A Firewall for your Radical Network

firewall

/ˈfʌɪəwɔːl/

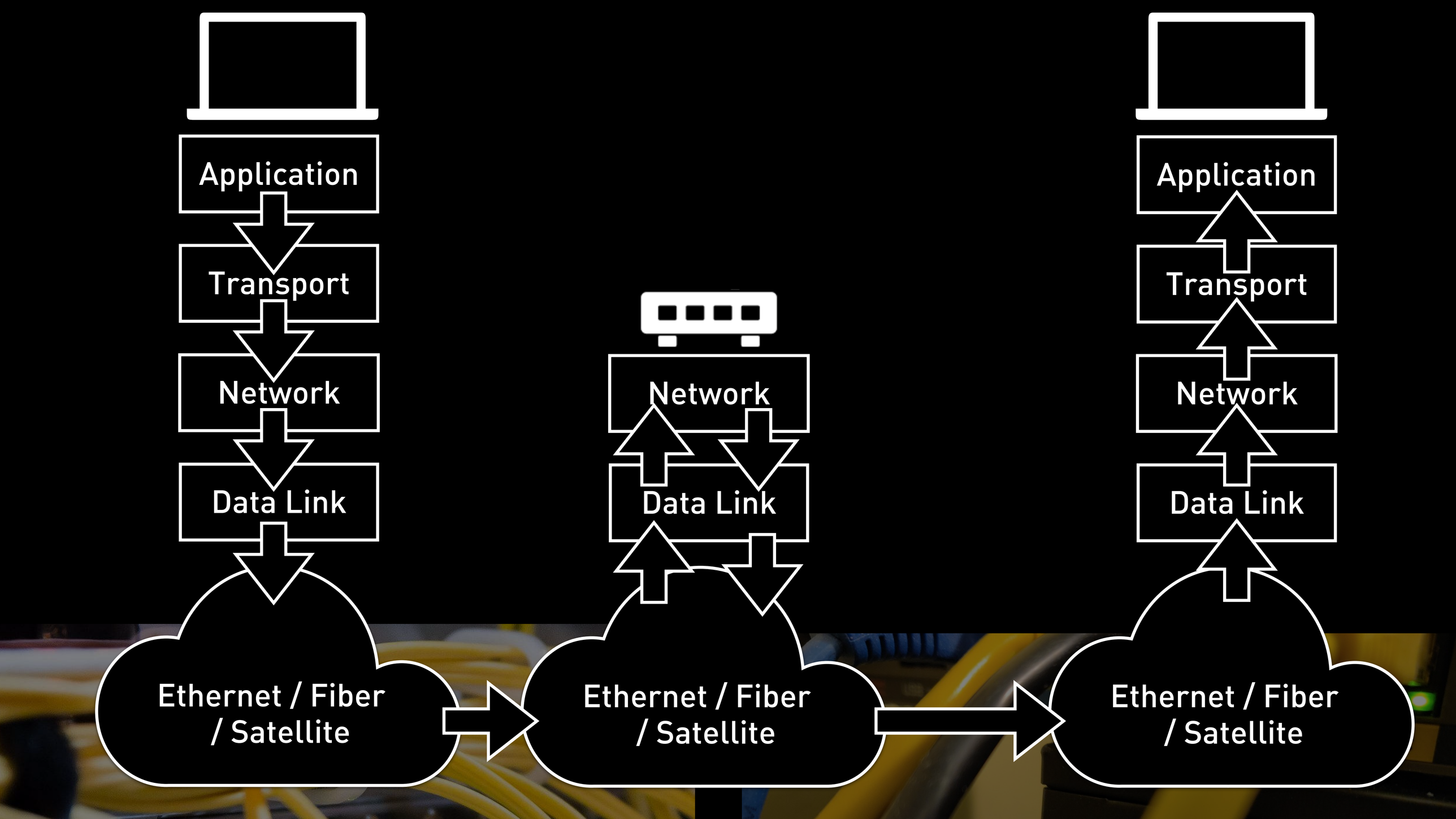
noun

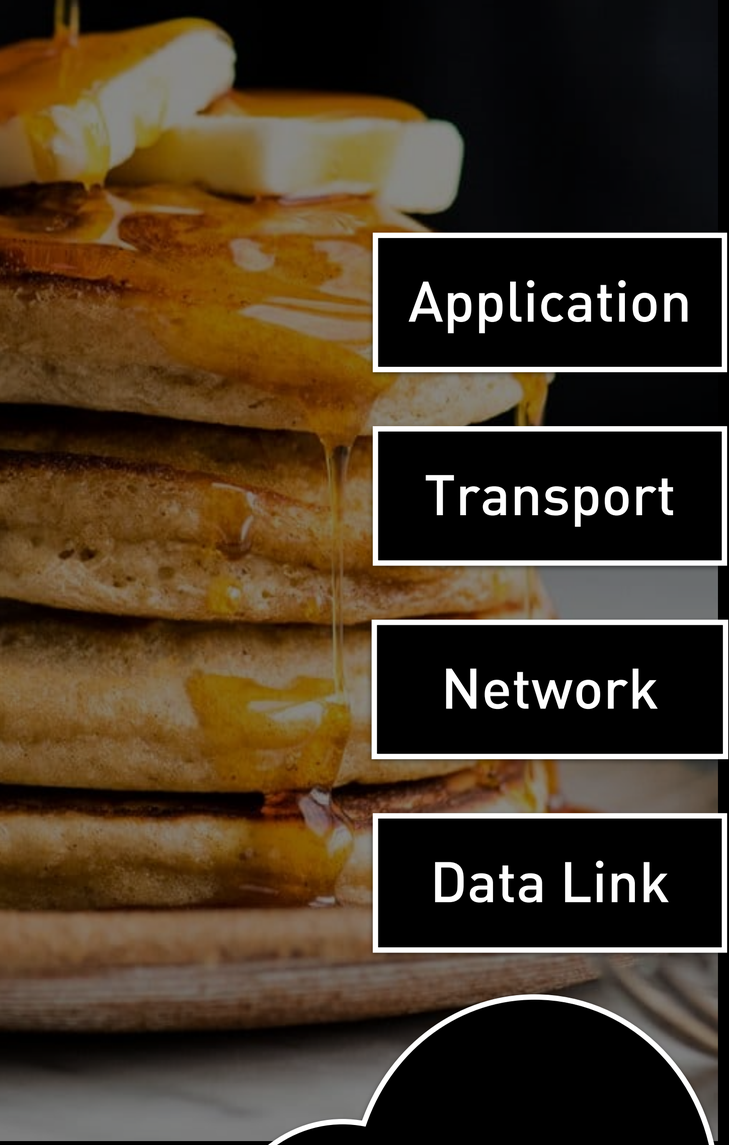
1. a wall or partition designed to inhibit or prevent the spread of fire.

verb

2. COMPUTING: protect a network or system from unauthorized access with a firewall. *"a firewalled network"*

Network Layers


Application
Transport
Network
Data Link
Ethernet / Fiber / Satellite
Network
Data Link
Ethernet / Fiber / Satellite
Application
Transport
Network
Data Link
Ethernet / Fiber / Satellite



Application

Transport

Network

Data Link

Ethernet / Fiber / Satellite

HTTP

TCP

UDP

IP

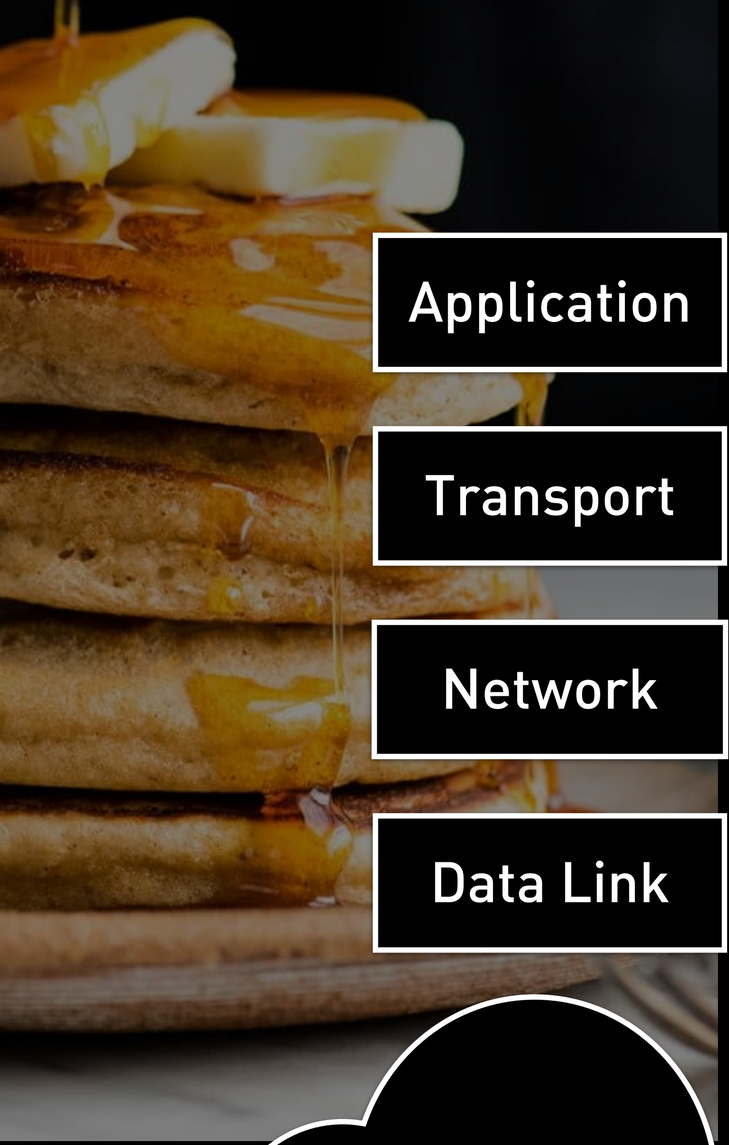
ICMP

ARP

Tunnels

Ethernet Frame

MAC



Application

Transport

Network

Data Link

Ethernet / Fiber / Satellite

HTTP

TCP

UDP

IP

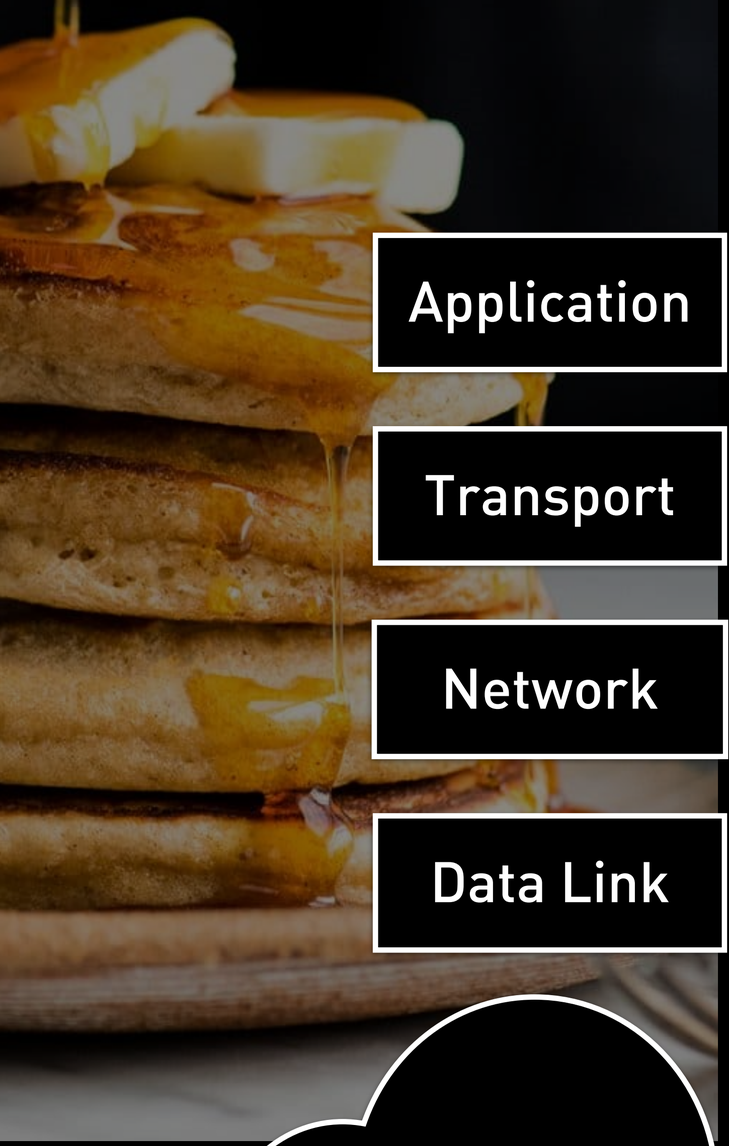
ICMP

ARP

Tunnels

Ethernet Frame

MAC



Application

Transport

Network

Data Link

Ethernet / Fiber / Satellite

HTTP

TCP

UDP

IP

ICMP

ARP

Tunnels

Ethernet Frame

MAC

IP Address

Let's look up a public IP address!


