DNS

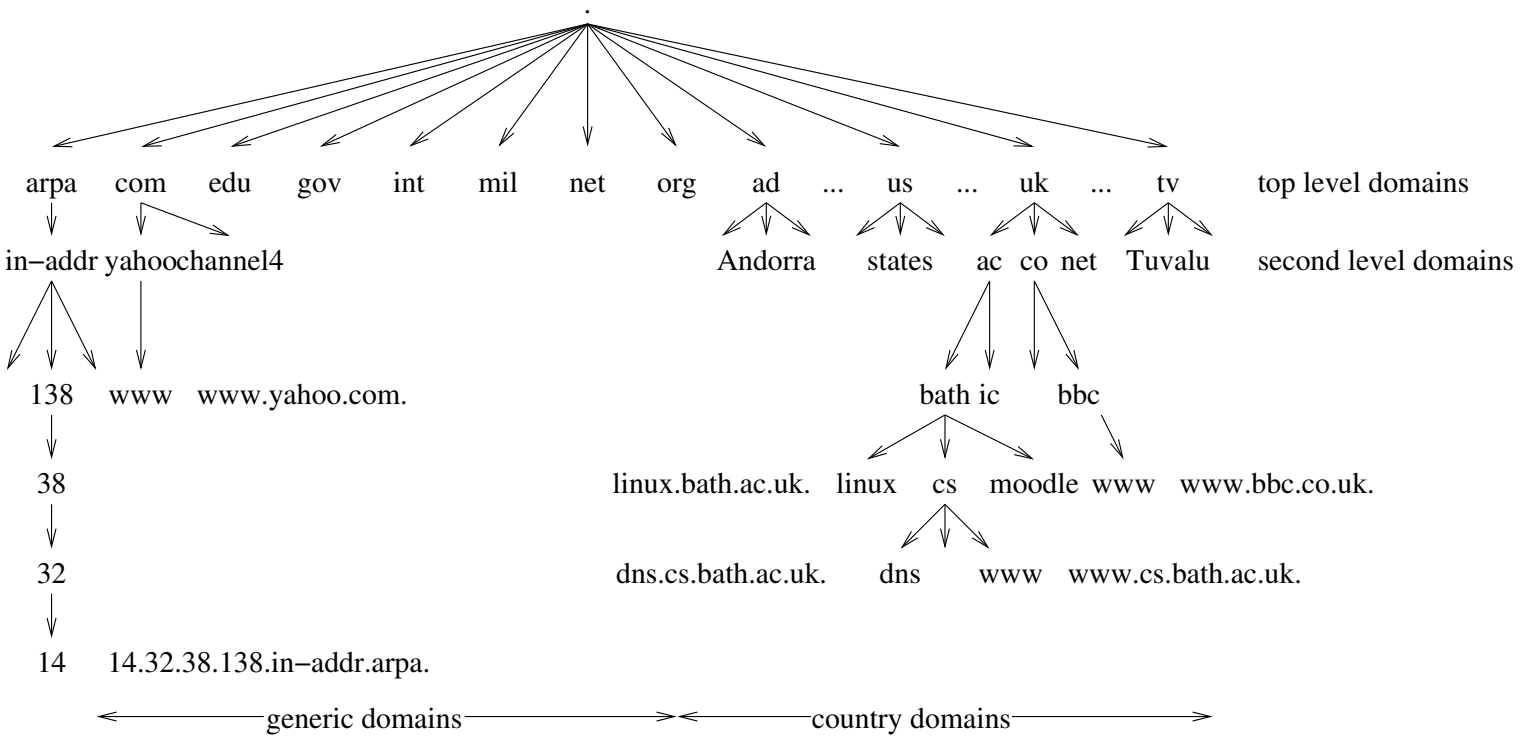
Sometimes we want to do the reverse lookup: given an IP address find a name

DNS

Sometimes we want to do the reverse lookup: given an IP address find a name

There is a part of the tree dedicated to this with TLD `arpa`

DNS



The DNS hierarchy

DNS

If we have the IP address `138.38.32.14` we can do a request for `14.32.38.138.in-addr.arpa`

DNS

If we have the IP address `138.38.32.14` we can do a request for `14.32.38.138.in-addr.arpa`

The same kind of recursive lookup as before will lead us to finding that Bath is responsible for `38.138.in-addr.arpa`

