Routing

This is simple to implement and run, but such protocols have a *slow convergence* problem

Routing

This is simple to implement and run, but such protocols have a *slow convergence* problem

This means that if the network changes (e.g., a link is broken, or a new link is made) it takes many interchanges of information for the routers to adjust to the new routes

Routing

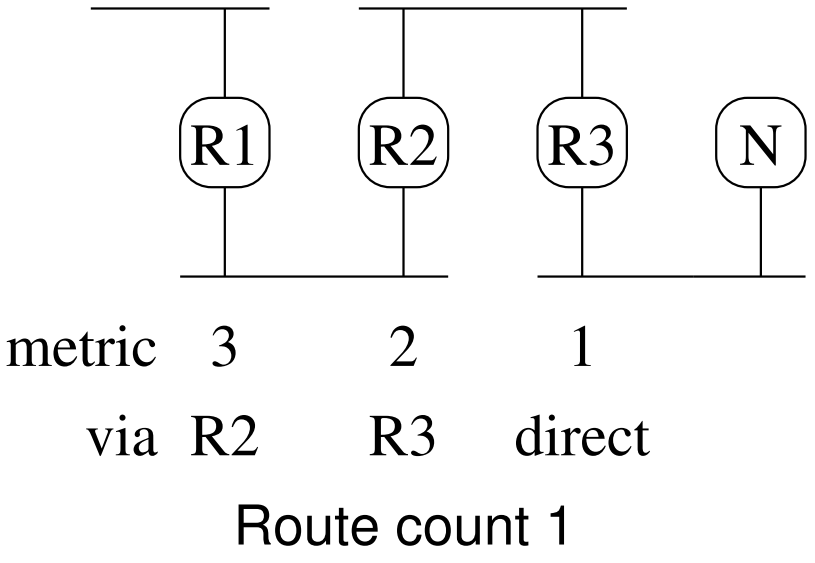
This is simple to implement and run, but such protocols have a *slow convergence* problem

This means that if the network changes (e.g., a link is broken, or a new link is made) it takes many interchanges of information for the routers to adjust to the new routes

And this can manifest in bad ways

Routing

Slow Convergence

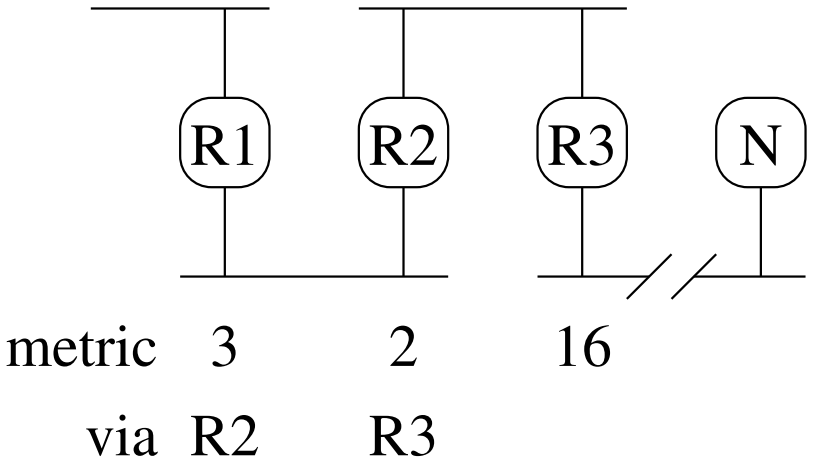


Route count 1

R3 knows a route to network N of hop count 1;

Routing

Slow Convergence

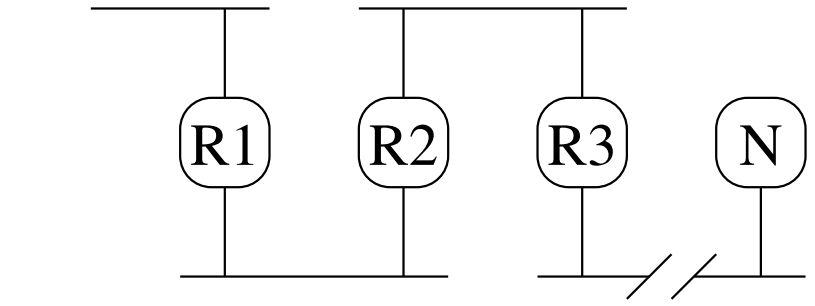


Break in route

After a break in the network R3 finds that route no longer works;

Routing

Slow Convergence

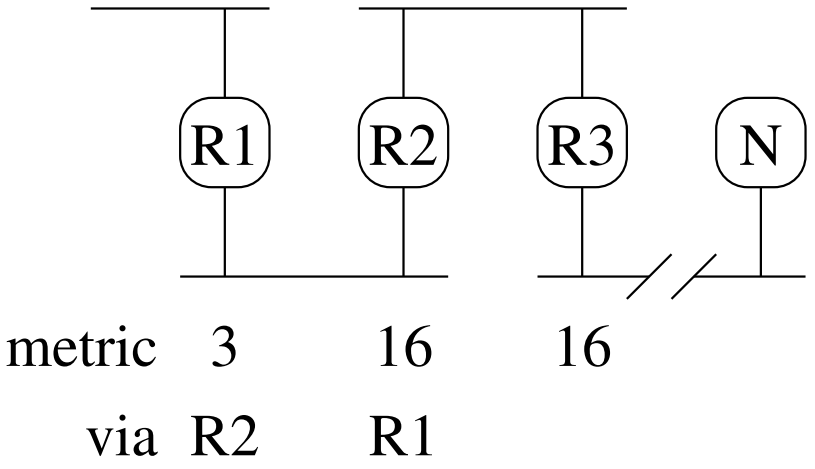


No route

So it sends a message to its neighbours (R2) saying “no route to N”. It uses a count of 16, which is interpreted as infinity;