➜ ~
LUNAR
digital VT100


➜ ~
LUNAR
digital VT100

Let’s look up

our own

IP address!


~
LUNAR
digital VT100

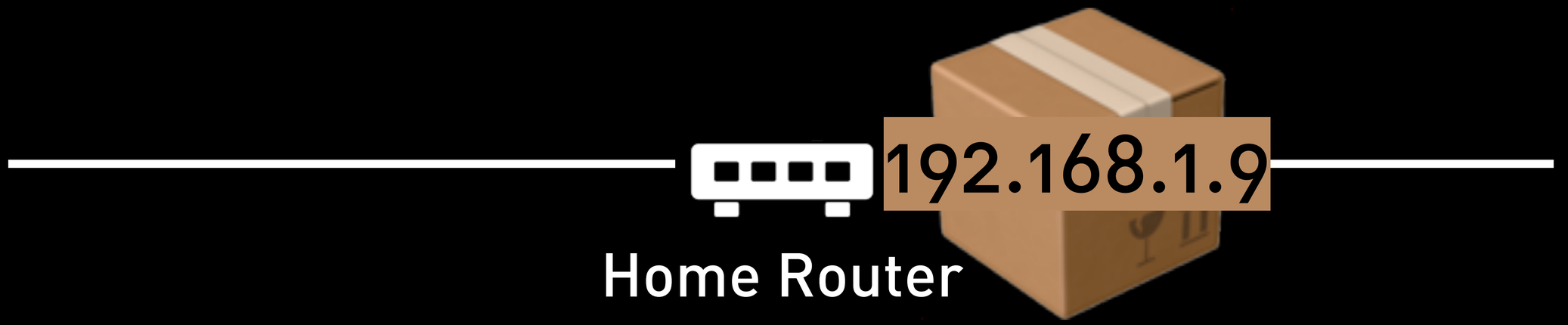
Let’s send some
test packet!


~
LUNAR
digital VT100

Building Block 1:

How NAT works


90.187.37.21
Home Router


192.168.1.9
Home Router


90.187.37.21
Home Router

Public IP Address

90.187.37.21

x.x.x.x

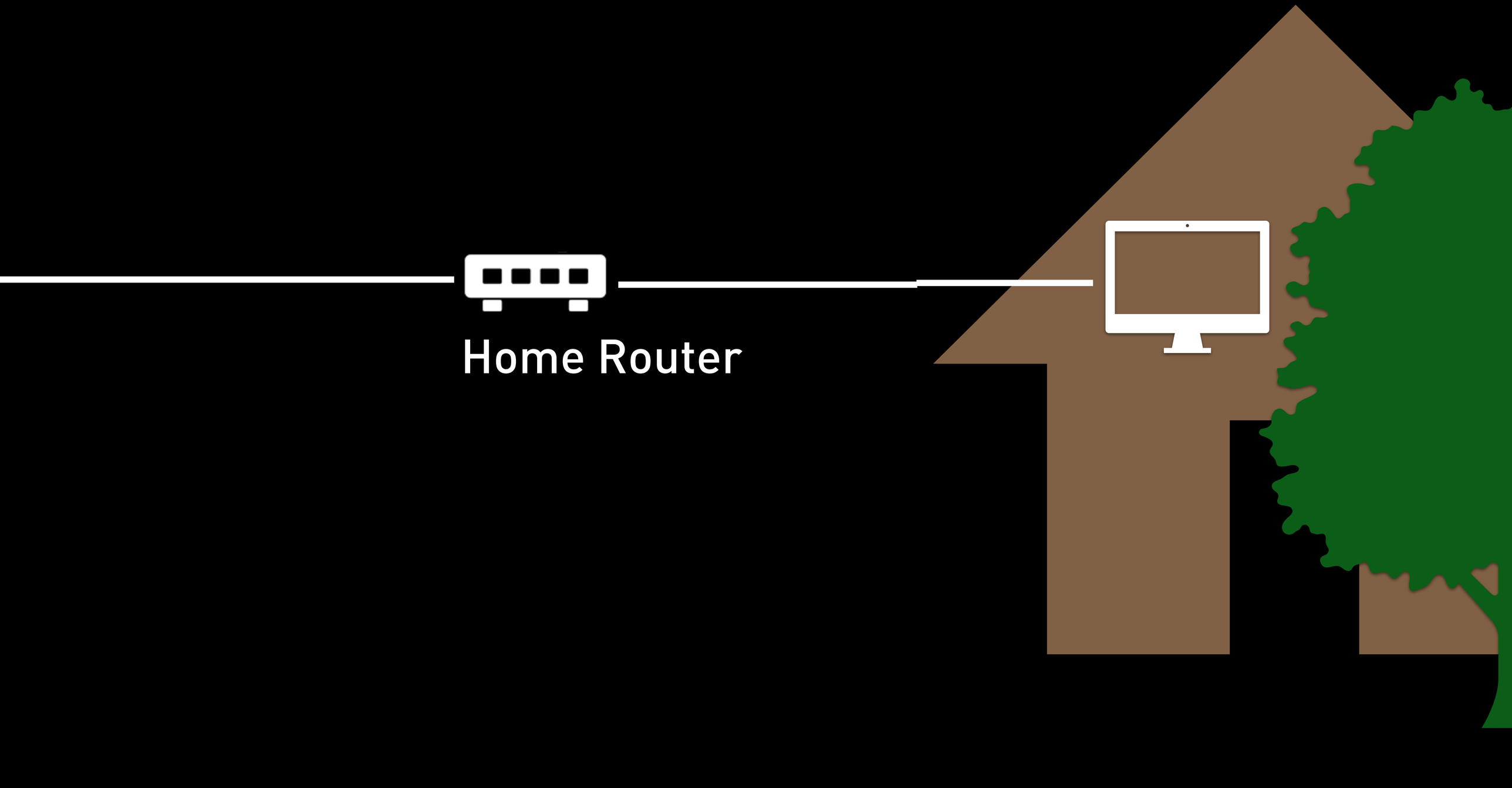
Publicly registered!!!