DNS

If we have the IP address `138.38.32.14` we can do a request for `14.32.38.138.in-addr.arpa`

The same kind of recursive lookup as before will lead us to finding that Bath is responsible for `38.138.in-addr.arpa`

And a couple more steps takes us to a *pointer* (PTR) request for `14.32.38.138.in-addr.arpa` from the Bath server

DNS

If we have the IP address `138.38.32.14` we can do a request for `14.32.38.138.in-addr.arpa`

The same kind of recursive lookup as before will lead us to finding that Bath is responsible for `38.138.in-addr.arpa`

And a couple more steps takes us to a *pointer* (PTR) request for `14.32.38.138.in-addr.arpa` from the Bath server

We get `clan.bath.ac.uk`

DNS

DNS does

DNS

DNS does

- A address: name to IP address

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)
- AAAA address: name to IPv6 address

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)
- AAAA address: name to IPv6 address
- SOA start of authority: name to responsible name server

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)
- AAAA address: name to IPv6 address
- SOA start of authority: name to responsible name server
- MX mail server: name to a mail server for that domain. E.g., `bath.ac.uk` used to have mail server `138.38.32.14`

DNS

DNS does

- A address: name to IP address
- PTR pointer: IP address to name (IPv6 uses the `ip6.arpa` branch)
- AAAA address: name to IPv6 address
- SOA start of authority: name to responsible name server
- MX mail server: name to a mail server for that domain. E.g., `bath.ac.uk` used to have mail server `138.38.32.14`

And many more: about 50 in total, though few are used frequently

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

When a new organisation plugs into the Internet, it brings along its own DNS servers, thus increasing the scale of DNS

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

When a new organisation plugs into the Internet, it brings along its own DNS servers, thus increasing the scale of DNS

Lookup usually takes a few milliseconds: caching of all levels in the local DNS server is a big help here

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

When a new organisation plugs into the Internet, it brings along its own DNS servers, thus increasing the scale of DNS

Lookup usually takes a few milliseconds: caching of all levels in the local DNS server is a big help here

DNS is actually a many-many relationship of names and addresses

DNS

DNS is remarkably successful and scales easily to the billions of hosts on the Internet

When a new organisation plugs into the Internet, it brings along its own DNS servers, thus increasing the scale of DNS

Lookup usually takes a few milliseconds: caching of all levels in the local DNS server is a big help here

DNS is actually a many-many relationship of names and addresses

- One address can have several names

DNS

Both `news.bbc.co.uk` and `newswww.bbc.net.uk` resolve to `212.58.226.33`

```
news.bbc.co.uk.          1619    IN  CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.      77      IN  A       212.58.226.33
```

DNS

Both `news.bbc.co.uk` and `newswww.bbc.net.uk` resolve to `212.58.226.33`

```
news.bbc.co.uk.          1619    IN  CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.      77      IN  A       212.58.226.33
```

`news.bbc.co.uk` is an *alias* for the real *canonical name* `newswww.bbc.net.uk`

DNS

Both `news.bbc.co.uk` and `newswww.bbc.net.uk` resolve to `212.58.226.33`

```
news.bbc.co.uk.          1619    IN  CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.      77      IN  A       212.58.226.33
```

`news.bbc.co.uk` is an *alias* for the real *canonical name* `newswww.bbc.net.uk`

This allows us tricks, like moving the Web server to different hosts, even different ISPs or different countries, but keeping its public name the same

DNS

- One name can have several IP addresses associated. This allows *load sharing*, e.g., a Web server can actually be several machines spread about anywhere in the world

DNS

```
www.microsoft.com.        68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285     IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.      285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.193.254
```

DNS

```
www.microsoft.com.       68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285    IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.     285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.193.254
```

`www.microsoft.com` is an alias: its canonical name is `toggle.www.ms.akadns.net`;

DNS

```
www.microsoft.com.       68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285    IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.     285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.   285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.   285     IN  A       207.46.193.254
```

But that is an alias for `g.www.ms.akadns.net`;

DNS

```
www.microsoft.com.        68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285     IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.      285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.193.254
```

And that is an alias for `lb1.www.ms.akadns.net`;