Routing

Slow Convergence

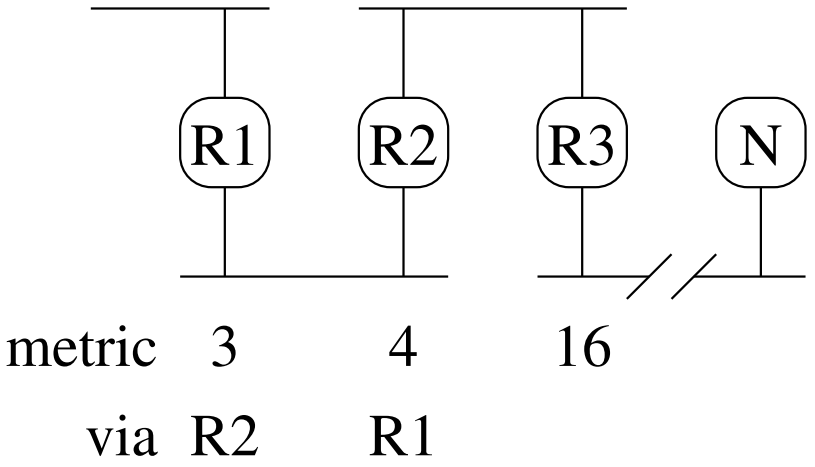


Update routing table

R2 updates its routing table;

Routing

Slow Convergence

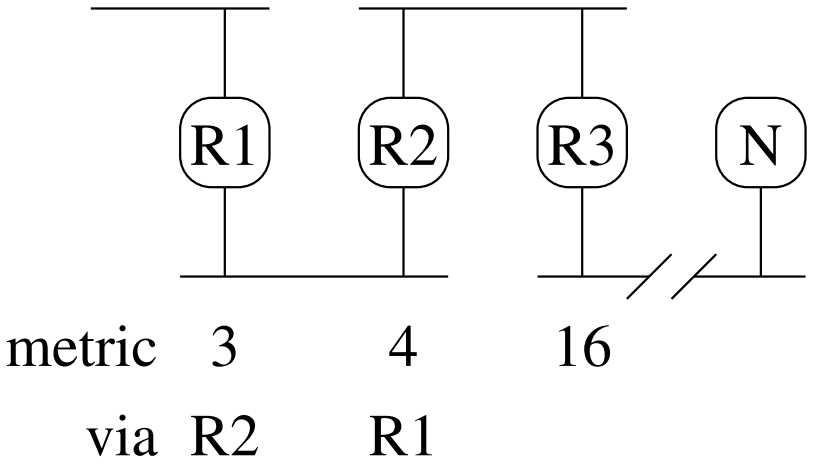


New route for R2?

But R2 also gets a periodic update message from R1 saying “route of 3 hops”;

Routing

Slow Convergence

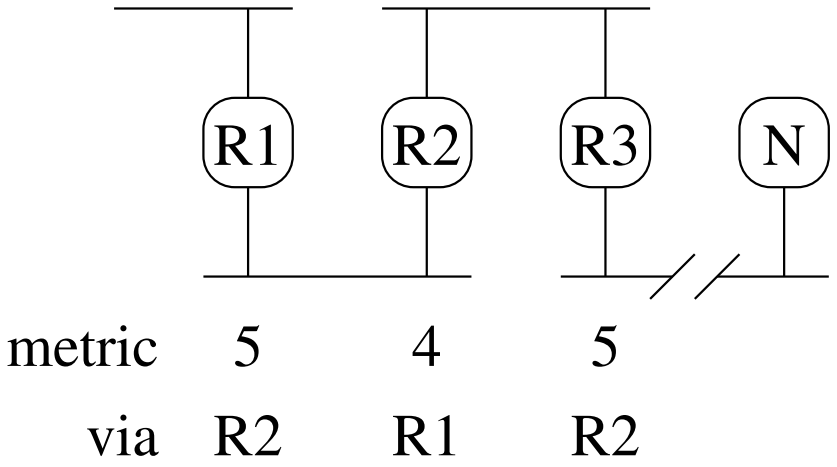


R2 update again

So R2 now thinks the best route is via R1, 4 hops;