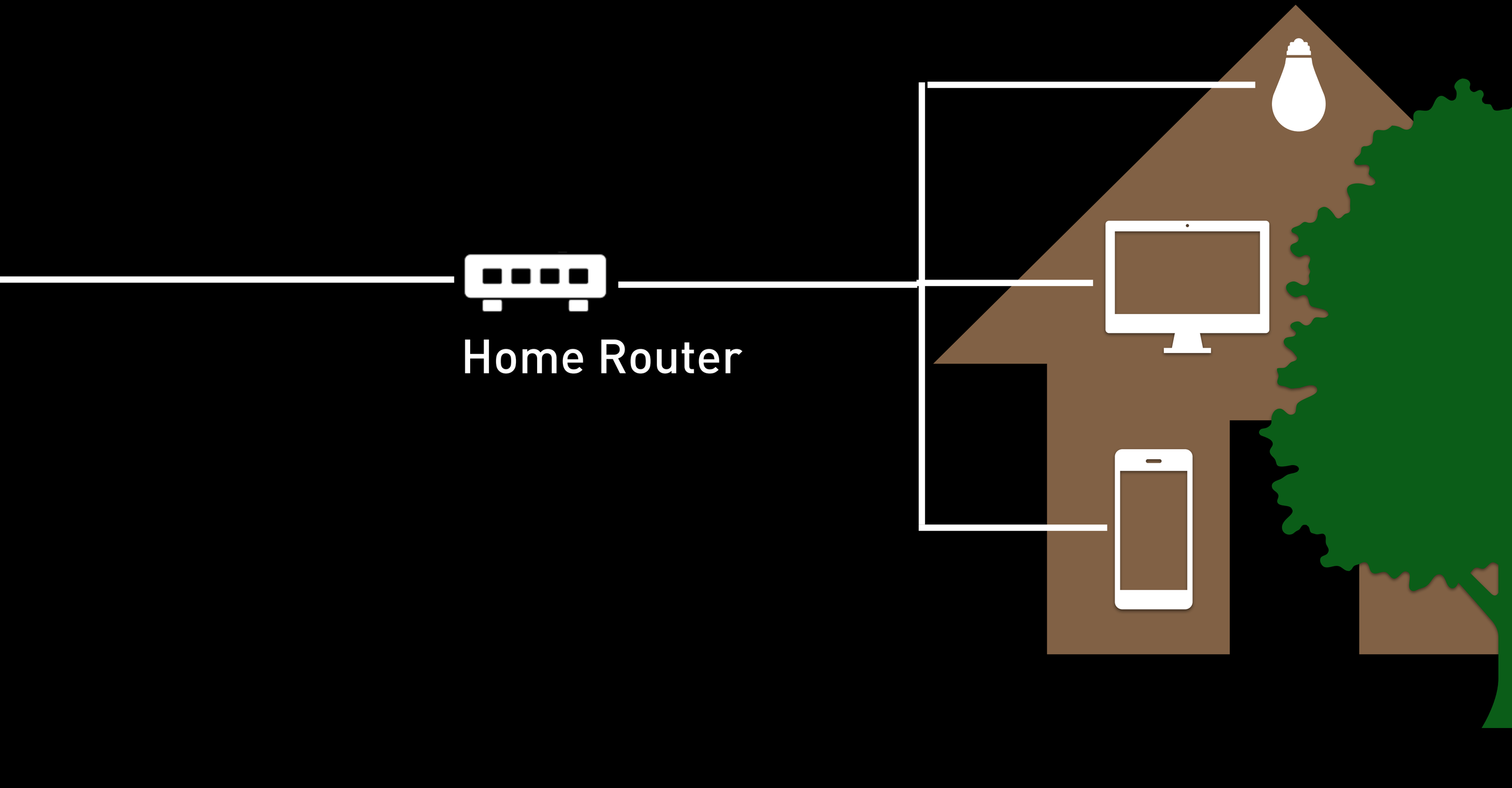
Private IP Address

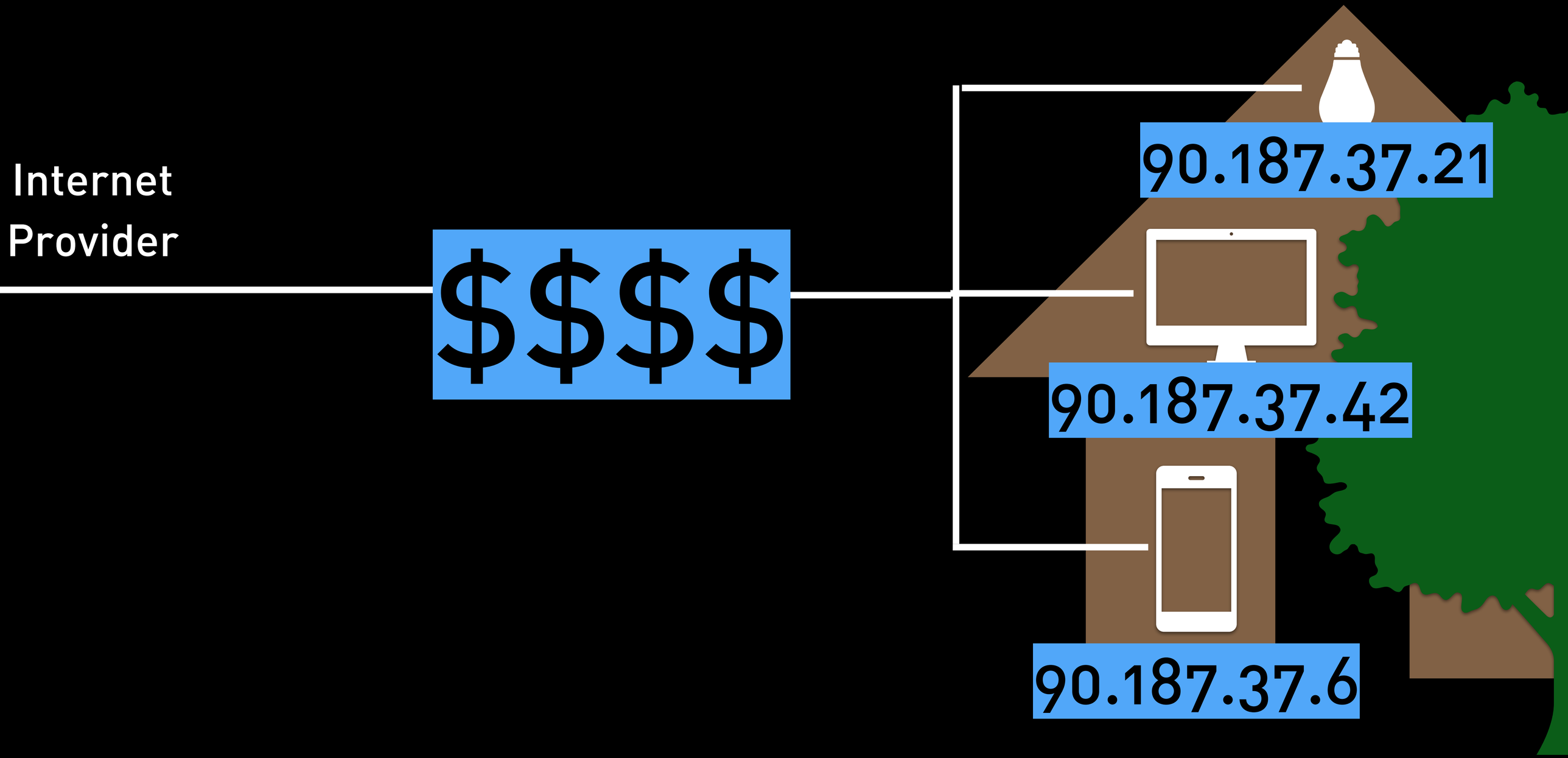
192.168.1.9

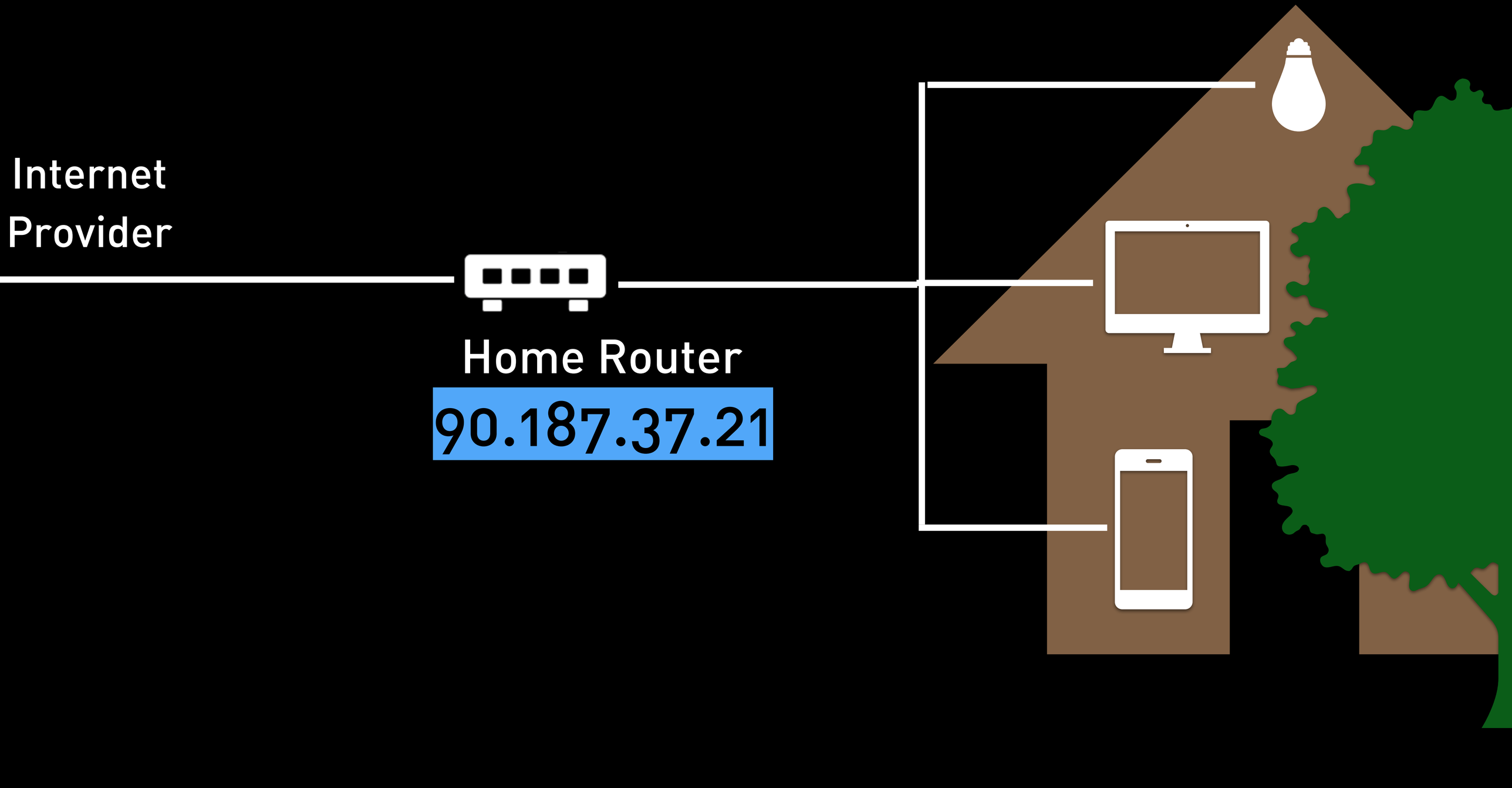
10.x.x.x

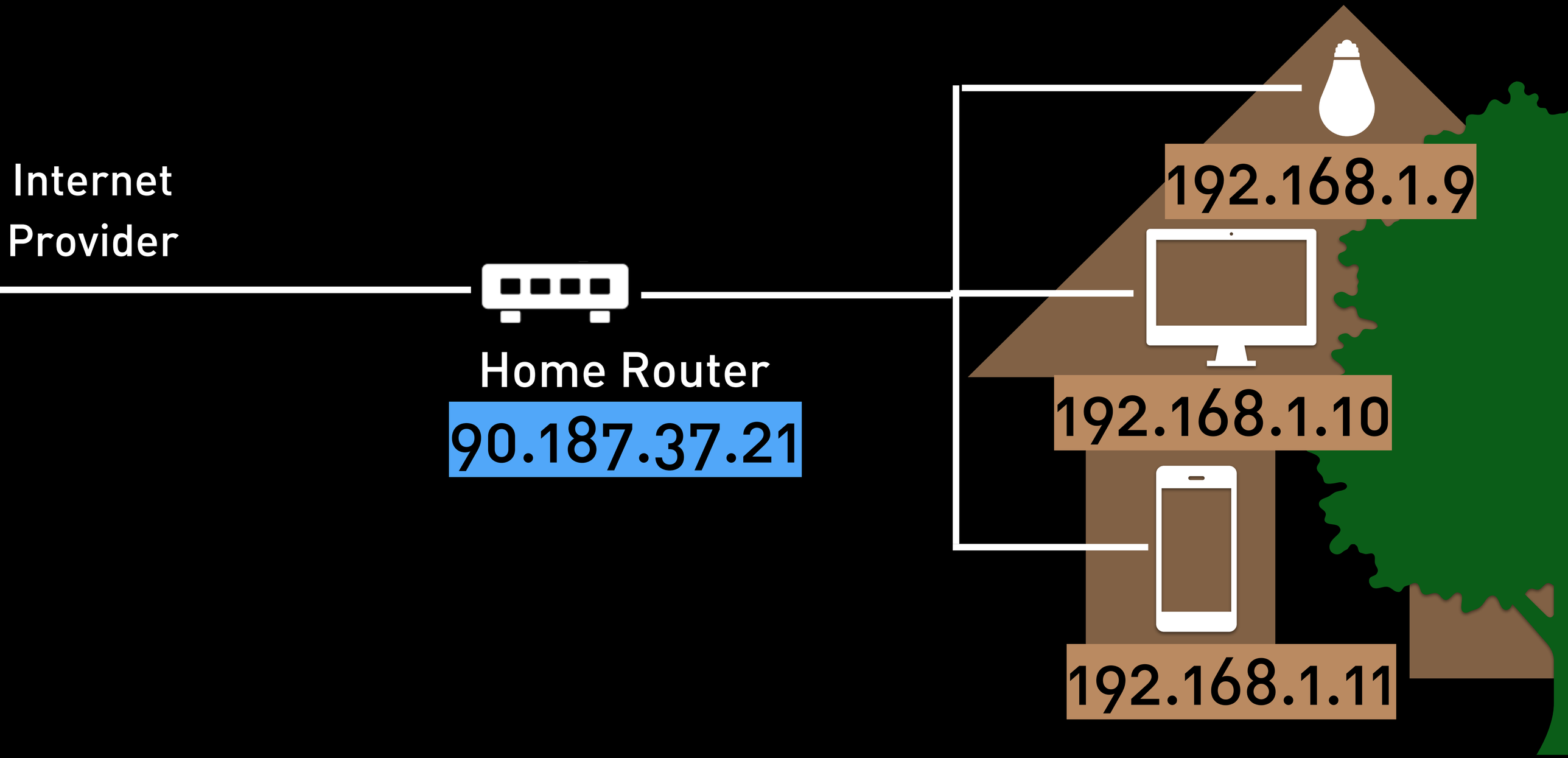
192.168.x.x

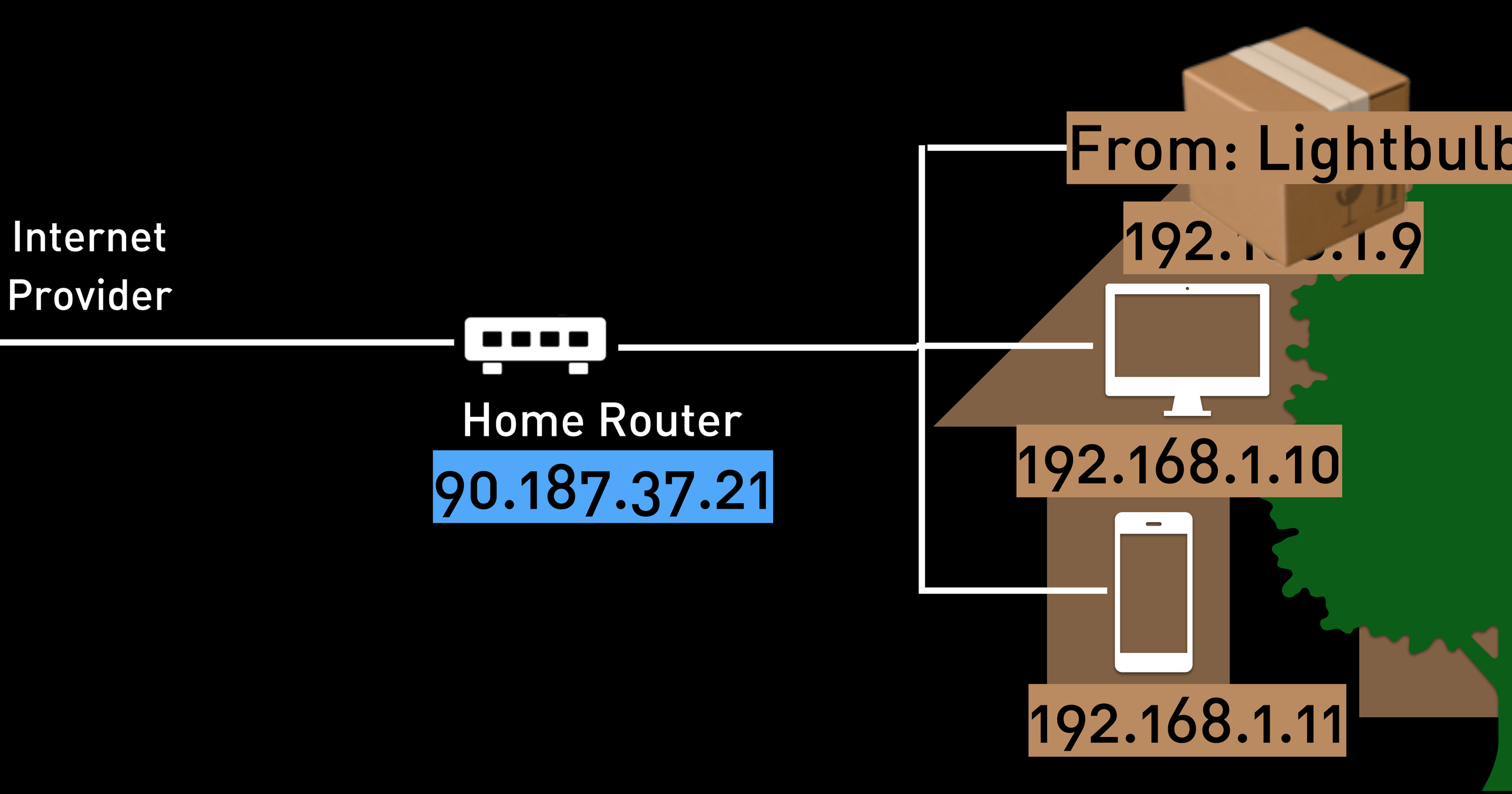

Home Router

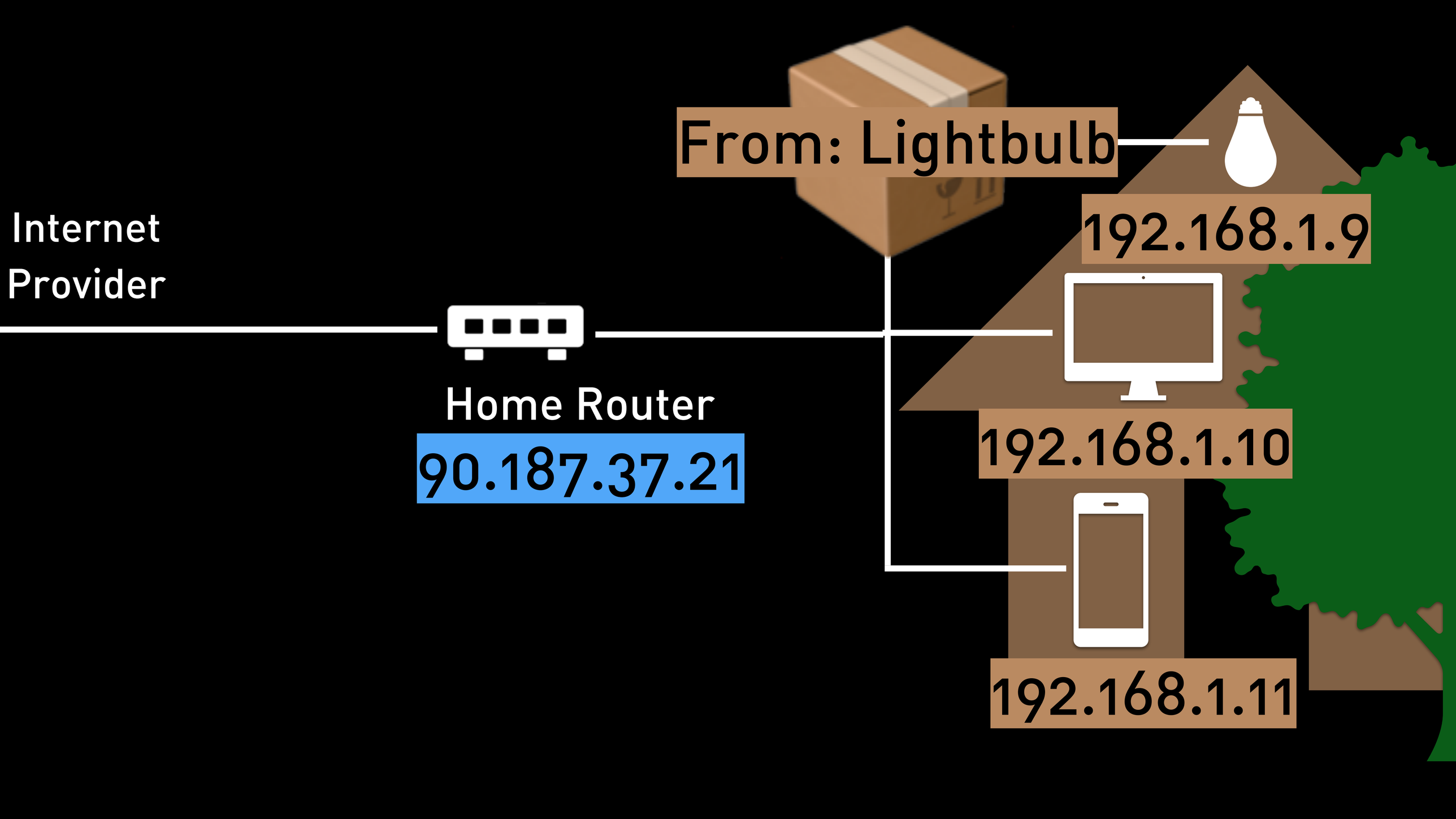

Home Router

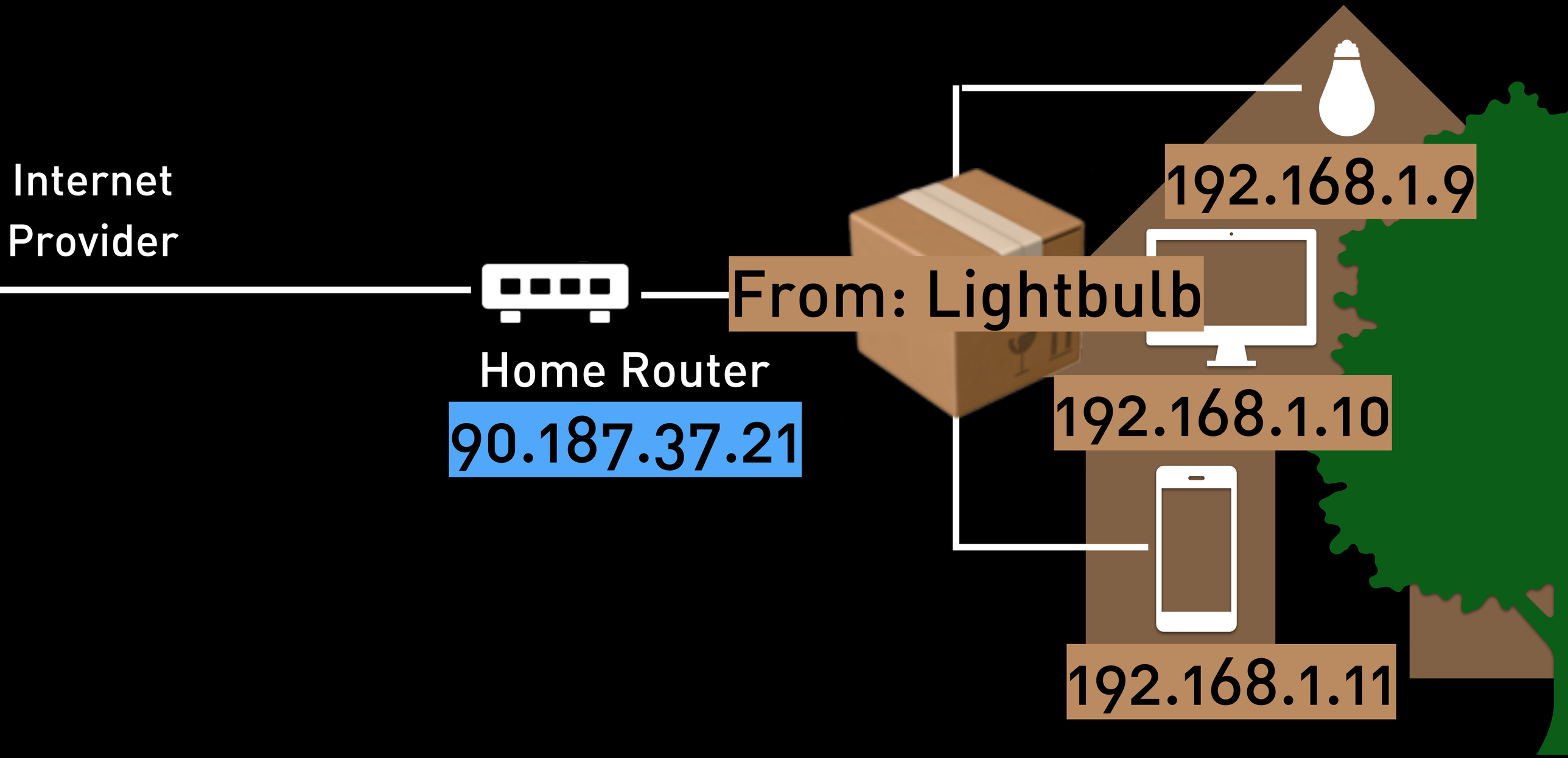

Internet
Provider
$$$$
90.187.37.21
90.187.37.42
90.187.37.6

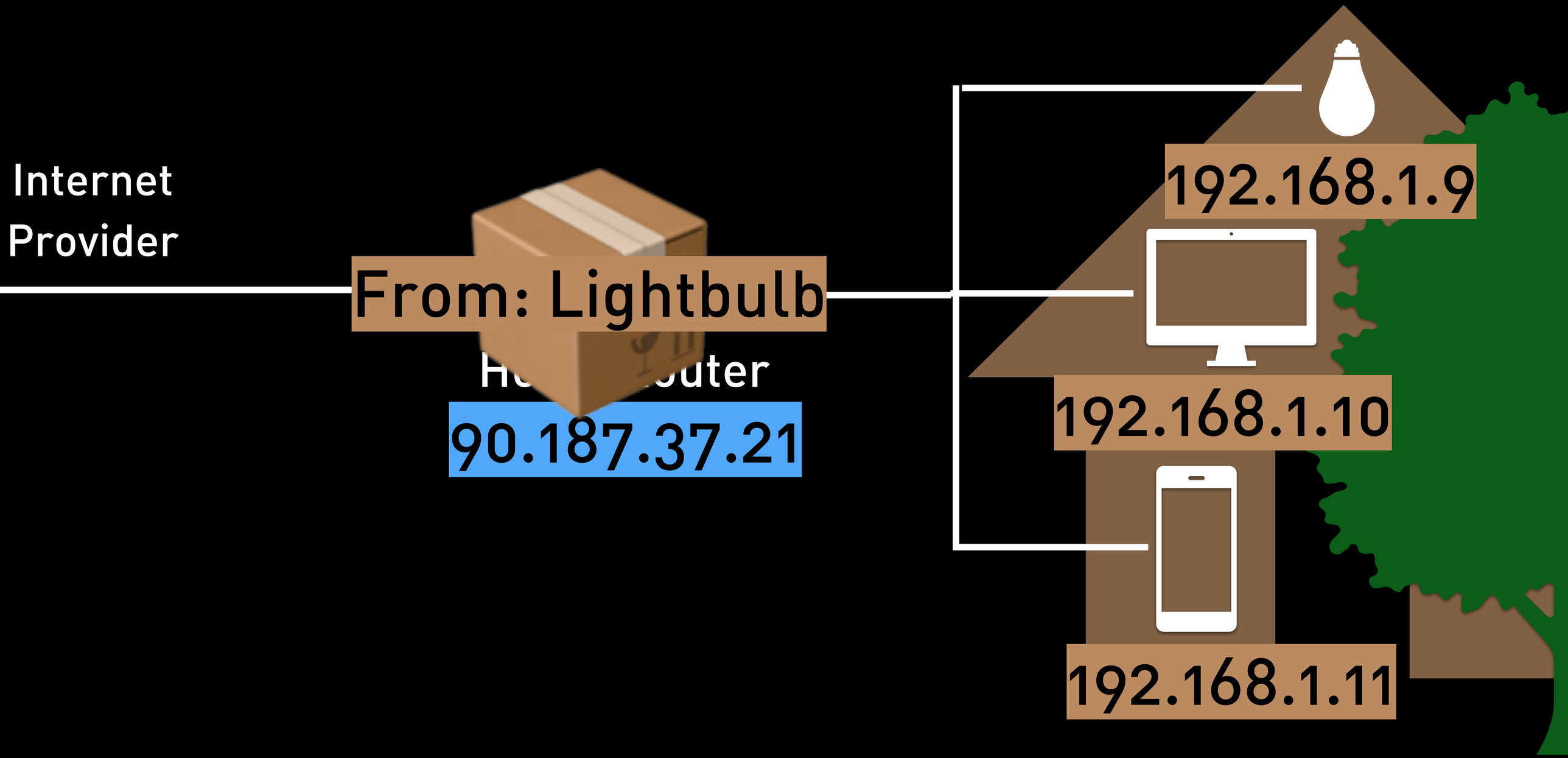

Internet
Provider
Home Router
90.187.37.21

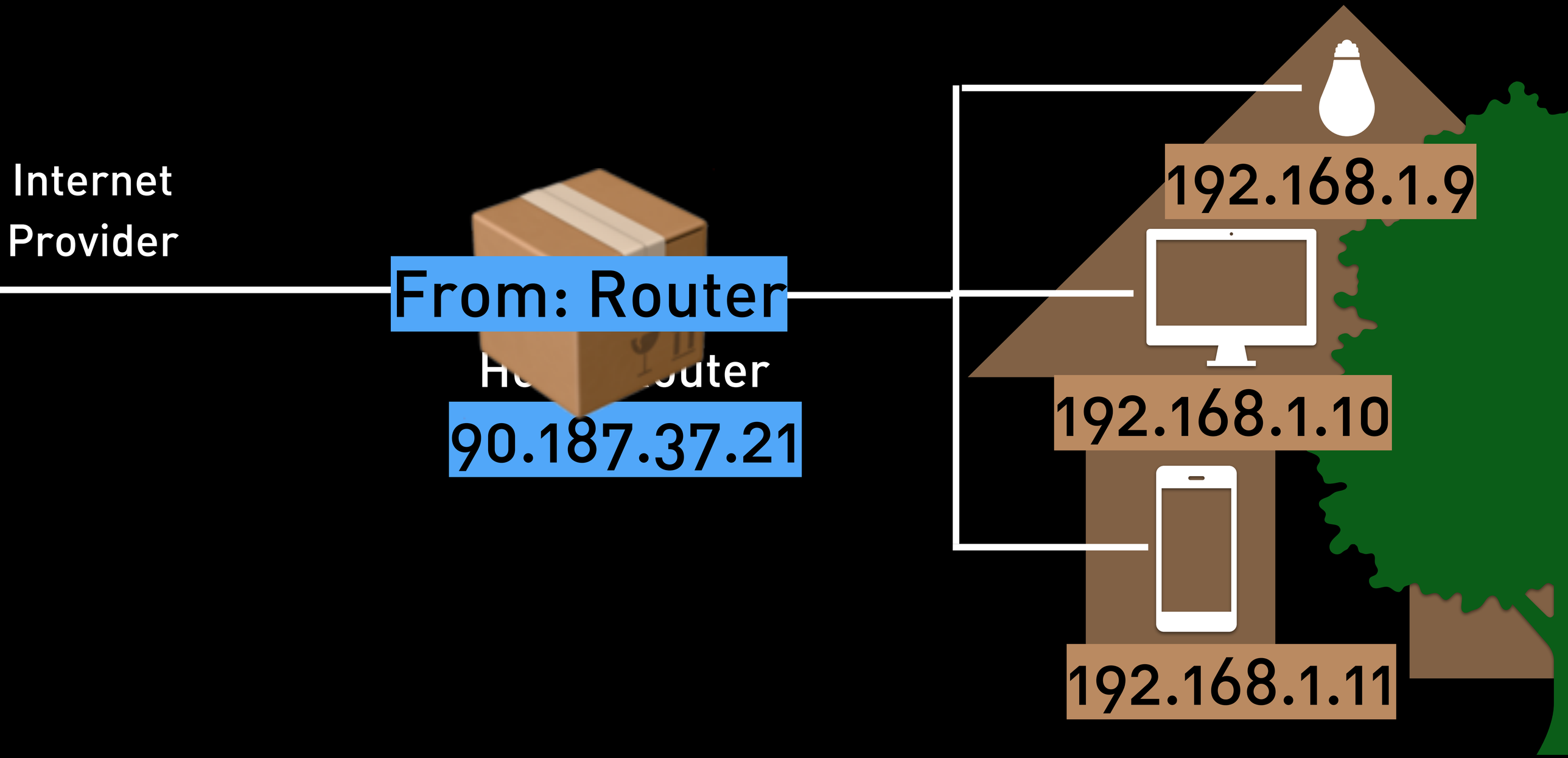

Internet
Provider
Home Router
90.187.37.21
192.168.1.9
192.168.1.10
192.168.1.11


From: Lightbulb
192.168.1.9
Internet
Provider
Home Router
90.187.37.21
192.168.1.10
192.168.1.11