DNS

```
www.microsoft.com.        68      IN  CNAME   toggle.www.ms.akadns.net.
toggle.www.ms.akadns.net. 285     IN  CNAME   g.www.ms.akadns.net.
g.www.ms.akadns.net.      285     IN  CNAME   lb1.www.ms.akadns.net.
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.190
lb1.www.ms.akadns.net.    285     IN  A       207.46.19.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.192.254
lb1.www.ms.akadns.net.    285     IN  A       207.46.193.254
```

And that refers to four different addresses

DNS

And a DNS server can give out different lookups dependent on who is asking

DNS

And a DNS server can give out different lookups dependent on who is asking

A recent lookup for `www.microsoft.com` returned `104.78.177.250` from one location, and `2.18.85.172` from another

DNS

And a DNS server can give out different lookups dependent on who is asking

A recent lookup for `www.microsoft.com` returned `104.78.177.250` from one location, and `2.18.85.172` from another

This allows for load sharing; and also can be used for *geofencing*: giving different services to clients in different places (e.g., videos that are licensed only for certain regions)

DNS

Exercise Redo these lookups to see how they currently turn out

Exercise Compare having DNS give out multiple IP addresses for a given name against giving out different addresses for different clients against using anycast addresses

DNS

In programs, if we need to look up an address, we can use a simple function call to `getaddrinfo()` (previously `gethostbyname()`) that hides all this complexity from us

DNS

In programs, if we need to look up an address, we can use a simple function call to `getaddrinfo()` (previously `gethostbyname()`) that hides all this complexity from us

Exercise Old code using `gethostbyname()` is one of the sticking points in the transition to IPv6. Read about this

DNS

In programs, if we need to look up an address, we can use a simple function call to `getaddrinfo()` (previously `gethostbyname()`) that hides all this complexity from us

Exercise Old code using `gethostbyname()` is one of the sticking points in the transition to IPv6. Read about this

Under Linux the `dig` tool can be used to do direct lookups

DNS

```
% dig news.bbc.co.uk
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 15281
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;news.bbc.co.uk.                        IN      A

;; ANSWER SECTION:
news.bbc.co.uk.         193     IN      CNAME   newswww.bbc.net.uk.
newswww.bbc.net.uk.     76      IN      A       212.58.226.73

;; AUTHORITY SECTION:
bbc.net.uk.             85129   IN      NS      ns0.thdo.bbc.co.uk.
bbc.net.uk.             85129   IN      NS      ns0.rbsov.bbc.co.uk.

;; ADDITIONAL SECTION:
ns0.thdo.bbc.co.uk.     9490    IN      A       212.58.224.20
ns0.rbsov.bbc.co.uk.    9490    IN      A       212.58.227.47
```

DNS

Quite often a DNS server will reply with more information than we requested, e.g., the lookup of the CNAME `newswww.bbc.net.uk` in the above

DNS

Quite often a DNS server will reply with more information than we requested, e.g., the lookup of the CNAME `newswww.bbc.net.uk` in the above

This means we don't have to do an additional query to get the actual IP address we were looking for