Routing

Slow Convergence

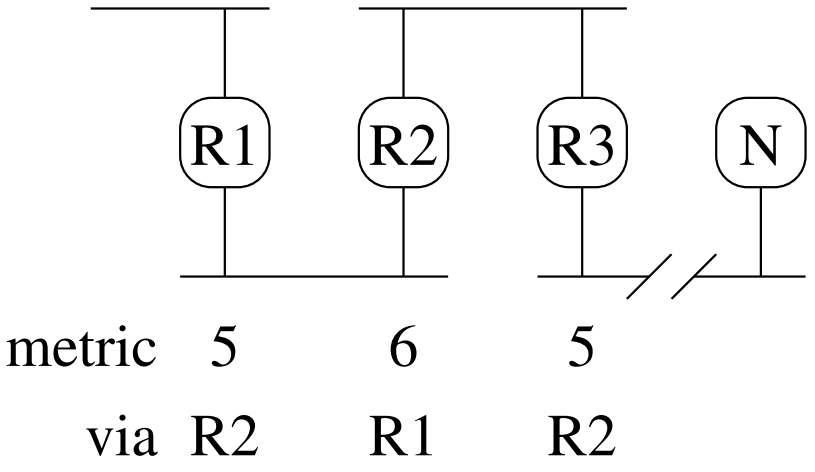


Another broadcast

And when R2 sends its periodic update message “4” to R1 and R3, R1 now thinks there is a route via R2 of 5 hops; and R3 thinks there is a route of 5 hops via R2;

Routing

Slow Convergence

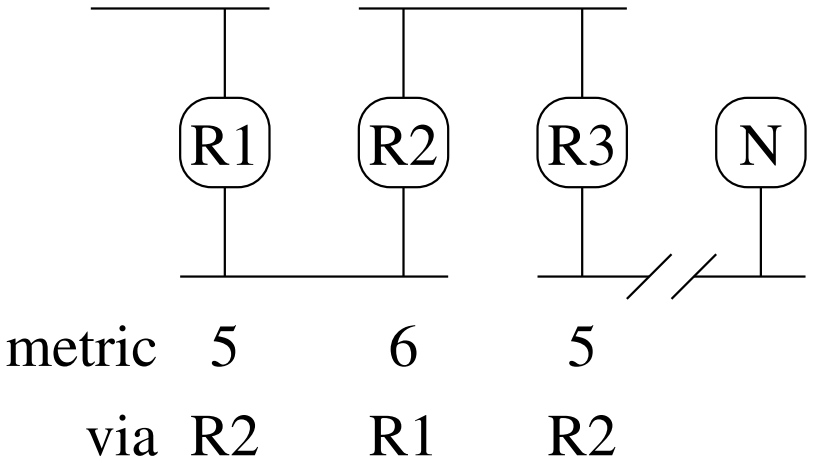


Counts grow slowly

After the next update, R2 thinks there is a route via R3 of 6 hops;

Routing

Slow Convergence

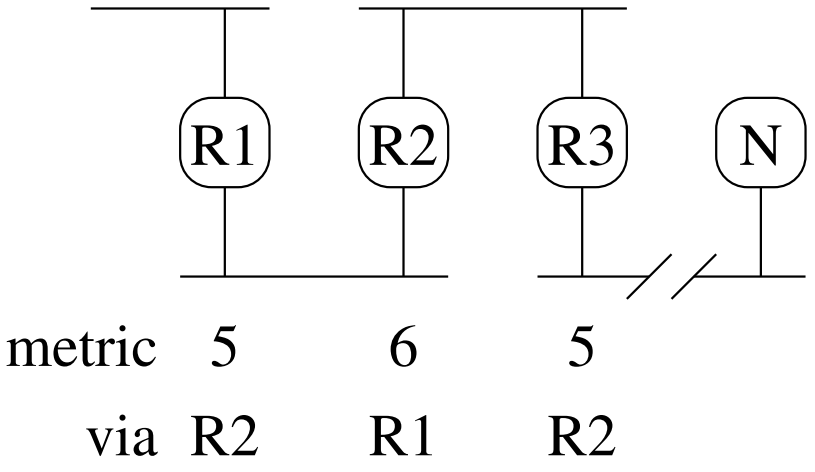


Counts grow slowly

And so on;

Routing

Slow Convergence

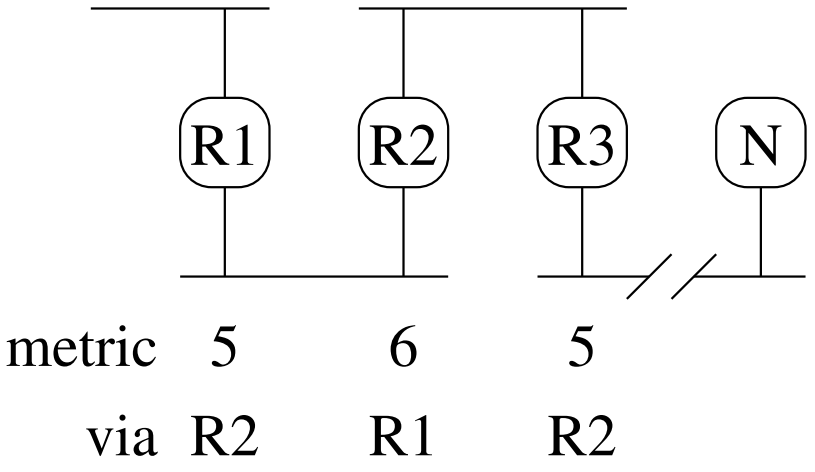


Counts grow slowly

Eventually the hop count reaches 16, i.e., no route, and so this route is dropped;

