Web Security

Week 2 - DNS, Cookies, Sessions

Old Dominion University
Department of Computer Science
CS 495/595 Spring 2022
Michael L. Nelson <mln@cs.odu.edu>

2022-01-24

DNS

Resolving hostnames

```
scorpii:/home/mln % nslookup stanford.edu
Server:         127.0.0.53
Address:   127.0.0.53#53

Non-authoritative answer:
Name: stanford.edu
Address: 171.67.215.200

scorpii:/home/mln % nslookup odu.edu
Server:         127.0.0.53
Address:   127.0.0.53#53

Non-authoritative answer:
Name: odu.edu
Address: 128.82.112.29
```

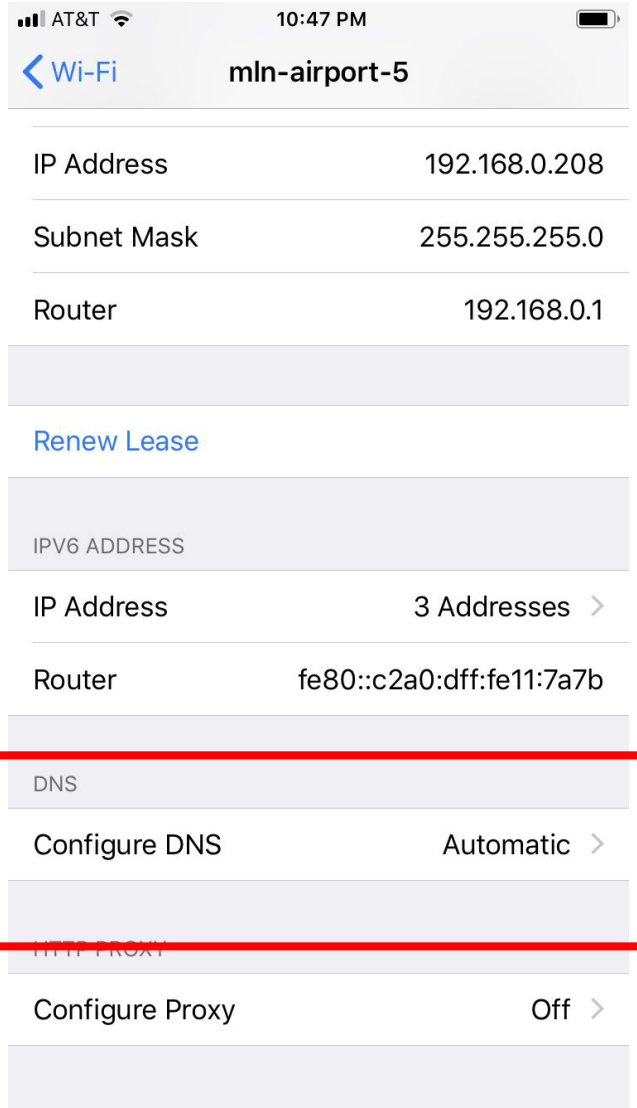
```
mln2@Michaels-MacBook-Air ~ % nslookup stanford.edu
Server:         2001:578:3f::30
Address:   2001:578:3f::30#53

Non-authoritative answer:
Name: stanford.edu
Address: 171.67.215.200

mln2@Michaels-MacBook-Air ~ % nslookup odu.edu
Server:         2001:578:3f::30
Address:   2001:578:3f::30#53

Non-authoritative answer:
Name: odu.edu
Address: 128.82.112.29
```

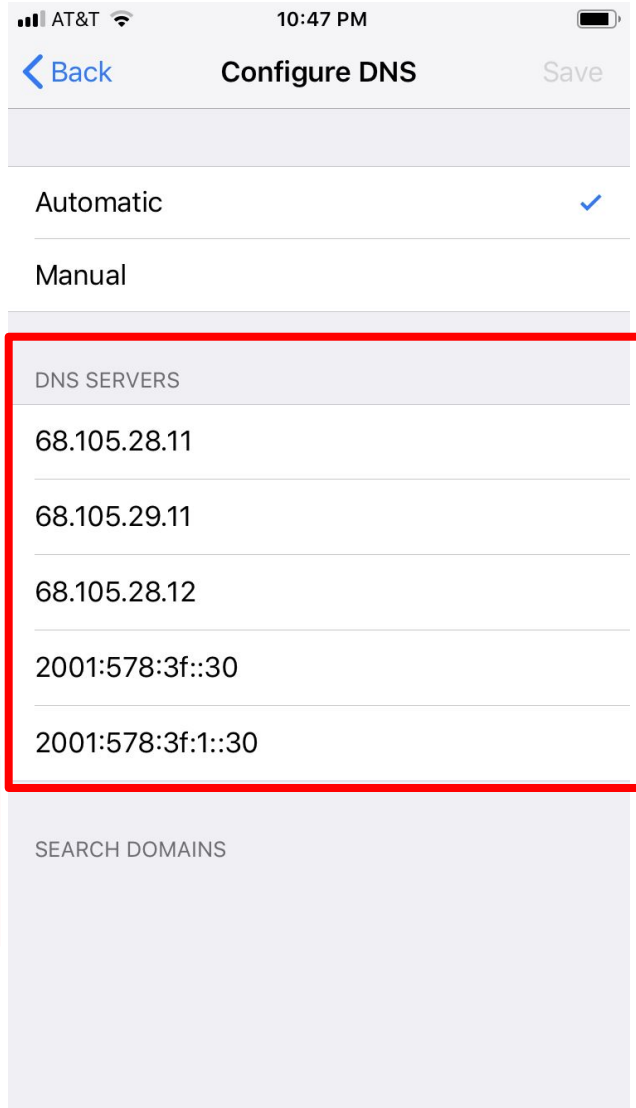
Resolving hostnames

```
scorpii:/home/mln % cd /etc

scorpii:/etc % more hosts
127.0.0.1   localhost
172.18.12.33scorpii.infra.cs.odu.eduscorpii

# The following lines are desirable for IPv6 capable
hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

```
scorpii:/etc % more /etc/resolv.conf
# This file is managed by man:systemd-resolved(8). Do
not edit.
#
# This is a dynamic resolv.conf file for connecting
local clients to the
# internal DNS stub resolver of systemd-resolved.
This file lists all
# configured search domains.
#
# Run "systemd-resolve --status" to see details about
the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file
directly, but only through the
# symlink at /etc/resolv.conf. To manage
man:resolv.conf(5) in a different way,
# replace this symlink by a static file or a
different symlink.
#
# See man:systemd-resolved.service(8) for details
about the supported modes of
# operation for /etc/resolv.conf.