From: Lightbulb
Internet Provider
Home Router
90.187.37.21
192.168.1.9
192.168.1.10
192.168.1.11


Internet
Provider
Home Router
90.187.37.21
From: Lightbulb
192.168.1.9
192.168.1.10
192.168.1.11


Internet
Provider
From: Lightbulb
90.187.37.21
192.168.1.9
192.168.1.10
192.168.1.11


Internet
Provider
From: Router
90.187.37.21
192.168.1.9
192.168.1.10
192.168.1.11

IN THE FUTURE™:

2003:cd:ebd1:f200:144b:4e7e:6c7:57bc

? : ? : ? : ? : ? : ? : ? : ?

0 - ffff

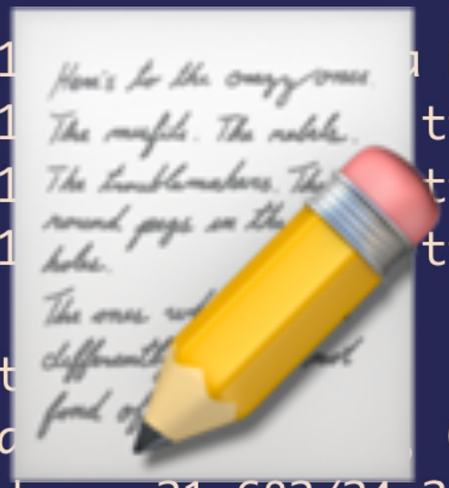
3400 000 000 000 000 000 000 000 000 000 000 000 000 000 000 000

Building Block 2:

Network Protocols

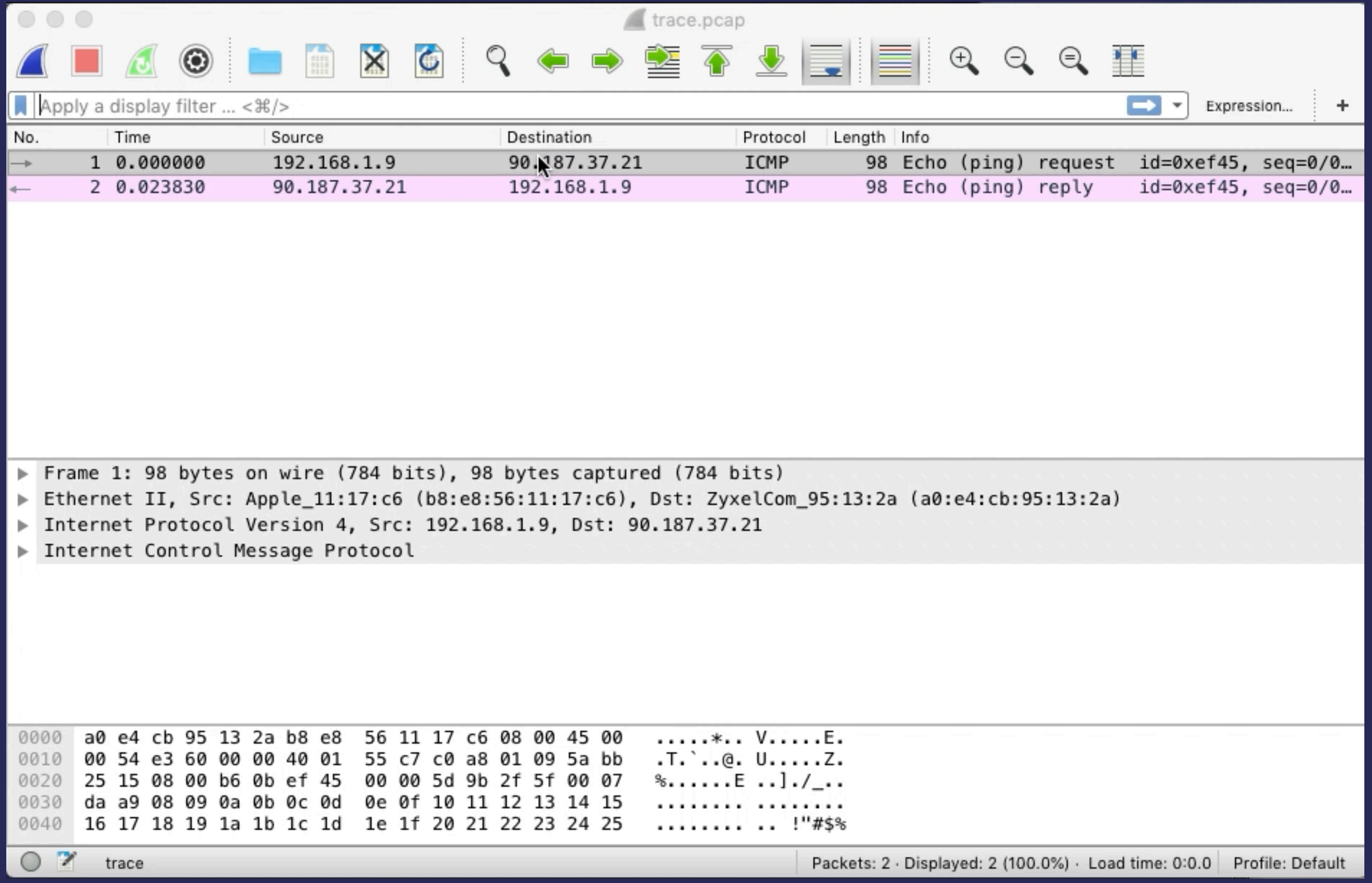
```
➜  ~ ping bobkonf.de
PING bobkonf.de (185.199.111.153): 56 data bytes
64 bytes from 185.199.111.153: icmp_seq=0 ttl=56 time=21.682 ms
64 bytes from 185.199.111.153: icmp_seq=1 ttl=56 time=28.356 ms
64 bytes from 185.199.111.153: icmp_seq=2 ttl=56 time=22.823 ms
^C
--- bobkonf.de ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 21.682/24.287/28.356/2.915 ms
➜  ~
```

```
➜  ~ ping bobkonf.de
PING bobkonf.de (185.199.11                    bytes
64 bytes from 185.199.111.1                    ttl=56 time=21.682 ms
64 bytes from 185.199.111.1                    ttl=56 time=28.356 ms
64 bytes from 185.199.111.1                    ttl=56 time=22.823 ms
^C
--- bobkonf.de ping statist
3 packets transmitted, 3 pa                    0.0% packet loss
round-trip min/avg/max/stddev = 21.682/24.287/28.356/2.915 ms
➜  ~
```



