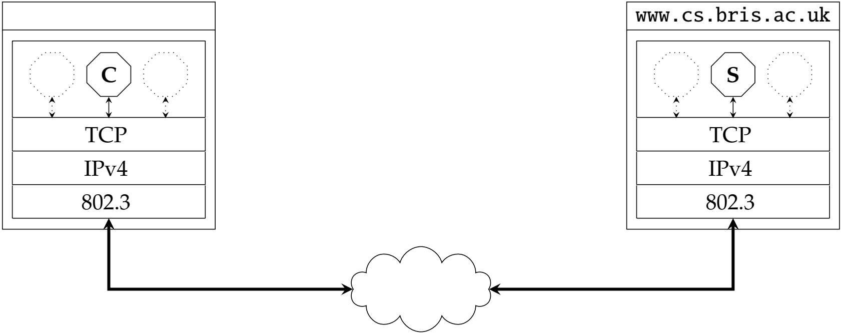
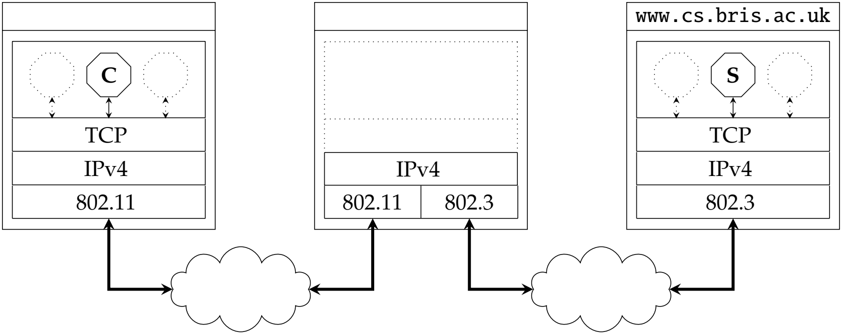


- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,
  1. addressing,
  2. forwarding, and
  3. routing,

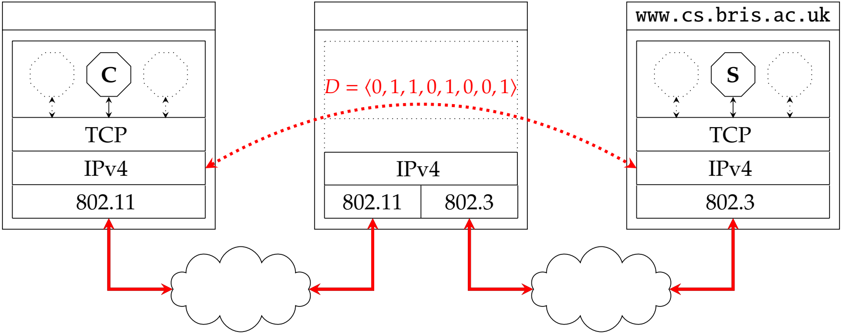
  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.



- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,

  1. addressing,
  2. forwarding, and
  3. routing,

  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.



$F = H_{802.11} \parallel H_{IPv4} \parallel \langle 0,1,1,0,1,0,0,1 \rangle \parallel T_{802.11}$

- Goal: investigate the **network layer** (or *inter*-networking more generally) e.g.,
  1. addressing,
  2. forwarding, and
  3. routing,

  st. we can transmit (structured) **packets** through an inter-network of multiple connected LANs.

An Aside: "This Jen, is *the Internet*"

---

http://www.youtube.com/watch?v=iDbyYGrswtg

An Aside: "This Jen, is *the Internet*"

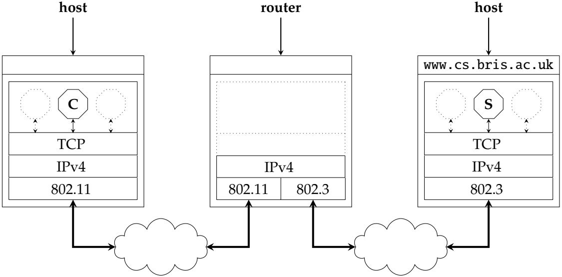
http://www.youtube.com/watch?v=iDbyYGrswtg

Concepts (1)

Definition

An inter-network is formed from entities such as

1. **End System (ES)**, i.e., a **host**,
2. **Intermediate System (IS)**, e.g., a **router**, and/or
3. **Autonomous System (AS)**, i.e., a collection of connected nodes under the control of one operator.