Routing

Slow Convergence

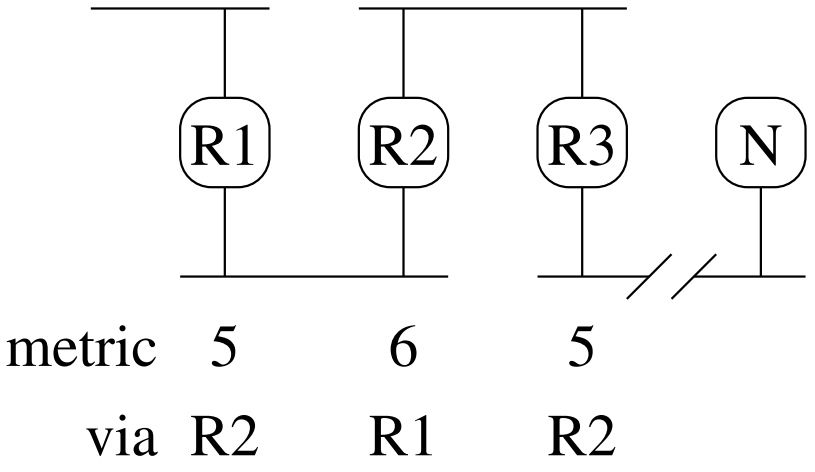


Counts grow slowly

This is called the *count to infinity* problem;

Routing

Slow Convergence



Counts grow slowly

If there was a valid route, it might take a long time to converge to that route

Routing

Meanwhile real data packets are bouncing forwards and back between the routers

Routing

Meanwhile real data packets are bouncing forwards and back between the routers

The local information that distance vector provides is not enough

Routing

Meanwhile real data packets are bouncing forwards and back between the routers

The local information that distance vector provides is not enough

RIP uses distance vector and this is a real problem for it

Routing

Meanwhile real data packets are bouncing forwards and back between the routers

The local information that distance vector provides is not enough

RIP uses distance vector and this is a real problem for it

So RIP should only used on small networks that are fairly stable

Routing

Meanwhile real data packets are bouncing forwards and back between the routers

The local information that distance vector provides is not enough

RIP uses distance vector and this is a real problem for it

So RIP should only used on small networks that are fairly stable

Link state protocols, e.g., OSPF, converge faster, but need more complicated graph traversal algorithms to determine best routes

BGP

BGP is a *path vector* variation of distance vector: this includes the path (multiple hops) to the destination, which can be used to spot the loops that lead to count-to-infinity

BGP

BGP is a *path vector* variation of distance vector: this includes the path (multiple hops) to the destination, which can be used to spot the loops that lead to count-to-infinity

ASs do not change very much so slow convergence is not such a big problem anyway

BGP

BGP is a *path vector* variation of distance vector: this includes the path (multiple hops) to the destination, which can be used to spot the loops that lead to count-to-infinity

ASs do not change very much so slow convergence is not such a big problem anyway

Exercise Read about path vector systems

BGP

BGP does have other problems, particularly authentication

BGP

BGP does have other problems, particularly authentication

Through accident or malice it is easy to trick BGP

BGP

BGP does have other problems, particularly authentication

Through accident or malice it is easy to trick BGP

For example, it would be relatively easy to get BGP to transit data through an evil third party

BGP

BGP does have other problems, particularly authentication

Through accident or malice it is easy to trick BGP

For example, it would be relatively easy to get BGP to transit data through an evil third party

Also, see the problem with the route to Youtube, earlier

BGP

Exercise Read about the 2018 hack on the cryptocurrency website `MyEtherWallet.com` that started by subverting BGP to send DNS traffic to a rogue server

Exercise Read about the BGP problem of April 2021, where Vodafone Idea (AS55410) published bad routes

Exercise Read about the proposed *Resource Public Key Infrastructure* (RPKI), RFC6810

Exercise Read about the *Mutually Agreed Norms for Routing Security* (MANRS) initiative for ISPs and routing exchange operators

BGP

Exercise Read about RIP

Exercise Read about Dijkstra's algorithm for finding shortest paths in a graph; and OSPF which uses this algorithm

DNS

Next: back to IP addresses (again!)

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

The Internet would probably be a lot less easy to use if we had to refer to `212.58.233.253` rather than `www.bbc.co.uk`

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

The Internet would probably be a lot less easy to use if we had to refer to `212.58.233.253` rather than `www.bbc.co.uk`

The IP addresses are essential as they are hardware independent and have structure that aids routing

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

The Internet would probably be a lot less easy to use if we had to refer to `212.58.233.253` rather than `www.bbc.co.uk`

The IP addresses are essential as they are hardware independent and have structure that aids routing

But the names are what makes things like the Web usable

DNS

Next: back to IP addresses (again!)