nameserver 127.0.0.53
options edns0
search infra.cs.odu.edu cs.odu.edu
```

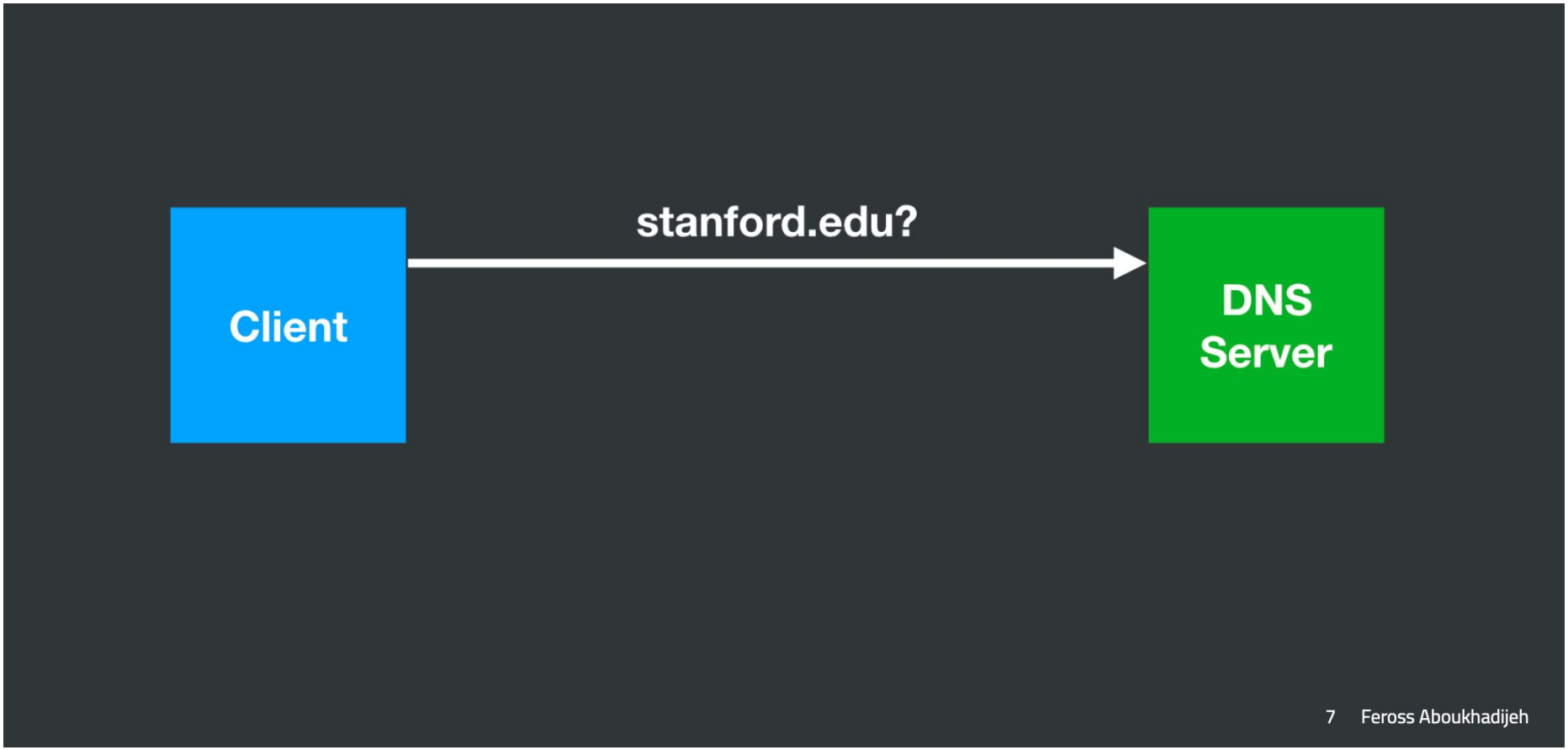
On my iPhone


AT&T
10:47 PM
Wi-Fi
mln-airport-5
IP Address
192.168.0.208
Subnet Mask
255.255.255.0
Router
192.168.0.1
Renew Lease
IPV6 ADDRESS
IP Address
3 Addresses
Router
fe80::c2a0:dff:fe11:7a7b
DNS
Configure DNS
Automatic
Configure Proxy
Off

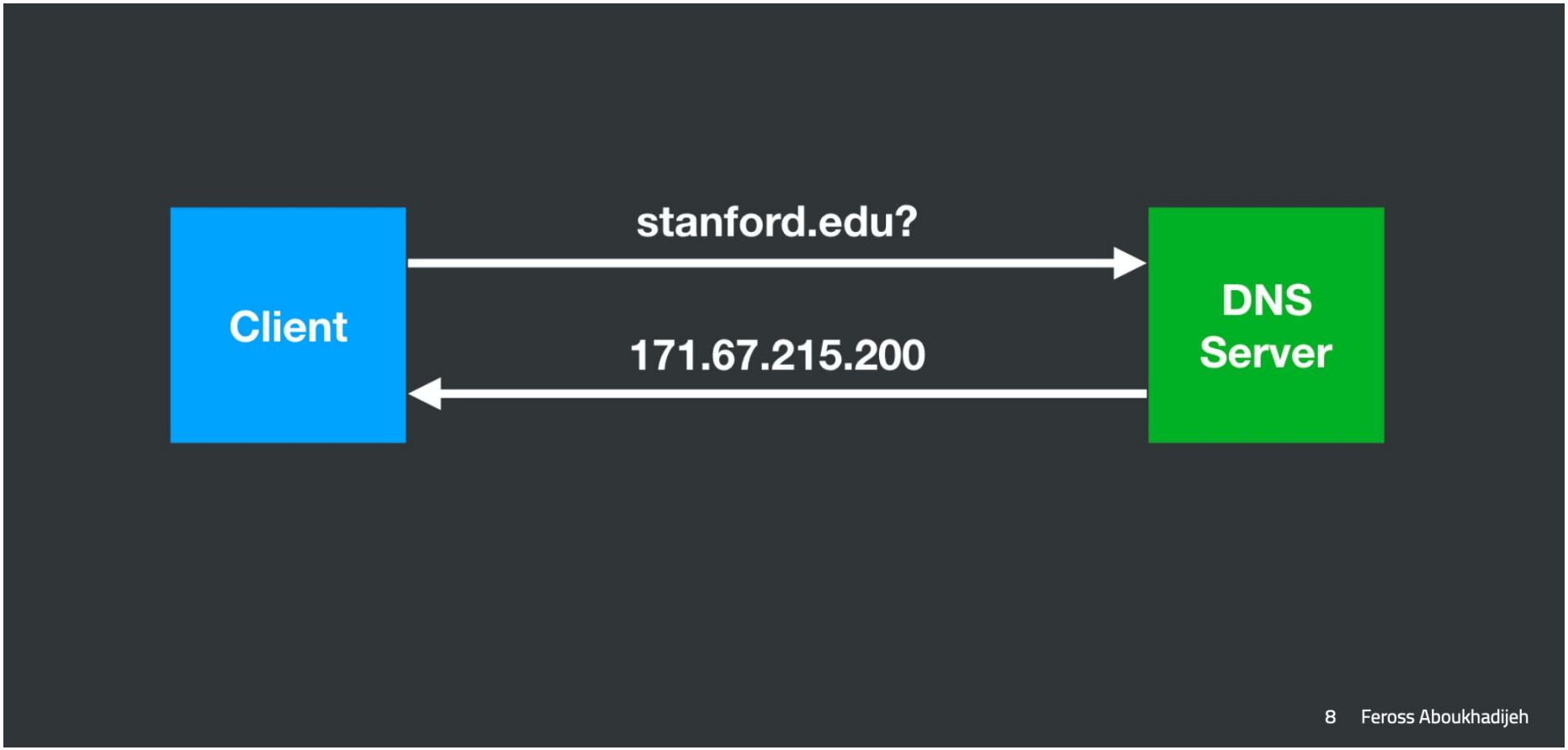

AT&T
10:47 PM
Back
Configure DNS
Save
Automatic
Manual
DNS SERVERS
68.105.28.11
68.105.29.11
68.105.28.12
2001:578:3f::30
2001:578:3f:1::30
SEARCH DOMAINS

Domain Name System

Domain Name System

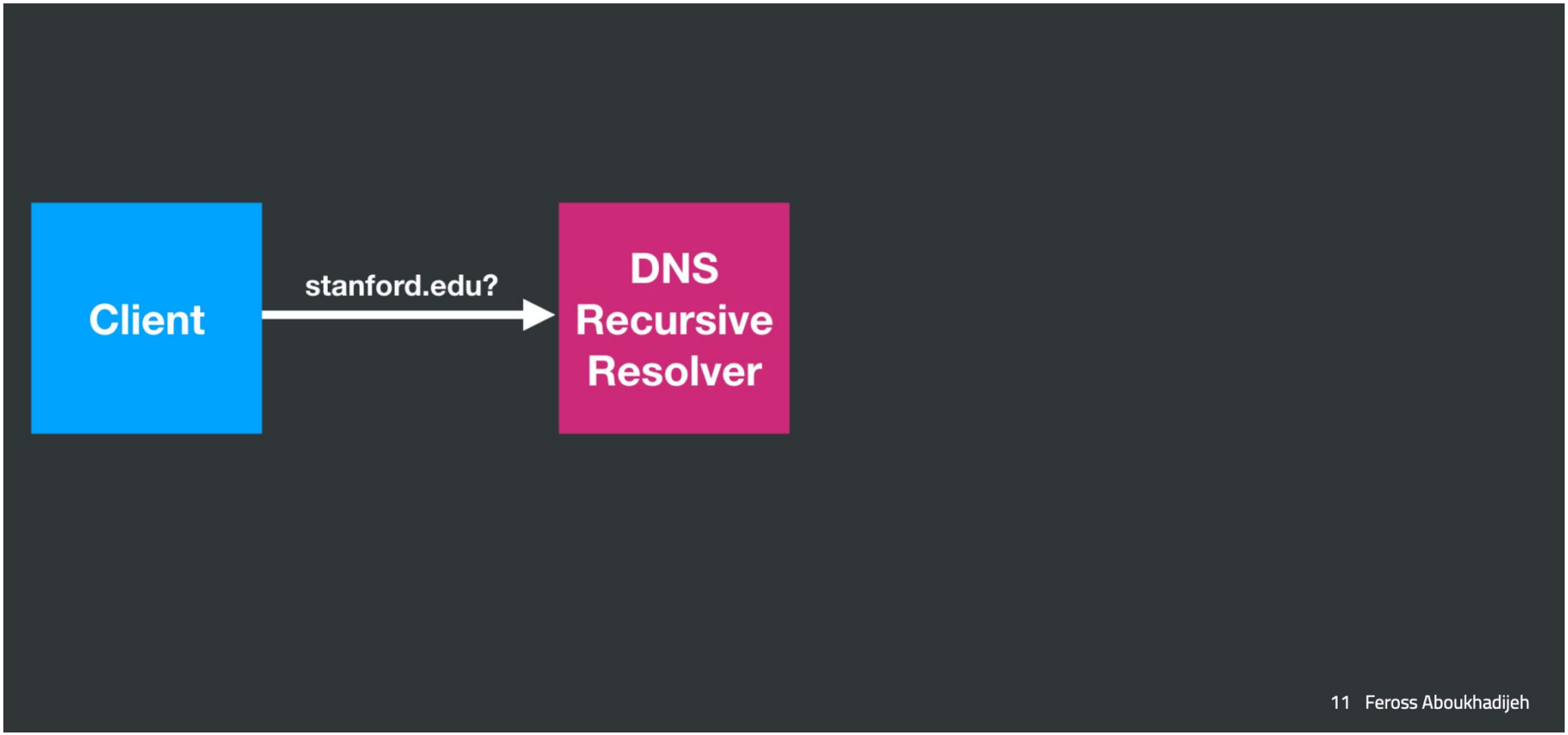


Domain Name System

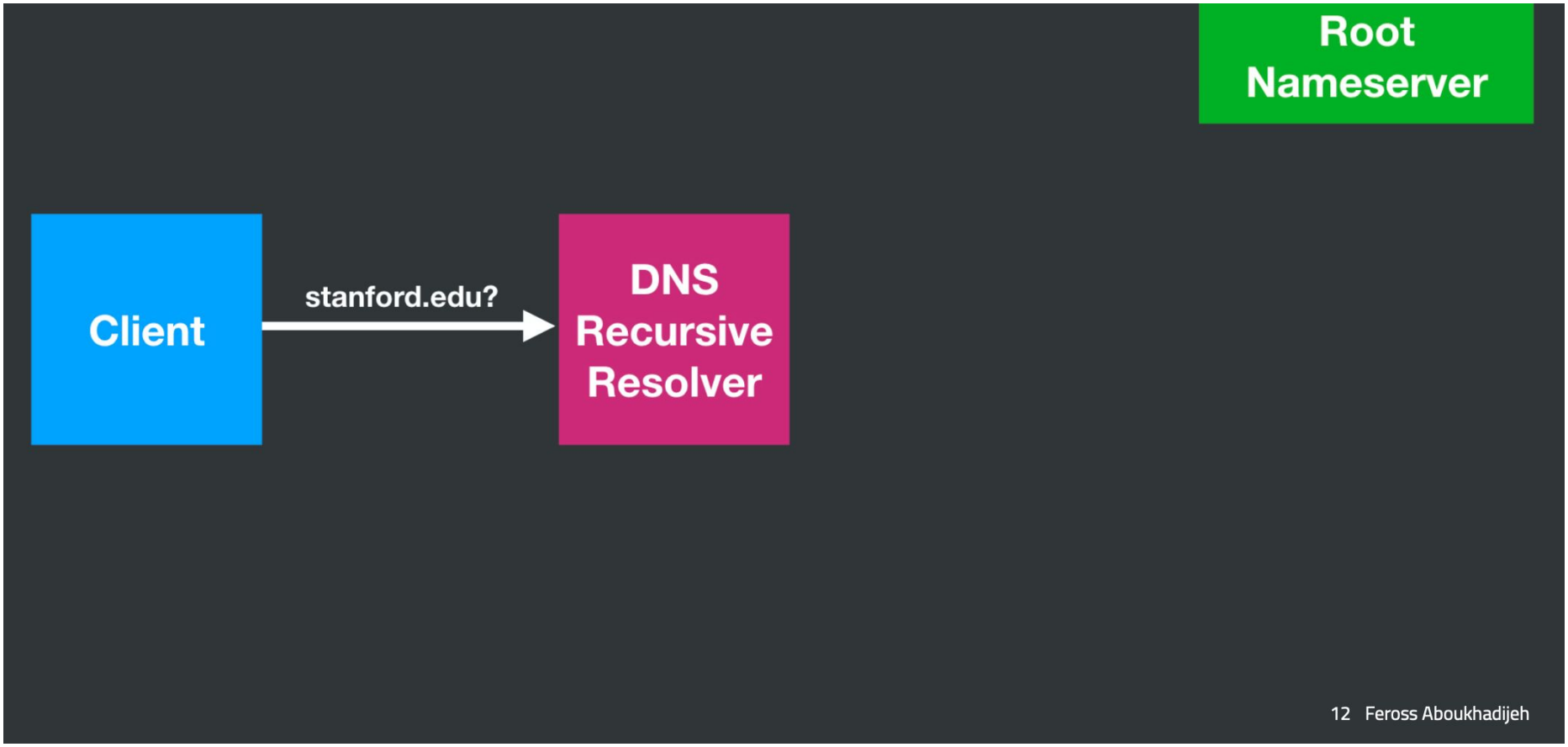


DNS Recursive Resolution

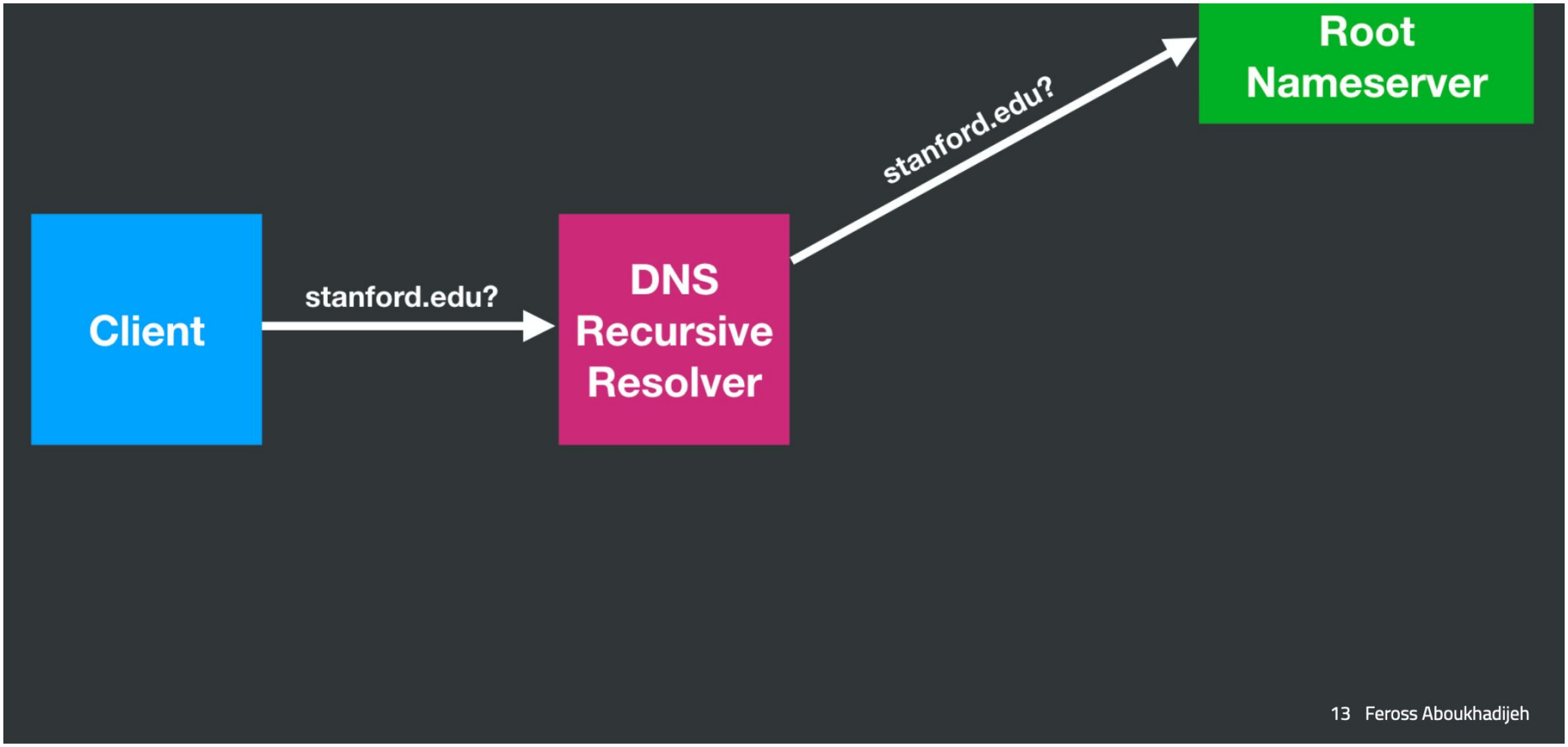
DNS Recursive Resolution



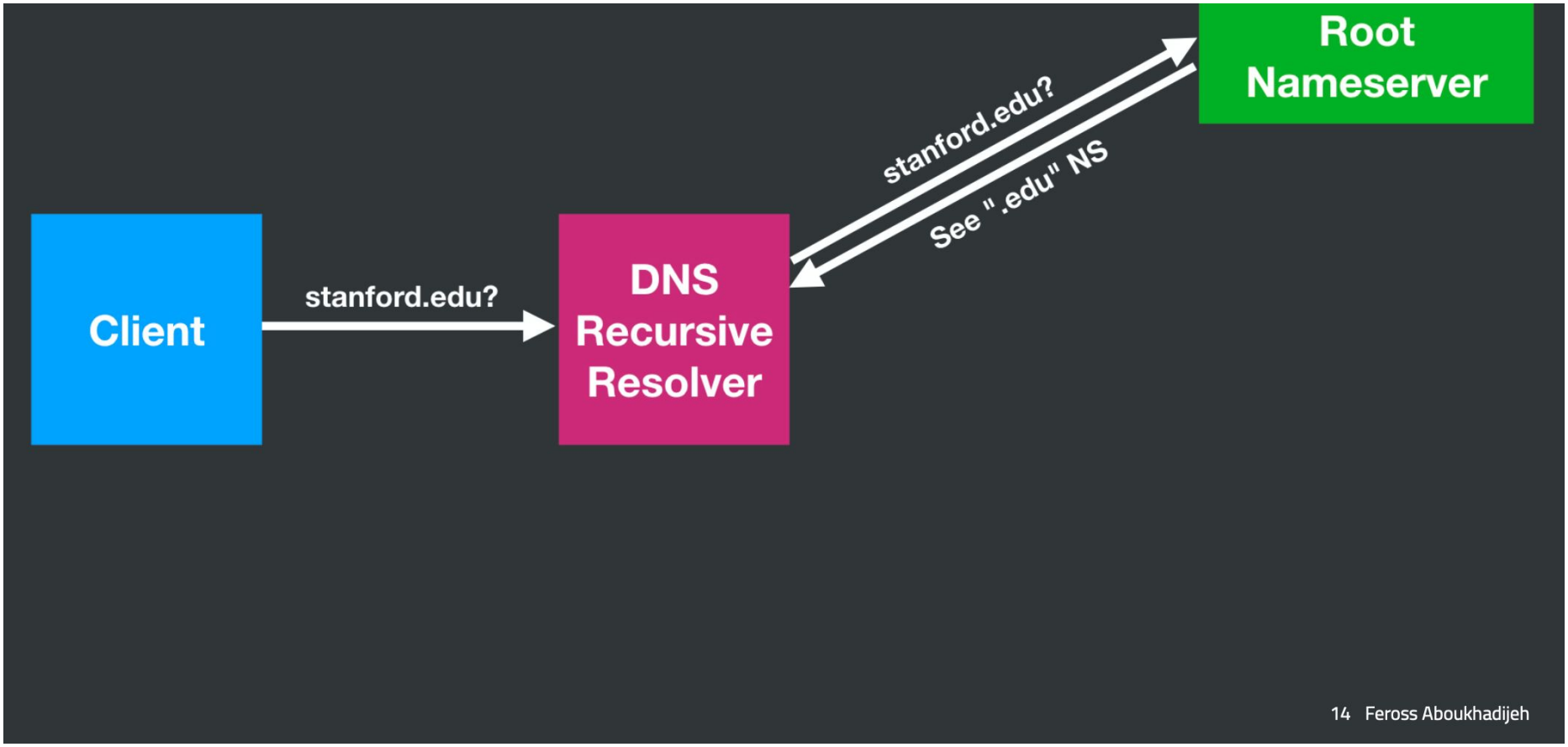
DNS Recursive Resolution



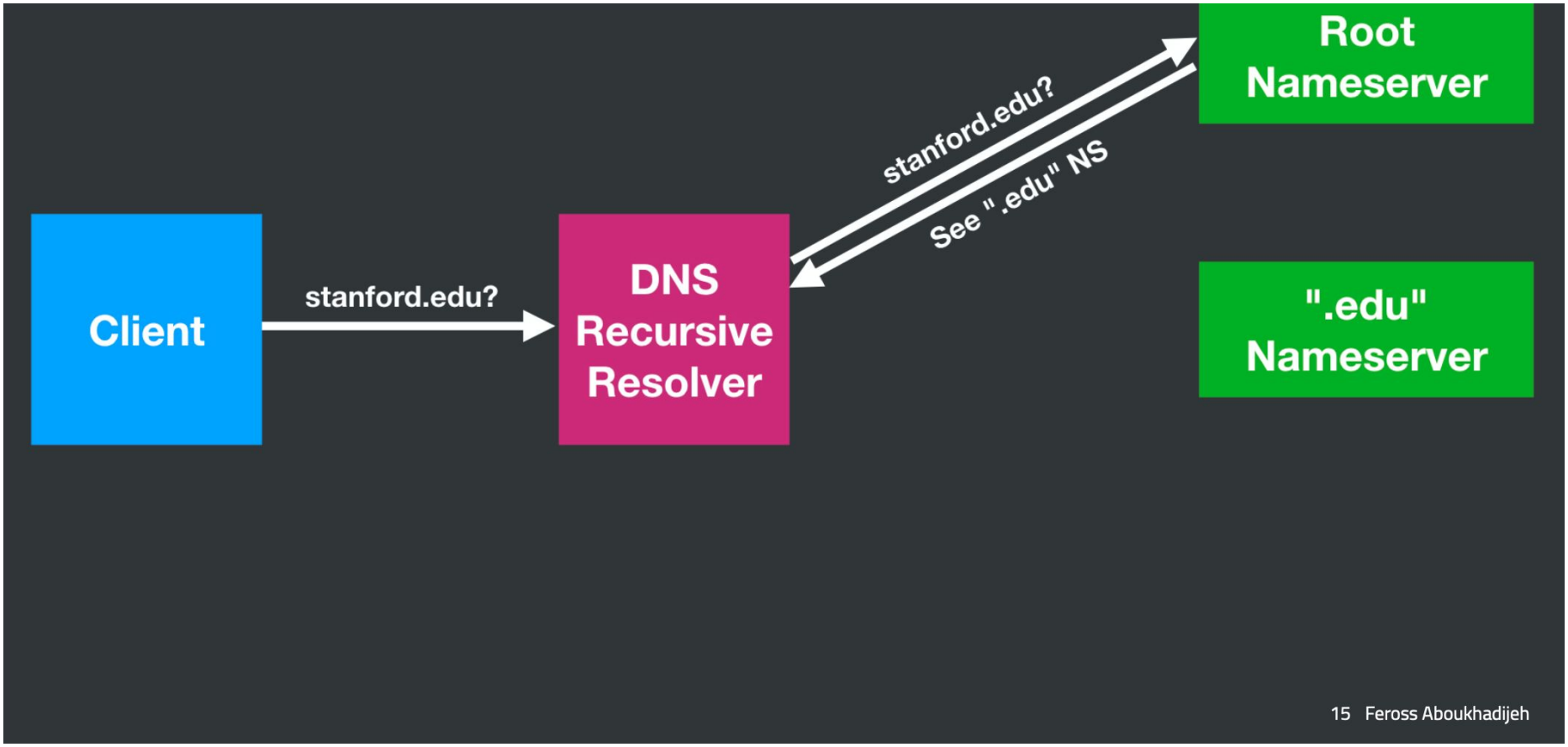
DNS Recursive Resolution



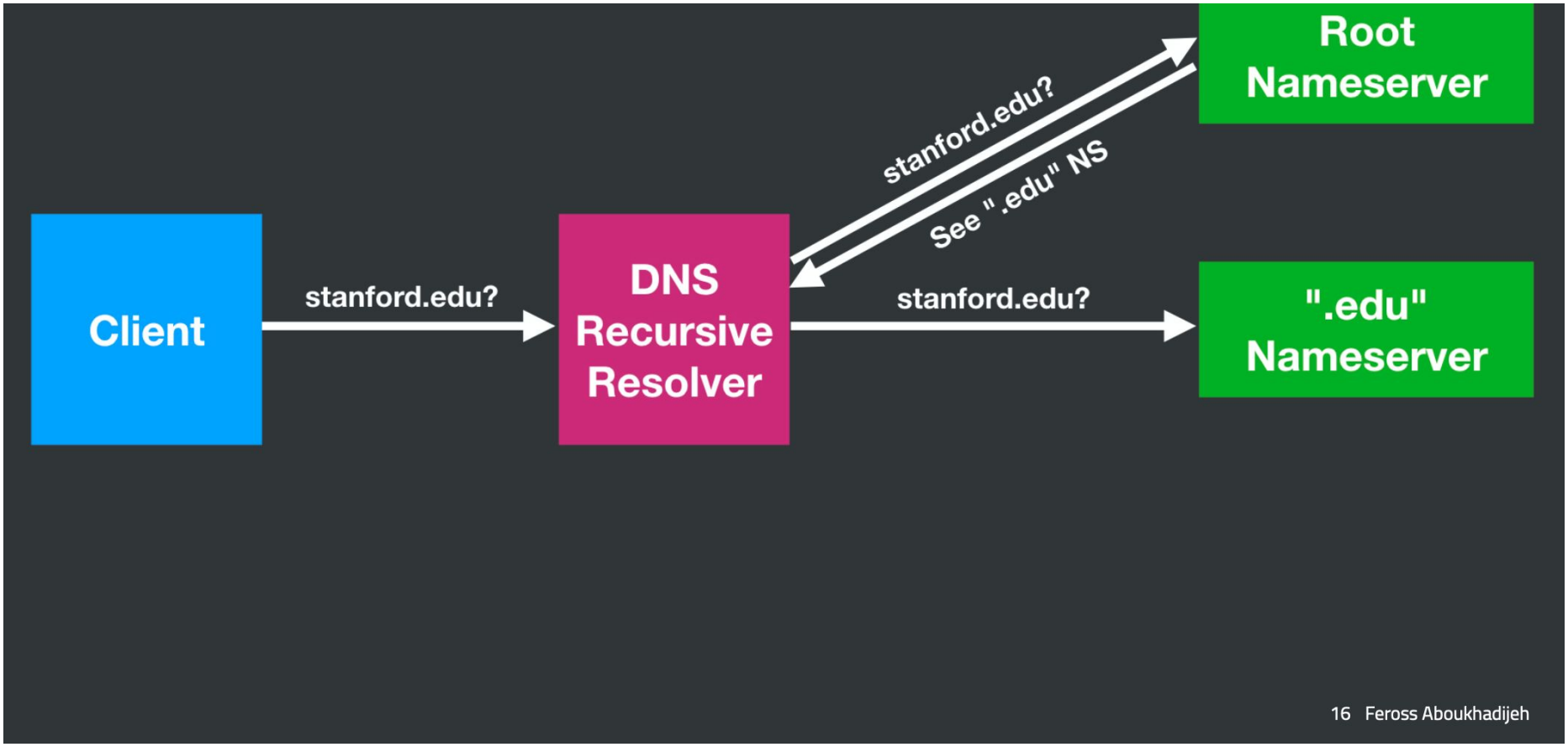
DNS Recursive Resolution