DNS

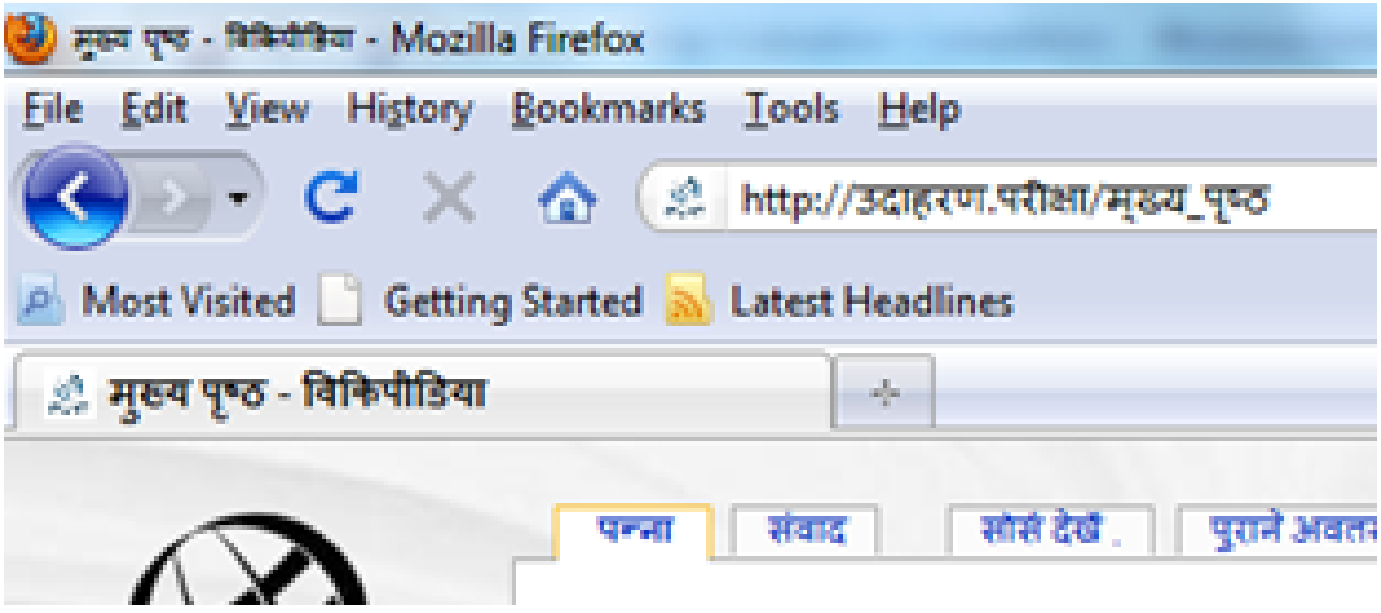
```
% dig -x 212.58.226.73
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 32413
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 3

;; QUESTION SECTION:
;73.226.58.212.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
73.226.58.212.in-addr.arpa. 50081 IN    PTR     newslb305.telhc.bbc.co.uk.

;; AUTHORITY SECTION:
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.thdo.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.thny.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns1.bbc.co.uk.
226.58.212.in-addr.arpa. 50081  IN      NS      ns.bbc.co.uk.

;; ADDITIONAL SECTION:
ns1.bbc.co.uk.          311     IN      A       132.185.132.21
ns1.thny.bbc.co.uk.     32106   IN      A       212.58.227.48
ns1.thdo.bbc.co.uk.     33051   IN      A       212.58.224.21
```

DNS

```
% dig +trace www.google.com
.                       180695  IN      NS      E.ROOT-SERVERS.NET.
.                       180695  IN      NS      J.ROOT-SERVERS.NET.
.                       180695  IN      NS      I.ROOT-SERVERS.NET.
...
com.                    172800  IN      NS      A.GTLD-SERVERS.NET.
com.                    172800  IN      NS      B.GTLD-SERVERS.NET.
com.                    172800  IN      NS      C.GTLD-SERVERS.NET.
...
google.com.             172800  IN      NS      ns1.google.com.
google.com.             172800  IN      NS      ns2.google.com.
google.com.             172800  IN      NS      ns3.google.com.
...
www.google.com.         604800  IN      CNAME   www.l.google.com.
l.google.com.           86400   IN      NS      b.l.google.com.
l.google.com.           86400   IN      NS      d.l.google.com.
...
```

DNS

DNS labels can be in arbitrary character sets, not just Latin



A non-Latin DNS name

From `blog.icann.org`

DNS

DNS is a very successful protocol: fast and it's relatively easy for system administrators to manage the DNS servers

DNS

DNS is a very successful protocol: fast and it's relatively easy for system administrators to manage the DNS servers

DNS names are a big source of money, so often a source of contention over who should be in control of what

DNS

DNS is a very successful protocol: fast and it's relatively easy for system administrators to manage the DNS servers

DNS names are a big source of money, so often a source of contention over who should be in control of what

Exercise Read about the controversies behind the introduction of new top-level DNS labels like `me` and `search`

DNS

Exercise Try looking up

- AAAA for `bath.ac.uk`
- AAAA for `facebook.com`
- SOA for `bath.ac.uk`
- A for `moodle.bath.ac.uk`
- And so on

DNS

Exercise And

- MX for `bath.ac.uk`

What is the Uni's mail server physical location?

What are the privacy/security/legal aspects?

Then try MX for `bath.edu`, an address the University uses for alumni (graduates and past staff)

DNS

Exercise Read about the public DNS servers like `1.1.1.1`, `8.8.8.8` and others; why would you want to use them over your local DNS server?

Exercise What are the privacy/security/legal aspects of using public servers?