IP addresses (v4 or v6) are bunches of bits: but we are used to addressing machines by names such as `linux.bath.ac.uk`

The Internet would probably be a lot less easy to use if we had to refer to `212.58.233.253` rather than `www.bbc.co.uk`

The IP addresses are essential as they are hardware independent and have structure that aids routing

But the names are what makes things like the Web usable

Note we now have *three* kinds of addresses: physical, network and human

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

In the early Internet every host had a file containing the names and addresses of all machines on the Internet

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

In the early Internet every host had a file containing the names and addresses of all machines on the Internet

A simple look into this table sufficed

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

In the early Internet every host had a file containing the names and addresses of all machines on the Internet

A simple look into this table sufficed

We can't do that now, though!

DNS

So we have a repeat of the hardware-software address gap that ARP is for: there must also be a mechanism for turning `linux.bath.ac.uk` into `138.38.3.40`

In the early Internet every host had a file containing the names and addresses of all machines on the Internet

A simple look into this table sufficed

We can't do that now, though!

Exercise The file lives on as `/etc/hosts` under Unix. Look at this file

DNS

An ARP-style discovery broadcast would have to go to the entire Internet, so this approach is also infeasible

DNS

An ARP-style discovery broadcast would have to go to the entire Internet, so this approach is also infeasible

So there is another protocol, the *domain name system* (DNS) to do this

DNS

An ARP-style discovery broadcast would have to go to the entire Internet, so this approach is also infeasible

So there is another protocol, the *domain name system* (DNS) to do this

There is no sensible, secure and economic way to have a single database that contains all the names and addresses on the Internet, so this information has to be distributed amongst a large number of machines called DNS servers

DNS

An ARP-style discovery broadcast would have to go to the entire Internet, so this approach is also infeasible

So there is another protocol, the *domain name system* (DNS) to do this

There is no sensible, secure and economic way to have a single database that contains all the names and addresses on the Internet, so this information has to be distributed amongst a large number of machines called DNS servers

That is, machines running a DNS server program

Aside

In the usual way, when talking about these things, we blur the distinction between the service and the host it lives on. The host might well be running other services, such as DCHP, as well

Aside

In the usual way, when talking about these things, we blur the distinction between the service and the host it lives on. The host might well be running other services, such as DCHP, as well

Exercise Find out the services your home Access Point runs

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

`bath.ac.uk` is a network in the domain `ac.uk`, and so on

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

`bath.ac.uk` is a network in the domain `ac.uk`, and so on

`ac.uk` is the name for the JANET network, and is in the domain `uk`

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

`bath.ac.uk` is a network in the domain `ac.uk`, and so on

`ac.uk` is the name for the JANET network, and is in the domain `uk`

`uk` is in the domain `.` (*dot* or *root*)

DNS

Names are hierarchical: `linux.bath.ac.uk` is a name of a machine in the *domain* `bath.ac.uk`

`bath.ac.uk` is a network in the domain `ac.uk`, and so on

`ac.uk` is the name for the JANET network, and is in the domain `uk`

`uk` is in the domain `.` (*dot* or *root*)

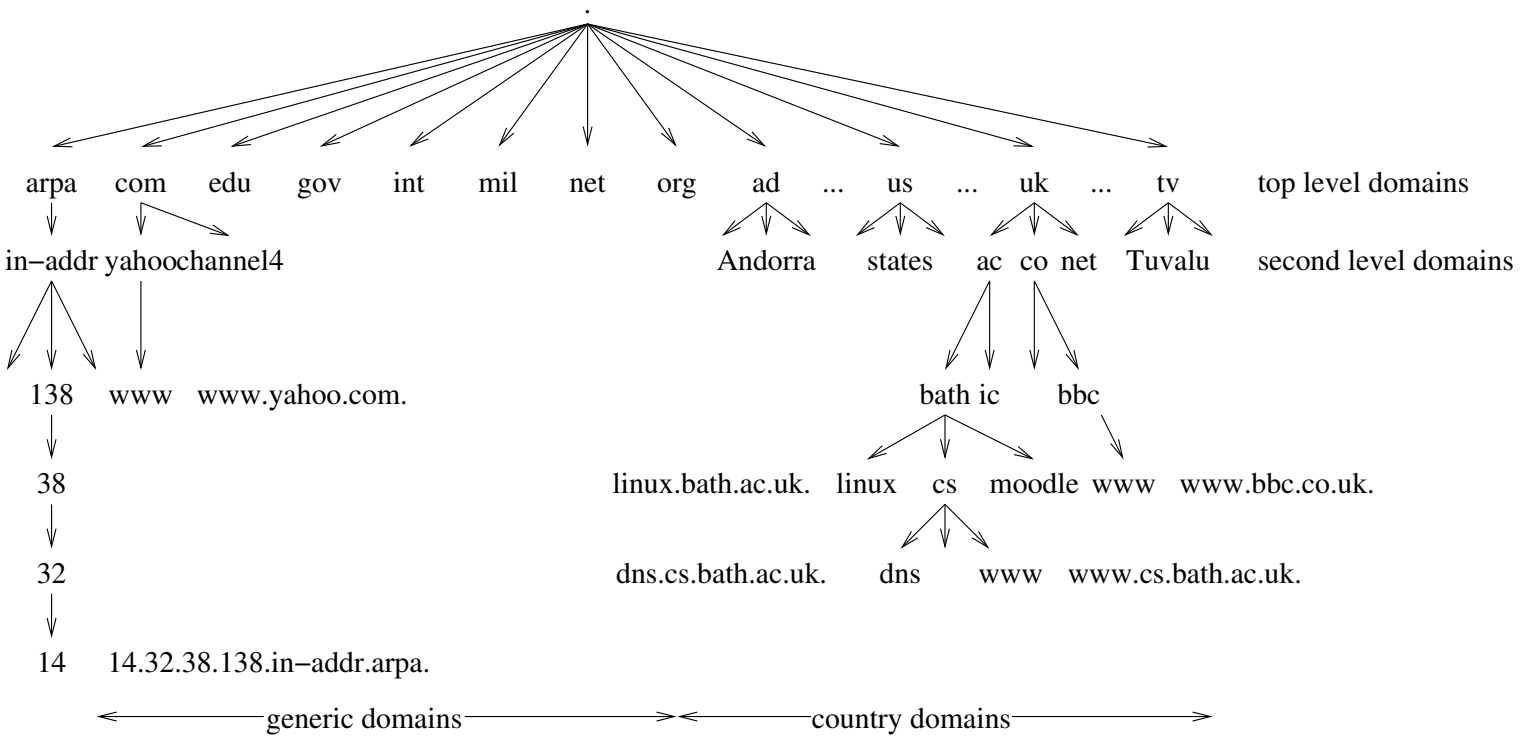
Each node in this tree is called a *label*