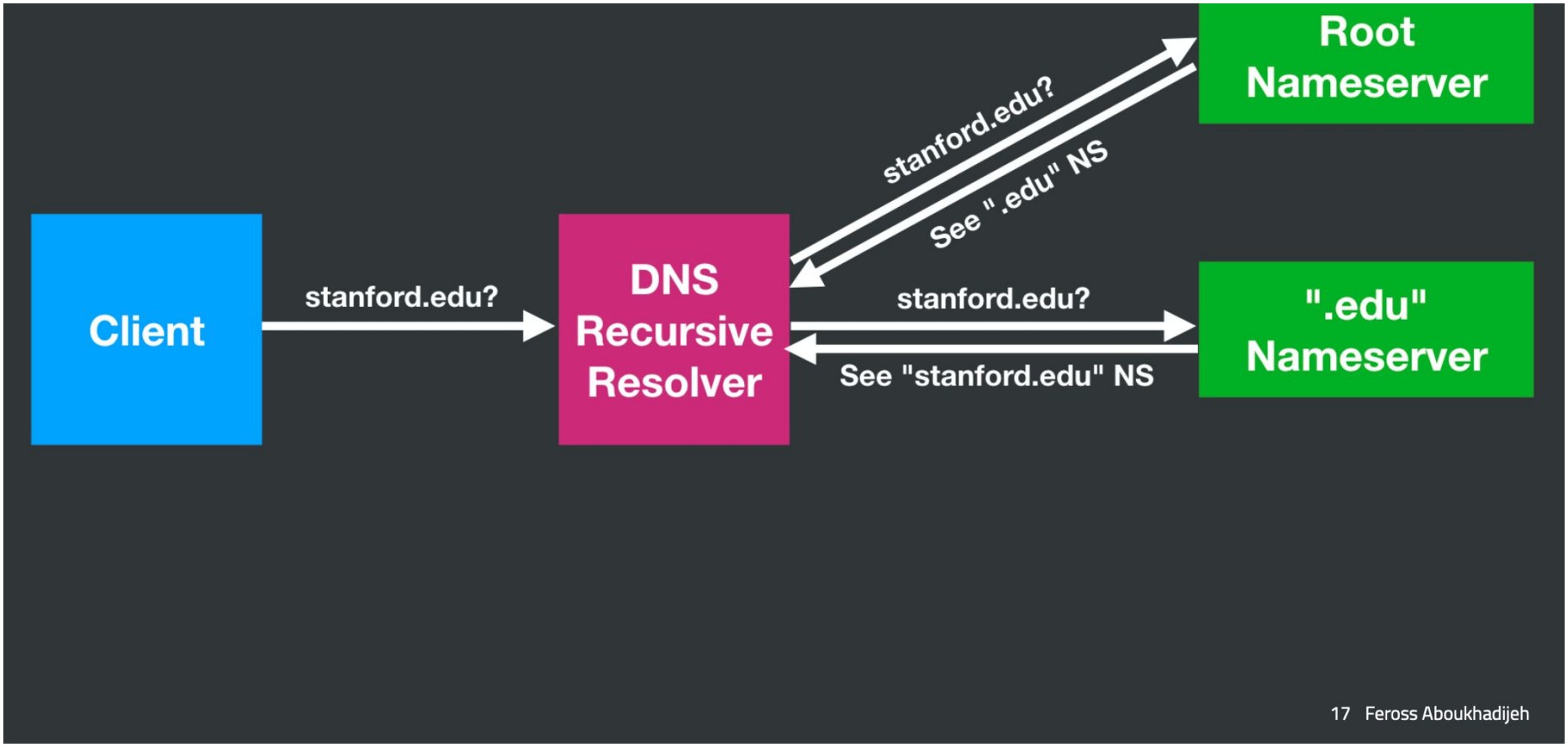
DNS Recursive Resolution



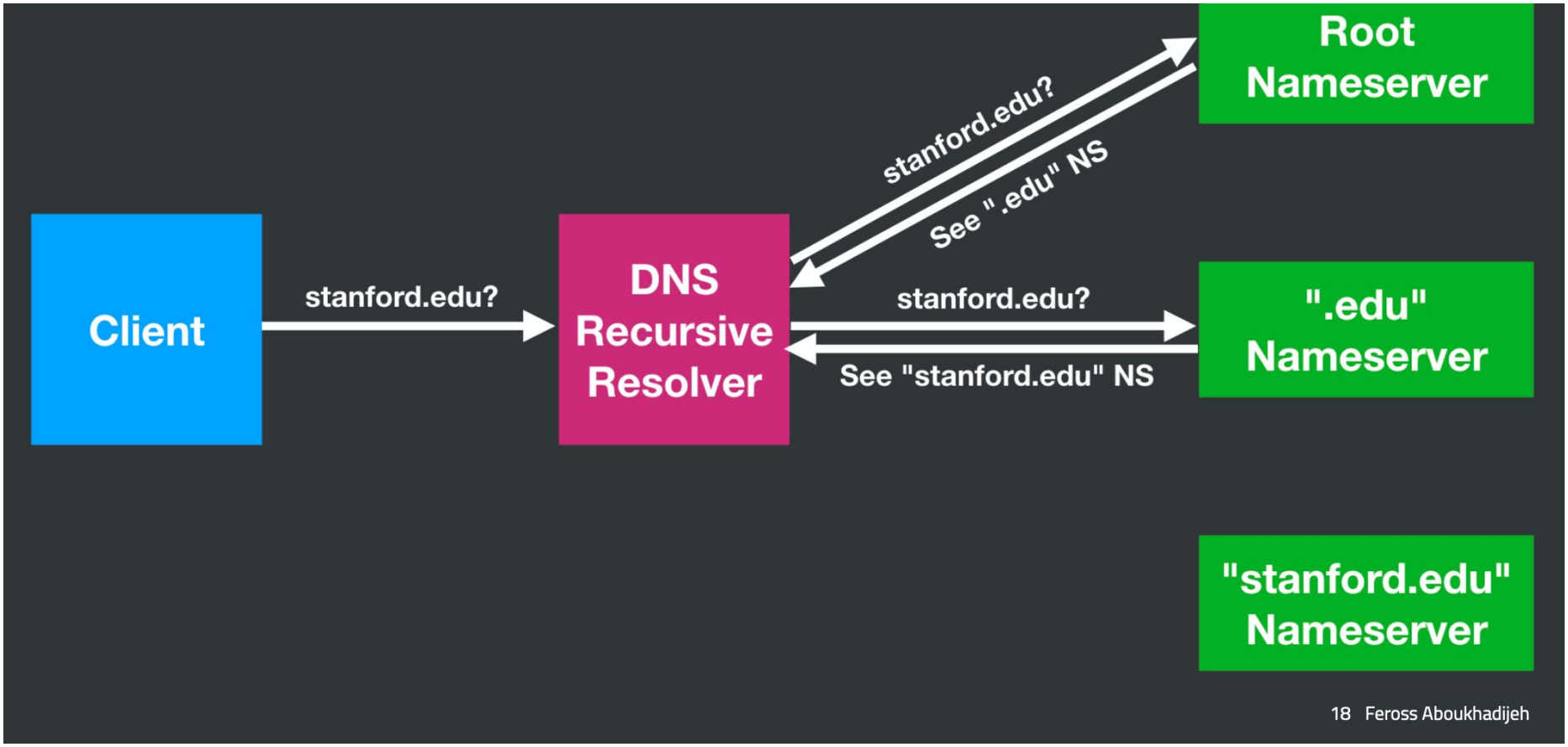
DNS Recursive Resolution



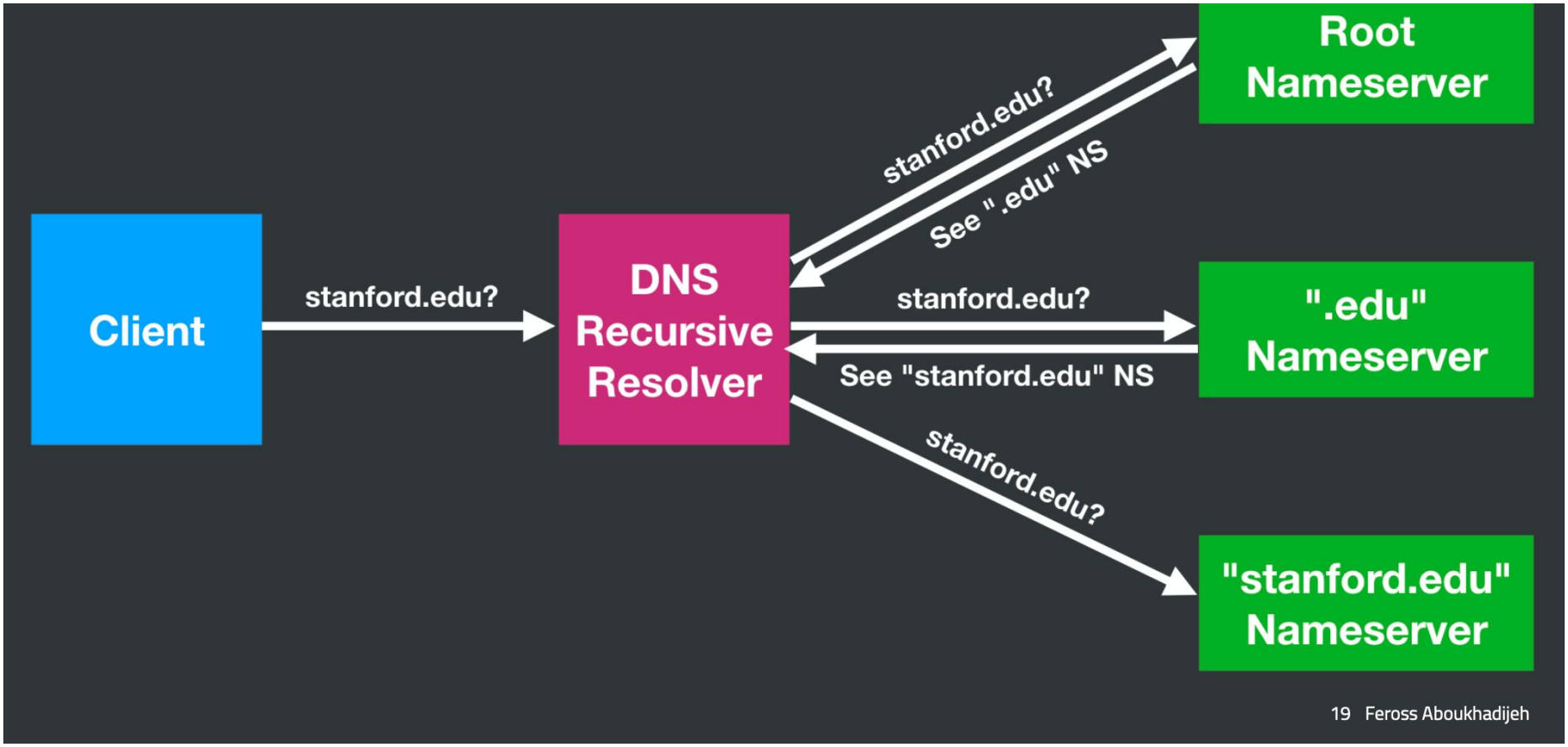
DNS Recursive Resolution



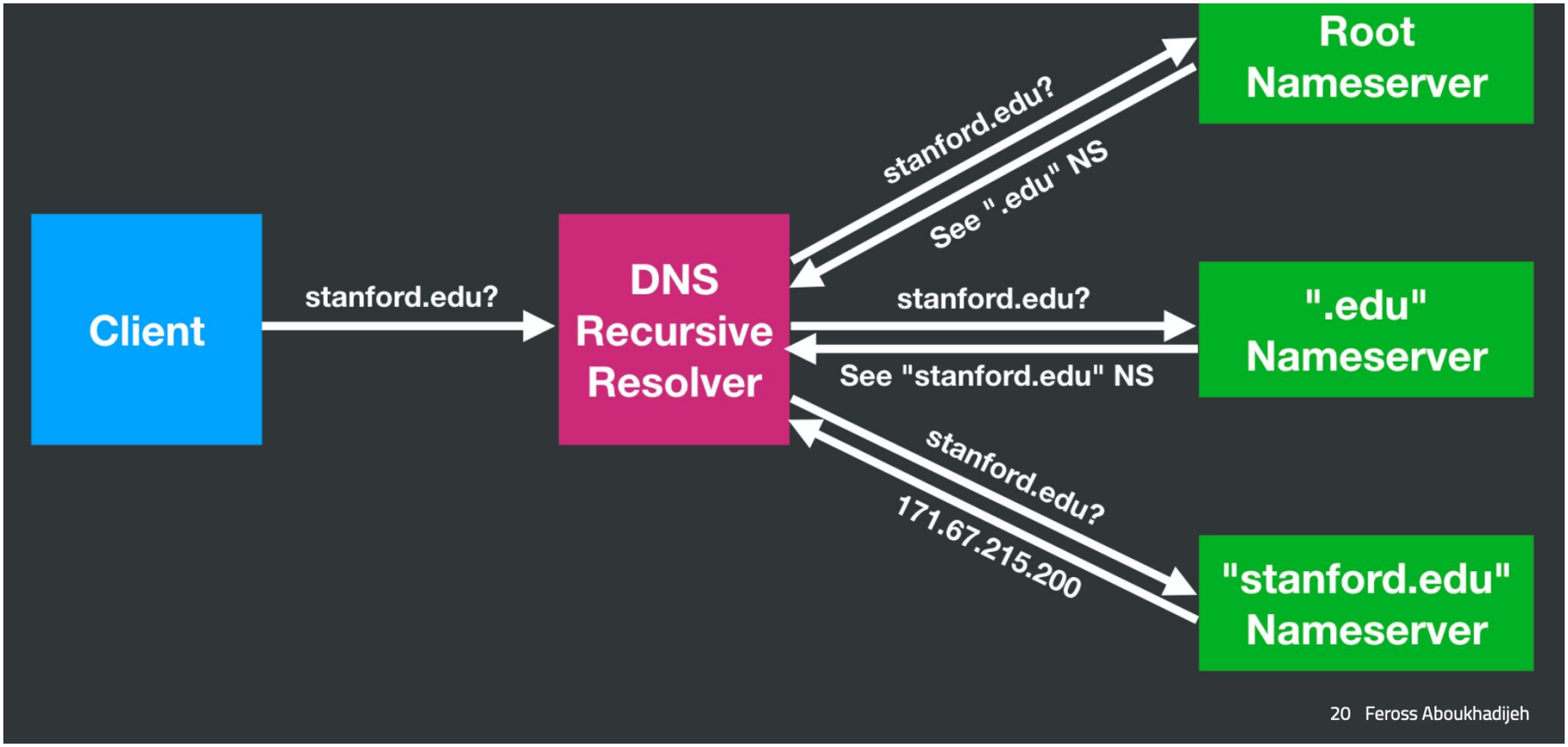
DNS Recursive Resolution



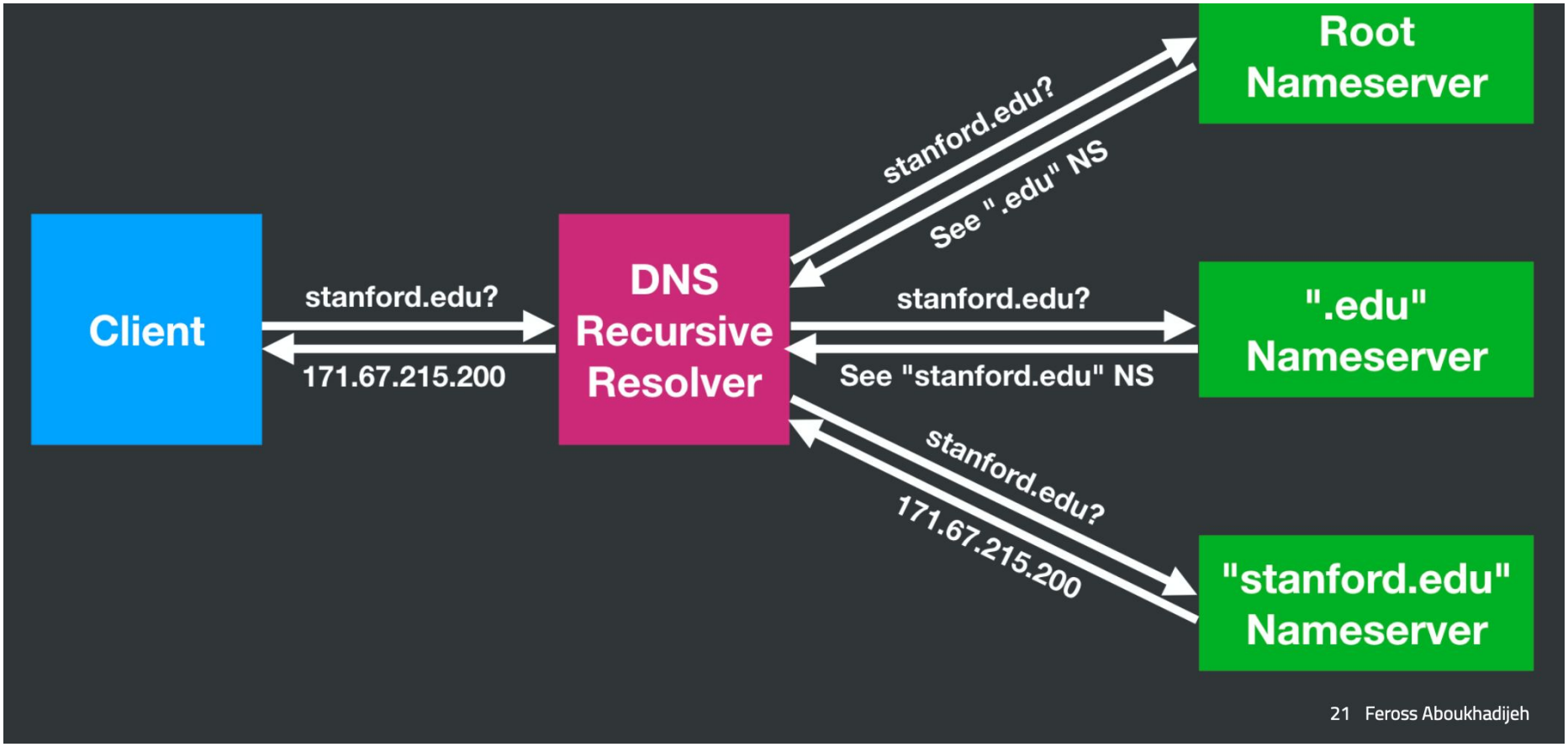
DNS Recursive Resolution



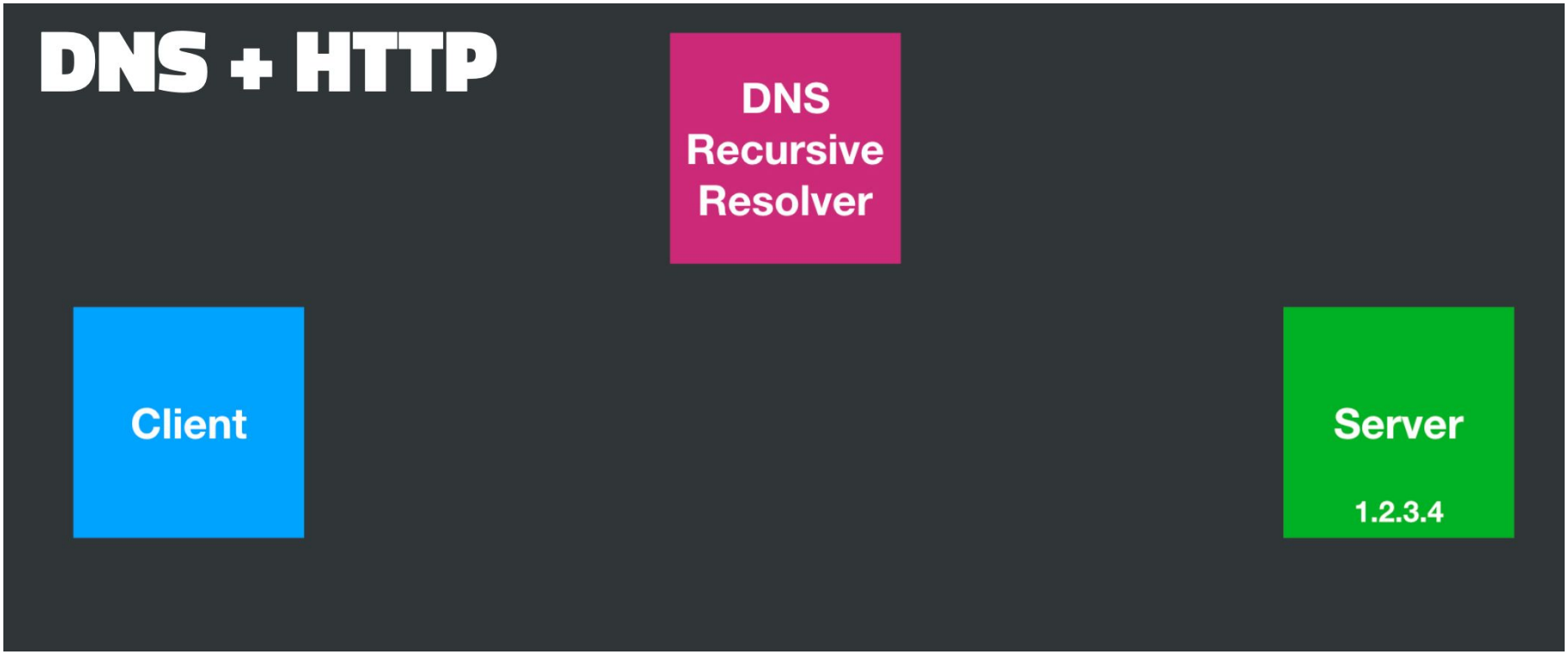
DNS Recursive Resolution



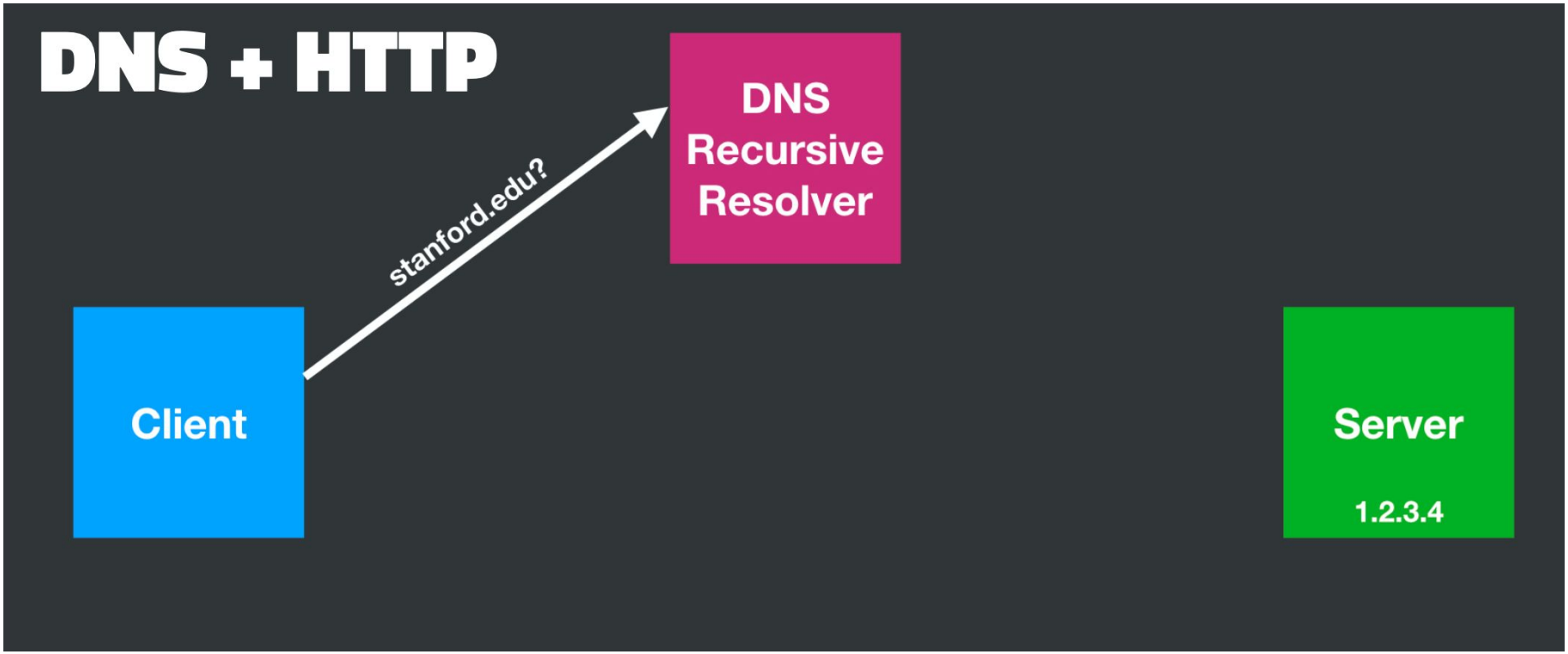
DNS Recursive Resolution



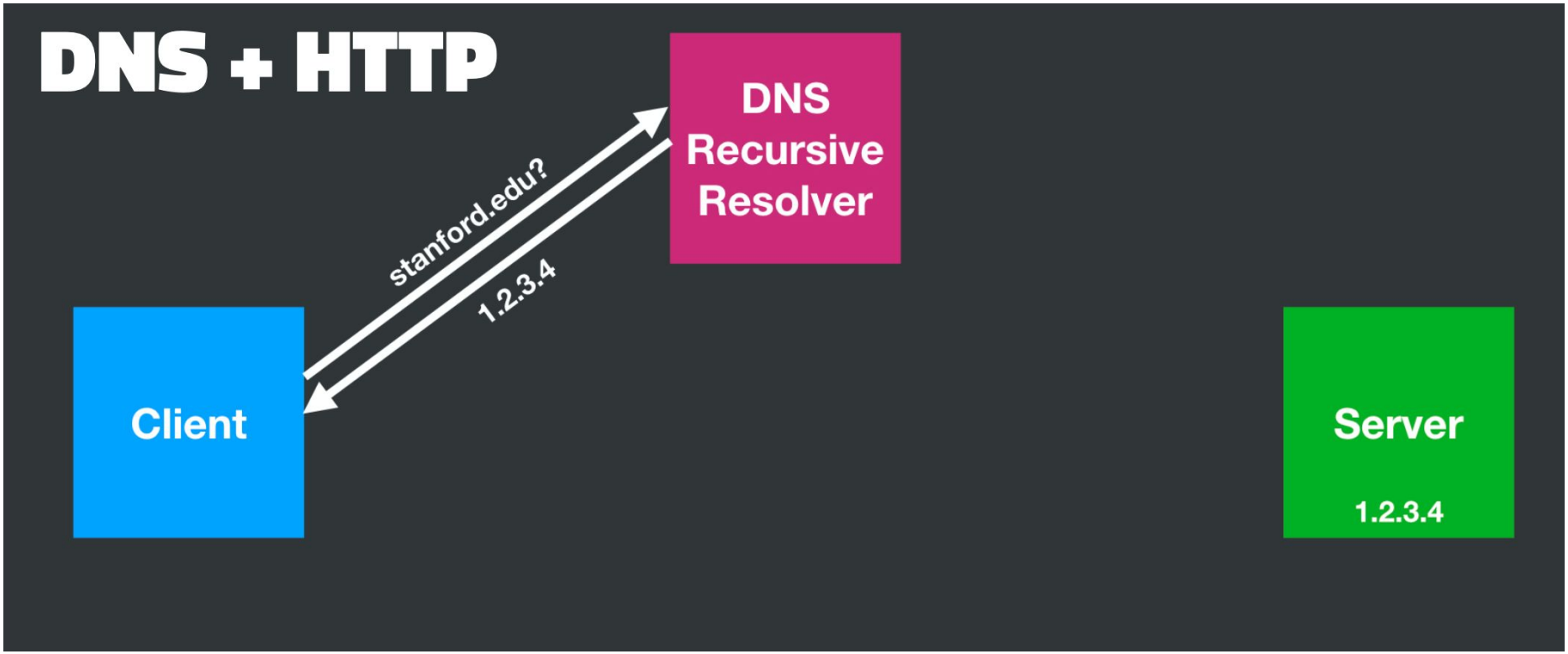
DNS Resolution Before HTTP Connections



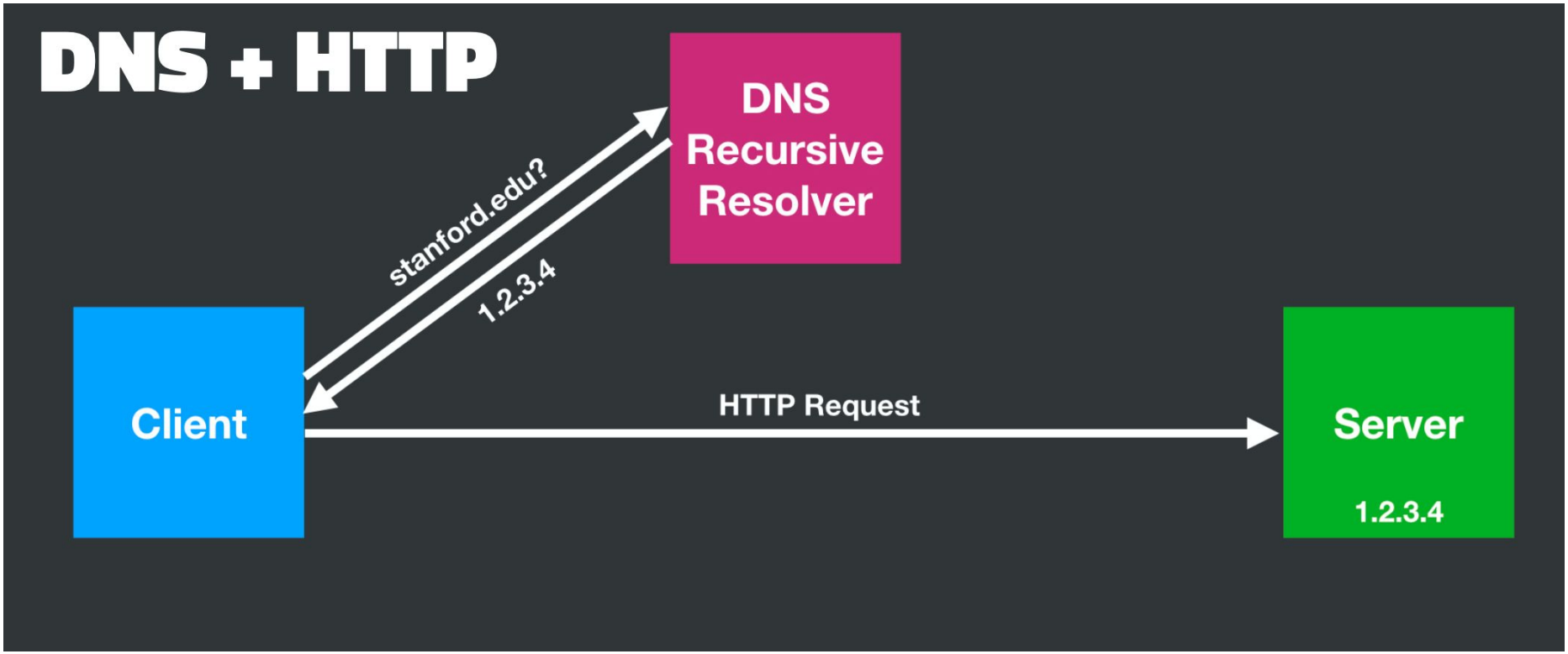
DNS Resolution Before HTTP Connections



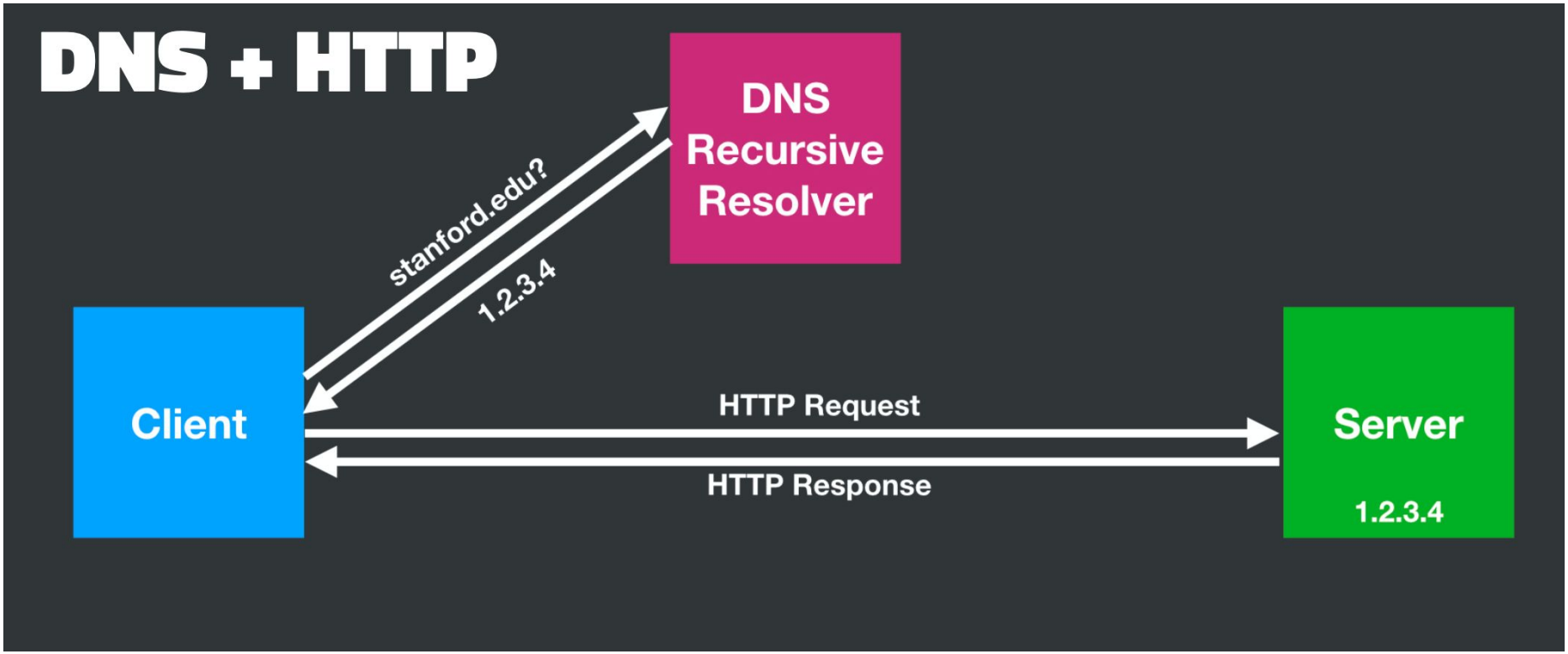
DNS Resolution Before HTTP Connections



DNS Resolution Before HTTP Connections



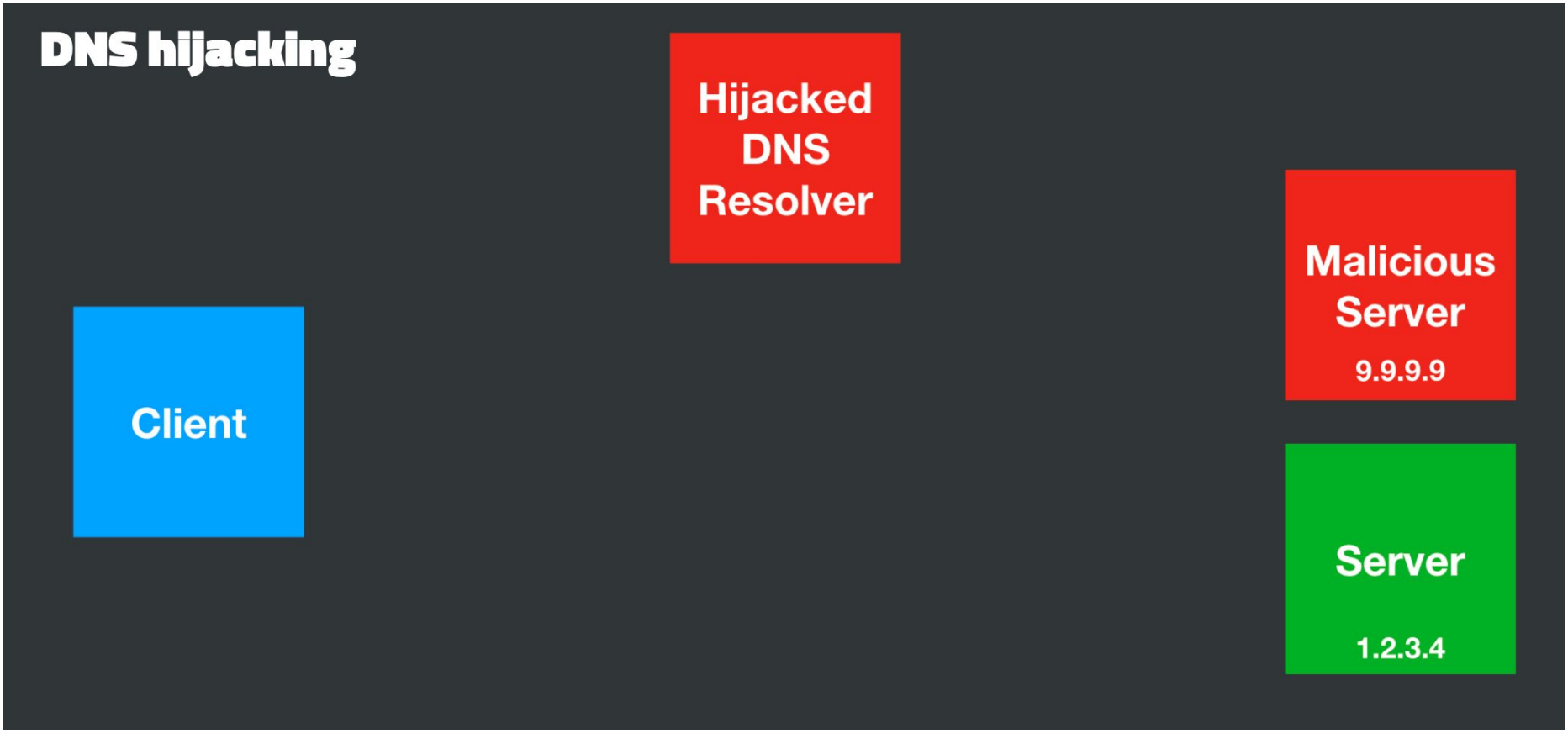
DNS Resolution Before HTTP Connections



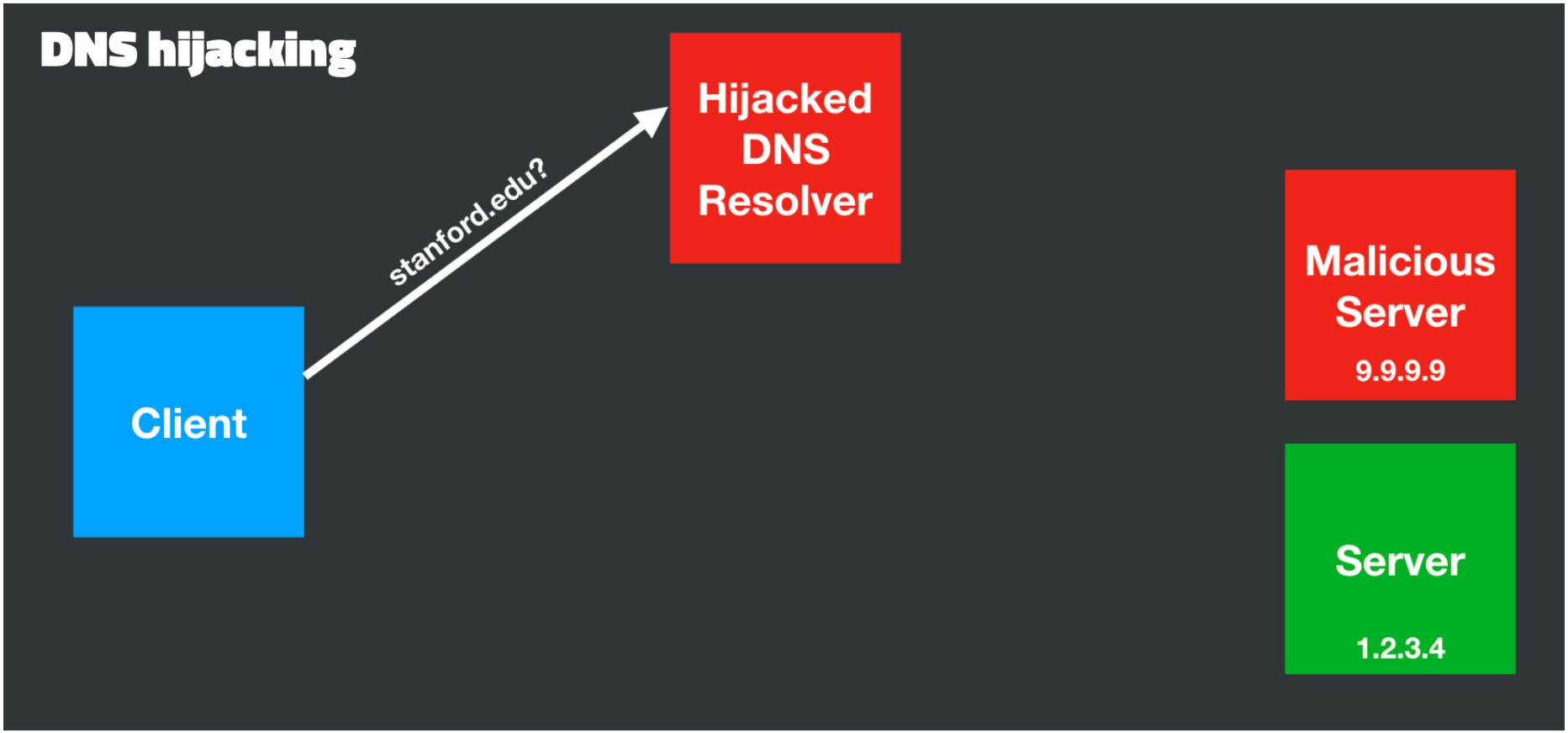
DNS Hijacking

- Attacker changes DNS records of target to point to own IP address
- All site visitors are directed to attacker's web server