Exercise Find out how to buy a DNS name

DNS

DNS requests are normally sent using UDP, and fit in a single UDP datagram

DNS

DNS requests are normally sent using UDP, and fit in a single UDP datagram

When the reply fits within 512 bytes, UDP is used for the reply by the server

DNS

DNS requests are normally sent using UDP, and fit in a single UDP datagram

When the reply fits within 512 bytes, UDP is used for the reply by the server

To keep datagrams small, if the reply is more than 512 bytes, the server sends a reply with a “truncated” flag set, and the client resends the request but using TCP

DNS

DNS requests are normally sent using UDP, and fit in a single UDP datagram

When the reply fits within 512 bytes, UDP is used for the reply by the server

To keep datagrams small, if the reply is more than 512 bytes, the server sends a reply with a “truncated” flag set, and the client resends the request but using TCP

Now, UDP is fast but unreliable, and 512 byte datagrams won’t be fragmented (recall: must be able to send 576 bytes IP), so there is little complication if a DNS datagram is lost: the source will just send a new request

DNS

UDP is preferred as it is fast with little overhead; while a TCP connection has a considerable setup cost (see later)

DNS

UDP is preferred as it is fast with little overhead; while a TCP connection has a considerable setup cost (see later)

So small and fast wherever possible; but slower and reliable if needed

DNS

DNS is good, but has problems

DNS

DNS is good, but has problems

- There is no security or authentication: if I get a reply saying that `138.38.32.14` is the address for `www.bath.ac.uk` can I trust this?

DNS

DNS is good, but has problems

- There is no security or authentication: if I get a reply saying that `138.38.32.14` is the address for `www.bath.ac.uk` can I trust this?
- Some server on the lookup path might have been subverted and made to hand out the wrong address; or a router may have carefully rewritten a DNS reply

DNS

DNS is good, but has problems

- There is no security or authentication: if I get a reply saying that `138.38.32.14` is the address for `www.bath.ac.uk` can I trust this?
- Some server on the lookup path might have been subverted and made to hand out the wrong address; or a router may have carefully rewritten a DNS reply
- Then I could receive a spoof IP address leading to someone else's web page: a problem if, say, I was looking for the page of a bank or shop

DNS

DNS is good, but has problems

- There is no security or authentication: if I get a reply saying that `138.38.32.14` is the address for `www.bath.ac.uk` can I trust this?
- Some server on the lookup path might have been subverted and made to hand out the wrong address; or a router may have carefully rewritten a DNS reply
- Then I could receive a spoof IP address leading to someone else's web page: a problem if, say, I was looking for the page of a bank or shop
- The spoof page could be made to look identical to the real bank web page, inviting me to enter my id and password

DNS

Exercise The security company RSA was attacked by DNS spoofing. Read about this

Exercise April 2018: routes to Amazon DNS servers were faked so that DNS requests were sent to fake servers. Read about this

Exercise Your home connection probably uses your ISP as a DNS server. ISPs have been known to rewrite DNS replies. They also might block access to other public DNS servers. Read about this

DNS

A partial solution exists in *Secure DNS* (DNSSec), which uses cryptography to authenticate DNS lookups

DNS

A partial solution exists in *Secure DNS* (DNSSec), which uses cryptography to authenticate DNS lookups

Not much in use as it was introduced when people still thought that cryptography was too expensive to use

DNS

And neither DNS nor DNSSec provide any privacy: anyone listening can see what hosts you are trying to contact by monitoring your DNS requests

DNS

And neither DNS nor DNSSec provide any privacy: anyone listening can see what hosts you are trying to contact by monitoring your DNS requests

So there has been a move by some people to use DNS over TLS (see TLS later and RFC7858) that encrypts DNS requests and replies

DNS

And neither DNS nor DNSSec provide any privacy: anyone listening can see what hosts you are trying to contact by monitoring your DNS requests

So there has been a move by some people to use DNS over TLS (see TLS later and RFC7858) that encrypts DNS requests and replies

This has a fair overhead over plain DNS, but provides both authentication and privacy

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

This sounds silly as HTTPS is a heavyweight protocol designed to move large Web pages, not lightweight DNS requests

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

This sounds silly as HTTPS is a heavyweight protocol designed to move large Web pages, not lightweight DNS requests

But:

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

This sounds silly as HTTPS is a heavyweight protocol designed to move large Web pages, not lightweight DNS requests

But:

- The overhead can be managed fairly well (see HTTP *Keep-Alive* and TLS reconnections)

DNS

More surprisingly (at first), there is also DNS over HTTPS (DoH)

This sounds silly as HTTPS is a heavyweight protocol designed to move large Web pages, not lightweight DNS requests

But:

- The overhead can be managed fairly well (see HTTP *Keep-Alive* and TLS reconnections)
- While an uncooperative ISP could block the DNS over TLS assigned port (853) the HTTPS port (443) cannot be blocked without a lot of collateral damage to normal browsing

DNS

Exercise CIDR makes PTR lookups harder as netmasks, and therefore the delegations, are no longer on a byte boundary. Read RFC2317 on how CNAMES are used to solve this problem

Exercise Read about how (or if) DoH is supported in your browser

Exercise Read about *DNS over Twitter* and *DNS over SMS*

Transport Layer

We now move up a layer: the Transport Layer

Transport Layer

We now move up a layer: the Transport Layer

The Internet Protocol has three main protocols that run on top of IP: two are for data, one for control

Transport Layer

We now move up a layer: the Transport Layer

The Internet Protocol has three main protocols that run on top of IP: two are for data, one for control

The data protocols are complementary

- one is fast, unreliable, connectionless: UDP
- the other is more sophisticated, reliable and connection-oriented: TCP

Transport Layer

We now move up a layer: the Transport Layer

The Internet Protocol has three main protocols that run on top of IP: two are for data, one for control

The data protocols are complementary

- one is fast, unreliable, connectionless: UDP
- the other is more sophisticated, reliable and connection-oriented: TCP

The control protocol, ICMP, we have already seen and is usually considered as part of the network layer

Transport Layer

We now move up a layer: the Transport Layer

The Internet Protocol has three main protocols that run on top of IP: two are for data, one for control

The data protocols are complementary

- one is fast, unreliable, connectionless: UDP
- the other is more sophisticated, reliable and connection-oriented: TCP

The control protocol, ICMP, we have already seen and is usually considered as part of the network layer

Other data protocols exist in this layer, but TCP and UDP are currently the important ones

Transport Layer

Both UDP and TCP use the concept of *ports*

Transport Layer

Both UDP and TCP use the concept of *ports*

On a single server machine there can be many programs running, web, email, and so on: how does a client indicate which service it wants from the server?

Transport Layer

Both UDP and TCP use the concept of *ports*

On a single server machine there can be many programs running, web, email, and so on: how does a client indicate which service it wants from the server?

And when a reply packet arrives back at a client, how does the OS know which of the many processes running on the client that packet should be delivered to?

Transport Layer

Both UDP and TCP use the concept of *ports*

On a single server machine there can be many programs running, web, email, and so on: how does a client indicate which service it wants from the server?

And when a reply packet arrives back at a client, how does the OS know which of the many processes running on the client that packet should be delivered to?

This is done by ports

Transport Layer

Both UDP and TCP use the concept of *ports*

On a single server machine there can be many programs running, web, email, and so on: how does a client indicate which service it wants from the server?

And when a reply packet arrives back at a client, how does the OS know which of the many processes running on the client that packet should be delivered to?

This is done by ports

A port is just a 16 bit integer: 1-65535

Transport Layer

Every TCP and UDP connection has a *source port* and a *destination port*

Transport Layer

Every TCP and UDP connection has a *source port* and a *destination port*

When a service starts
— i.e., a program that will deal with the service starts —
it *listens* on a port
— i.e., it informs the operating system that it wishes to receive data from packets directed to that port number

Transport Layer

Every TCP and UDP connection has a *source port* and a *destination port*

When a service starts
— i.e., a program that will deal with the service starts —
it *listens* on a port
— i.e., it informs the operating system that it wishes to receive data from packets directed to that port number

E.g., an email server may indicate it wants packets addressed to TCP port 25; a browser would listen on port 80 (and 443)

Transport Layer

The OS checks that port is not already being used by another program, and subsequently ensures packets with that destination port are sent to that service program

Transport Layer

The OS checks that port is not already being used by another program, and subsequently ensures packets with that destination port are sent to that service program

So when a TCP packet with destination port 25 arrives its data will be given to the email program

Transport Layer

The OS checks that port is not already being used by another program, and subsequently ensures packets with that destination port are sent to that service program

So when a TCP packet with destination port 25 arrives its data will be given to the email program