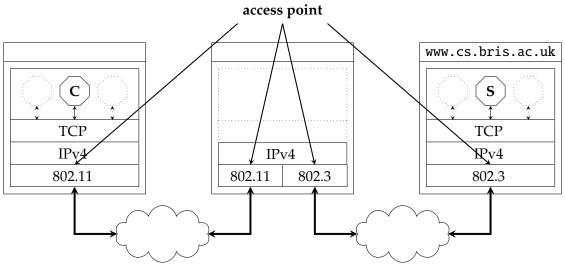


Concepts (1)

Definition

Each network layer **access point** is identified by an **address**:

- unlike link layer addresses, network layer addresses need to be *globally* unique,
- one address per host may be insufficient: we need an address per point of access, so per NIC for example.



Concepts (1)

Definition

Transmission of packets through an inter-network is based on two tasks, namely

1. **routing**, i.e., looking at the destination address in a packet and deciding on the next **hop** (toward said destination), and
2. **forwarding**, i.e., actually transmitting the packet to the next hop.

IPv4 (1)

- **Internet Protocol (IP)** [13] is an inter-networking lingua franca ...
- ... rather than services per se, it deals with *abstraction*: it offers
  - unicast (one-to-one),
  - broadcast (one-to-many, i.e., one-to-“all available”),
  - multicast (one-to-some, i.e., one to-“all selected”), and
  - anycast (one-to-nearest)

  packet delivery models, each of which is

  - unreliable,
  - connection-less (i.e., stateless), and
  - “best effort” (i.e., no QoS guarantees)

  but, crucially, allows hererogenaity wrt. lower and higher layers.

IPv4 (2)

Algorithm

Given a packet provided by the lower, link layer:

1. validate header (e.g., use checksum to check for errors),
2. process options in header,
3. check destination address: if the packet is for this host

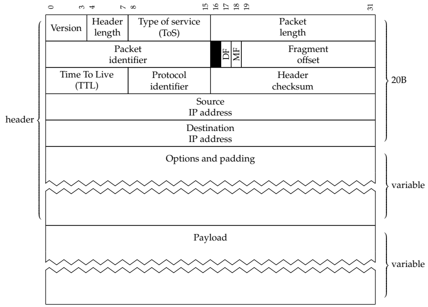
   3.1 buffer fragments and apply reassembly process, then eventually
   3.2 provide payload to a higher layer (per protocol field)

   otherwise, assuming we want to forward the packet

   3.3 check TTL, and drop if exceeded,
   3.4 look-up next hop in forwarding table,
   3.5 apply fragmentation and header update processes, (e.g., decrement TTL, recompute checksum),
   3.6 transmit packet(s) via lower, link layer

and if/when errors occur, signal them appropriately.

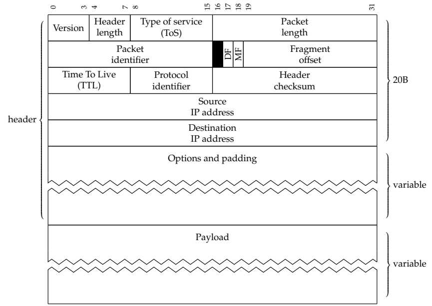
Data Structure (IPv4 packet [13, Section 3.1])



The data structure includes:

- A packet version number (allowing protocol evolution).
- A header length (measured in 32-bit words), formally termed **Internet Header Length (IHL)**.
- A packet length (measured in 8-bit octets), which includes the header.
- A packet identifier.

Data Structure (IPv4 packet [13, Section 3.1])



The data structure includes:

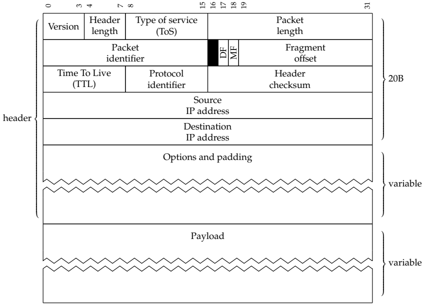
- A set of service options, originally defined as
  - 2-bit reserved,
  - 3-bit priority level,
  - 1-bit reliability (normal or high),
  - 1-bit latency (normal or low),
  - 1-bit throughput (normal or high)

  but now depreciated in favour of

  1. **Differentiated Services (DS)** [11] and
  2. **Explicit Congestion Notification (ECN)** [14]

  fields.