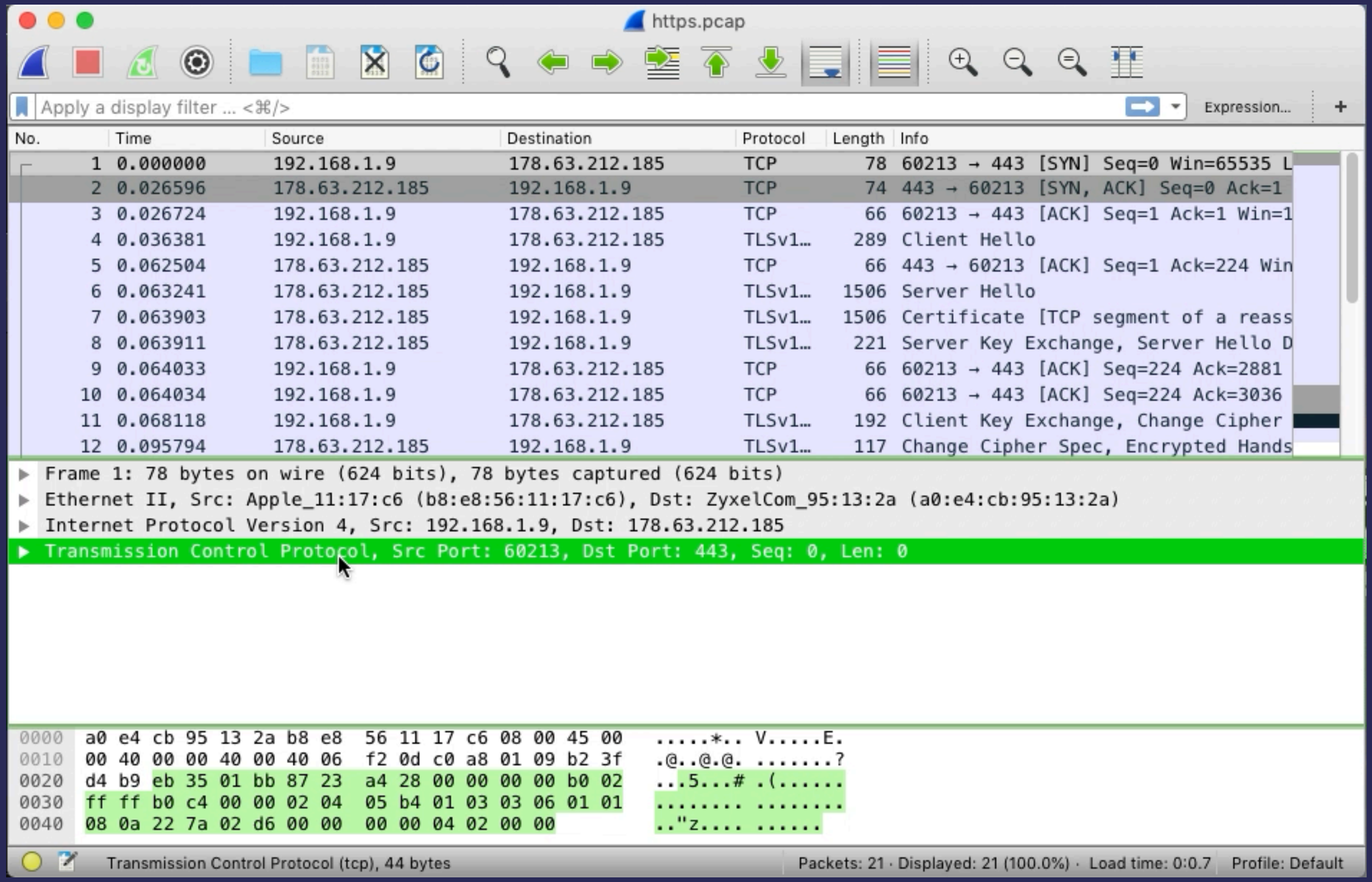
```
→ ~ sudo tcpdump -e -n -i en0 icmp -w ping.pcap
```

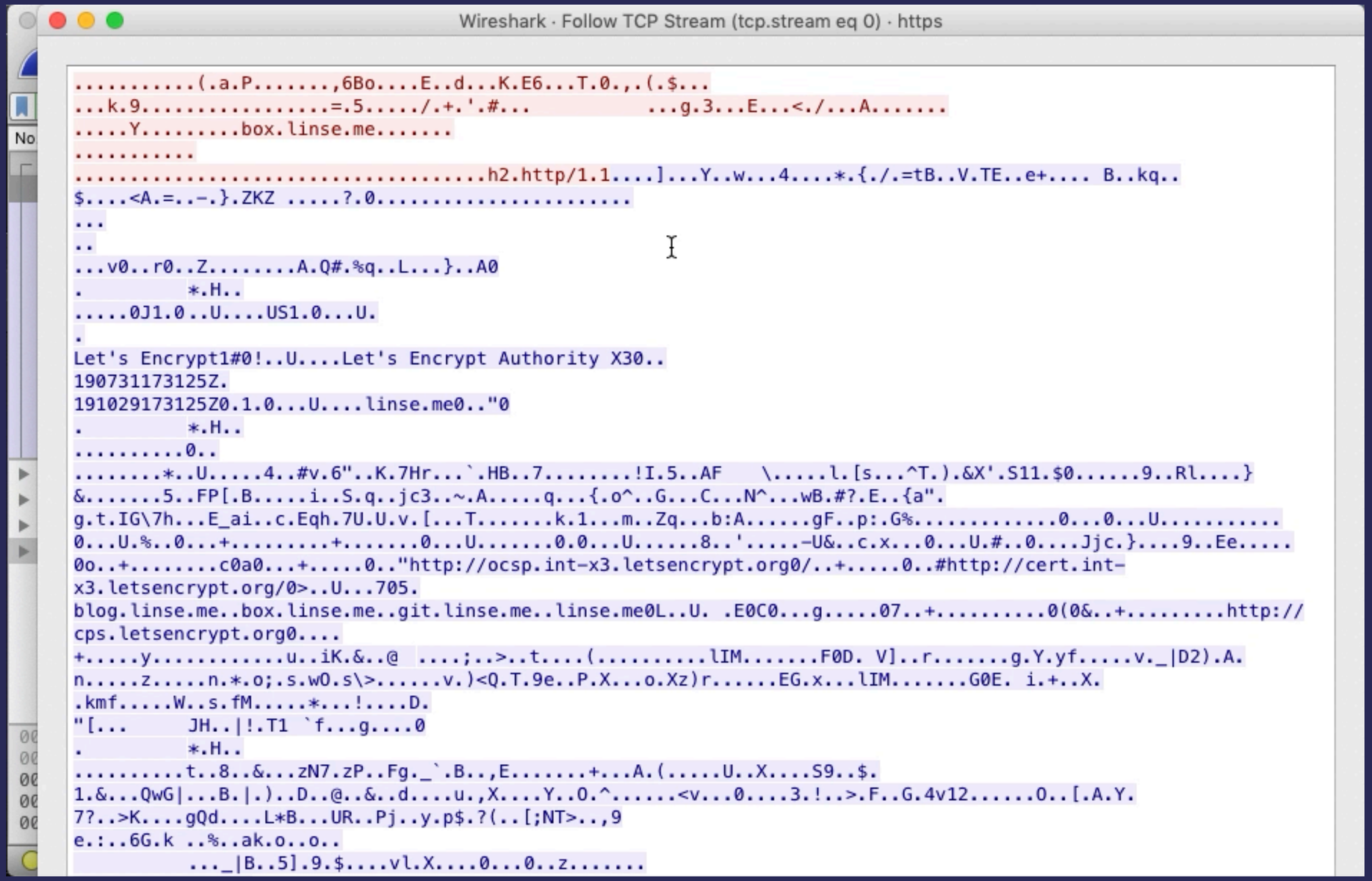
```
→ ~ wireshark ping.pcap
```


trace.pcap
Apply a display filter ... <⌘/>
Expression...
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.1.9 90.187.37.21 ICMP 98 Echo (ping) request id=0xef45, seq=0/0…
2 0.023830 90.187.37.21 192.168.1.9 ICMP 98 Echo (ping) reply id=0xef45, seq=0/0…
Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: Apple_11:17:c6 (b8:e8:56:11:17:c6), Dst: ZyxelCom_95:13:2a (a0:e4:cb:95:13:2a)
Internet Protocol Version 4, Src: 192.168.1.9, Dst: 90.187.37.21
Internet Control Message Protocol
0000 a0 e4 cb 95 13 2a b8 e8 56 11 17 c6 08 00 45 00 .....*.. V.....E.
0010 00 54 e3 60 00 00 40 01 55 c7 c0 a8 01 09 5a bb .T.`..@. U.....Z.
0020 25 15 08 00 b6 0b ef 45 00 00 5d 9b 2f 5f 00 07 %.....E ..]./_..
0030 da a9 08 09 0a 0b 0c 0d 0e 0f 10 11 12 13 14 15 ....
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .... !"#$%
trace
Packets: 2 · Displayed: 2 (100.0%) · Load time: 0:0.0
Profile: Default

Secure Protocols?

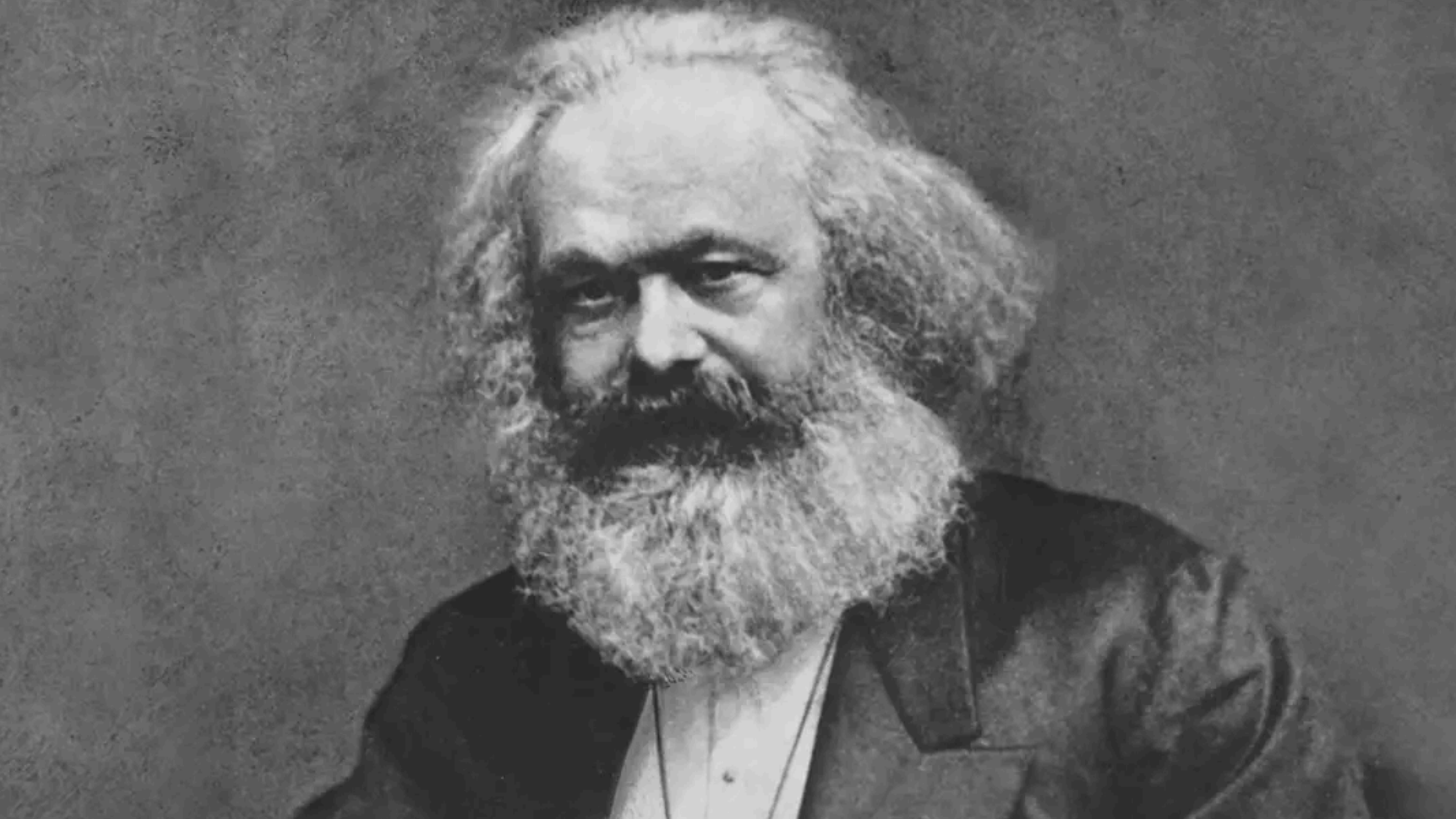
```
→ ~ curl https://box.linse.me/secret.txt
don't tell :)
```