An analogy: a host as a block of flats. To address a letter to a specific person you need both a building address (IP address) and a flat number (port)

Transport Layer

TCP and UDP ports are entirely separate: one service can be listening for for a TCP connection on a port and another service for UDP on the same port number

Transport Layer

TCP and UDP ports are entirely separate: one service can be listening for for a TCP connection on a port and another service for UDP on the same port number

The OS can distinguish the two as they are port within different protocols

Transport Layer

TCP and UDP ports are entirely separate: one service can be listening for for a TCP connection on a port and another service for UDP on the same port number

The OS can distinguish the two as they are port within different protocols

TCP and UDP are completely separate and do not interact at all (at the transport level)

Transport Layer

Certain *well-known* ports are reserved for certain services

- web server on port 80
- email server on port 25
- FTP on port 21
- hundreds of others. See `/etc/services` and RFC6335

Transport Layer

Certain *well-known* ports are reserved for certain services

- web server on port 80
- email server on port 25
- FTP on port 21
- hundreds of others. See `/etc/services` and RFC6335

A range of ports are reserved for privileged (root/administrator) programs; most are available to any program that wants to use them

Transport Layer

Certain *well-known* ports are reserved for certain services

- web server on port 80
- email server on port 25
- FTP on port 21
- hundreds of others. See `/etc/services` and RFC6335

A range of ports are reserved for privileged (root/administrator) programs; most are available to any program that wants to use them

Typically, port numbers under 1024 are reserved for privileged programs

Transport Layer

These associations of port numbers to services are purely convention and for convenience only: no port is special and you can run any service on any port

Transport Layer

These associations of port numbers to services are purely convention and for convenience only: no port is special and you can run any service on any port

It just means you don't have the extra problem of determining the port for, say, the web server: it is almost always 80

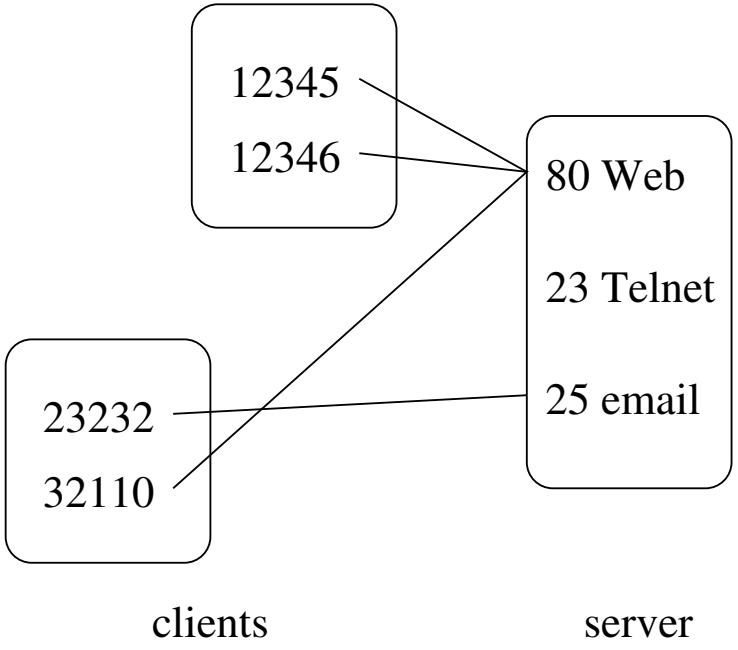
Transport Layer

These associations of port numbers to services are purely convention and for convenience only: no port is special and you can run any service on any port

It just means you don't have the extra problem of determining the port for, say, the web server: it is almost always 80

You can run a web server on port 25 if you wish: you will just confuse anyone who tries to send you email