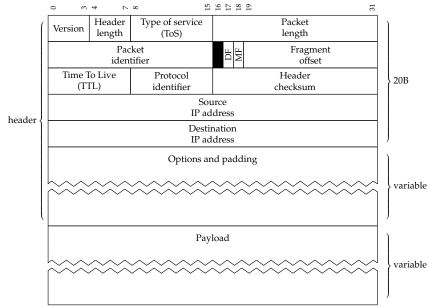
Data Structure (IPv4 packet [13, Section 3.1])



The data structure includes:

- A set of flags, including
  - 1-bit **More Fragments (MF)** flag, which marks final or non-final fragment(s),
  - 1-bit **Don't Fragment (DF)** flag, which prohibits fragmentation by the IP layer.
- A fragment offset (measured in 64-bit words), specifying where this packet payload stems from in the original, unfragmented packet.

Data Structure (**IPv4 packet** [13, Section 3.1])

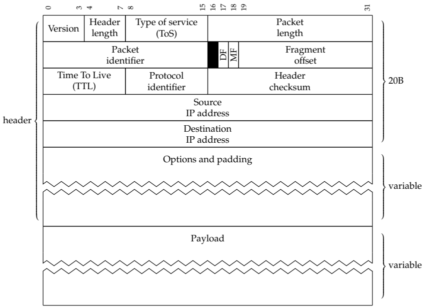


The data structure includes:

- A **Time To Live (TTL)** field specifying the point at which the packet is discarded (if not yet delivered to the destination).
- A 16-bit checksum (on header only) used to detect errors.

IPv4 (3)
Packets

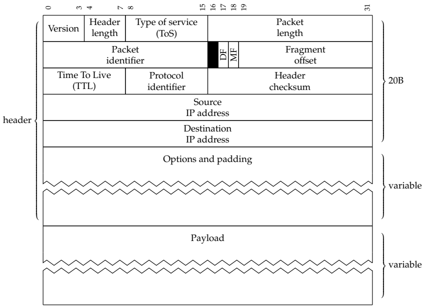
Data Structure (IPv4 packet [13, Section 3.1])



The data structure includes:

- A 32-bit source IP address.
- A 32-bit destination IP address.
- The protocol identifier, specifying which higher layer will receive the packet once it reaches the destination.

Data Structure (**IPv4 packet** [13, Section 3.1])



The data structure includes:

- A set of options (allowing protocol extensibility).
- Any padding required to ensure the header is a multiple of 32 bits.
- The payload.

Definition

Each access point is assigned a unique identifier called an **IP address**. Note that

- a given address x is simply an n bit integer,
- for IPv4 we have $n = 32$,
- IP addresses are often written in dotted-decimal notation, i.e., 4 decimal integers $0 \leq \hat{x}_i < 256$ each representing 8 bits of the address x.

Definition

IP addresses are hierarchical: a **prefix** l defines a block of addresses, called a **sub-network** (or **sub-net**), by dividing the address into

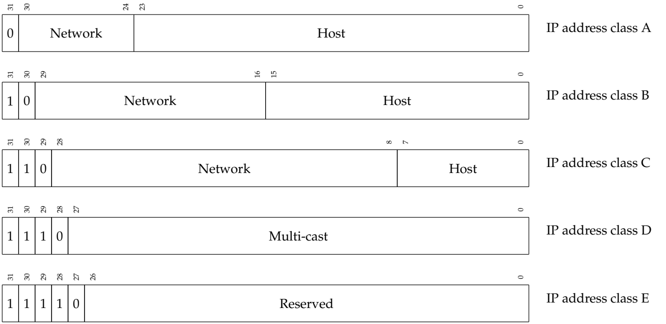
1. an l-bit **network identifier**, and
2. an $(n - l)$-bit **host identifier**

All host identifiers within a block share the same prefix, i.e., have the same network identifier. Note that smaller l means more host identifiers, so is less specific, while larger l means fewer host identifiers, so is more specific.

- Question: how do we know the prefix l?

IPv4 (9)
Addressing

- Question: how do we know the prefix l?
- Answer #1: pre-1993, using a class-based hierarchy, i.e.,



- Question: how do we know the prefix l?

- Answer #2: post-1993, using **Classless Inter-Domain Routing (CIDR)** [15], st.

  137.222.103.3/24

  explicitly denotes the fact $l = 24$.

- Question: how do we obtain an address x (or block thereof)?