DNS

`www.llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch.com`
is a valid name

Labels may be up to 63 bytes long

DNS



The DNS hierarchy

DNS

It is this tree we use to distribute the database

DNS

It is this tree we use to distribute the database

The root of the tree is called . (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

DNS

It is this tree we use to distribute the database

The root of the tree is called `.` (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

They manage the *top level domains* (TLDs), including `com` and `uk`

DNS

It is this tree we use to distribute the database

The root of the tree is called `.` (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

They manage the *top level domains* (TLDs), including `com` and `uk`

“Manage” means they keep the lookup tables for that level and they say who can get labels in the next level

DNS

It is this tree we use to distribute the database

The root of the tree is called `.` (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

They manage the *top level domains* (TLDs), including `com` and `uk`

“Manage” means they keep the lookup tables for that level and they say who can get labels in the next level

They get to charge money for labels in the next level

DNS

It is this tree we use to distribute the database

The root of the tree is called `.` (dot) and this label is currently managed by the *Internet Corporation for Assigned Names and Numbers* (ICANN)

They manage the *top level domains* (TLDs), including `com` and `uk`

“Manage” means they keep the lookup tables for that level and they say who can get labels in the next level

They get to charge money for labels in the next level

But the important bit is that they *delegate* management of lower labels to other organisations

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

Note that ICANN gets to say that Nominet is in charge of `uk`

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

Note that ICANN gets to say that Nominet is in charge of `uk`

Nominet delegates labels under `ac.uk` to the *Joint Information Systems Committee* (Jisc) (previously *United Kingdom Education and Research Networking Association* (UKERNA))

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

Note that ICANN gets to say that Nominet is in charge of `uk`

Nominet delegates labels under `ac.uk` to the *Joint Information Systems Committee* (Jisc) (previously *United Kingdom Education and Research Networking Association* (UKERNA))

Labels under `bath.ac.uk` are managed by the University of Bath

DNS

Labels under `uk` are managed by the UK *Network Information Centre* (NIC), currently run by a UK company called Nominet

Note that ICANN gets to say that Nominet is in charge of `uk`

Nominet delegates labels under `ac.uk` to the *Joint Information Systems Committee* (Jisc) (previously *United Kingdom Education and Research Networking Association* (UKERNA))

Labels under `bath.ac.uk` are managed by the University of Bath

Labels under `cs.bath.ac.uk` are managed by the Department of Computer Science

DNS

At each level, the relevant organisation keeps a list of labels it has delegated and who is responsible for them

DNS

At each level, the relevant organisation keeps a list of labels it has delegated and who is responsible for them

At the lowest levels, for the leaves of the tree (the hosts), the relevant organisation keeps the name-to-IP-address mapping of the hosts

DNS

At each level, the relevant organisation keeps a list of labels it has delegated and who is responsible for them

At the lowest levels, for the leaves of the tree (the hosts), the relevant organisation keeps the name-to-IP-address mapping of the hosts

So, starting a dot, we can work our way down the tree to find the machine we want

DNS

All this is done by *name servers*

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

E.g., `ns1.bath.ac.uk` is (the name of) a name server for the University

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

E.g., `ns1.bath.ac.uk` is (the name of) a name server for the University

Also there is `ns2.bath.ac.uk`, for resilience and spreading load

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

E.g., `ns1.bath.ac.uk` is (the name of) a name server for the University

Also there is `ns2.bath.ac.uk`, for resilience and spreading load

They contain replicas of the same information

DNS

All this is done by *name servers*

These are just computers that run the DNS protocol

E.g., `ns1.bath.ac.uk` is (the name of) a name server for the University

Also there is `ns2.bath.ac.uk`, for resilience and spreading load

They contain replicas of the same information

A few years ago, there was a convention of having an off-site replica, too, e.g., Bath used `ns2.ja.net`