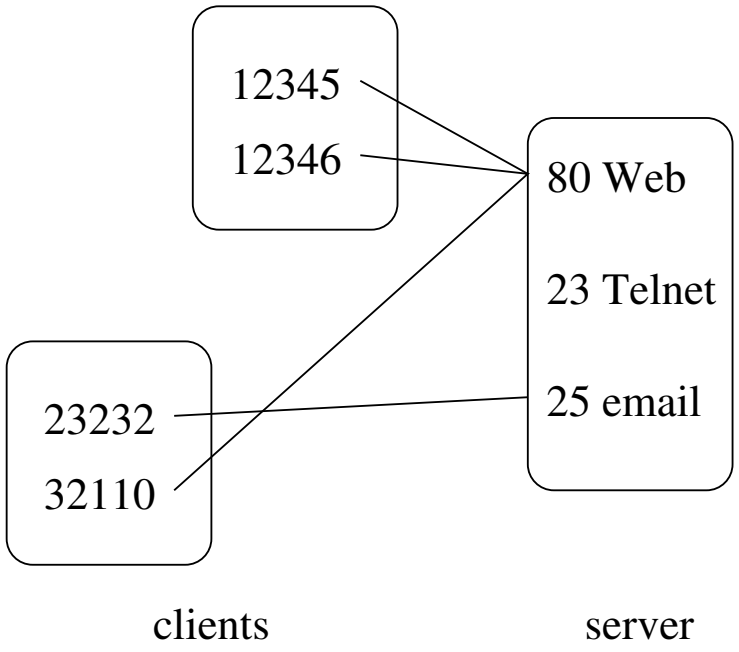
Transport Layer



Transport layer ports

Ports also enable multiple simultaneous connections between two machines, e.g., fetching several web pages

Transport Layer



Transport layer ports

Ports also enable multiple simultaneous connections between two machines, e.g., fetching several web pages

The source port (destination port on the returning packet) allows the client OS to identify which packet belongs to which client program

Transport Layer

Source ports are usually chosen afresh “at random” (usually: just increment by 1 for each time) for each new connection and are called *ephemeral* ports as they only live for the duration of the connection

Transport Layer

Source ports are usually chosen afresh “at random” (usually: just increment by 1 for each time) for each new connection and are called *ephemeral* ports as they only live for the duration of the connection

There is no technical difference between ephemeral and well-known ports, just the way they are used

Transport Layer

Source ports are usually chosen afresh “at random” (usually: just increment by 1 for each time) for each new connection and are called *ephemeral* ports as they only live for the duration of the connection

There is no technical difference between ephemeral and well-known ports, just the way they are used

The quad

source address
source port
destination address
destination port

specifies a connection uniquely: the hosts involved and the processes on those hosts

Transport Layer

The pair (source address, source port) is often called a *socket*

Transport Layer

The pair (source address, source port) is often called a *socket*

A full quad is then called a *socket pair*

Transport Layer

The pair (source address, source port) is often called a *socket*

A full quad is then called a *socket pair*

Both TCP and UDP have port fields early in their headers: this is so that the port numbers are included in the “IP header plus 8 bytes of data” that an ICMP error contains

Transport Layer

The pair (source address, source port) is often called a *socket*

A full quad is then called a *socket pair*

Both TCP and UDP have port fields early in their headers: this is so that the port numbers are included in the “IP header plus 8 bytes of data” that an ICMP error contains

Thus the OS can identify which process an ICMP belongs to

Transport Layer

The pair (source address, source port) is often called a *socket*

A full quad is then called a *socket pair*

Both TCP and UDP have port fields early in their headers: this is so that the port numbers are included in the “IP header plus 8 bytes of data” that an ICMP error contains

Thus the OS can identify which process an ICMP belongs to

And a non-initial IP fragment won’t have such identifying information, so this is why ICMPs are not generated for errors involving such fragments

Transport Layer

NAT, again

And ports are how NAT firewalls do their magic of matching returning reply packets to request packets

Transport Layer

NAT, again

And ports are how NAT firewalls do their magic of matching returning reply packets to request packets

It keeps a list of private (internal) socket pairs against public (external) socket pairs

Transport Layer

NAT, again

And ports are how NAT firewalls do their magic of matching returning reply packets to request packets

It keeps a list of private (internal) socket pairs against public (external) socket pairs

And this is enough to match up replies with requests

Transport Layer

NAT, again

And ports are how NAT firewalls do their magic of matching returning reply packets to request packets

It keeps a list of private (internal) socket pairs against public (external) socket pairs

And this is enough to match up replies with requests

Exercise Read about *port address translation*

Transport Layer

NAT, again

And ports are how NAT firewalls do their magic of matching returning reply packets to request packets

It keeps a list of private (internal) socket pairs against public (external) socket pairs

And this is enough to match up replies with requests

Exercise Read about *port address translation*

Exercise Sometime we wish to allow an external host to initiate a connection with a private host. Read about *port forwarding*