- Question: how do we obtain an address x (or block thereof)?
- Answer #1: use an assigned **public address block**
  1. network identifiers are assigned by **Internet Assigned Numbers Authority (IANA)**, e.g.,

     137.222.0.0/16

     was hierarchically assigned via IANA → RIPE → UoB, and
  2. host identifiers are assigned within the (sub-)network, e.g.,

     137.222.103.3

     was statically assigned to `snowy.cs.bris.ac.uk`

  which *is* globally unique.

- Question: how do we obtain an address x (or block thereof)?

- Answer #2: use *any* **private address block** [8] e.g.,

  $$192.168.0.0/16$$

  and *any* address in it, e.g.,

  $$192.168.1.123$$

  which *isn't* globally unique!

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.

IPv4 (11)
Forwarding

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- Observation:
  - all hosts on the same (sub-)network share a prefix, so
  - maintain a **forwarding table** that maps prefixes to next hop

  st. the table remains compact iff. sensible prefixing is used.

- Goal: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- Problem: prefixes in the table might overlap.
- Solution: select the *longest* matching prefix that applies; this allows
  - special-case behaviours (via more-specific prefix, e.g., 137.222.103.3/32), and
  - default behaviours (via less-specific prefix, e.g., 0.0.0.0/0).

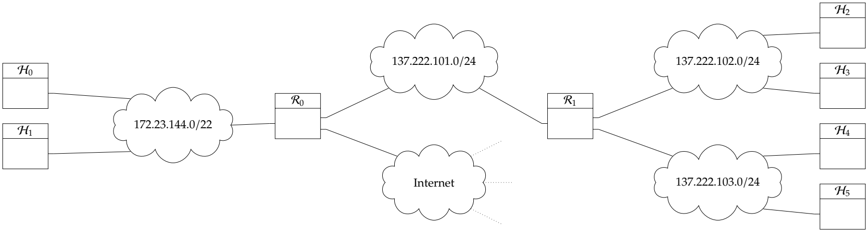
- **Goal**: forward a packet P from source $\mathcal{H}_i$ on next hop toward destination $\mathcal{H}_j$.
- **Observation**: we only need to know what the next hop is, so
  - have the routers make (global) routing decisions, and
  - have the hosts follow a simple rule:
    1. communicate locally with destination if it is on the same sub-network, otherwise
    2. forward to nearest router as next hop toward destination

i.e., improve scalability by leveraging hierarchy within topology and addressing.

- Example: forward a packet

$$P = H_{IPv4}[\mathrm{src} = \mathcal{H}_0, \mathrm{dst} = \mathcal{H}_2] \parallel D$$

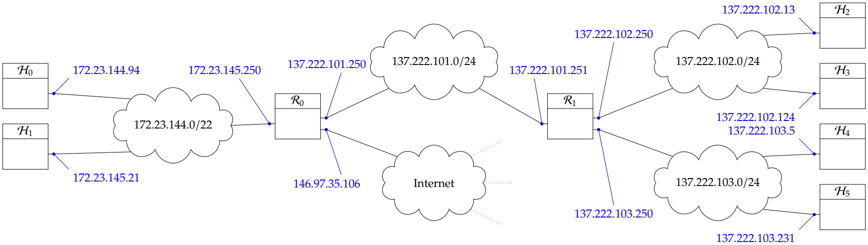
through some (purely hypothetical) inter-network.



- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13] \parallel D$$

through some (purely hypothetical) inter-network.

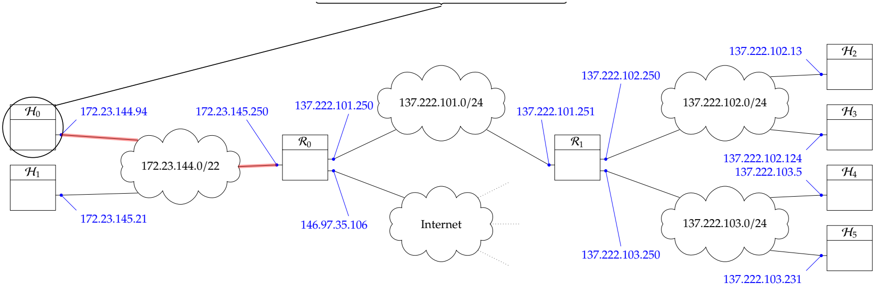


- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 64] \parallel D$$

through some (purely hypothetical) inter-network.