DNS

DNS servers do two things

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

(b) Help the local hosts when they need to do a lookup of non-local addresses

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

(b) Help the local hosts when they need to do a lookup of non-local addresses

The former is “just” a lookup in a database

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

(b) Help the local hosts when they need to do a lookup of non-local addresses

The former is “just” a lookup in a database

If a host on the Bath network needs to look up a local name, `linux.bath.ac.uk`, say, it sends a request to one of the local servers, e.g., `ns2.bath.ac.uk`

DNS

DNS servers do two things

(a) Keep the table of name to IP address mappings for the domain

(b) Help the local hosts when they need to do a lookup of non-local addresses

The former is “just” a lookup in a database

If a host on the Bath network needs to look up a local name, `linux.bath.ac.uk`, say, it sends a request to one of the local servers, e.g., `ns2.bath.ac.uk`

And the local server will look it up and return the answer

DNS

How does that host know which machine to ask? In particular, how does it know the local DNS server's IP address?

DNS

How does that host know which machine to ask? In particular, how does it know the local DNS server's IP address?

Every host has the IP addresses of one or more local nameservers, typically in a file (e.g., `/etc/resolv.conf`) that was set up by an administrator or was created by DHCP (or SLAAC)

DNS

How does that host know which machine to ask? In particular, how does it know the local DNS server's IP address?

Every host has the IP addresses of one or more local nameservers, typically in a file (e.g., `/etc/resolv.conf`) that was set up by an administrator or was created by DHCP (or SLAAC)

So it can send the DNS request to one of these servers

DNS

The second function is more interesting: we shall look at an example

DNS

The second function is more interesting: we shall look at an example

If a host in the University requires a name lookup of a non-local name, say `news.bbc.co.uk`, it sends a DNS request to the **local** DNS server, `ns1.bath.ac.uk`, say

DNS

In this example, the Bath server does not have responsibility for the name `news.bbc.co.uk`, but it will helpfully do a lookup for us. It will run down the DNS tree in a *recursive lookup*

DNS

In this example, the Bath server does not have responsibility for the name `news.bbc.co.uk`, but it will helpfully do a lookup for us. It will run down the DNS tree in a *recursive lookup*

The Bath server `ns1` sends a *start of authority* (SOA) request to a random *top level* server to find who is responsible for the `uk` label

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root

They have names like `a.root-servers.net` (`198.41.0.4`)

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root

They have names like `a.root-servers.net` (`198.41.0.4`)

And the local nameserver has the IP addresses of these machines in a file: otherwise we would never get started!

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root

They have names like `a.root-servers.net` (`198.41.0.4`)

And the local nameserver has the IP addresses of these machines in a file: otherwise we would never get started!

The selected top level server responds to the Bath nameserver with something like `ns1.nic.uk`, together with its IP address

DNS

At the top level, there are about 80 machines dotted about the world that serve the DNS root

They have names like `a.root-servers.net` (`198.41.0.4`)

And the local nameserver has the IP addresses of these machines in a file: otherwise we would never get started!

The selected top level server responds to the Bath nameserver with something like `ns1.nic.uk`, together with its IP address

This a machine that is responsible for `uk` labels

DNS

The Bath server now sends a SOA request to `ns1.nic.uk` for `co.uk`

DNS

The Bath server now sends a SOA request to `ns1.nic.uk` for `co.uk`

It gets a reply `ns1.nic.uk`. It just so happens this is responsible for both `uk` and `co.uk`

DNS

The Bath server now sends a SOA request to `ns1.nic.uk` for `co.uk`

It gets a reply `ns1.nic.uk`. It just so happens this is responsible for both `uk` and `co.uk`

So the server sends a SOA request to `ns1.nic.uk` for `bbc.co.uk`

DNS

The Bath server now sends a SOA request to `ns1.nic.uk` for `co.uk`

It gets a reply `ns1.nic.uk`. It just so happens this is responsible for both `uk` and `co.uk`