- Motivation
  - Phishing
  - Revenue through ads, cryptocurrency mining, etc.
- How do they do it?

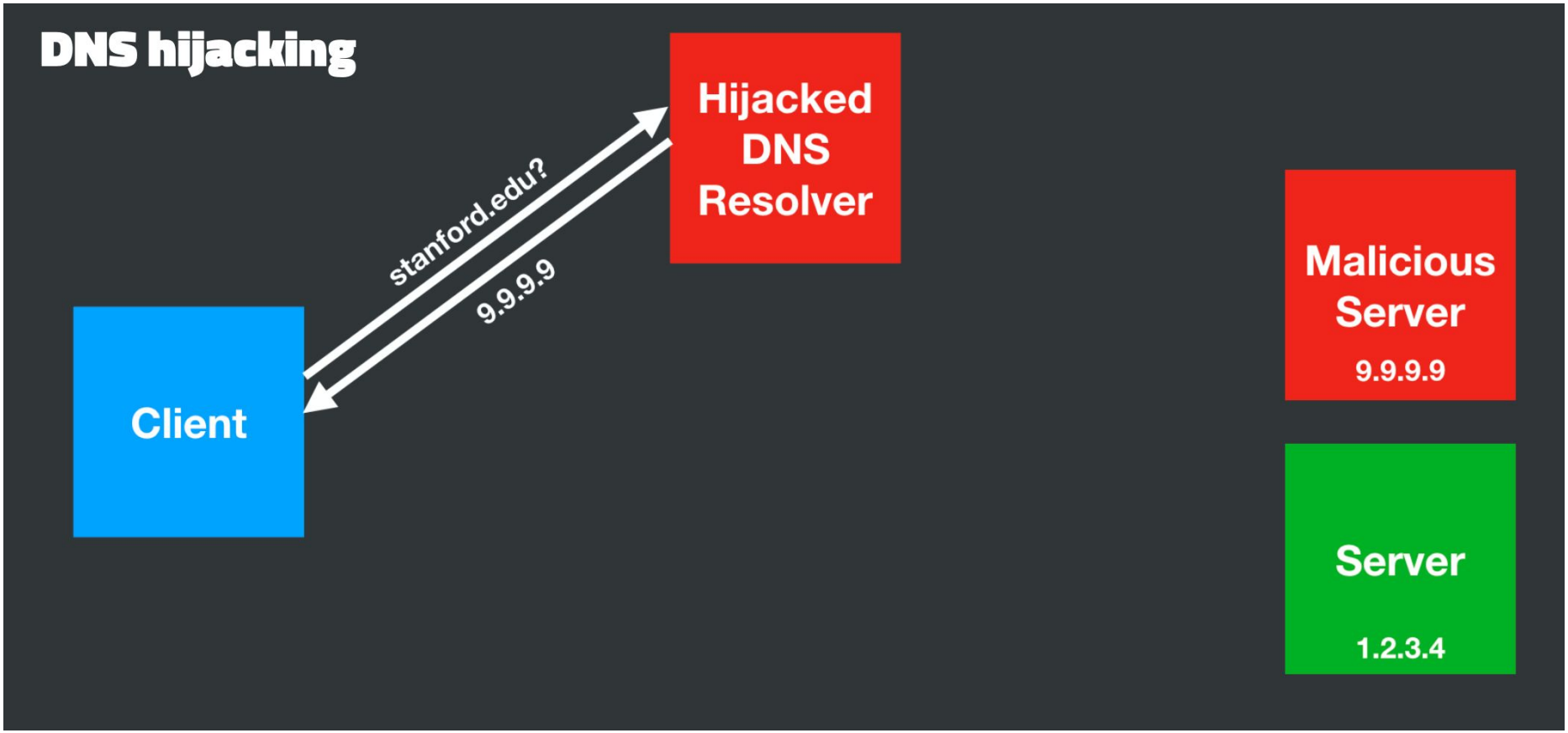
DNS Hijacking



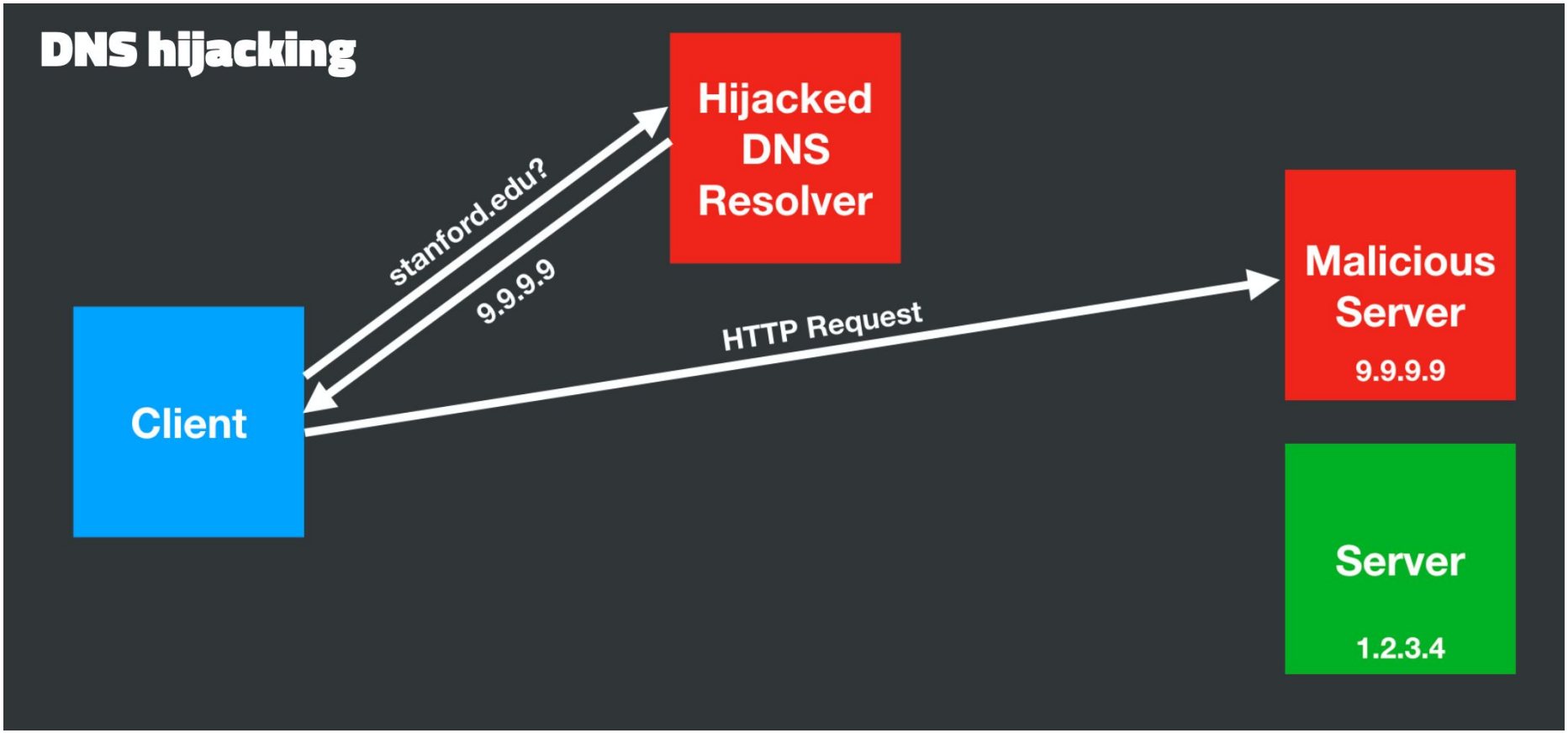
DNS Hijacking



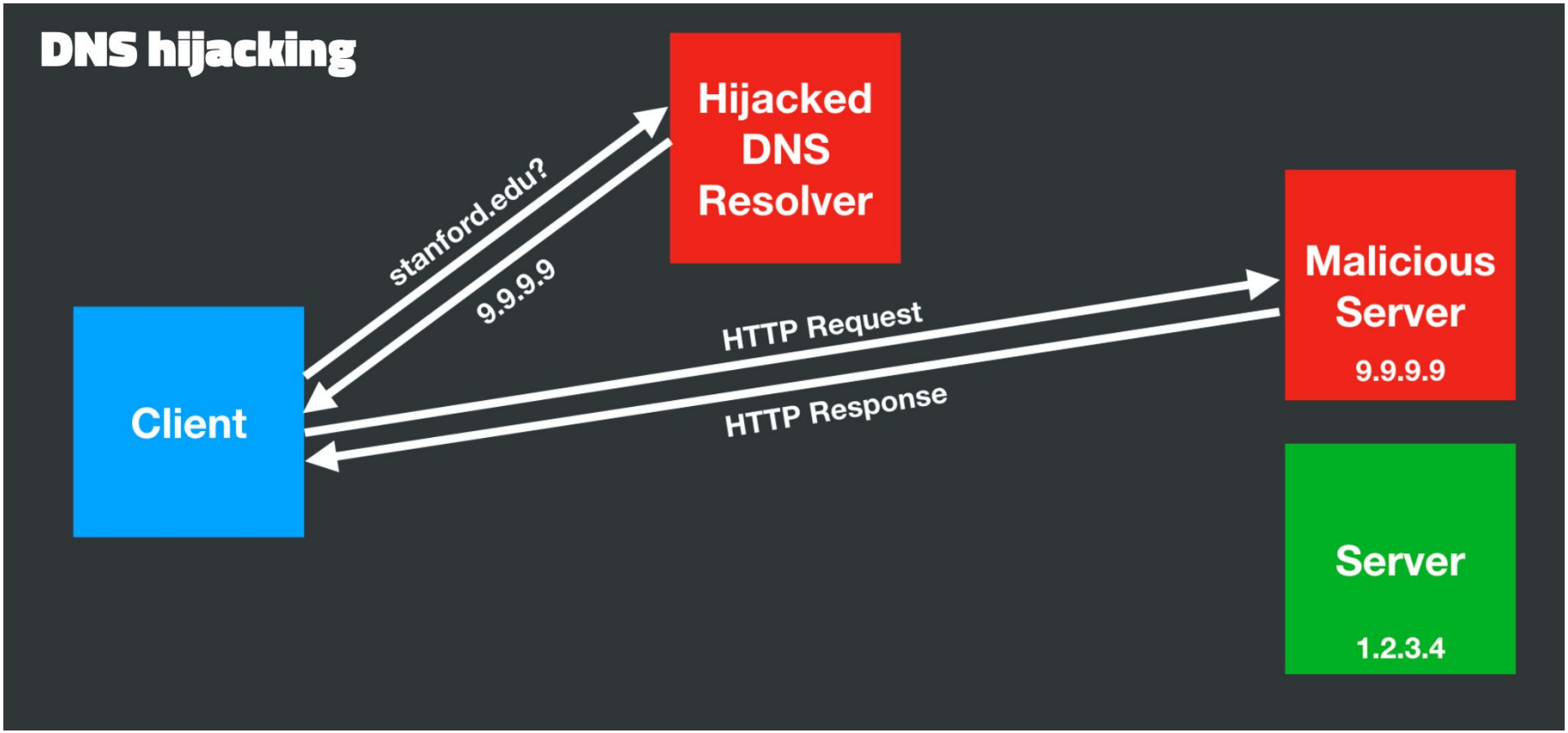
DNS Hijacking



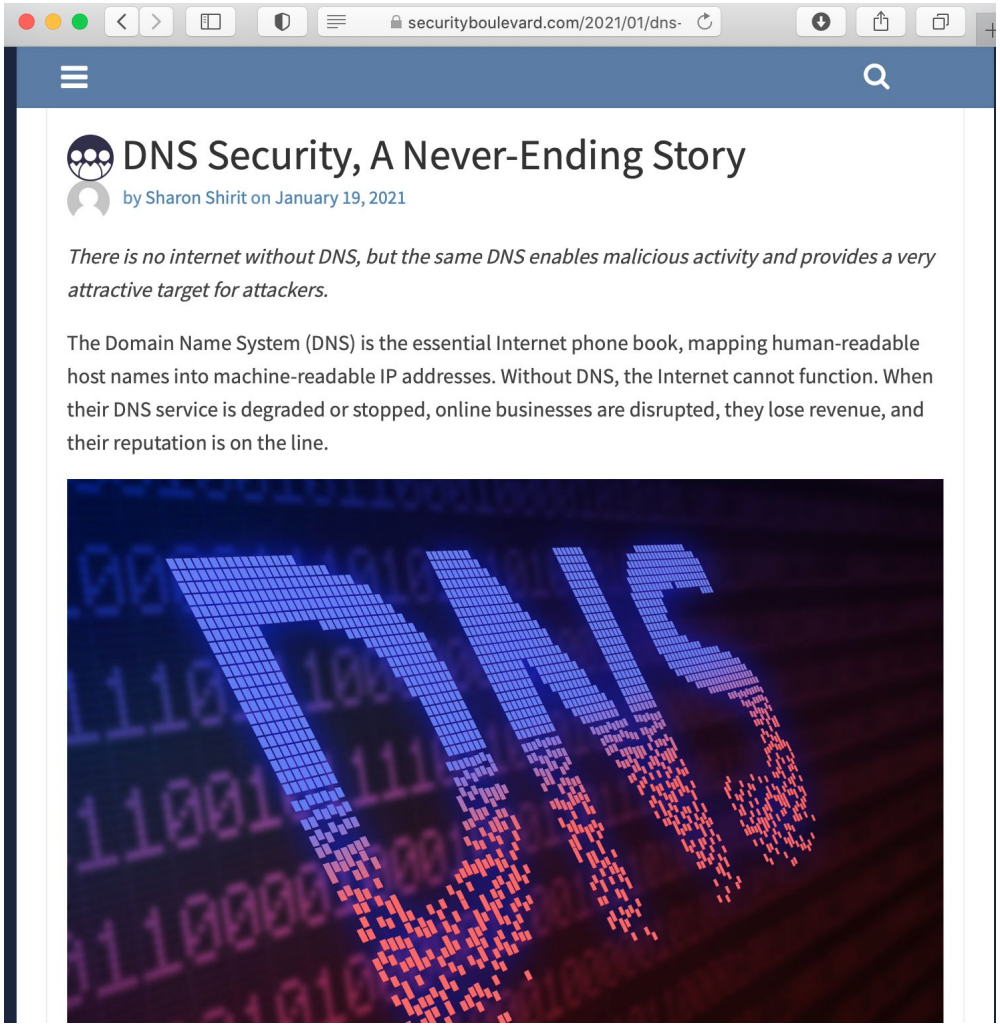
DNS Hijacking



DNS Hijacking

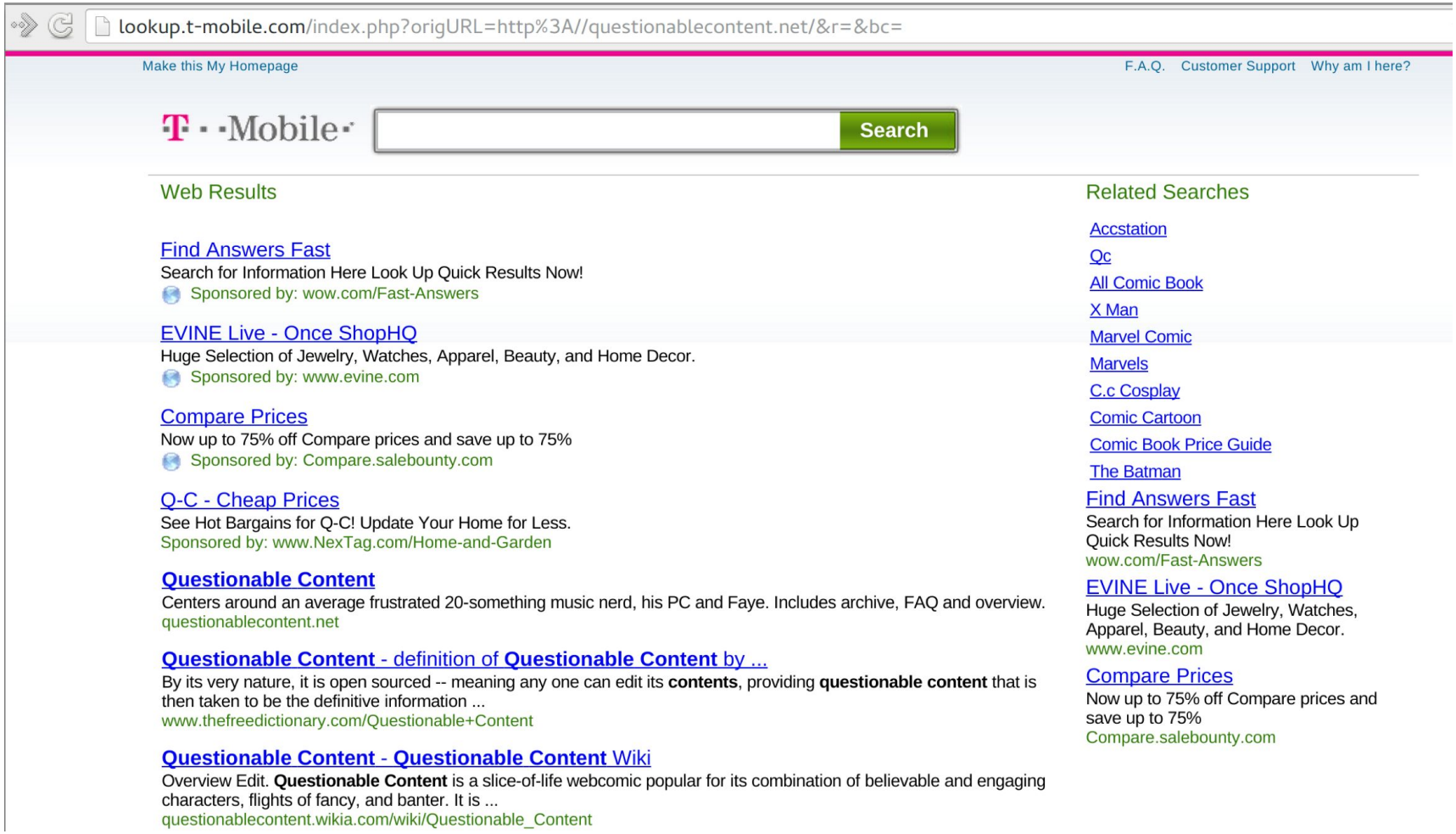


DNS attacks won’t go away



https://securityboulevard.com/2021/01/dns-security-a-never-ending-story/

DNS hijack from your ISP

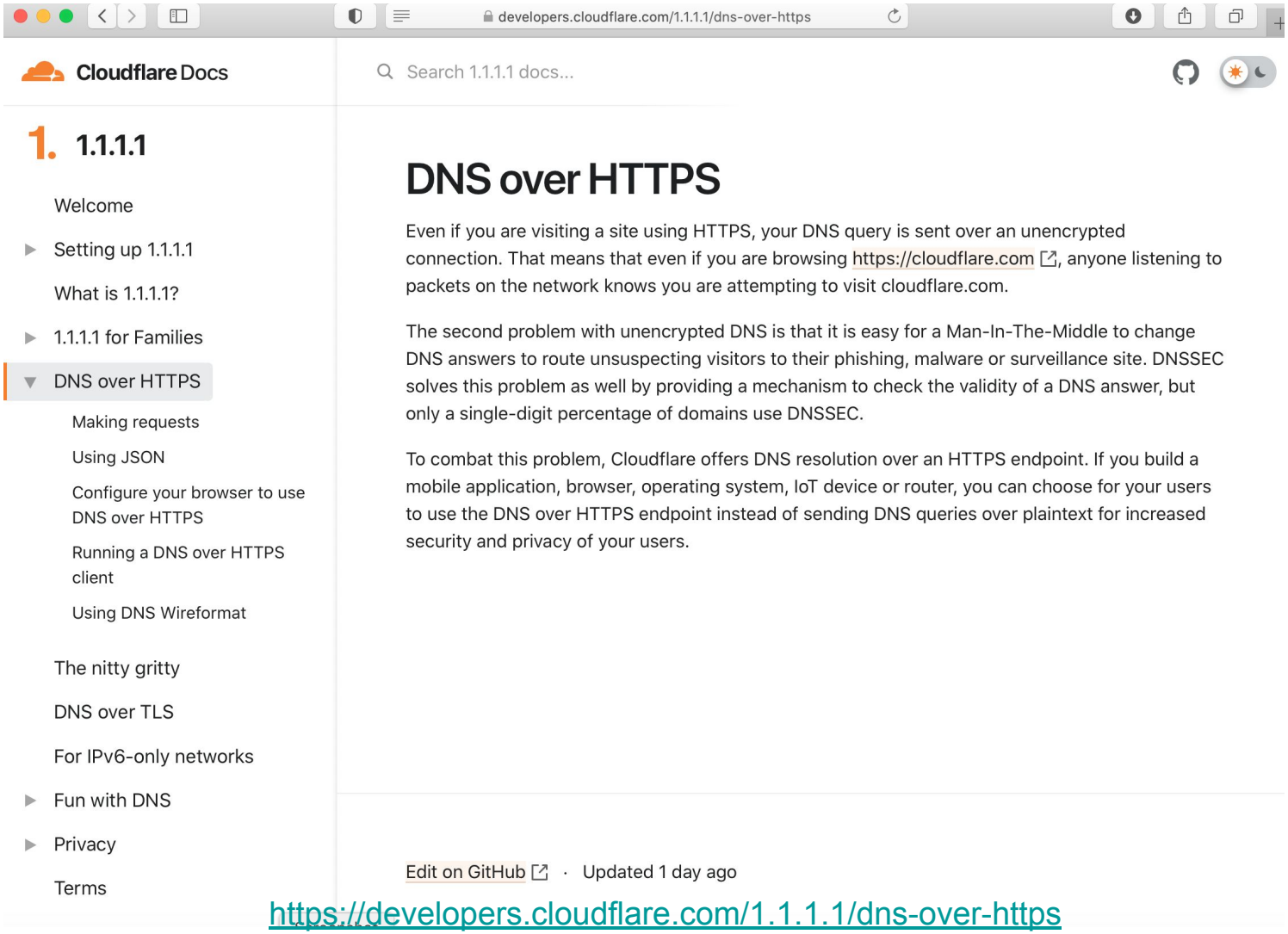


I was going to show an example from cox.net (my home provider), but now it seems like they no longer do this?

DNS Privacy

- Queries are in plaintext
- ISPs have been known to sell this data
- **Pro tip**: Consider switching your DNS settings to Cloudflare (1.1.1.1) or another provider with a good privacy policy

DNS over HTTPS



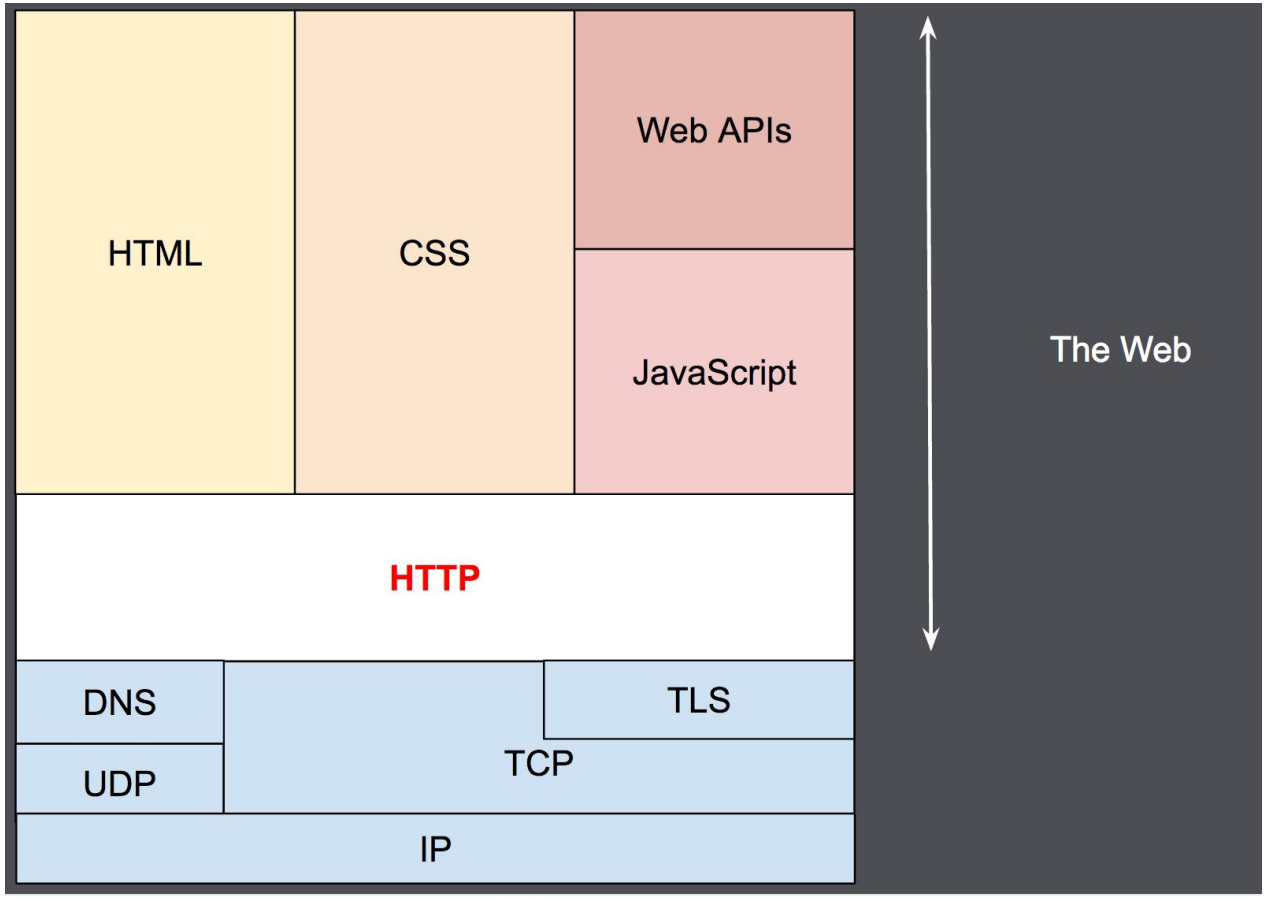
https://developers.cloudflare.com/1.1.1.1/dns-over-https

More curl… curl is important...

