@${SOCSK_SRV}" $BRIDGE
<html><h1>Hello proxied World.</h1></html>
```

Dispatcher-SOCKS5 is not SOCKS5

When establishing a connection to a PT Dispatcher,
it uses SOCKS5 for passing parameters and establishing
a point to point TCP connection.

Should you need a Proxy you have to run one as Server App.

Android App PT deployment



Credits

- Font Awesome Icons – CC-BY-4.0

also check out:

- https://pluggabletransports.info
- https://guardianproject.info

That’s it.

Michael Pöhn <michael@guardianproject.info>

PGP: 3DBD BA23 810A EE37 7CC8 E9D7 C843 2463 5610 899F