So the server sends a SOA request to `ns1.nic.uk` for `bbc.co.uk`

It gets `ns.bbc.co.uk`

DNS

It now knows the server reponsible for the address we want

DNS

It now knows the server reponsible for the address we want

The server sends an *address* (A) request for `news.bbc.co.uk` to `ns.bbc.co.uk`

DNS

It now knows the server reponsible for the address we want

The server sends an *address* (A) request for `news.bbc.co.uk` to `ns.bbc.co.uk`

It gets the IP address `194.130.56.40`

DNS

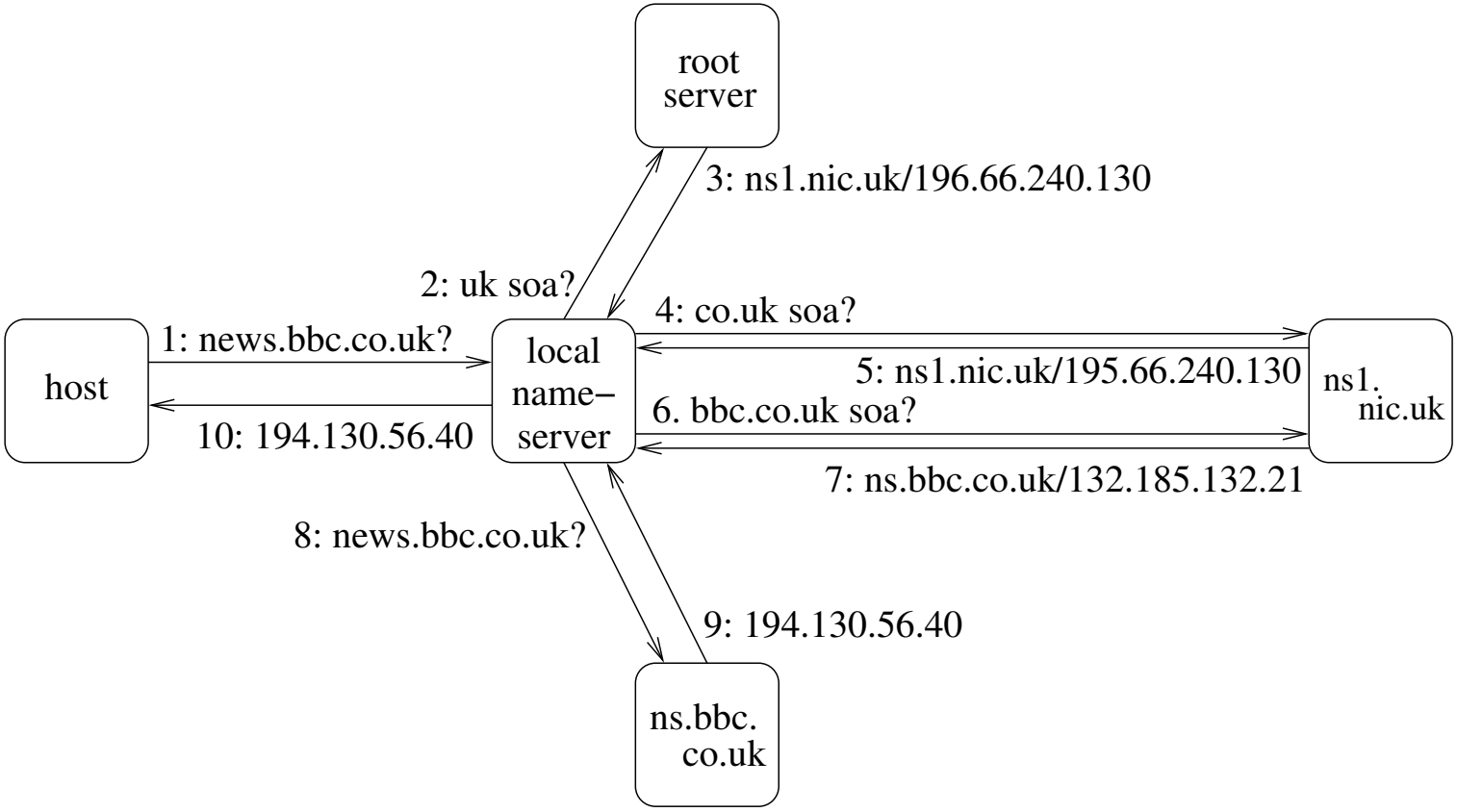
It now knows the server reponsible for the address we want

The server sends an *address* (A) request for `news.bbc.co.uk` to `ns.bbc.co.uk`

It gets the IP address `194.130.56.40`

Finally, the Bath server can relay this back to the original requesting host

DNS



DNS lookup

DNS

Of course, all these responses are cached by the local server so that it doesn't have to go through a complete lookup every time

DNS

Of course, all these responses are cached by the local server so that it doesn't have to go through a complete lookup every time

The next request for `news.bbc.co.uk` can be answered directly by the local nameserver

DNS

Of course, all these responses are cached by the local server so that it doesn't have to go through a complete lookup every time

The next request for `news.bbc.co.uk` can be answered directly by the local nameserver

Similarly, given a request for `www.bbc.co.uk`, it can go directly to `ns.bbc.co.uk`

DNS

As name mappings are not permanent, each reply has a *time to live* attached to it: this indicates how long the server should keep the information before asking again

DNS

As name mappings are not permanent, each reply has a *time to live* attached to it: this indicates how long the server should keep the information before asking again

Stable, long lived associations will have a long TTL

DNS

As name mappings are not permanent, each reply has a *time to live* attached to it: this indicates how long the server should keep the information before asking again

Stable, long lived associations will have a long TTL

Short lived associations will have a short TTL

DNS

The original asking host could do this lookup process itself but the benefit of using the organisation's name server is that it might well have done some or all of the steps already for some other request and cached them

DNS

The original asking host could do this lookup process itself but the benefit of using the organisation's name server is that it might well have done some or all of the steps already for some other request and cached them

This means a faster response and a decrease in network traffic

DNS

The original asking host could do this lookup process itself but the benefit of using the organisation's name server is that it might well have done some or all of the steps already for some other request and cached them

This means a faster response and a decrease in network traffic

And less work for the host