```
canis:/home/mln % curl  -v -I https://www.odu.edu/
*   Trying 128.82.112.29 ...
* TCP_NODELAY set
* Connected to www.odu.edu ( 128.82.112.29) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: serialNumber=232982; jurisdictionC=US; jurisdictionST=Virginia; businessCategory=Government Entity;
C=US; postalCode=23529; ST=VA; L=Norfolk; street=4600 Elkhorn Ave; O=Old Dominion University; OU=ITS; OU=COMODO
EV Multi-Domain SSL; CN=www.odu.edu
*  start date: Jun  5 00:00:00 2019 GMT
*  expire date: Jun  4 23:59:59 2021 GMT
*  subjectAltName: host "www.odu.edu" matched cert's "www.odu.edu"
*  issuer: C=GB; ST=Greater Manchester; L=Salford; O=COMODO CA Limited; CN=COMODO RSA Extended Validation Secure
Server CA
*  SSL certificate verify ok.
> HEAD / HTTP/1.1
> Host: www.odu.edu
> User-Agent: curl/7.58.0
> Accept: */*
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Date: Thu, 28 Jan 2021 16:52:04 GMT
Date: Thu, 28 Jan 2021 16:52:04 GMT
< Server: Apache/2.4.6 (Red Hat Enterprise Linux)
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
< Vary: Host
Vary: Host
< Accept-Ranges: bytes
Accept-Ranges: bytes
< Connection: close
Connection: close
```

TCP
DNS
TLS
HTTP Request
HTTP Response

We now have a better definition of what is “the web”

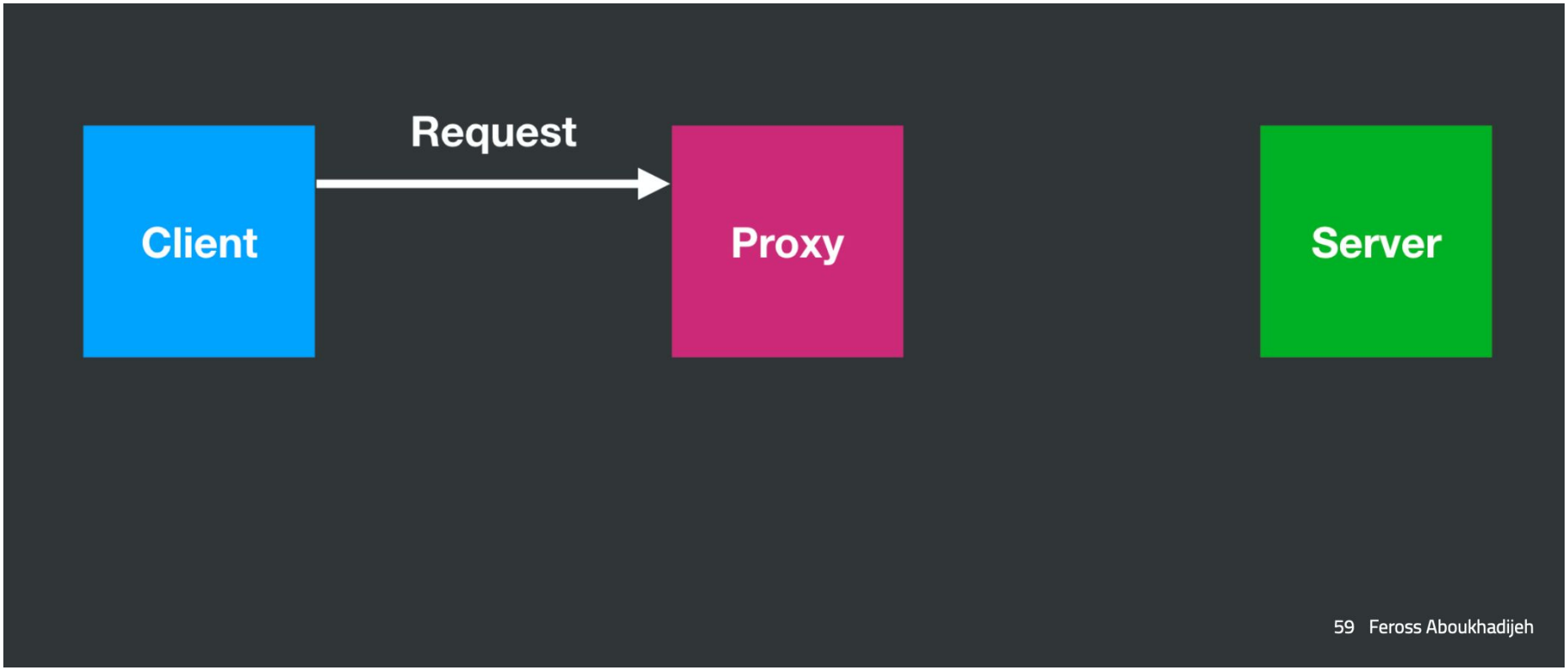


https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview

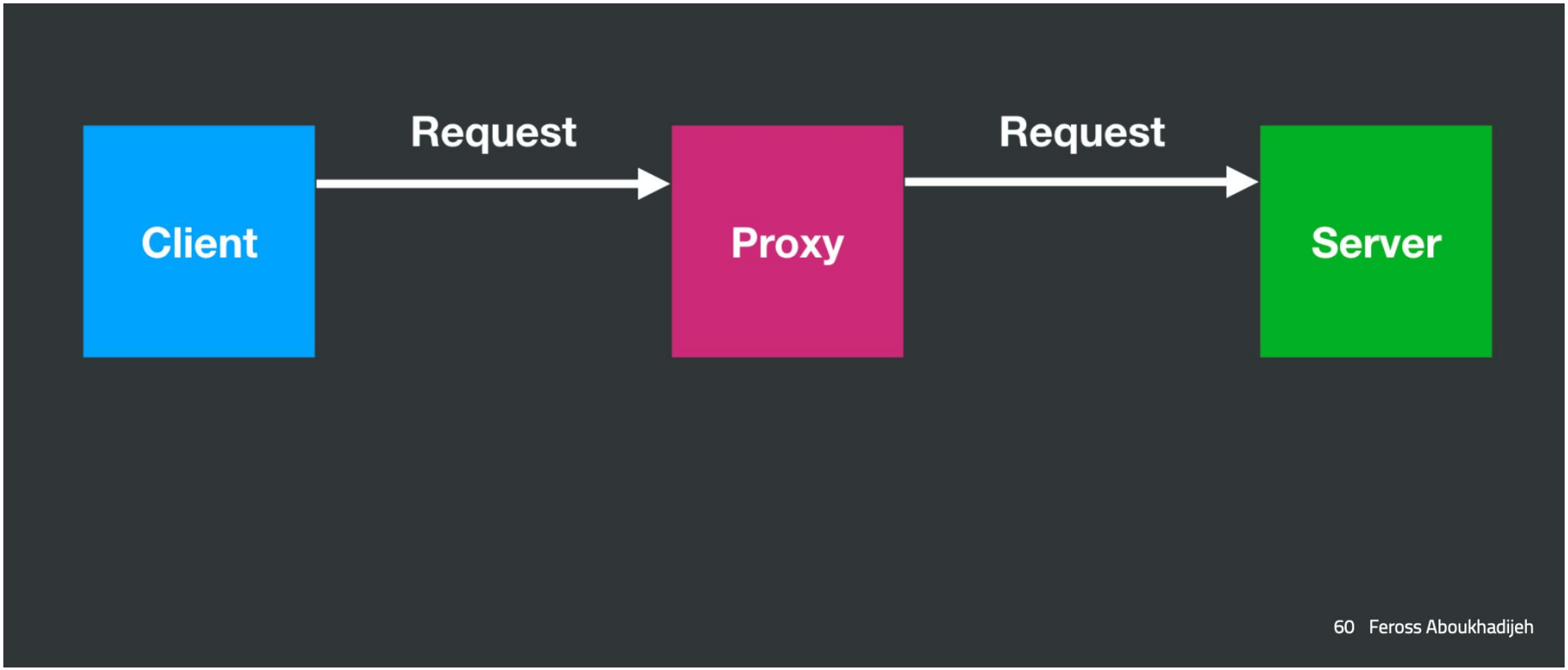
HTTP Proxies

- Can cache content
- Can block content (e.g., malware, adult content)
- Can modify content
- Can sit in front of many servers ("reverse proxy")

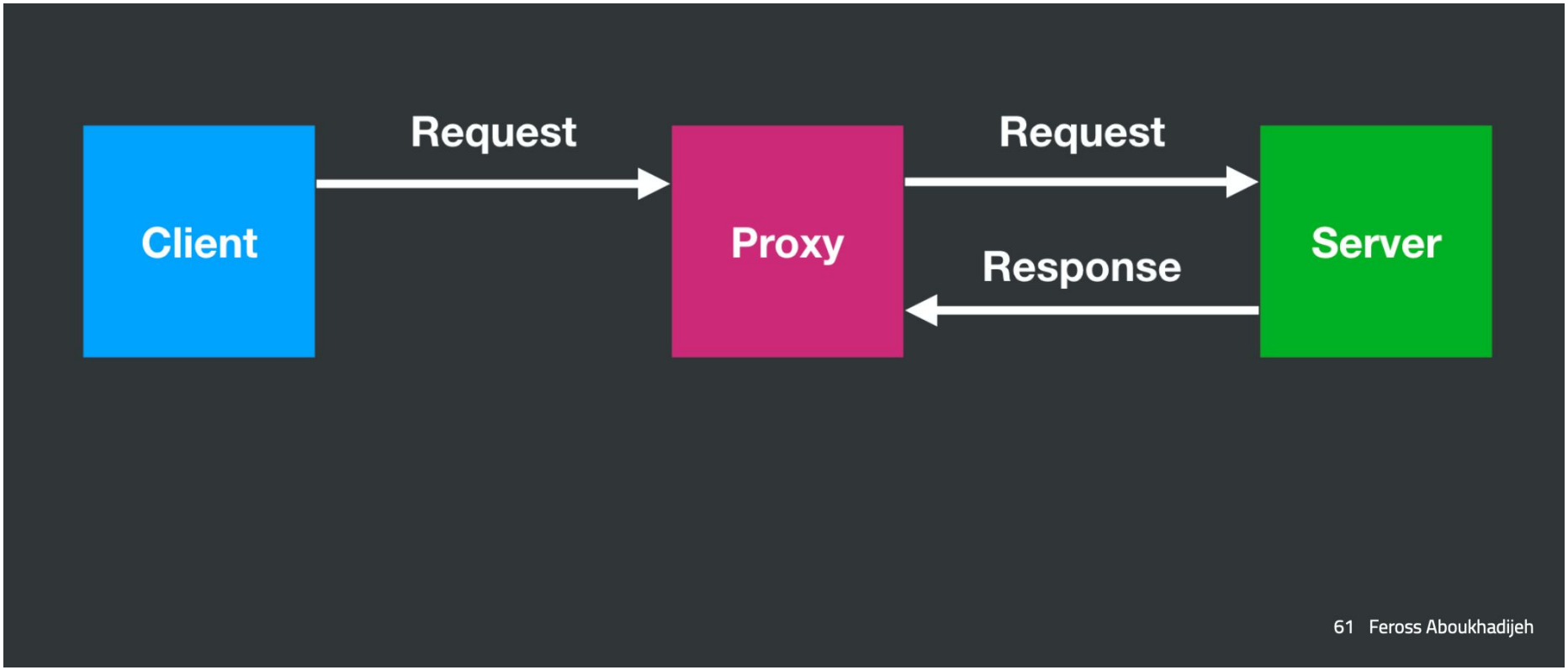
HTTP with a proxy server



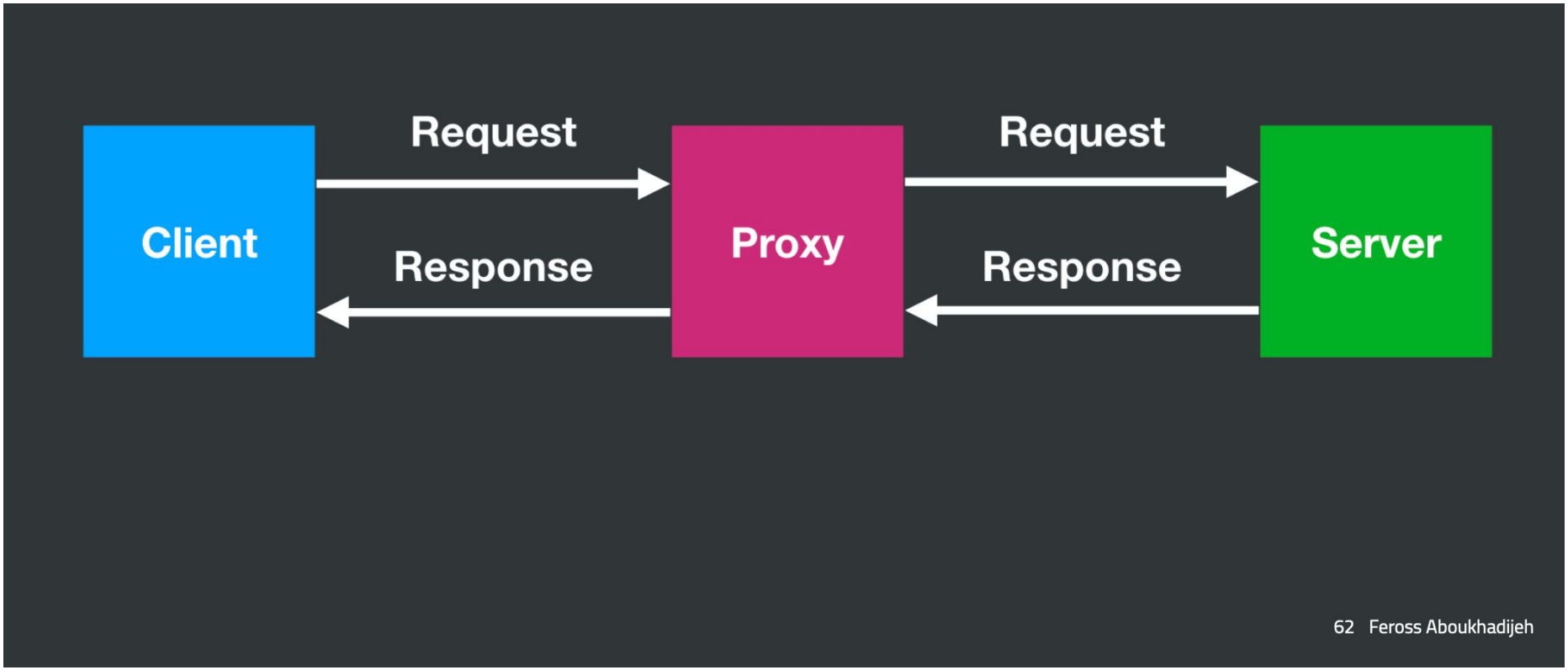
HTTP with a proxy server



HTTP with a proxy server



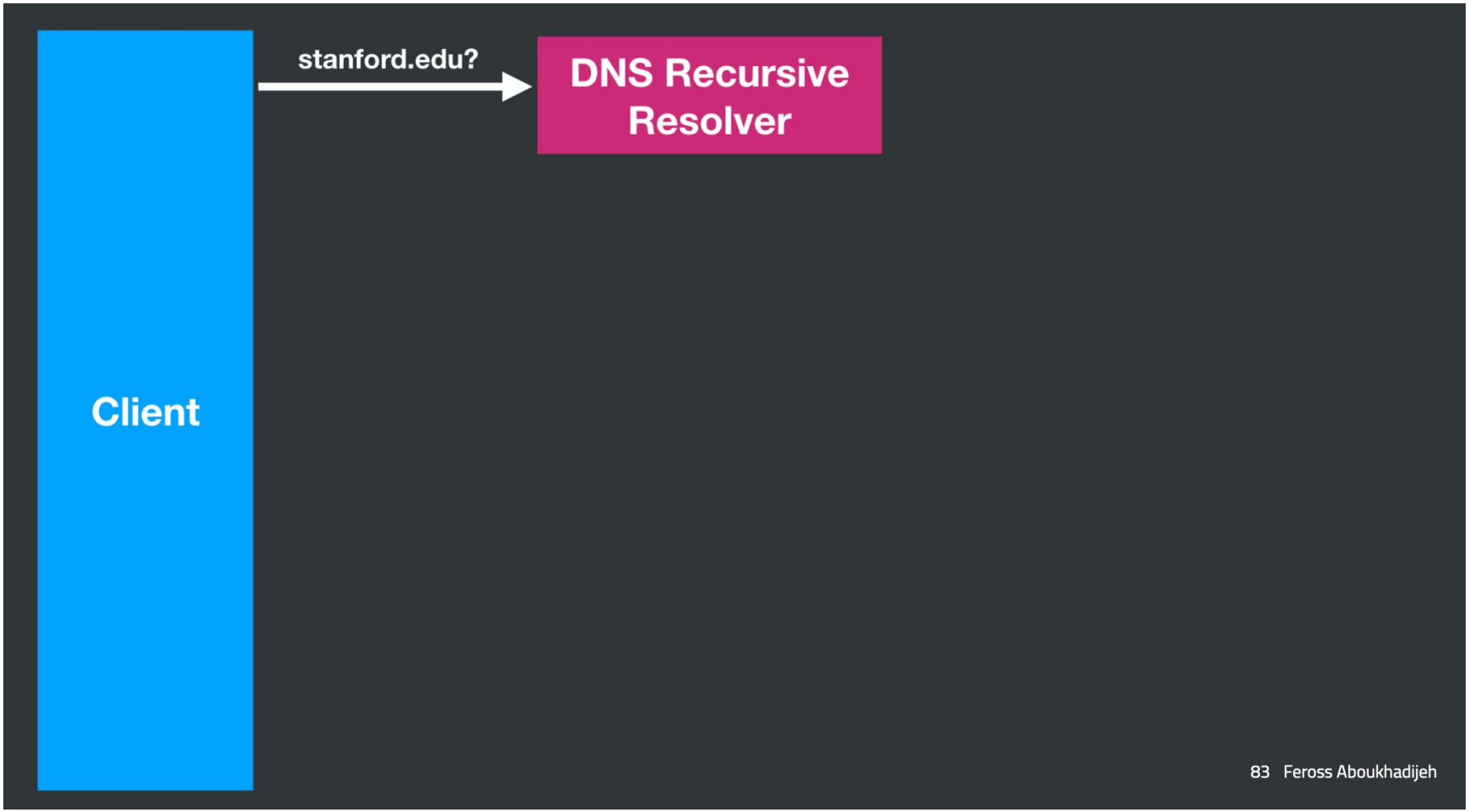
HTTP with a proxy server



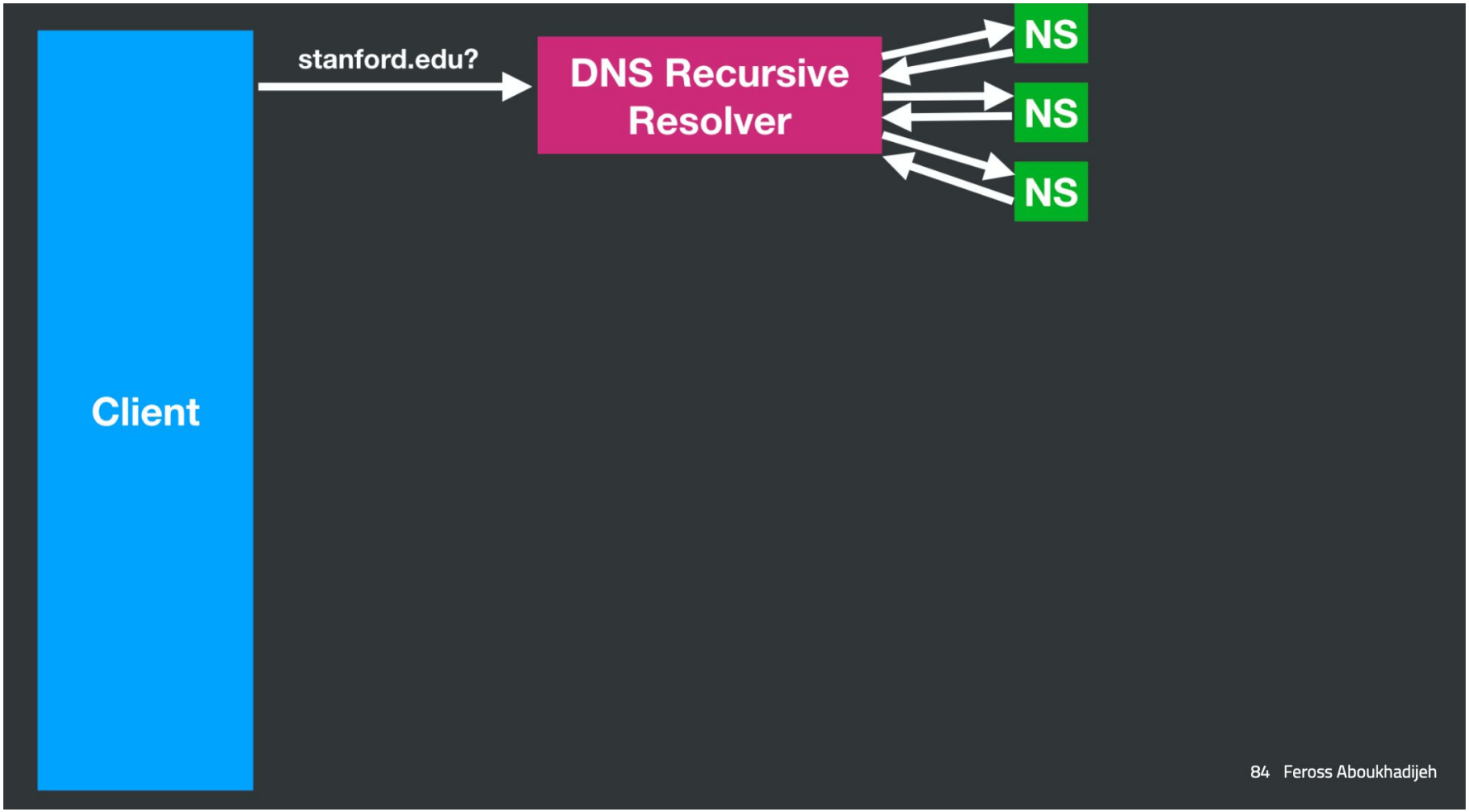
Web stack

Putting it together

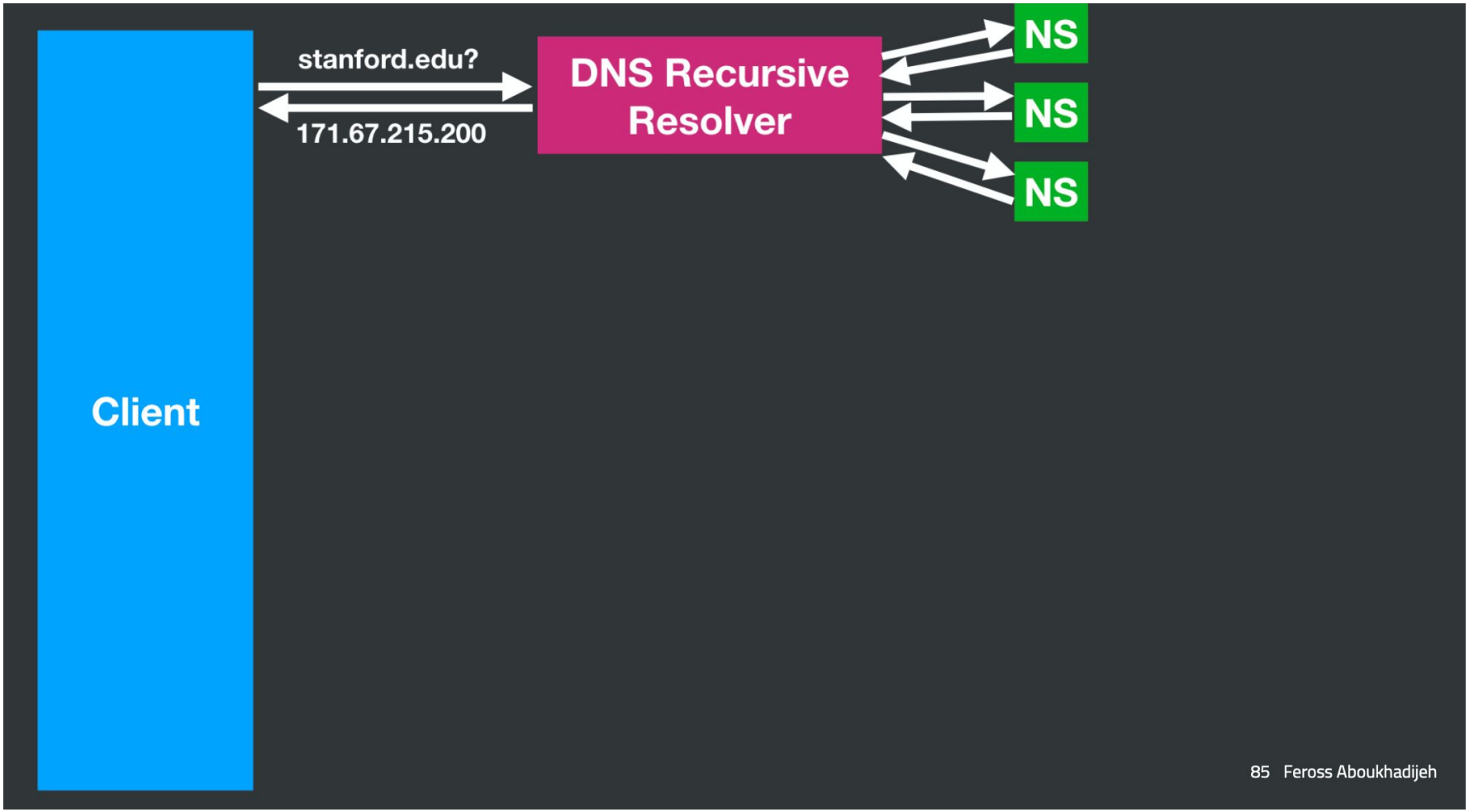
Putting it together



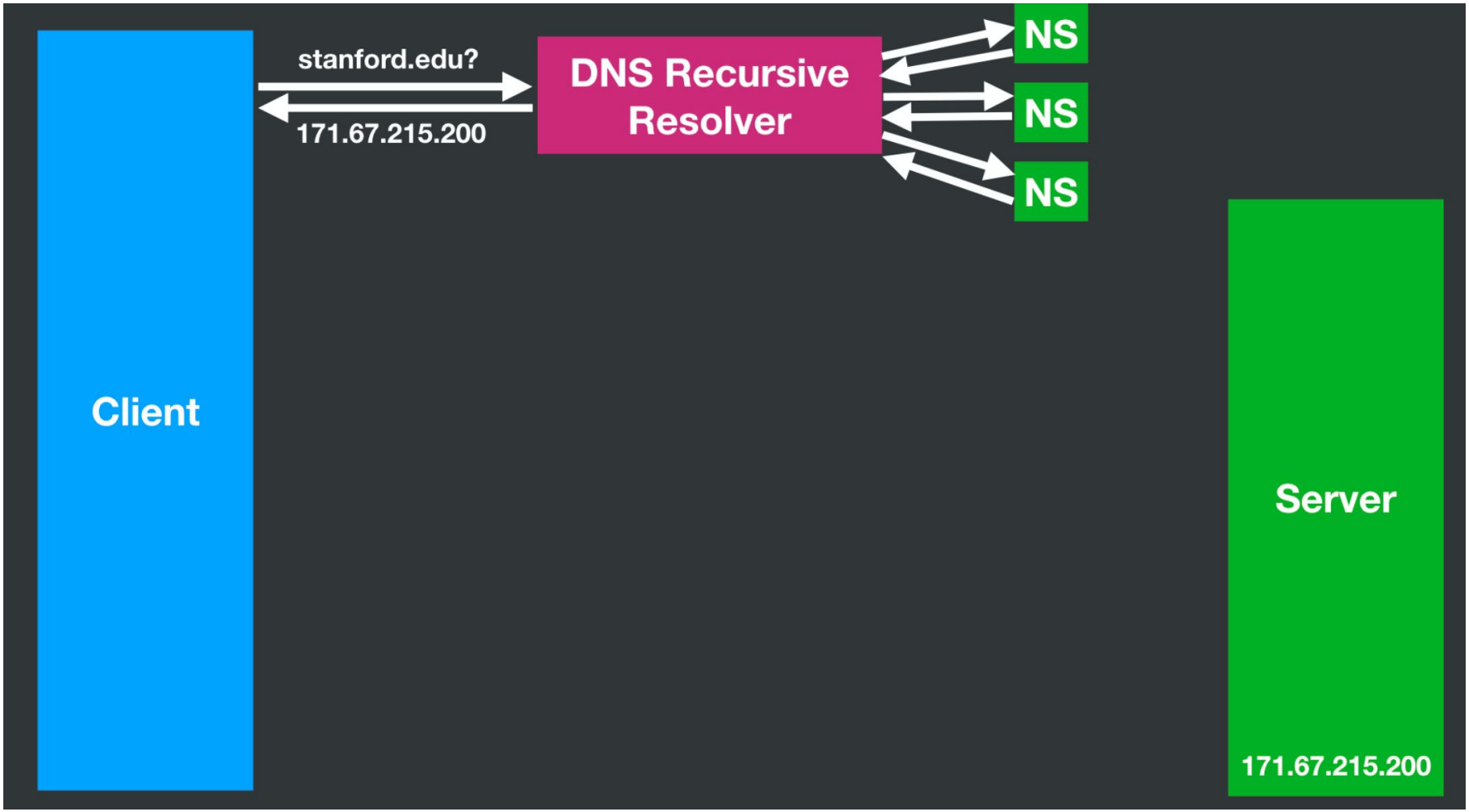
Putting it together



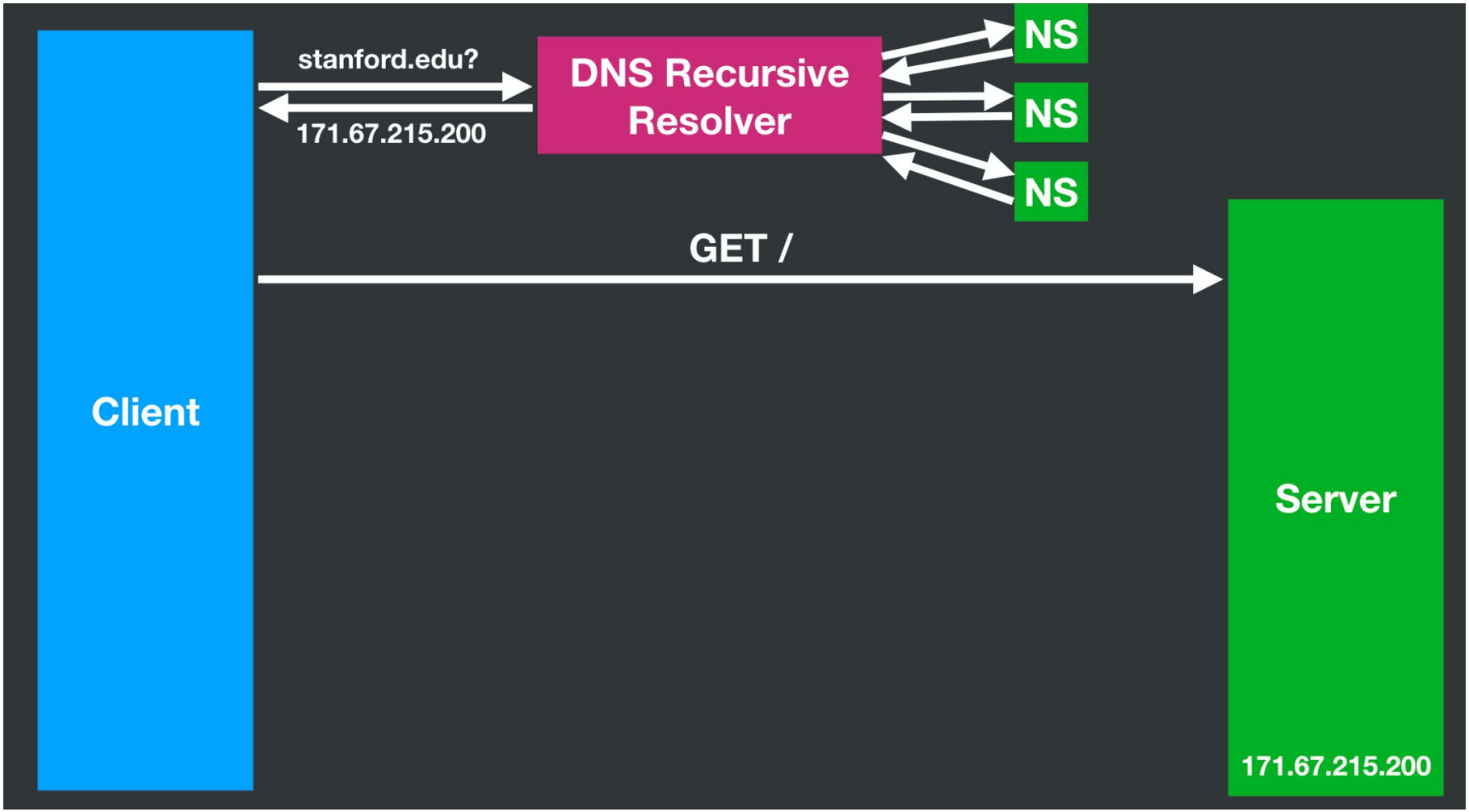
Putting it together



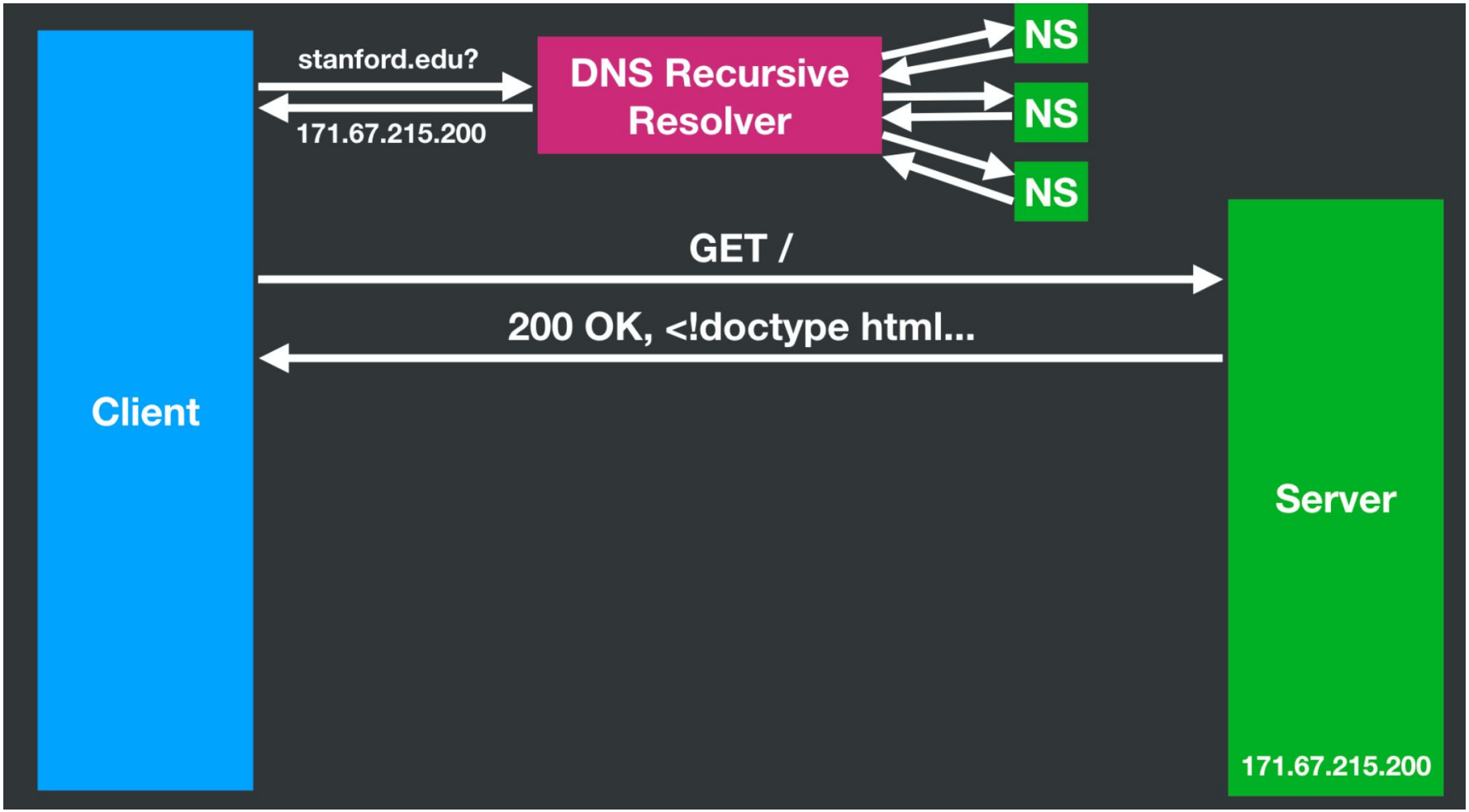
Putting it together



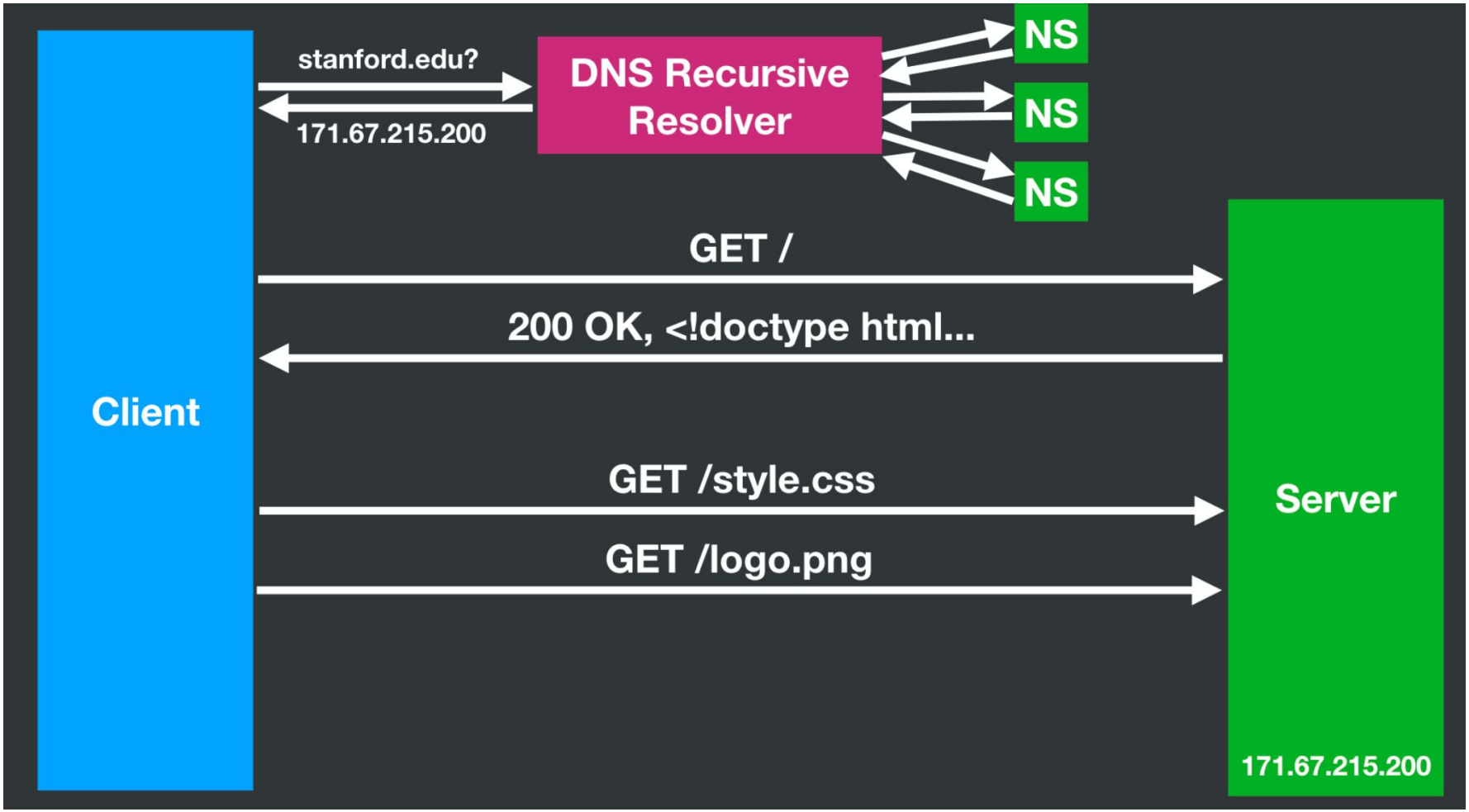
Putting it together



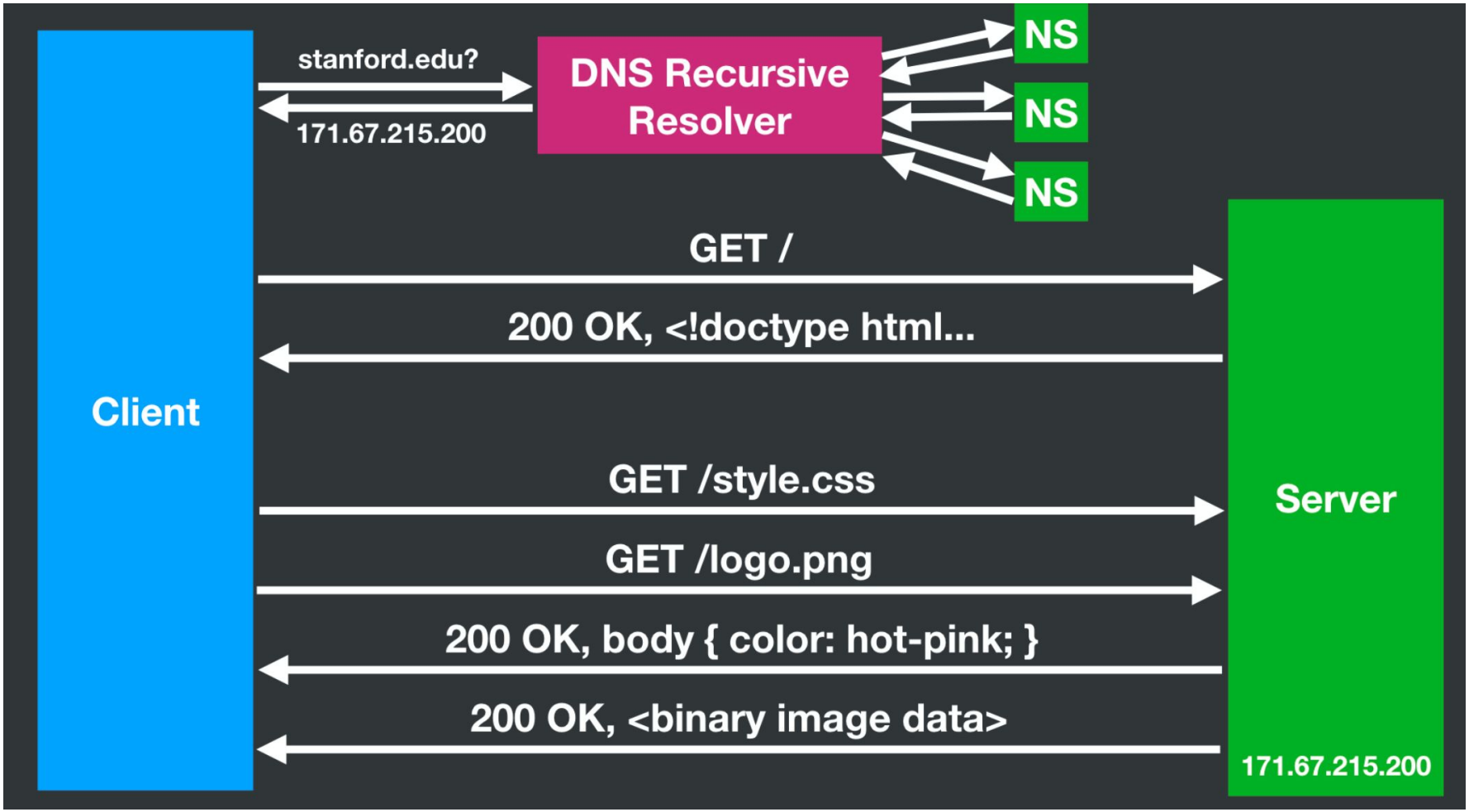
Putting it together



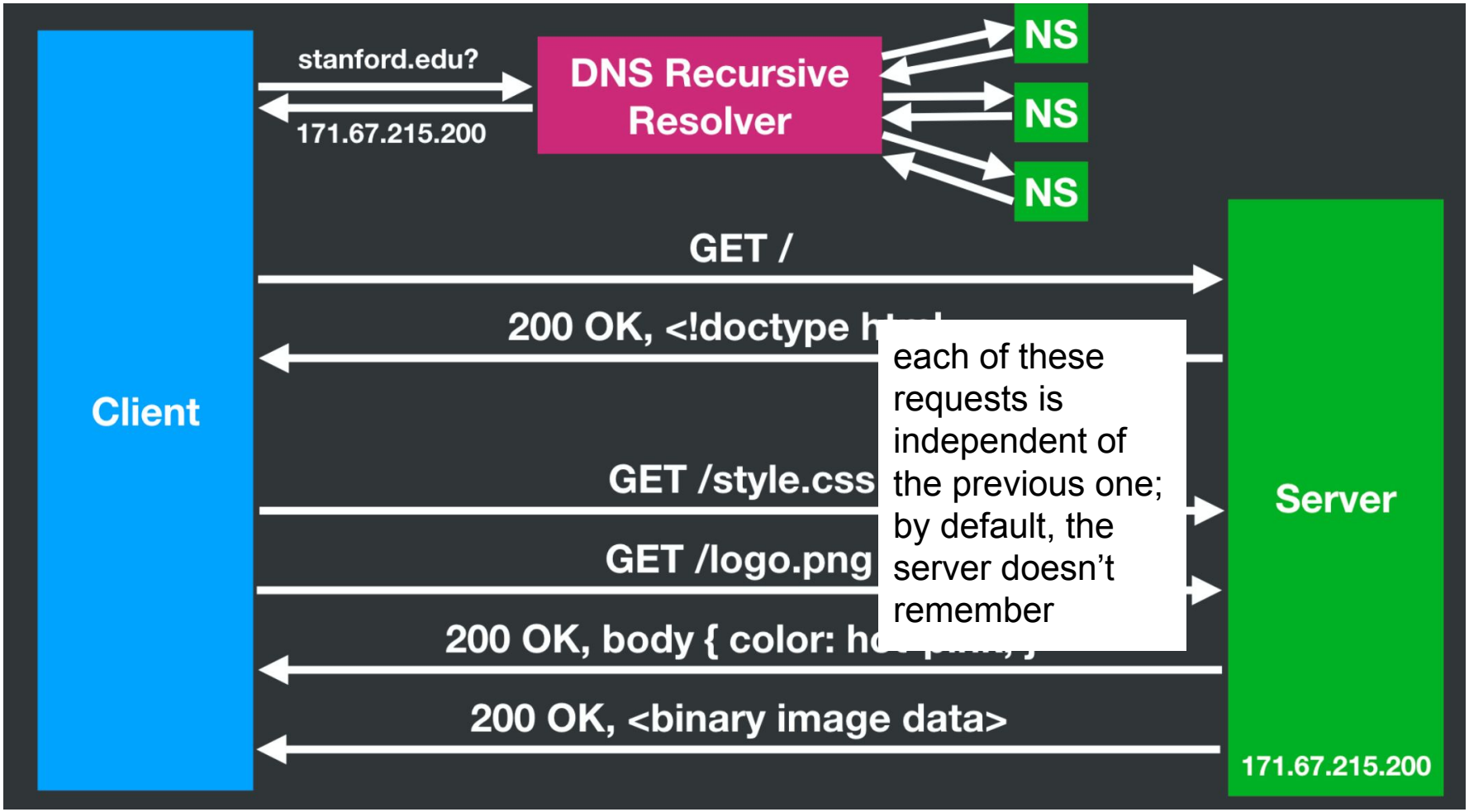
Putting it together



Putting it together

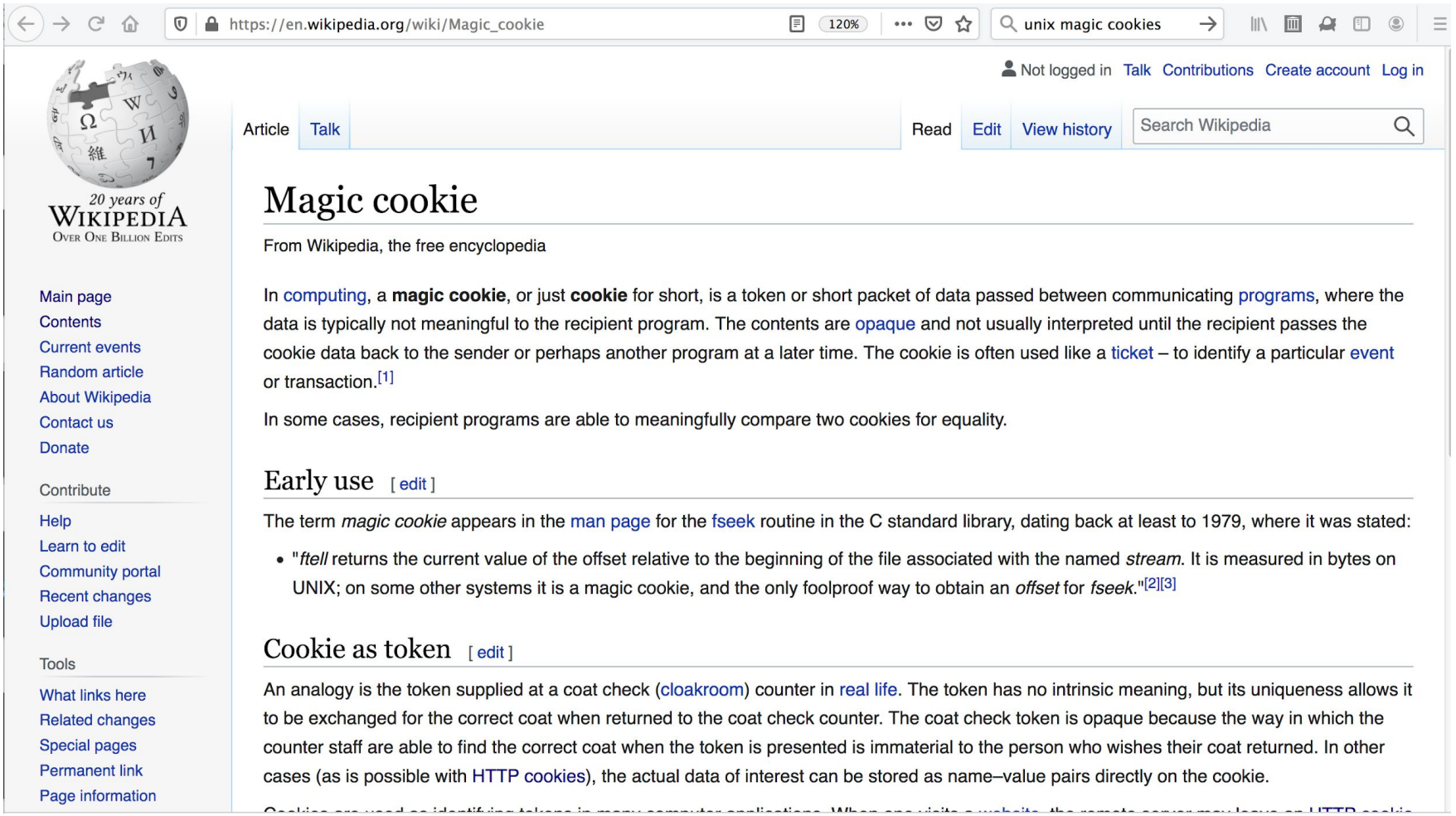


HTTP is *stateless*



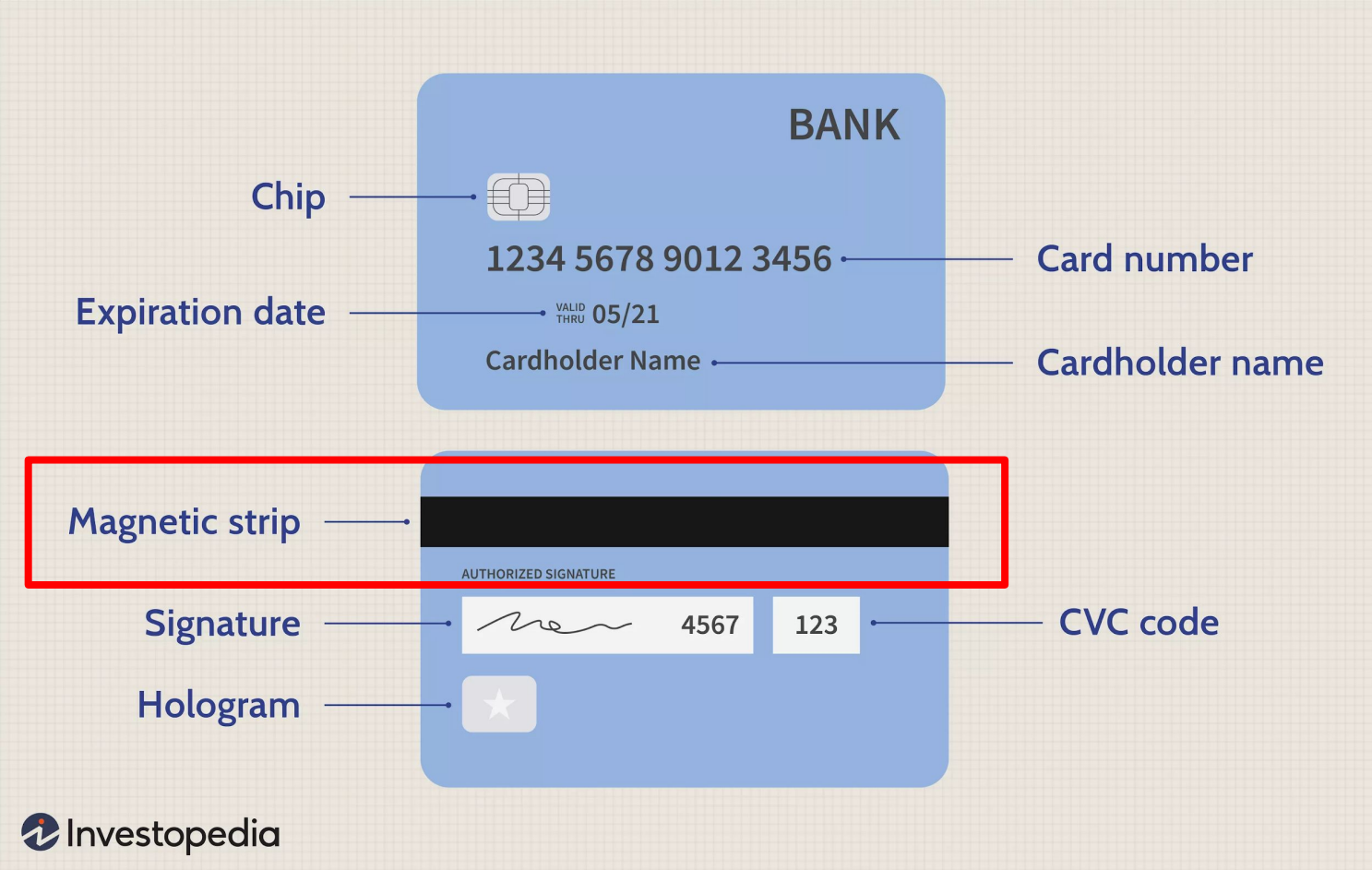
Cookies

Why “cookies”?



https://en.wikipedia.org/wiki/Magic_cookie

Real world opaque data structures



https://www.investopedia.com/terms/c/creditcard.asp

Cookies in the wild

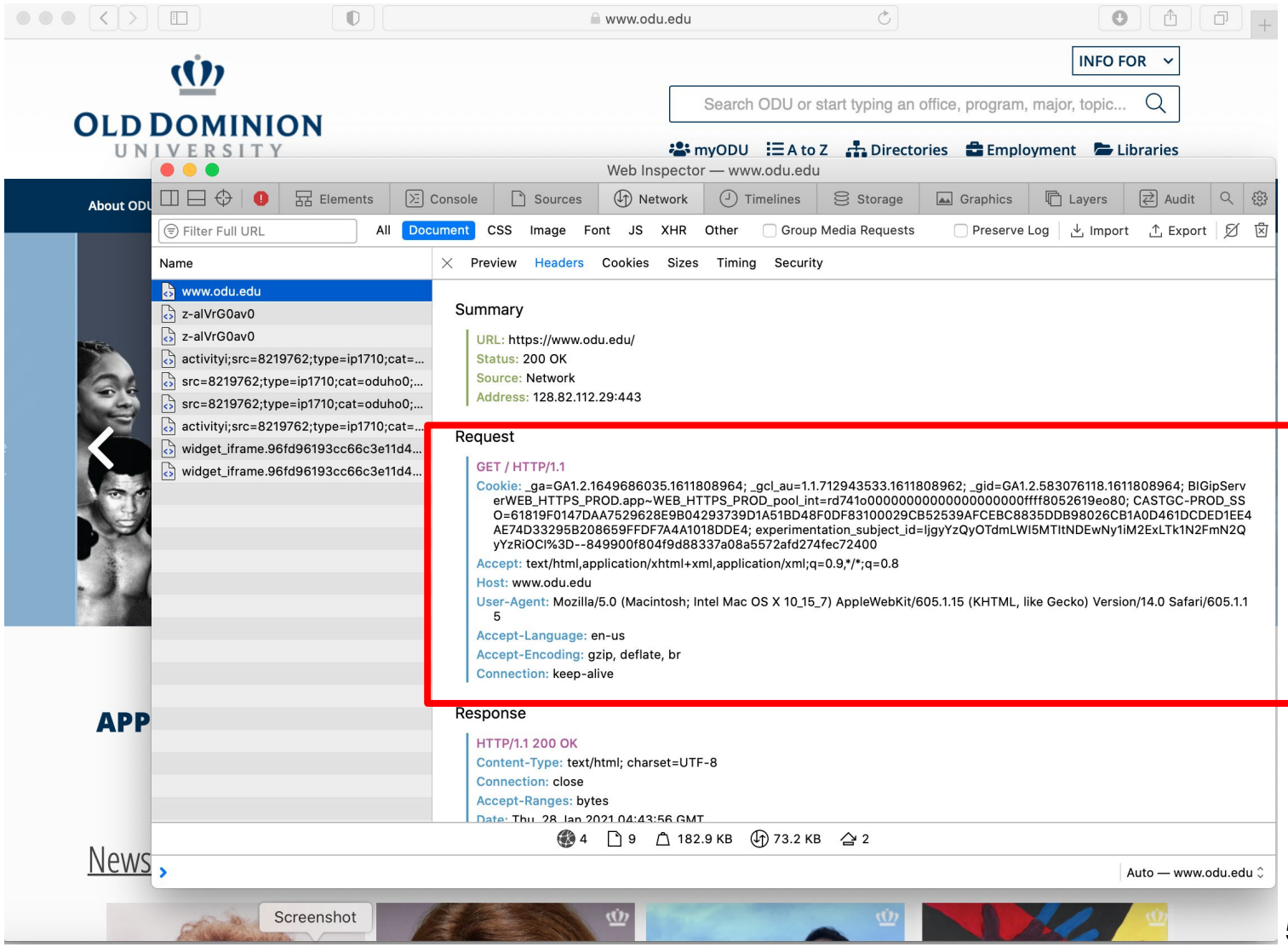
```
% curl -I https://www.odu.edu/
HTTP/1.1 200 OK
Date: Thu, 28 Jan 2021 04:30:58 GMT
Server: Apache/2.4.6 (Red Hat Enterprise Linux)
Vary: Host
Accept-Ranges: bytes
Connection: close
Content-Type: text/html; charset=UTF-8
Set-Cookie: BIGipServerWEB_HTTPS_PROD.app~WEB_HTTPS_PROD_pool_int=rd741o00000000000000000ffff8052619eo80;
path=/; Httponly; Secure
% curl -I https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
date: Thu, 28 Jan 2021 04:34:47 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:34:47 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:34:47 GMT; path=/; domain=.google.com; Secure
set-cookie:
NID=208=cOFUezC7N6Pxds_6eh3UEYUw17ON0B78ja0GYq8ZIGvg8dqlfuS1NZS5kCJK0QEtciSRWceu0ZFPHS7WMd-2xcKsF_3BWgdOUcKBG
NW9mcwH6_1WWnaoTY4k-ugIG_JzvtQCSMr5Naonq-gkTf6KCmYuy_AwJenAiEN89KNC-TY; expires=Fri, 30-Jul-2021 04:34:47
GMT; path=/; domain=.google.com; HttpOnly
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
```

Cookies are opaque: don't parse them, just send them back