https.pcap
Apply a display filter ... <⌘/>
Expression...
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.1.9 178.63.212.185 TCP 78 60213 → 443 [SYN] Seq=0 Win=65535 L
2 0.026596 178.63.212.185 192.168.1.9 TCP 74 443 → 60213 [SYN, ACK] Seq=0 Ack=1
3 0.026724 192.168.1.9 178.63.212.185 TCP 66 60213 → 443 [ACK] Seq=1 Ack=1 Win=1
4 0.036381 192.168.1.9 178.63.212.185 TLSv1… 289 Client Hello
5 0.062504 178.63.212.185 192.168.1.9 TCP 66 443 → 60213 [ACK] Seq=1 Ack=224 Win
6 0.063241 178.63.212.185 192.168.1.9 TLSv1… 1506 Server Hello
7 0.063903 178.63.212.185 192.168.1.9 TLSv1… 1506 Certificate [TCP segment of a reass
8 0.063911 178.63.212.185 192.168.1.9 TLSv1… 221 Server Key Exchange, Server Hello D
9 0.064033 192.168.1.9 178.63.212.185 TCP 66 60213 → 443 [ACK] Seq=224 Ack=2881
10 0.064034 192.168.1.9 178.63.212.185 TCP 66 60213 → 443 [ACK] Seq=224 Ack=3036
11 0.068118 192.168.1.9 178.63.212.185 TLSv1… 192 Client Key Exchange, Change Cipher
12 0.095794 178.63.212.185 192.168.1.9 TLSv1… 117 Change Cipher Spec, Encrypted Hands
Frame 1: 78 bytes on wire (624 bits), 78 bytes captured (624 bits)
Ethernet II, Src: Apple_11:17:c6 (b8:e8:56:11:17:c6), Dst: ZyxelCom_95:13:2a (a0:e4:cb:95:13:2a)
Internet Protocol Version 4, Src: 192.168.1.9, Dst: 178.63.212.185
Transmission Control Protocol, Src Port: 60213, Dst Port: 443, Seq: 0, Len: 0
0000 a0 e4 cb 95 13 2a b8 e8 56 11 17 c6 08 00 45 00 .....*.. V.....E.
0010 00 40 00 00 40 00 40 06 f2 0d c0 a8 01 09 b2 3f .@..@.@. ....?
0020 d4 b9 eb 35 01 bb 87 23 a4 28 00 00 00 00 b0 02 ...5...# .(......
0030 ff ff b0 c4 00 00 02 04 05 b4 01 03 03 06 01 01 ....
0040 08 0a 22 7a 02 d6 00 00 00 00 04 02 00 00 ..."z.... ......
Transmission Control Protocol (tcp), 44 bytes
Packets: 21 · Displayed: 21 (100.0%) · Load time: 0:0.7
Profile: Default


Wireshark · Follow TCP Stream (tcp.stream eq 0) · https
.....(.a.P.....,6Bo....E..d...K.E6...T.0.,.(.$...
...k.9.....=.5...../.+.'.#... ...g.3...E...<./...A.....
.....Y.....box.linse.me.....
.....
.....h2.http/1.1....]...Y..w...4....*.{./.=tB..V.TE..e+.... B..kq..
$....<A.=..-.}.ZKZ .....?.0.....
...
..
...v0..r0..Z.....A.Q#.%q..L...}..A0
. *.H..
.....0J1.0..U....US1.0...U.
.
Let's Encrypt1#0!..U....Let's Encrypt Authority X30..
190731173125Z.
191029173125Z0.1.0...U....linse.me0.."0
. *.H..
.....0..
.....*..U.....4..#v.6"..K.7Hr...`.HB..7.....!I.5..AF \.....l.[s...^T.).&X'.S11.$0.....9..Rl....}
&.....5..FP[.B.....i..S.q..jc3..~.A.....q...{.o^..G...C...N^...wB.#?.E..{a".
g.t.IG\7h...E_ai..c.Eqh.7U.U.v.[...T.....k.1...m..Zq...b:A.....gF..p:.G%.....0...0...U.....
0...U.%..0...+.....+.....0...U.....0.0...U.....8..'.....-U&..c.x...0...U.#..0....Jjc.}....9..Ee.....
0o..+.....c0a0...+.....0.."http://ocsp.int-x3.letsencrypt.org0/..+.....0..#http://cert.int-
x3.letsencrypt.org/0>..U...705.
blog.linse.me..box.linse.me..git.linse.me..linse.me0L..U. .E0C0...g.....07..+.....0(0&..+.....http://
cps.letsencrypt.org0....
+.....y.....u..iK.&..@ ....;..>..t....(.....lIM.....F0D. V]..r.....g.Y.yf.....v._|D2).A.
n.....z.....n.*.o;.s.wO.s\>.....v.)<Q.T.9e..P.X...o.Xz)r.....EG.x...lIM.....G0E. i.+..X.
.kmf.....W..s.fM.....*...!....D.
"[... JH..|!.T1 `f...g....0
. *.H..
.....t..8..&...zN7.zP..Fg._`.B..,E.....+...A.(.....U..X....S9..$.
1.&...QwG|...B.|.)..D..@..&..d....u.,X....Y..O.^.....<v...0....3.!..>.F..G.4v12.....O..[.A.Y.
7?..>K....gQd....L*B...UR..Pj..y.p$.?(..[;NT>..,9
e.:..6G.k ..%..ak.o..o..
  ..._|B..5].9.$....vl.X....0...0..z.....

Universe:

QubesOS

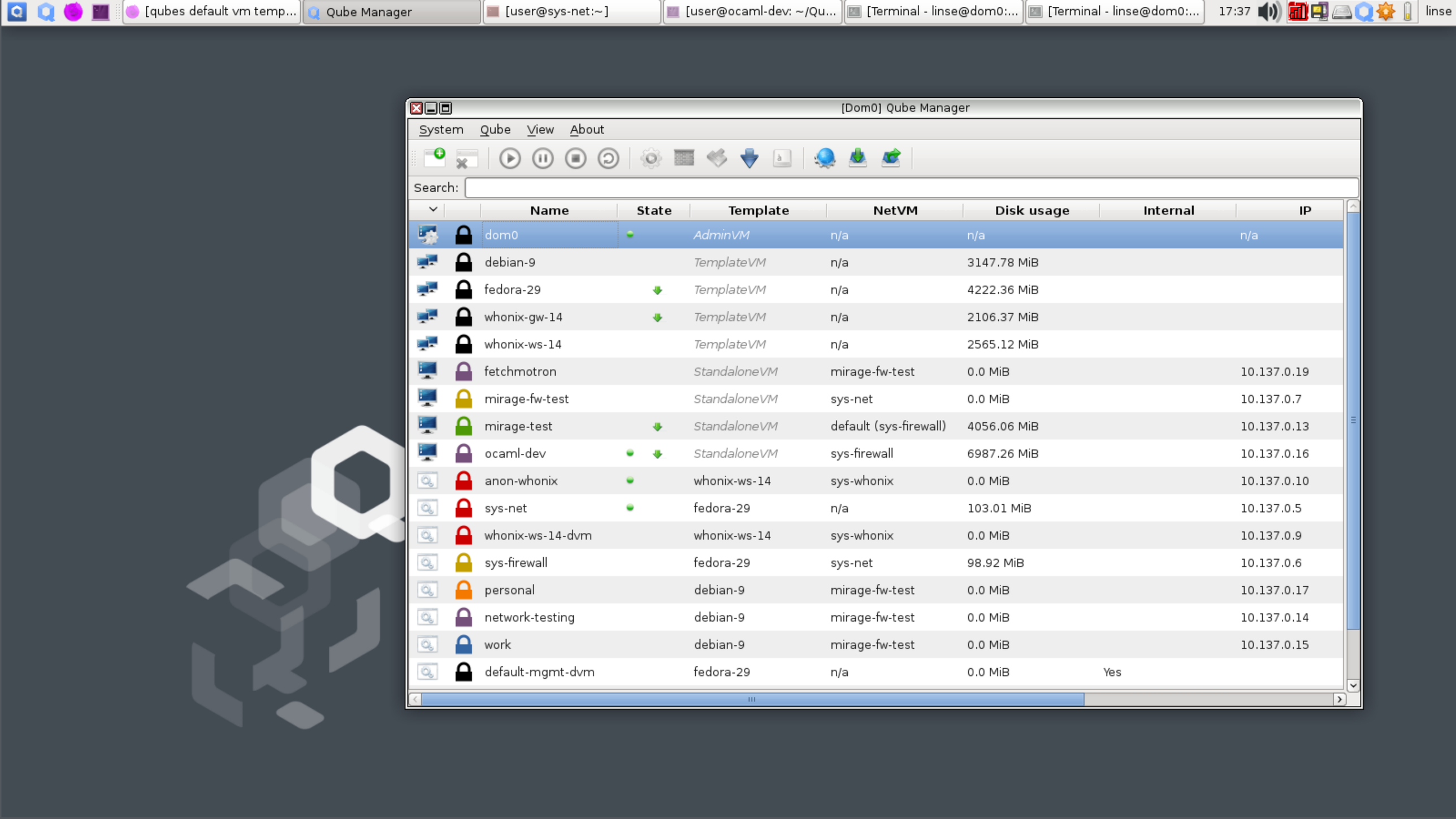


Universe:

QubesOS

Universe:

QubesOS


[qubes default vm temp...
Qube Manager
[user@sys-net:~]
[user@ocaml-dev: ~/Qu...
[Terminal - linse@dom0:...
[Terminal - linse@dom0:...
17:37
linse
[Dom0] Qube Manager
System
Qube
View
About
Search:
Name
State
Template
NetVM
Disk usage
Internal
IP
dom0 AdminVM n/a n/a n/a
debian-9 TemplateVM n/a 3147.78 MiB
fedora-29 TemplateVM n/a 4222.36 MiB
whonix-gw-14 TemplateVM n/a 2106.37 MiB
whonix-ws-14 TemplateVM n/a 2565.12 MiB
fetchmotron StandaloneVM mirage-fw-test 0.0 MiB 10.137.0.19
mirage-fw-test StandaloneVM sys-net 0.0 MiB 10.137.0.7
mirage-test StandaloneVM default (sys-firewall) 4056.06 MiB 10.137.0.13
ocaml-dev StandaloneVM sys-firewall 6987.26 MiB 10.137.0.16
anon-whonix whonix-ws-14 sys-whonix 0.0 MiB 10.137.0.10
sys-net fedora-29 n/a 103.01 MiB 10.137.0.5
whonix-ws-14-dvm whonix-ws-14 sys-whonix 0.0 MiB 10.137.0.9
sys-firewall fedora-29 sys-net 98.92 MiB 10.137.0.6
personal debian-9 mirage-fw-test 0.0 MiB 10.137.0.17
network-testing debian-9 mirage-fw-test 0.0 MiB 10.137.0.14
work debian-9 mirage-fw-test 0.0 MiB 10.137.0.15
default-mgmt-dvm fedora-29 n/a 0.0 MiB Yes

Qubes OS - Wikiped... | [Qube Manager] | [user@sys-net:~] | user@ocaml-dev: ~... | [user@ocaml-dev: ... | [Terminal - linse@d... | [Terminal - linse@d... | 14:23 | linse

[ocaml-dev] user@ocaml-dev: ~/cloned/qubes-mirage-firewall

File Edit View Search Terminal Help

```
  )

(* Main unikernel entry point (called from auto-generated main.ml). *)
let start _random clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in

  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexe
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clo
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexe
  (* Watch for shutdown requests from Qub
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request (
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns cloc
  let max_entries = Key_gen.nat_table_siz
  My_nat.create ~get_time ~max_entries >>

  (* Read network configuration from Qube
  Dao.read_network_config qubesDB >>= fun

  let rng i = R.generate i in

  Uplink.connect ~clock config >>= fun up

  let uplink' = Uplink.send_dns_client_qu
  let stack = (uplink', Lwt_mvar.create_e
  let dns_client = My_dns_client.Dns_clie

  let net_listener = network ~clock confi

  (* Report memory usage to XenStore *)
unikernel.ml
```

[ocaml-dev] Qubes OS - Wikipedia - Mozilla Firefox

Qubes OS - Wikipedia | Imgur: The magic of the

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

Not logged in Talk Contributions Create account Log in

Article Talk | Read Edit View history | Search Wikipedia

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file

Qubes OS

From Wikipedia, the free encyclopedia
(Redirected from Qubesos)

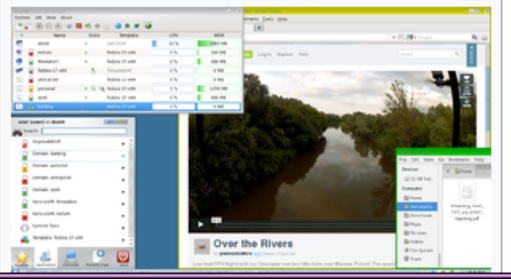
*"Qubes" redirects here. For the arcade game, see Q*bert's Qubes.*

Qubes OS is a security-focused desktop operating system that aims to provide security through isolation.[6] Virtualization is performed by Xen, and user environments can be based on Fedora, Debian, Whonix, and Microsoft Windows, among other operating systems.[7][8]

Contents [hide]

1 Security goals
2 System architecture overview
  2.1 Xen hypervisor and administrative domain (Dom0)
  2.2 Network domain
  2.3 Application Virtual Machines (AppVM)
3 Reception
4 See also
5 References

Qubes OS



Toolchest:

QubesOS ♥ MirageOS

trends in systems programming

microservices

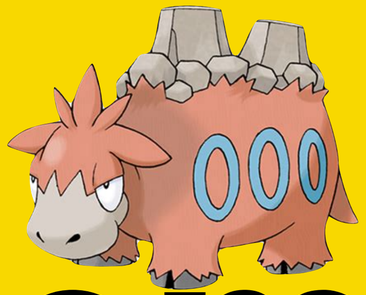
..unikernels

containerization

..library OS

virtualized hardware

.. targets hypervisor



OCaml

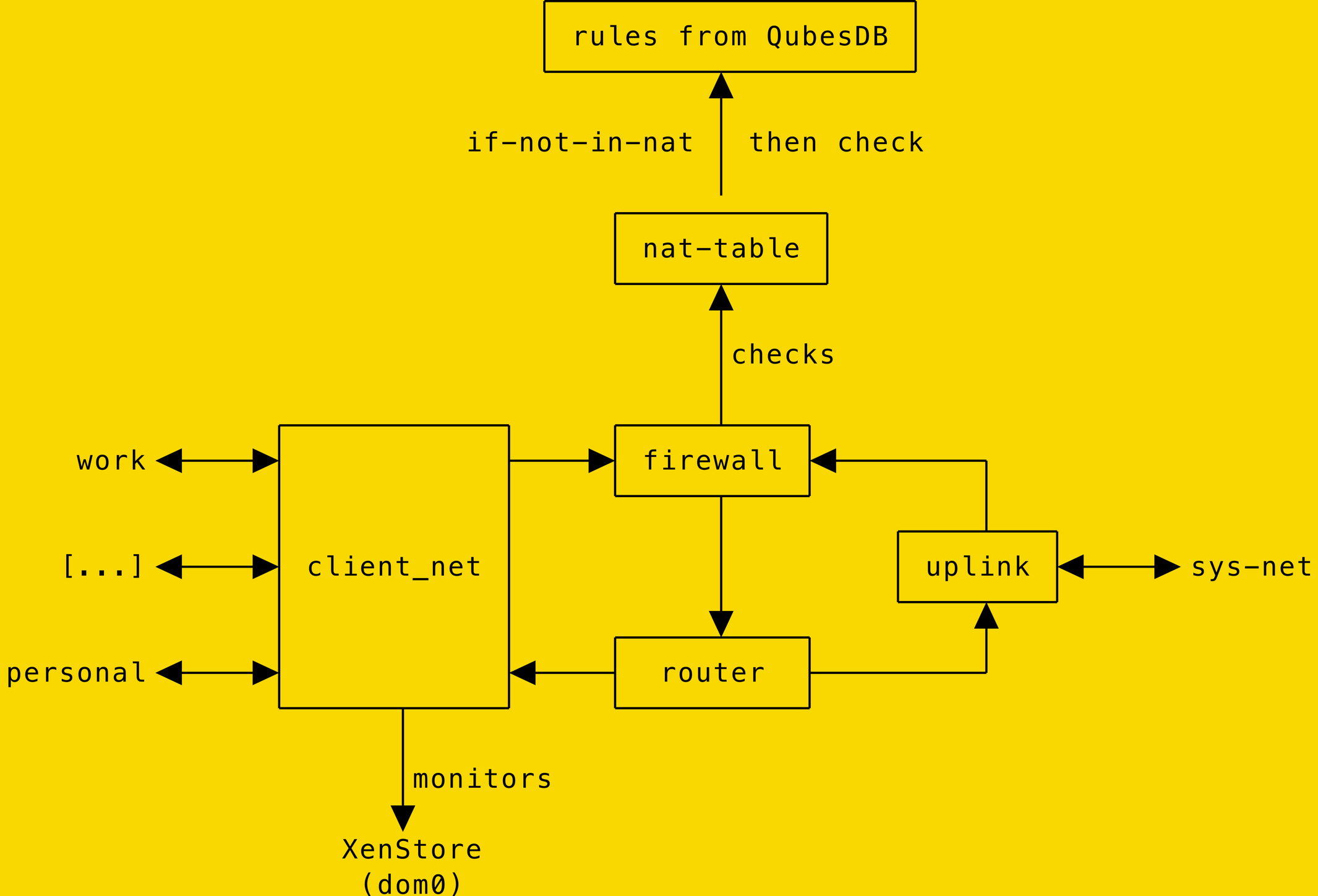
Unikernel

Library Operating System

Hypervisor

Firewall

`unikernel.ml`


rules from QubesDB
if-not-in-nat
then check
nat-table
checks
work
[...]
personal
client_net
firewall
router
uplink
sys-net
monitors
XenStore
(dom0)

```
qubes-mirage-firewall git:(master) → ~ wc -l **/*.ml
      14 cleanup.ml
     131 client_eth.ml
     170 client_net.ml
      27 command.ml
      44 config.ml
     159 dao.ml
     129 firewall.ml
      32 frameQ.ml
      48 fw_utils.ml
      53 memory_pressure.ml
      57 my_dns.ml
     109 my_nat.ml
      64 packet.ml
      16 ports.ml
      37 router.ml
     110 rules.ml
      27 test/config.ml
     357 test/unikernel.ml
      94 unikernel.ml
      97 uplink.ml
    1775 total
```

Assembly:

1: network listener

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

Network:
-clock
-NAT
-QubesDB

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

1. read network config

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

2. connect uplink

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

3. prepare client networking

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

4. route via NAT

```
open Lwt
open Qubes

let src = Logs.Src.create "unikernel" ~doc:"Main unikernel code"
module Log = (val Logs.src_log src : Logs.LOG)

module Main (Clock : Mirage_clock_lwt.MCLOCK) = struct
  module Uplink = Uplink.Make(Clock)

  (* Set up networking and listen for incoming packets. *)
  let network ~clock nat qubesDB =
    (* Read configuration from QubesDB *)
    let config = Dao.read_network_config qubesDB in
    (* Initialise connection to NetVM *)
    Uplink.connect ~clock config >>= fun uplink ->
    (* Report success *)
    Dao.set_iptables_error qubesDB "" >>= fun () ->
    (* Set up client-side networking *)
    let client_eth = Client_eth.create
      ~client_gw:config.Dao.clients_our_ip in
    (* Set up routing between networks and hosts *)
    let router = Router.create
      ~client_eth
      ~uplink:(Uplink.interface uplink)
      ~nat
    in
    (* Handle packets from both networks *)
    Lwt.choose [
      Client_net.listen router;
      Uplink.listen uplink router
    ]
```

5. handle packets

Assembly:

2: start function

```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

main entry point: calls start function

```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

1. note start time,
connect to QubesOS:
- remote exec interface
- GUI
- settings DB

```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

2. output startup time

```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

3. handle shutdown from Qubes

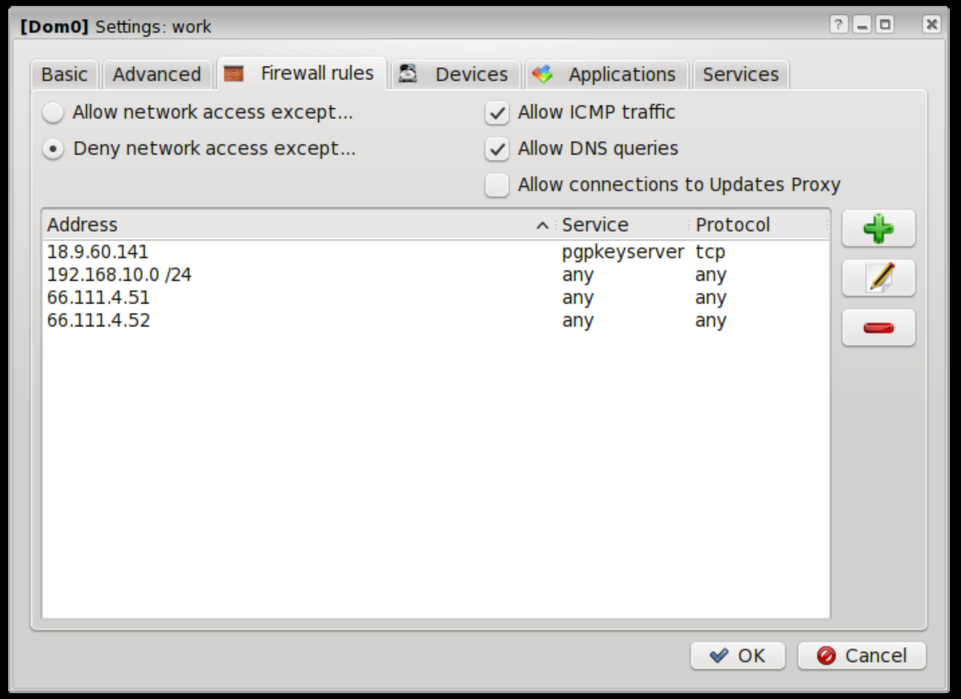
```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

4. prepare NAT table call network function

```
(* Main unikernel entry point (called from auto-generated main.ml). *)
let start clock =
  let start_time = Clock.elapsed_ns clock in
  (* Start qrexec agent, GUI agent and QubesDB agent in parallel *)
  let qrexec = RExec.connect ~domid:0 () in
  GUI.connect ~domid:0 () |> watch_gui;
  let qubesDB = DB.connect ~domid:0 () in
  (* Wait for clients to connect *)
  qrexec >>= fun qrexec ->
  let agent_listener = RExec.listen qrexec Command.handler in
  qubesDB >>= fun qubesDB ->
  let startup_time =
    let (-) = Int64.sub in
    let time_in_ns = Clock.elapsed_ns clock - start_time in
    Int64.to_float time_in_ns /. 1e9
  in
  Log.info (fun f -> f "QubesDB and qrexec agents connected in %.3f s" startup_time);
  (* Watch for shutdown requests from Qubes *)
  let shutdown_rq =
    OS.Lifecycle.await_shutdown_request () >>= fun (`Poweroff | `Reboot) ->
    return () in
  (* Set up networking *)
  let get_time () = Clock.elapsed_ns clock in
  let max_entries = Key_gen.nat_table_size () in
  My_nat.create ~get_time ~max_entries >>= fun nat ->
  let net_listener = network ~clock nat qubesDB in
  (* Report memory usage to XenStore *)
  Memory_pressure.init ();
  (* Run until something fails or we get a shutdown request. *)
  Lwt.choose [agent_listener; net_listener; shutdown_rq] >>= fun () ->
  (* Give the console daemon time to show any final log messages. *)
  OS.Time.sleep_ns (1.0 *. 1e9 |> Int64.of_float)
end
```

5. handle events, housekeeping

Configuring Your Firewall


[Dom0] Settings: work
Basic
Advanced
Firewall rules
Devices
Applications
Services
Allow network access except...
Deny network access except...
Allow ICMP traffic
Allow DNS queries
Allow connections to Updates Proxy
Address
Service
Protocol
18.9.60.141 pgpkeyserver tcp
192.168.10.0 /24 any any
66.111.4.51 any any
66.111.4.52 any any
OK
Cancel

Testing Your Firewall

Takeaways

- security in depth: not just patches
- try a few tools and look at traffic
- explore your network stack

- play with wireshark :D
- don’t block yourself :D

Thanks!!! ♥

Try it soon in QubesOS.

Thanks!!! ♥

Try it soon in QubesOS.

🙏 Thanks to
Marek Marczykowski-Górecki
from Invisible Things Lab!!!

Thanks!!! ♥

Try it soon in QubesOS.

We are **Stefanie Schirmer..**
linse.me
@linse

..and **Mindy Preston**
somerandomidiot.com
@yomimono

https://github.com/mirage/qubes-mirage-firewall