```
% curl -I https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
p3p: CP="This is not a P3P policy! See g.co/p3phelp for more info."
date: Thu, 28 Jan 2021 04:48:50 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:48:50 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:48:50 GMT; path=/; domain=.google.com; Secure
set-cookie:
NID=208=FZPAg05u4KrLWrvHY6rT-g47xszZmSuF1IUZBMQkKGGzcXcNKB5cWk6DrnMlijEqlwXCx5-D8K486Db6hwf_mRTpR2Za5ho2MyGQWut_40v
Qxgnh3eKN28a5Yw6_LwysvSIaaTz9s6lyhzK9J3b3VmhUlVkaopJo57XBcccXFS8; expires=Fri, 30-Jul-2021 04:48:50 GMT; path=/;
domain=.google.com; HttpOnly
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
mln2@Michaels-MacBook-Air ~ % curl -I -H "Cookie: 1P_JAR=2021-01-28-04" -H "Cookie:
NID=208=FZPAg05u4KrLWrvHY6rT-g47xszZmSuF1IUZBMQkKGGzcXcNKB5cWk6DrnMlijEqlwXCx5-D8K486Db6hwf_mRTpR2Za5ho2MyGQWut_40v
Qxgnh3eKN28a5Yw6_LwysvSIaaTz9s6lyhzK9J3b3VmhUlVkaopJo57XBcccXFS8" https://www.google.com/
HTTP/2 200
content-type: text/html; charset=ISO-8859-1
date: Thu, 28 Jan 2021 04:49:29 GMT
server: gws
x-xss-protection: 0
x-frame-options: SAMEORIGIN
expires: Thu, 28 Jan 2021 04:49:29 GMT
cache-control: private
set-cookie: 1P_JAR=2021-01-28-04; expires=Sat, 27-Feb-2021 04:49:29 GMT; path=/; domain=.google.com; Secure
alt-svc: h3-29=":443"; ma=2592000,h3-T051=":443"; ma=2592000,h3-Q050=":443"; ma=2592000,h3-Q046=":443";
ma=2592000,h3-Q043=":443"; ma=2592000,quic=":443"; ma=2592000; v="46,43"
```

see also: "--cookie" and "--cookie-jar"

Finding your cookies



Cookies can implement *sessions*

Cookies allow state to be passed between the user-agent and server

- Cookies are used by the server to implement sessions
- Goal: Server keeps a set of data related to a user's current "browsing session"
- Examples
  - Logins
  - Shopping carts
  - User tracking

Session skeleton

First HTTP request:

```
POST /login HTTP/1.1
Host: example.com

username=alice&password=password
```

HTTP response:

```
HTTP/1.1 200 OK
Set-Cookie: username=alice
Date: Tue, 24 Sep 2019 20:30:00 GMT

<!DOCTYPE html …
```

All future HTTP requests:

```
GET /page.html HTTP/1.1
Host: example.com
Cookie: username=alice;
```

note: in this example and in class demos, the cookies are readable. in real life, they're *opaque* and only make sense to the server.

Ambient Authority

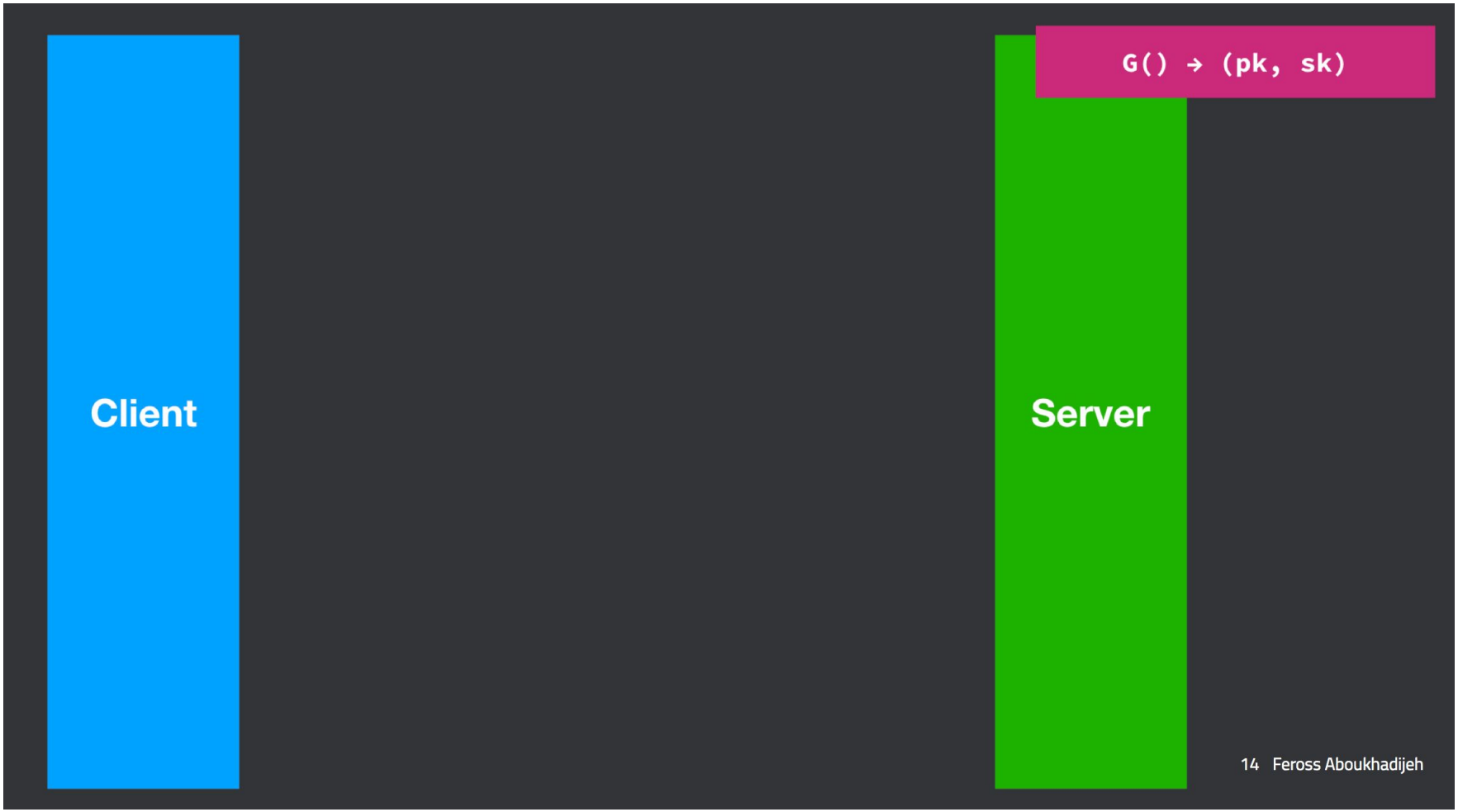
- Access control - Regulate who can view resources or take actions
- Ambient authority - Access control based on a global and persistent property of the requester
- The alternative is explicit authorization valid only for a specific action
- There are four types of ambient authority on the web
  - Cookies - most common, most versatile method
  - IP checking - library checks if you're on odu.edu network
  - Built-in HTTP authentication - rarely used
  - Client certificates - rarely used

Quick primer: signature schemes

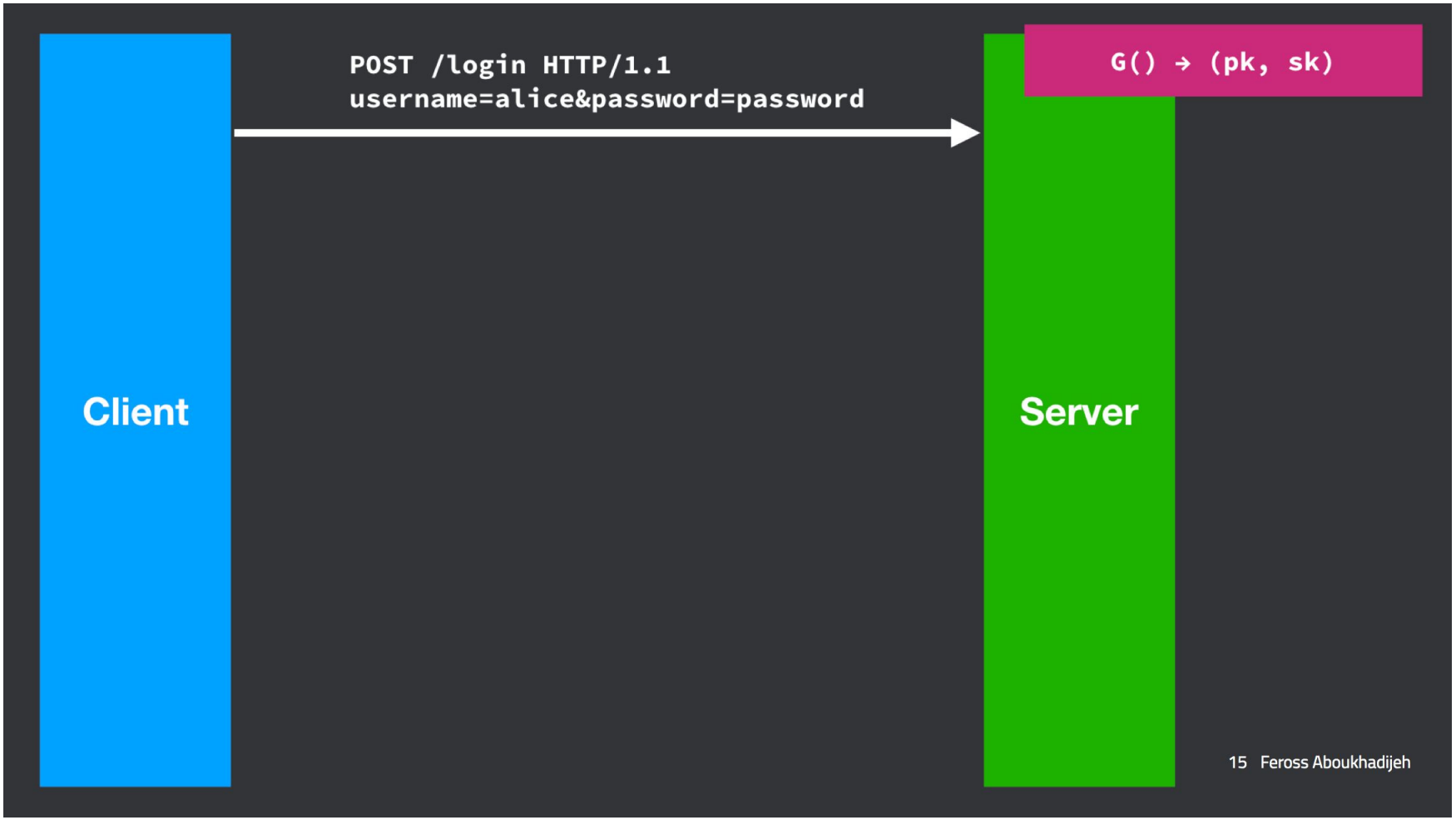
- Triple of algorithms (G, S, V)
  - G() → (pk, sk) - generator returns public key and secret key
  - S(sk, x) → t - signing returns a tag t for input x
  - V(pk, x, t) → accept|reject - checks validity of tag t for given input x
- Correctness property
  - V(pk, x, S(sk, x)) = accept should always be true
- Security property
  - V(pk, x, t) = accept should almost never be true when x and t are chosen by the attacker

Login and session

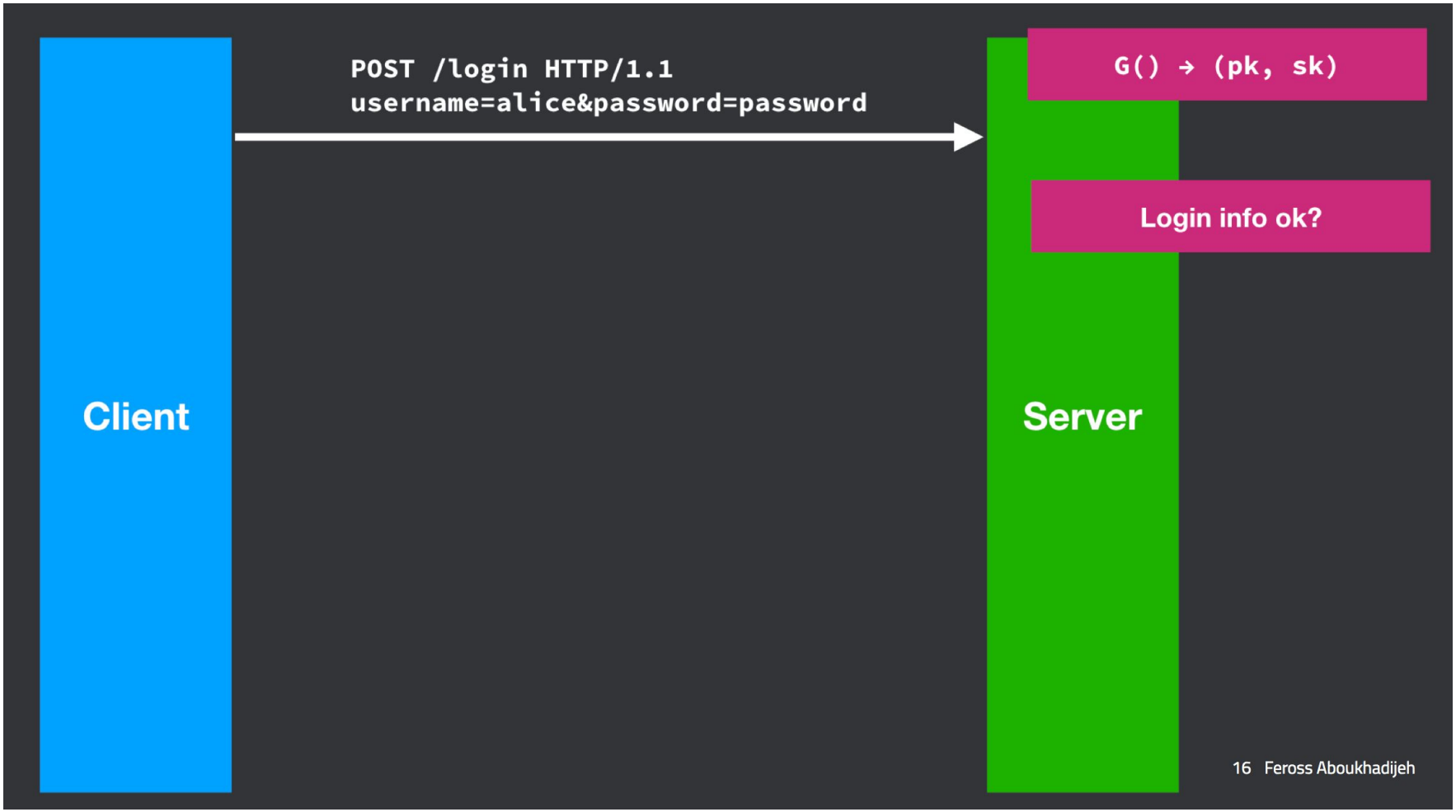
Login and session



Login and session

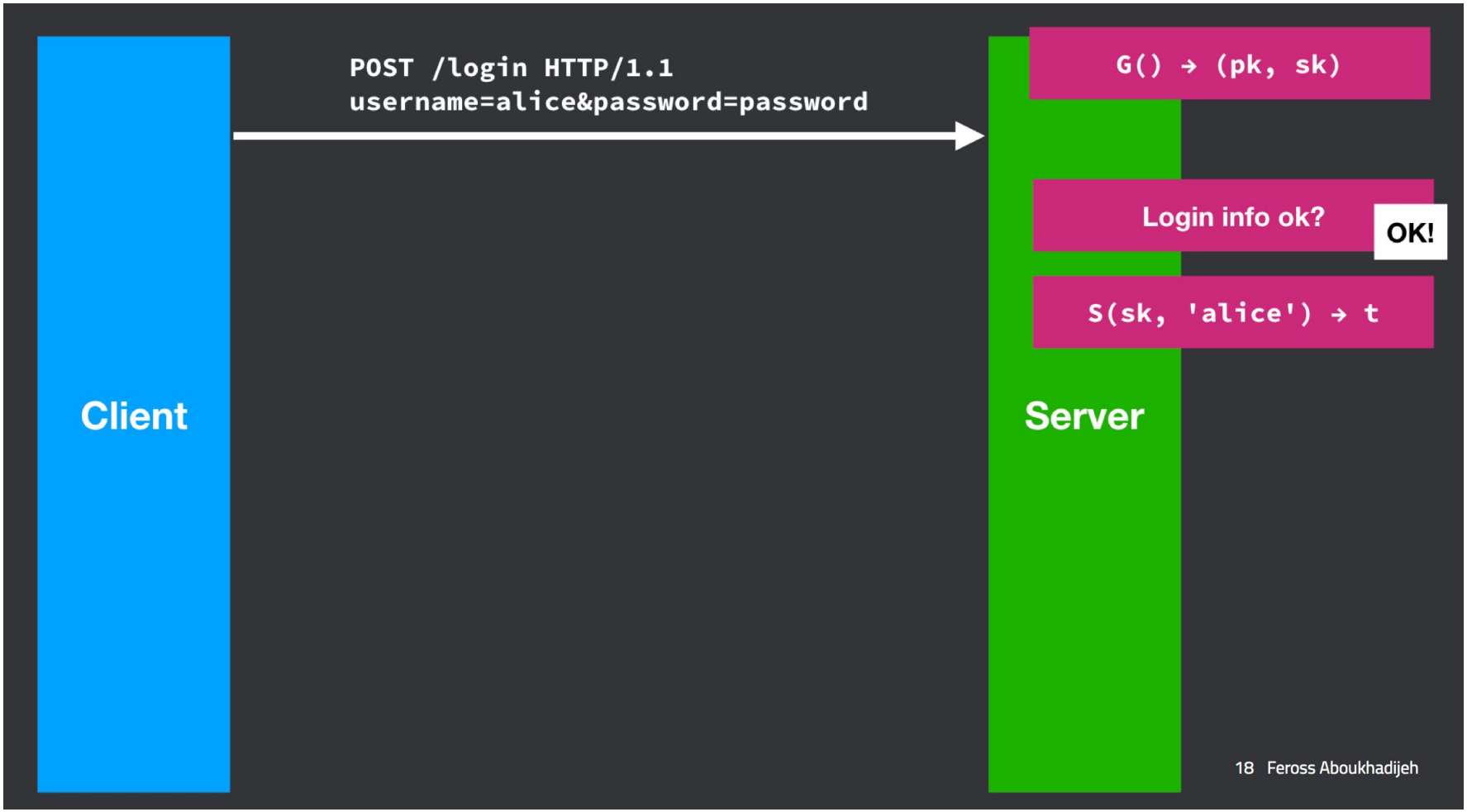


Login and session

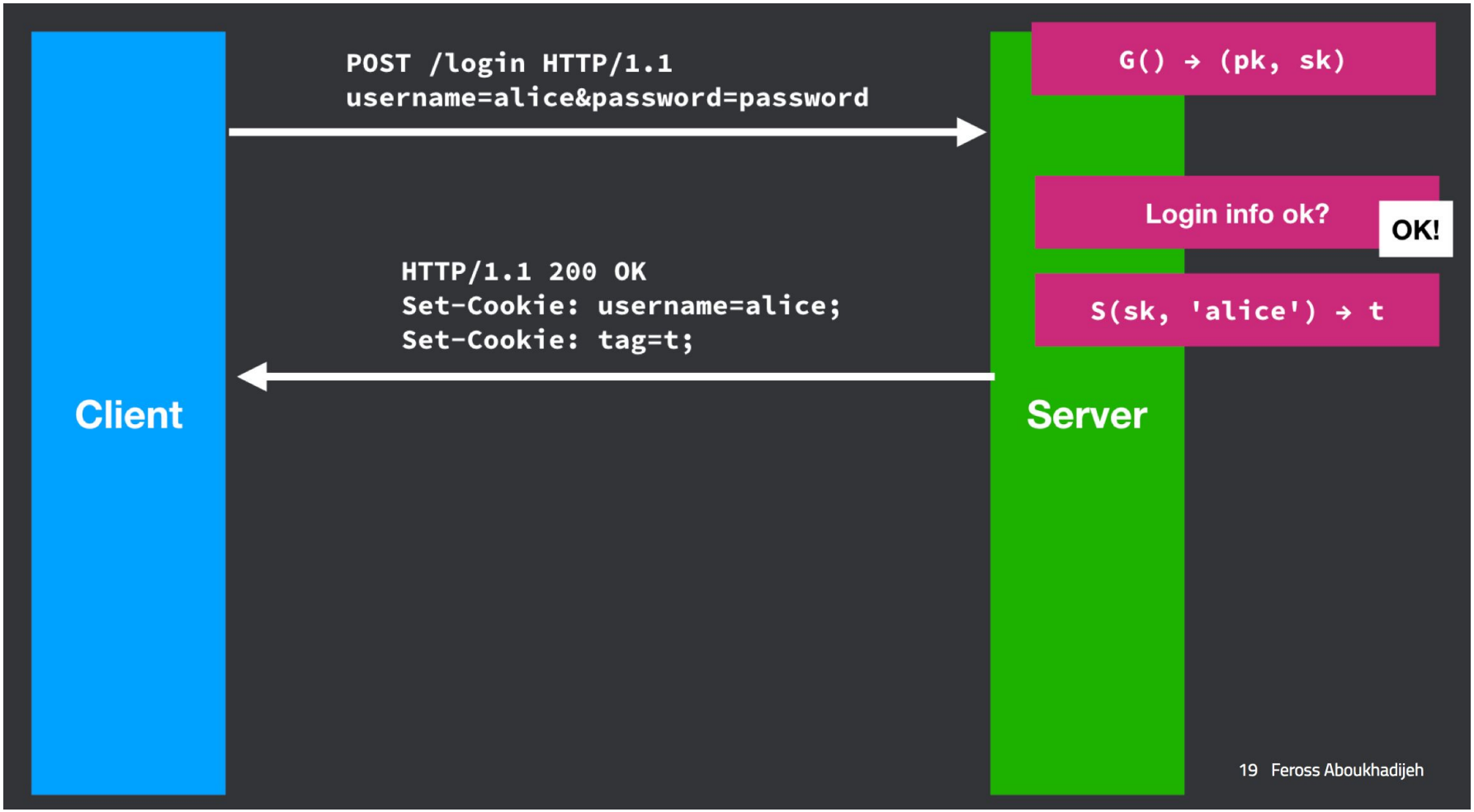


Login and session

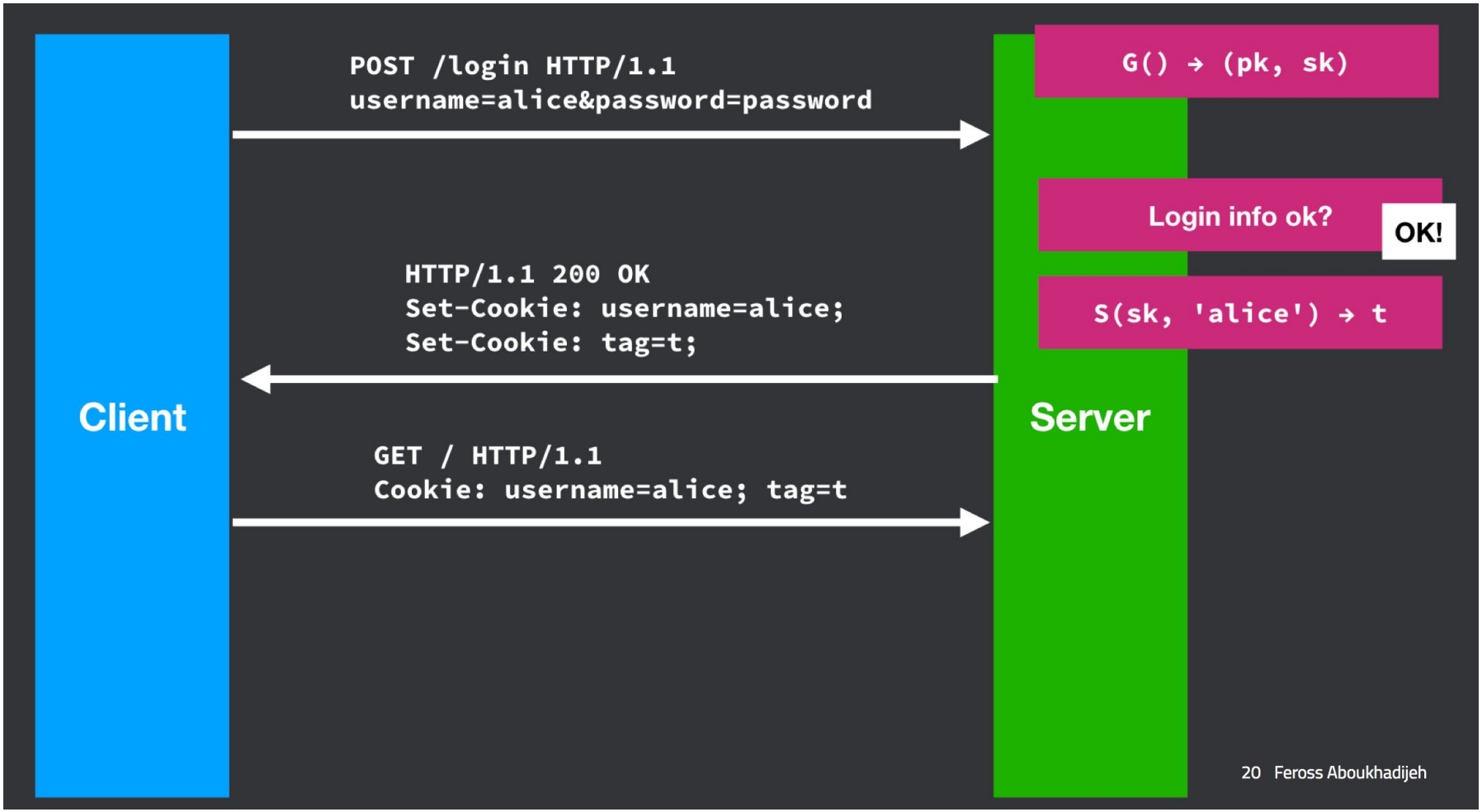
Login and session



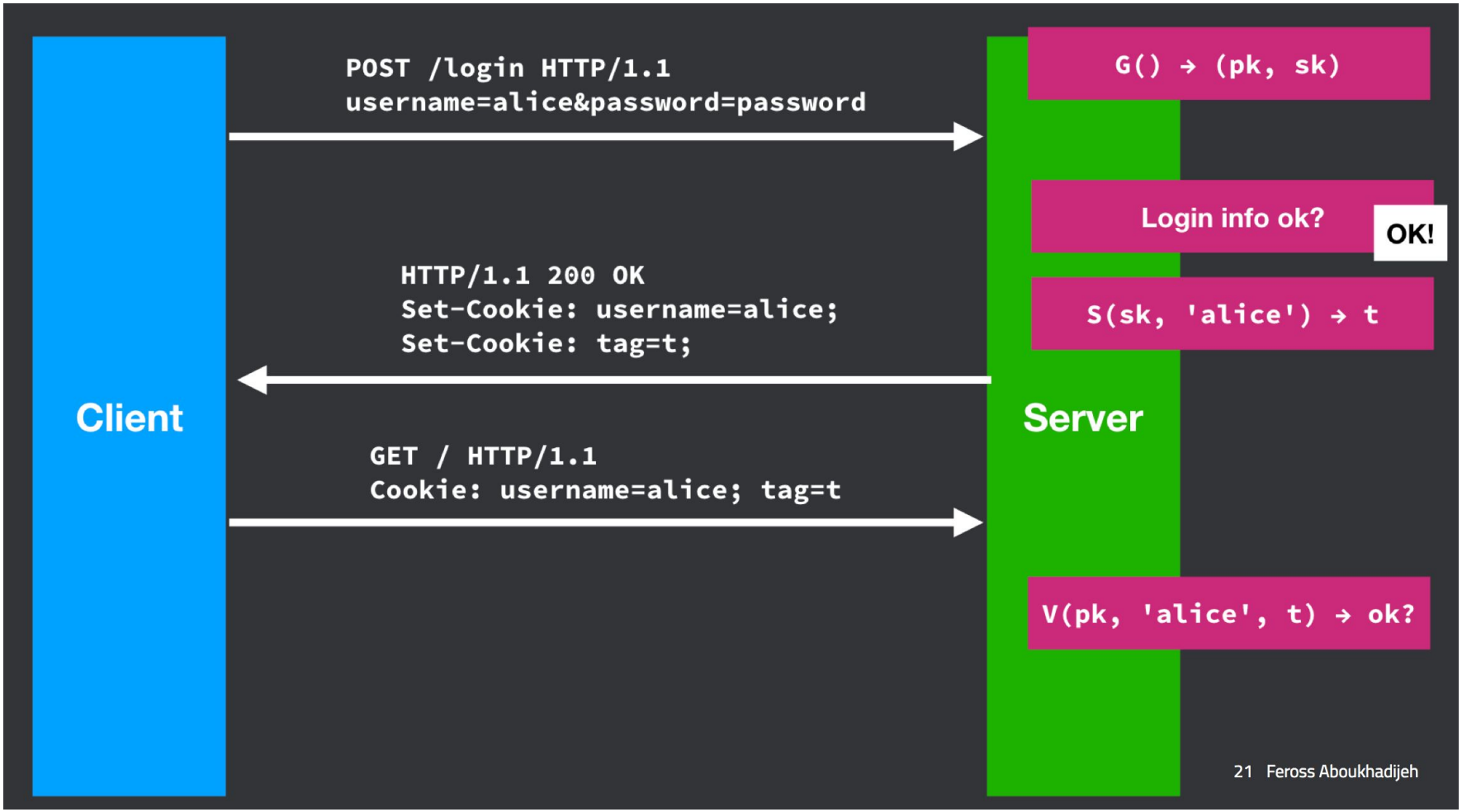
Login and session



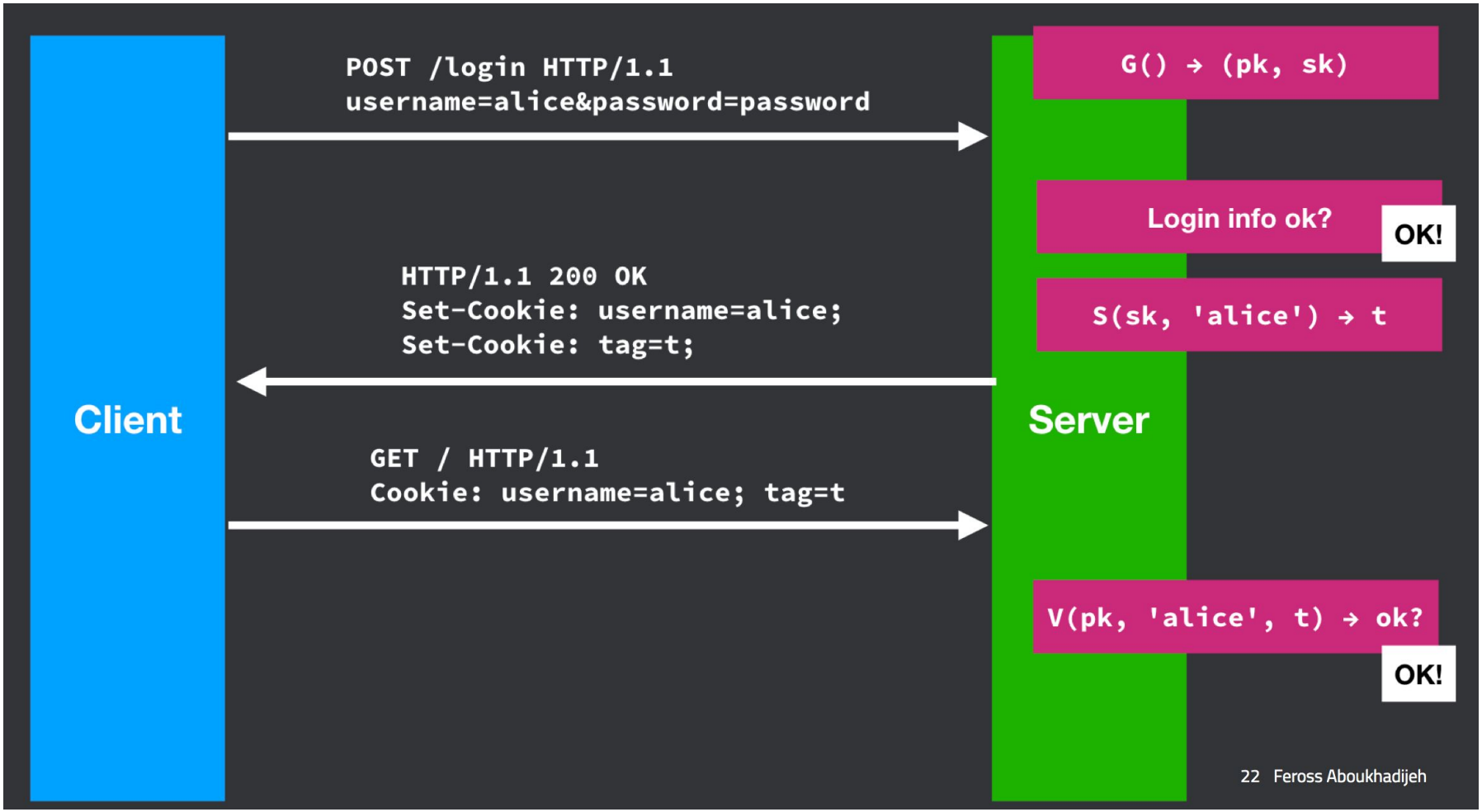
Login and session



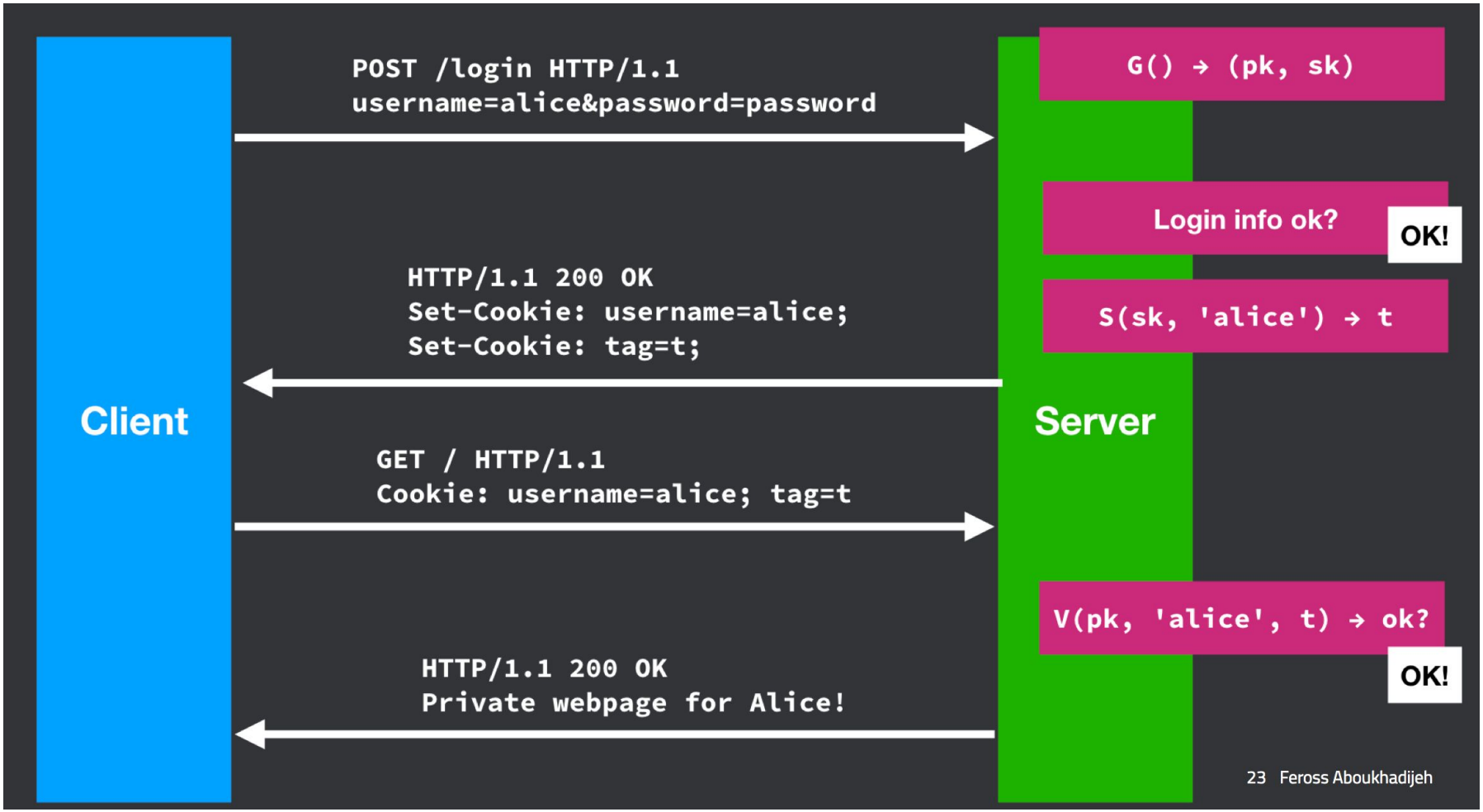
Login and session



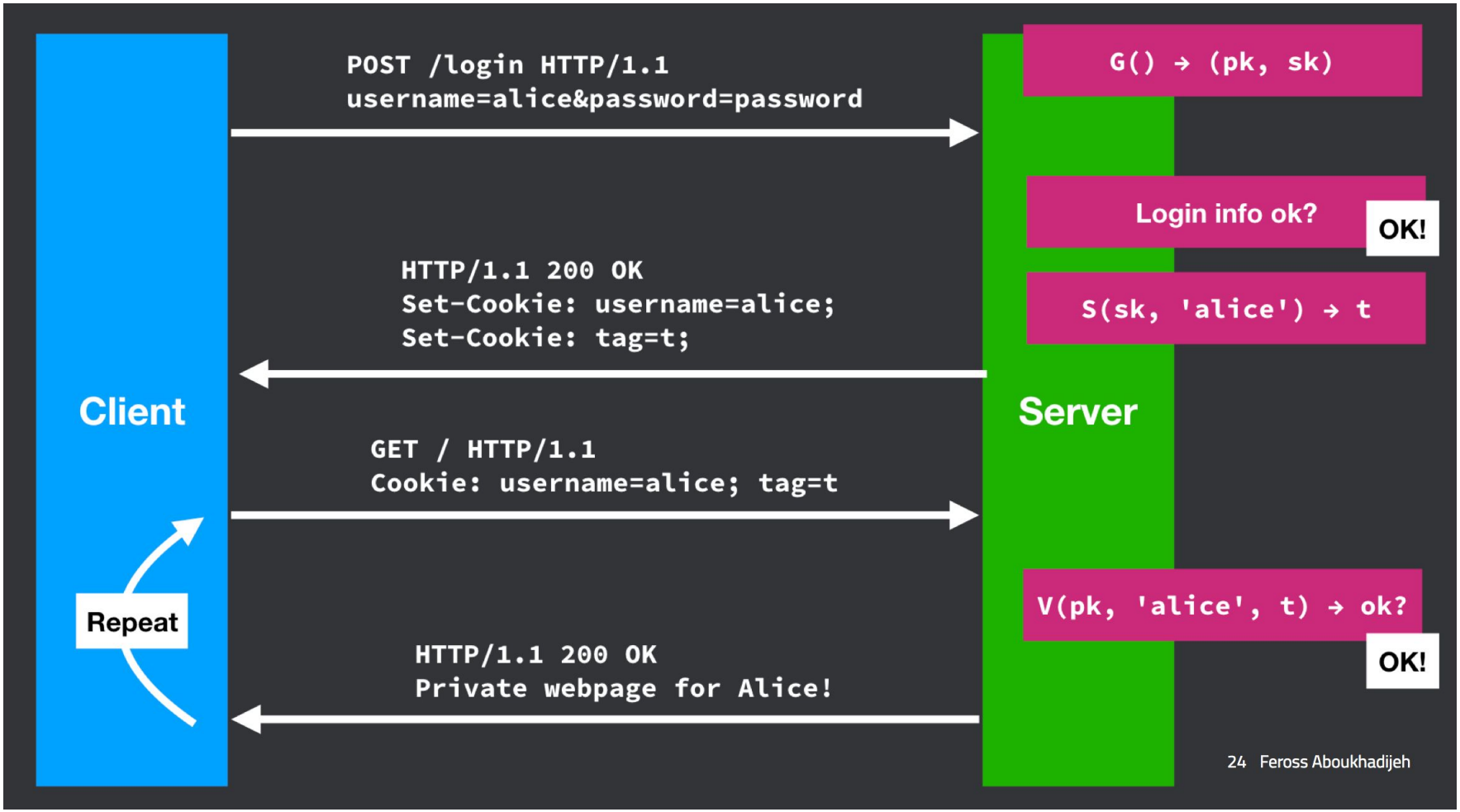
Login and session



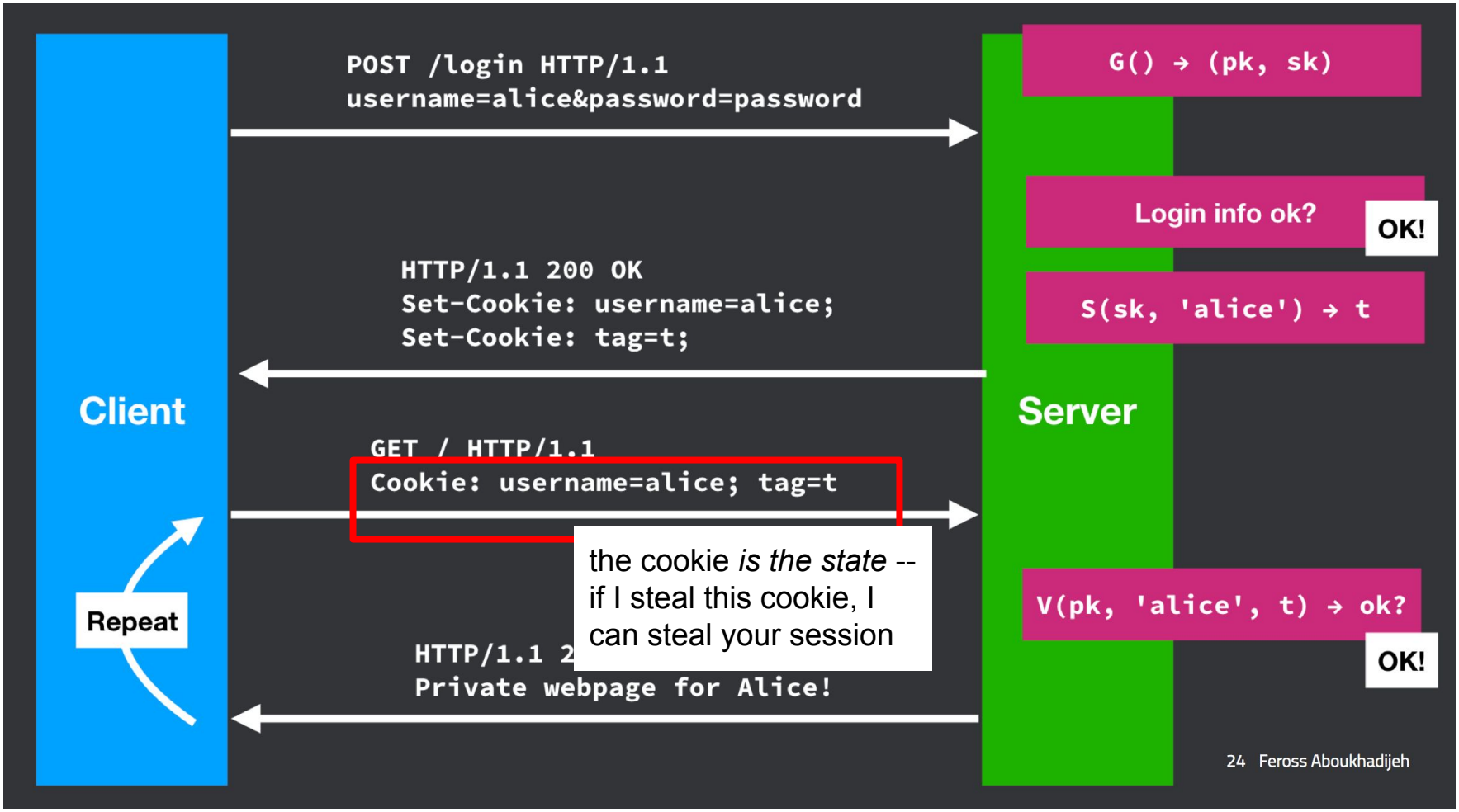
Login and session



Login and session



If I know the cookie, I can be Alice



History of Cookies

- Implemented in 1994 in Netscape and described in 4-page draft
- No spec for 17 years
  - Attempt made in 1997, but made incompatible changes (RFC 2109)
  - Another attempt in 2000 ("Cookie2", RFC 2965), same problem
  - Around 2011, another effort succeeded (RFC 6265)
- Ad-hoc design has led to interesting issues

Cookie attributes

```
canis:/home/mln % curl -I --silent https://www.google.com/ | grep -i "Cookie:"
set-cookie: 1P_JAR=2021-01-28-17;expires=Sat, 27-Feb-2021 17:08:10 GMT; path=/; domain=.google.com;Secure
set-cookie:
NID=208=QRqrD9wX6KJLXX5RH-pr8Q0UhqskKCjSAmnutkMRHT-QeI8E6pq3M5MVjIttWZBB4hXvoDUEubm0sv-n2ukDb43qt1Q8sk5qijlm9597Z0A5X
8BooOLE7Dewr5Z21s97V5gR1ovSWCKbHJm7npE_X1sKkkVHTCEuOwEMLoKlM;expires=Fri, 30-Jul-2021 17:08:10 GMT; path=/;
domain=.google.com; HttpOnly
```

- Expires - Specifies expiration date. If no date, then lasts for session
- Path - Scope the "Cookie" header to a particular request path prefix
  - e.g. Path=/docs will match /docs and /docs/Web/
  - Optimization only; do not use for security!
- Domain - Allows the cookie to be scoped to a domain broader than the domain that returned the Set-Cookie header
  - e.g. login.stanford.edu could set a cookie for stanford.edu

How long do cookies last?

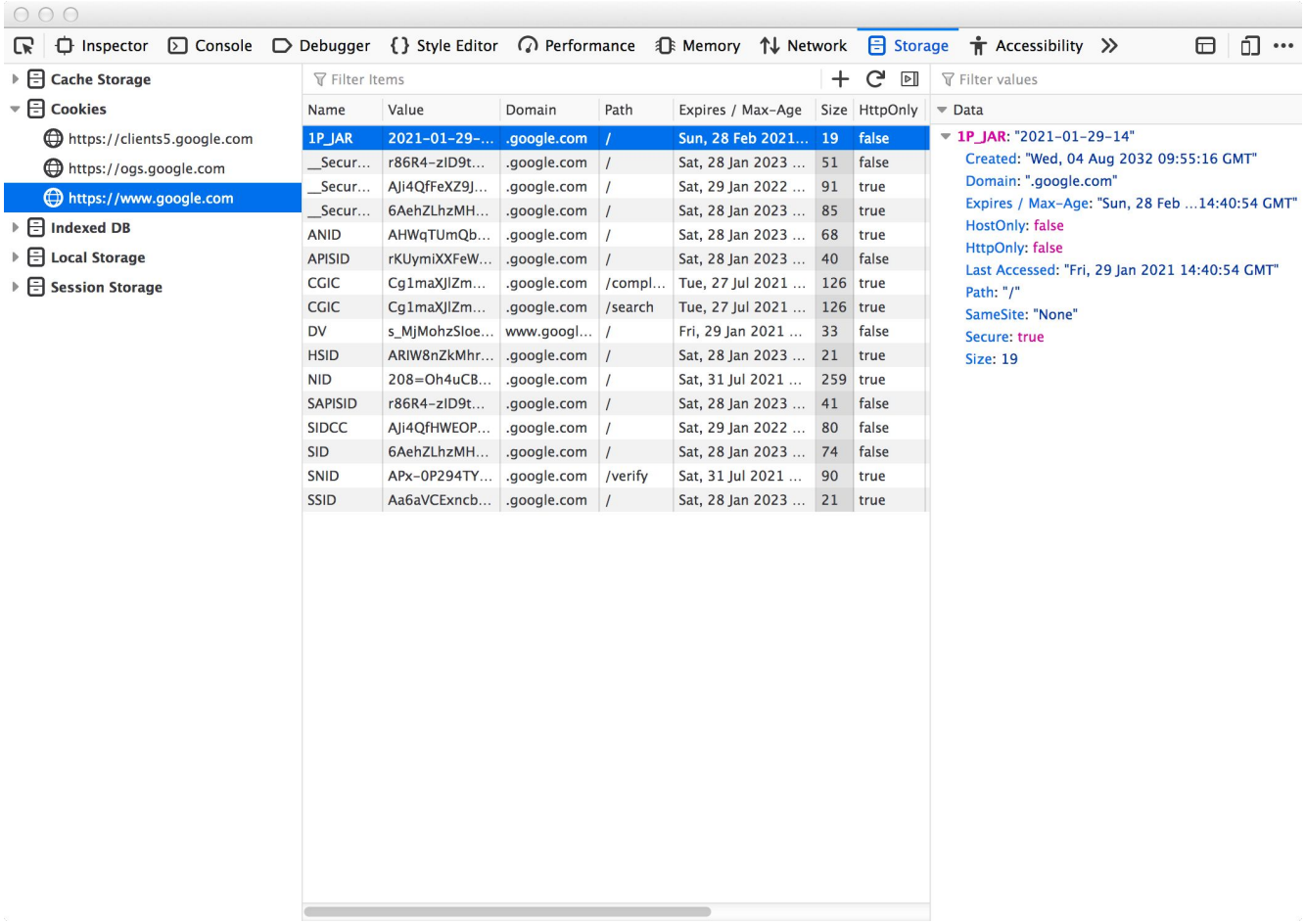
- Sites can set Expires to a very far-future date and the cookie will last until the user clears it.
  - 2007: "The Google Blog announced that Google will be shortening the expiration date of its cookies from the year 2038 to a two-year life cycle." – Search Engine Land
- When Expires not specified, lasts for current browser session
  - Caveat: Browsers do session restoring, so can last way longer

Server can reset (delete) the cookie

- Set cookie with same name and an expiration date in the past
  - cf. Unix epoch
- Cookie value can be omitted; from previous google.com example:

```
set-cookie: NID=; expires=Thu, 01 Jan 1970 00:00:00 GMT;
```

Use dev tools to access, set, delete cookies in your browser

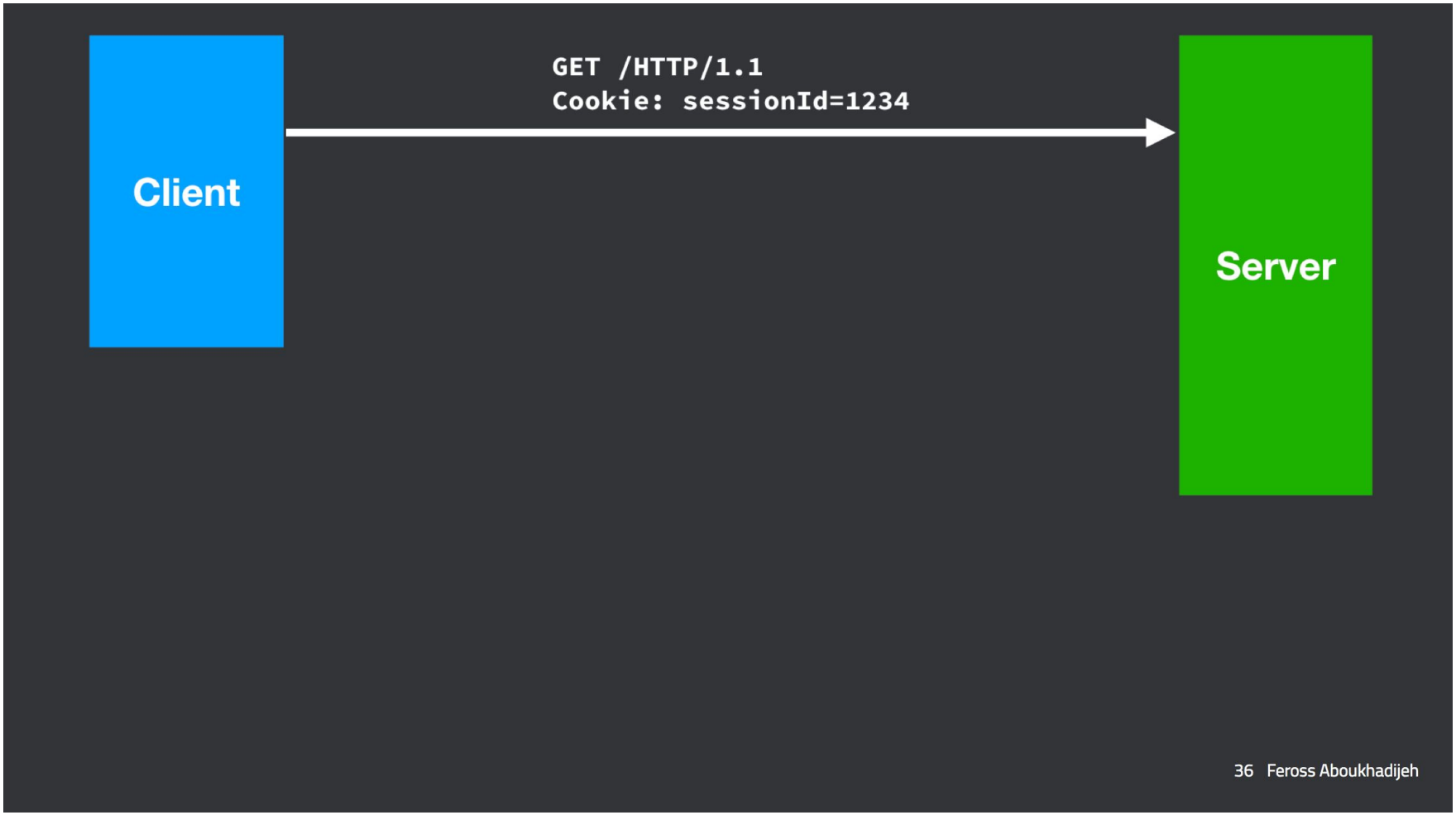


Session hijacking

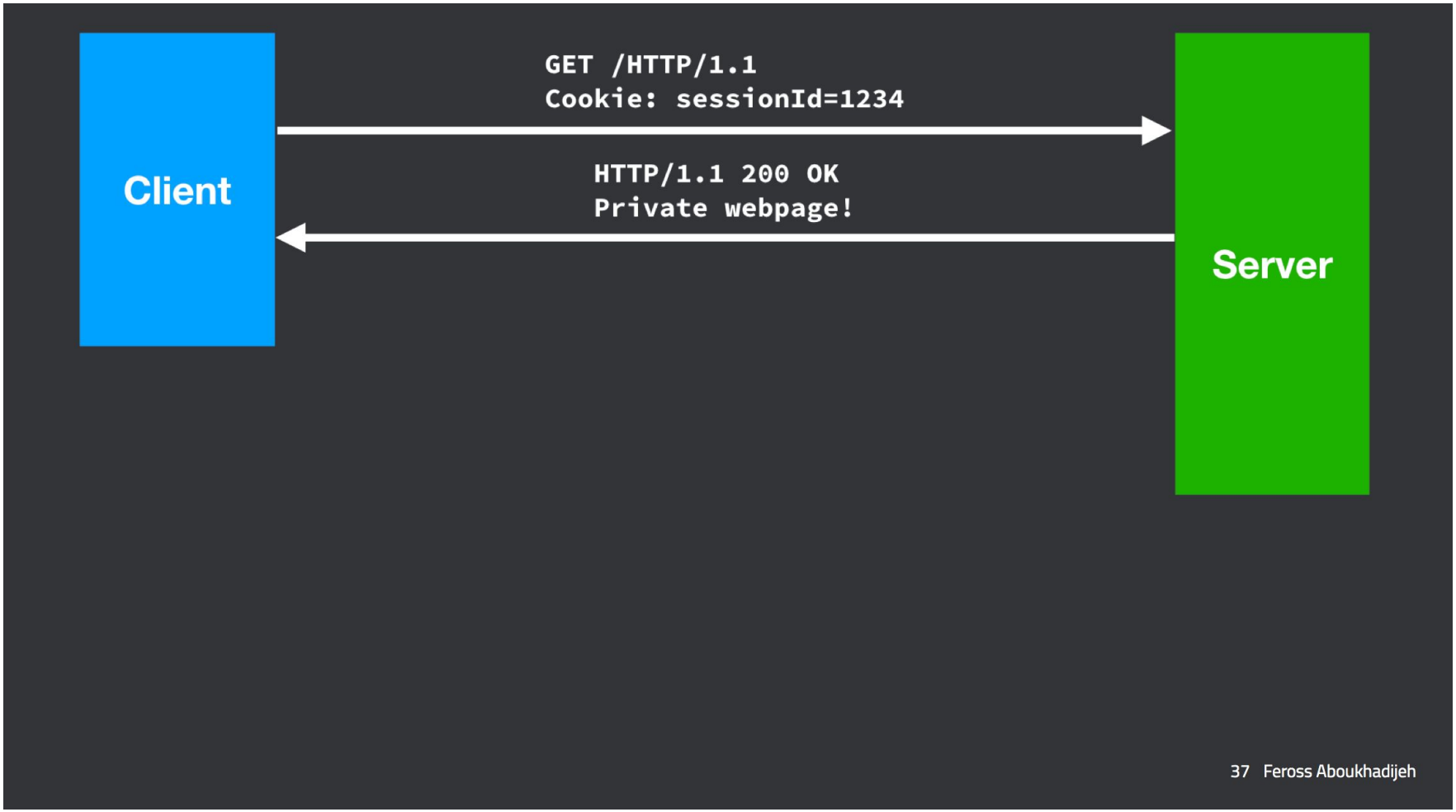
- Sending cookies over unencrypted HTTP is a *very bad idea*
  - If anyone sees the cookie, they can use it to hijack the user's session
  - Attacker sends victim's cookie as if it was their own
  - Server will be fooled -- it only “knows” about you *via the cookie*

A nice session

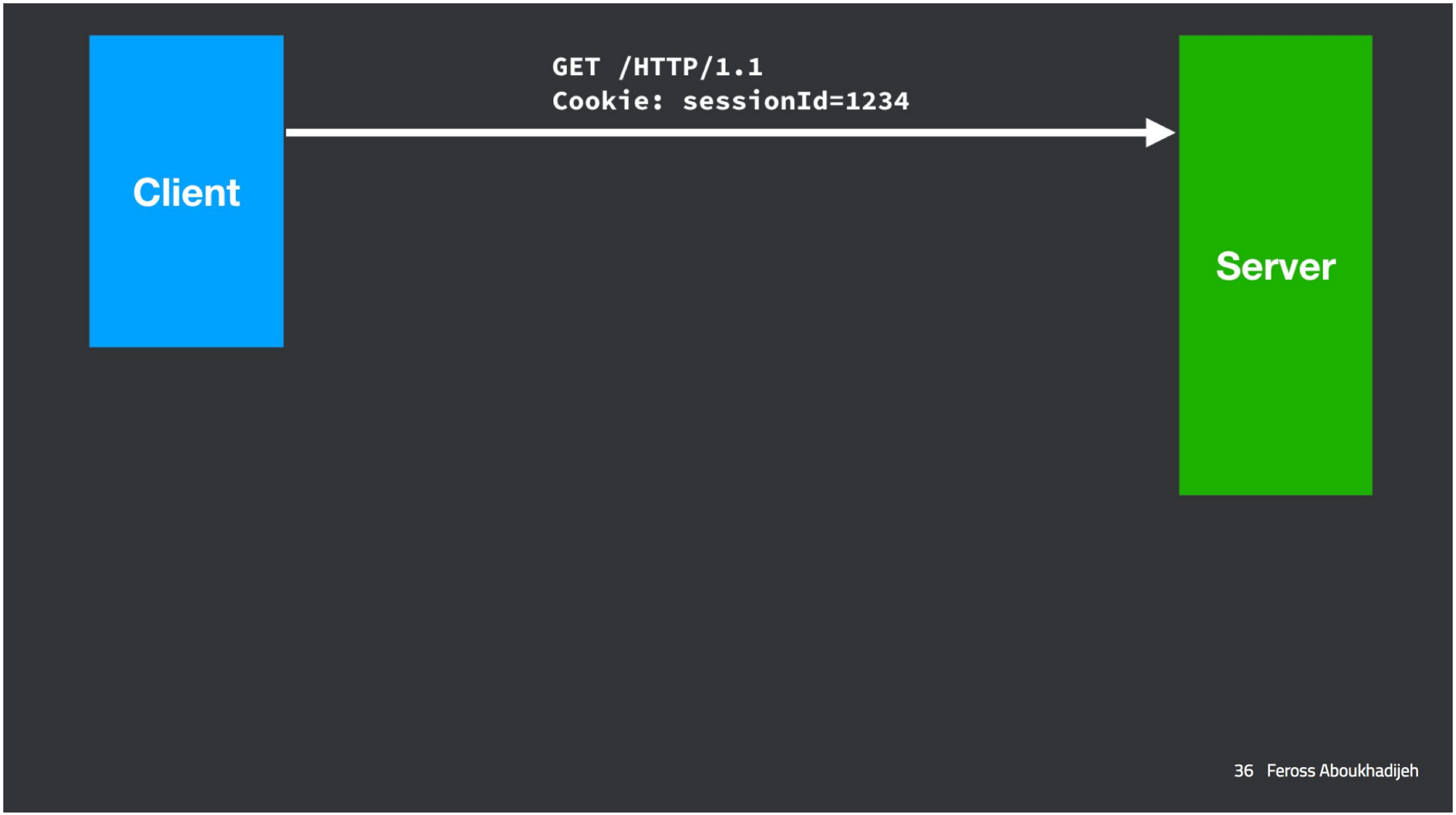
A nice session



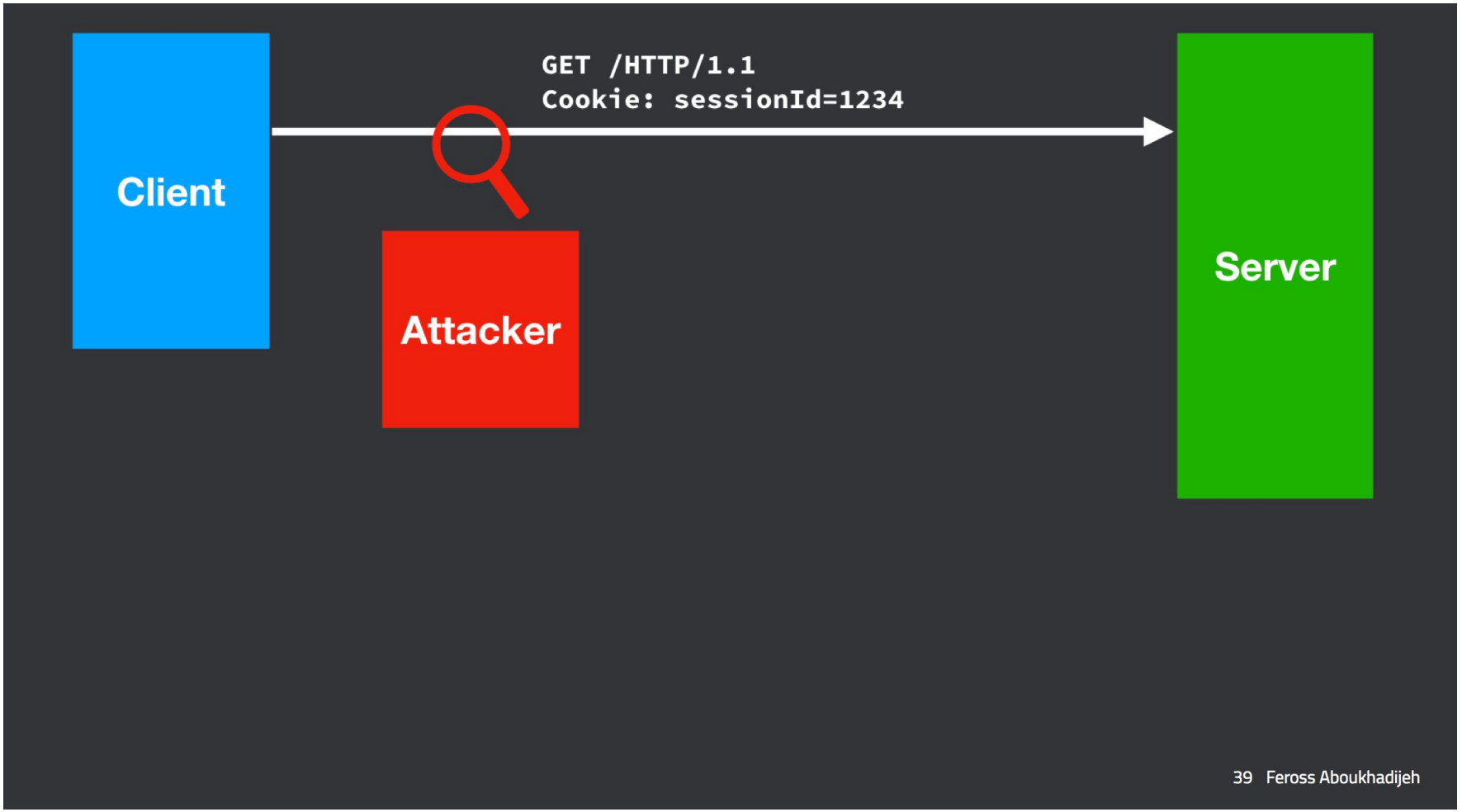
A nice session



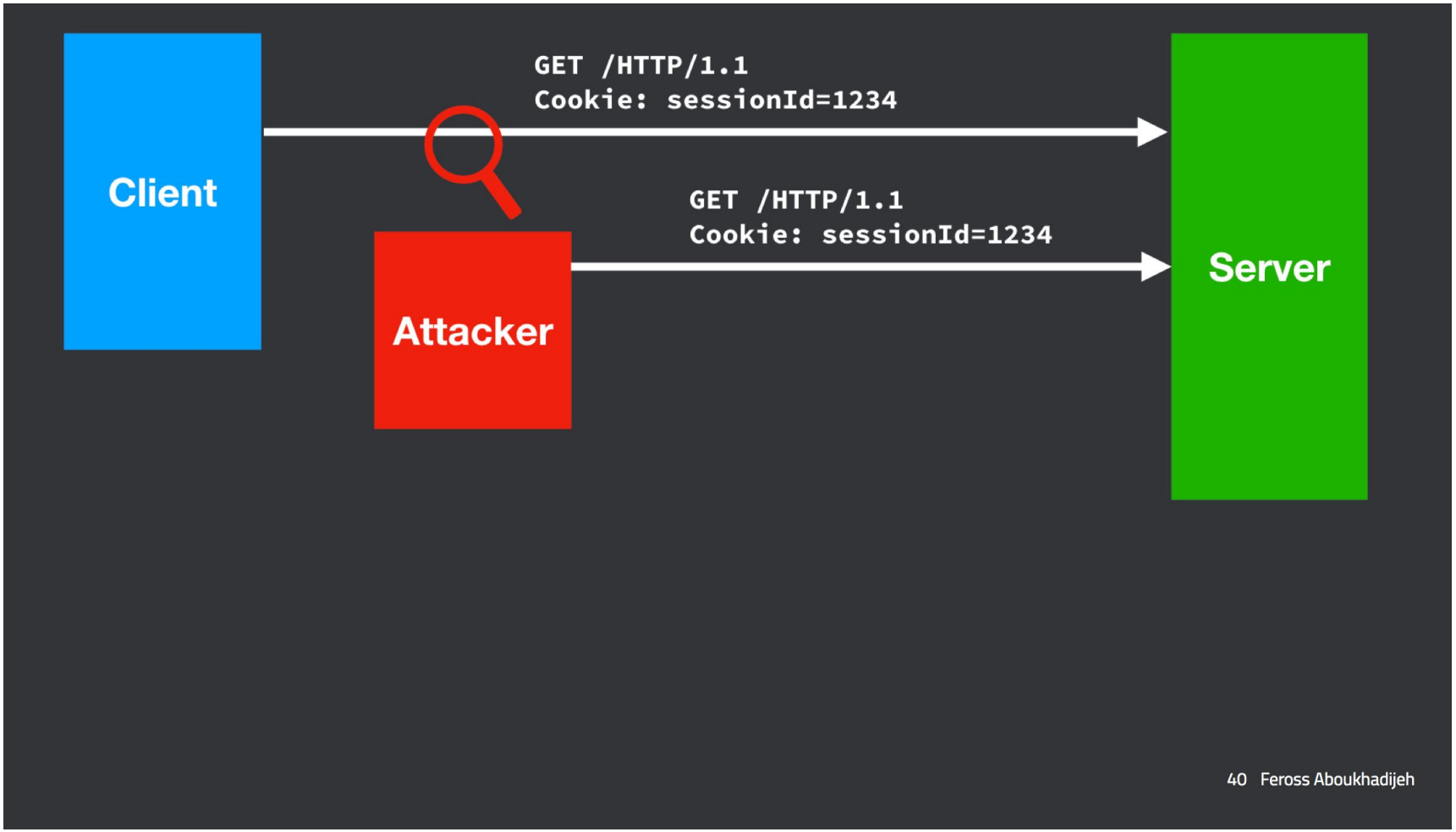
Steal your cookie



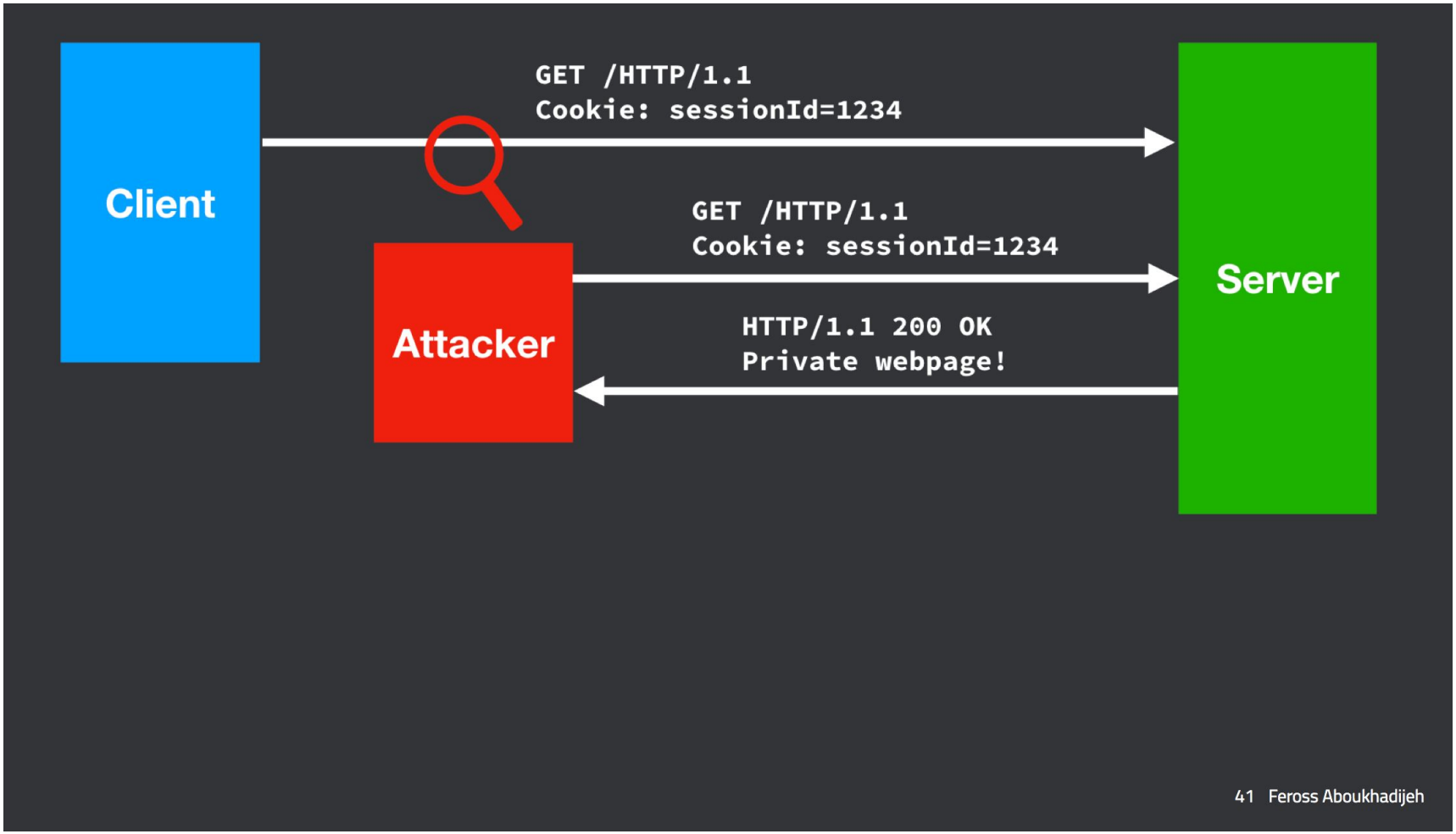
Steal your cookie



Steal your cookie



Steal your cookie

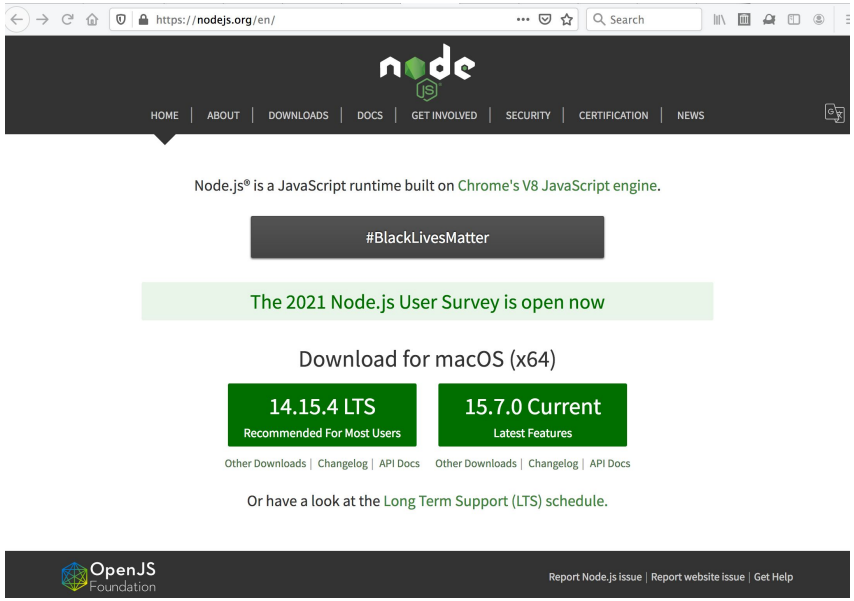


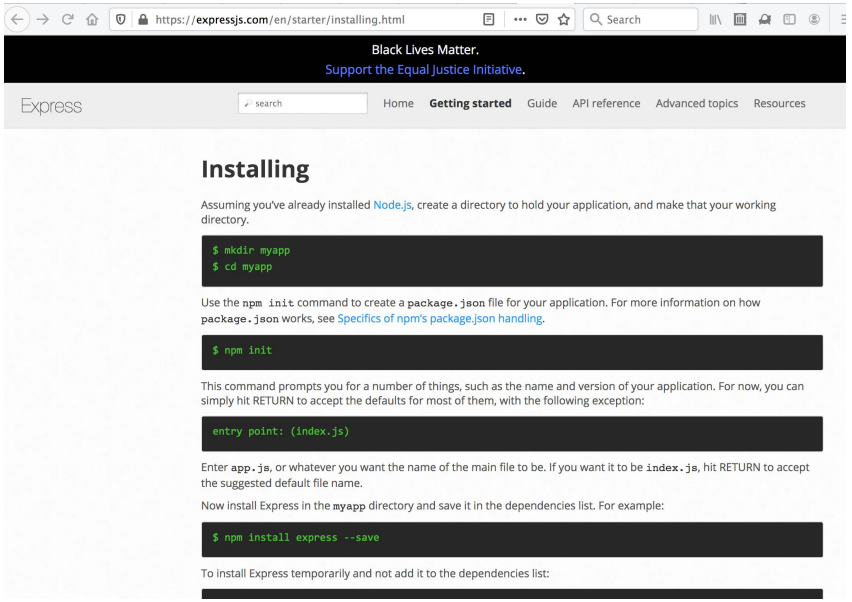
Session hijacking mitigation

- Use `Secure` cookie attribute to prevent cookie from being sent over unencrypted HTTP connections
  - `Set-Cookie: key=value; Secure`
- Even better: Use HTTPS for entire website

```
canis:/home/mln % curl -I --silent \
 http://www.google.com/ | grep -i "Secure"
set-cookie: 1P_JAR=2021-01-28-17;
expires=Sat, 27-Feb-2021 17:52:32 GMT;
path=/; domain=.google.com; Secure

canis:/home/mln % curl -I http://vt.edu
HTTP/1.1 301 Moved Permanently
Date: Thu, 28 Jan 2021 17:36:58 GMT
Server: Apache
Location: https://vt.edu/
Cache-Control: max-age=600
Expires: Thu, 28 Jan 2021 17:46:58 GMT
Content-Type: text/html; charset=iso-8859-1

canis:/home/mln % curl -I www.odu.edu
HTTP/1.0 301 Moved Permanently
Location: https://www.odu.edu/
Server: BigIP
Connection: Keep-Alive
Content-Length: 0
```

Setting up the world's worst bank with node.js





https://nodejs.org/en/

https://expressjs.com/en/starter/installing.html

use npm, included in the node.js distribution, to install packages you don't have

```
$ node bank-02.js
internal/modules/cjs/loader.js:883
  throw err;
  ^

Error: Cannot find module 'cookie-parser'
Require stack:
- /Users/mln/Desktop/cs595-s21/cs595-s21/slides/code/bank-02.js
      at Function.Module._resolveFilename
[lots of error messages deleted]
  code: 'MODULE_NOT_FOUND',
  requireStack: [ '/Users/mln/Desktop/cs595-s21/cs595-s21/slides/code/bank-02.js' ]
}
$ npm install cookie-parser
npm WARN code@1.0.0 No description
npm WARN code@1.0.0 No repository field.

+ cookie-parser@1.4.5
added 1 package from 2 contributors and audited 51 packages in 2.992s
found 0 vulnerabilities
$ node bank-02.js
```

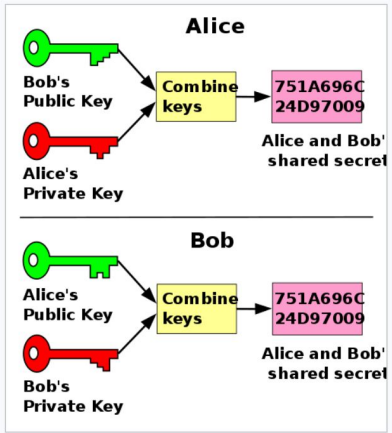
Who are Alice and Bob?!

https://en.wikipedia.org/wiki/Alice_and_Bob

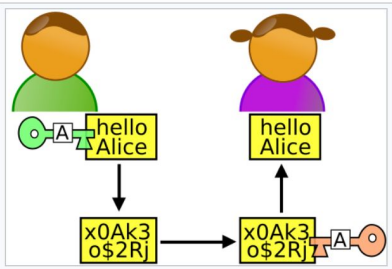
Cast of characters [edit]

The most common characters are Alice and Bob. Eve, Mallory, and Trent are also common names, and have fairly well-established "personalities" (or functions). The names often use rhyming mnemonics (for example, Eve, "eavesdropper"; Mallory, "malicious") where different players have different motives. Other names are much less common, and flexible in use. Sometimes the genders are alternated: Alice, Bob, Carol, Dave, Eve ...[13]

Alice and *Bob*	The original, generic characters. Generally, Alice and Bob want to exchange a message or cryptographic key.
Carol, *Carlos* or *Charlie*	A generic third participant.
Chuck	A third participant, usually of malicious intent.[14]
Craig	A *password cracker*, often encountered in situations with stored passwords.
Dan, *Dave* or *David*	A generic fourth participant.
Erin	A generic fifth participant, but rarely used, as "E" is usually reserved for Eve.
Eve	An *eavesdropper*, who is usually a passive attacker. While they can listen in on messages between Alice and Bob, they cannot modify them. In quantum cryptography, Eve may also represent



An example of an "Alice and Bob" used in cryptography.



An example of Alice and Bob used to explain public-key cryptography.

https://en.wikipedia.org/wiki/Alice_and_Bob