Address	Next hop
172.23.144.0/22	NIC via broadcast
0.0.0.0/0	NIC via 172.23.145.250

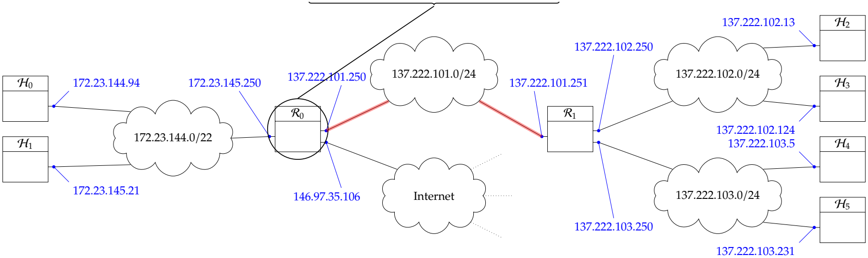


- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 63] \parallel D$$

through some (purely hypothetical) inter-network.

Address	Next hop
137.222.101.0/24	NIC #1 via broadcast
137.222.102.0/23	NIC #1 via 137.222.101.251
172.23.144.0/22	NIC #2 via broadcast
0.0.0.0/0	NIC #3

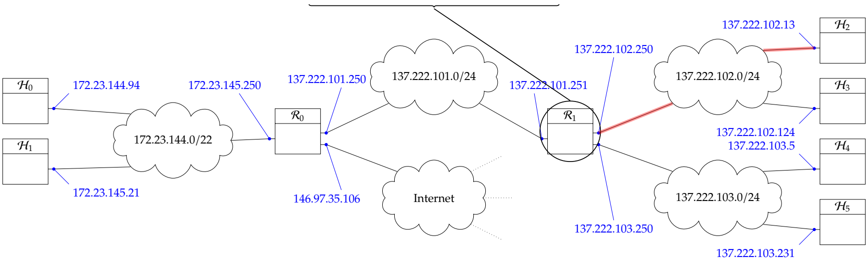


- Example: forward a packet

$$P = H_{IPv4}[\text{src} = 172.23.144.94, \text{dst} = 137.222.102.13, \text{TTL} = 62] \parallel D$$

through some (purely hypothetical) inter-network.

Address	Next hop
137.222.103.0/24	NIC #3 via broadcast
137.222.102.0/24	NIC #2 via broadcast
137.222.101.0/24	NIC #1 via broadcast
0.0.0.0/0	NIC #1 via 137.222.101.250



- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?
- "Solution" #1: produce an error, and drop the packet!

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

- Solution #2:

1. allow $\mathcal{H}_i$ to transmit packets of any length,
2. have each router **fragment** any outgoing packet that is larger than the associated MTU,
3. reassemble packet fragments at destination.

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

- Solution #3:
  1. force $\mathcal{H}_i$ to discover the **path MTU** between $\mathcal{H}_i$ and $\mathcal{H}_j$, then
  2. limit the size of packets transmitted by $\mathcal{H}_i$ so fragmentation is avoided.

- Problem:
  - each (sub-)network has a **Maximum Transmission Unit (MTU)**,
  - if $\mathcal{H}_i$ transmits an n-octet packet to $\mathcal{H}_j$, what if n is larger than some (intermediate) MTU?

noting that IPv4 hosts and routers

- must support reassembly [6, Section 3.3.2], and
- may support fragmentation [6, Section 3.3.3]

but modern implementations typically use path MTU discovery.

Algorithm (fragment)

At the source or an intermediate router, imagine there is a need to fragment some packet P:

1. If the DF flag in P is **true**, drop P.
2. Otherwise divide the payload into n fragments, F_i for $0 \leq i < n$, each of whose length (including header) is less than the MTU.
3. Copy the header from P into each F_i, and update both the offset and MF flags based on i.
4. Transmit each F_i.

Algorithm (reassemble)

At the destination *only*:

1. Buffer fragments with same identifier until they're all received, noting
   - they may be received out-of-order,
   - a reassembly time-out prevents indefinite buffering, and
   - a fragment F_i whose MF flag is **false** is the last one, so yields the total length.
2. Reconstruct the original packet P (at least the payload).
3. Process P as per normal, i.e., provide it to whatever transport layer protocol P indicates.

Concepts (1)

- Recall:
  - routing is the act of deciding a path used when forward packets from a given source to a given destination,
  - forwarding is a *local* process, routing is *global* in the sense it involves the whole (inter-)network,
  - goal is to make best use of connectivity and thus bandwidth: it can be viewed as a form of resource allocation.

Concepts (1)

- Recall:
  - routing is the act of deciding a path used when forward packets from a given source to a given destination,
  - forwarding is a *local* process, routing is *global* in the sense it involves the whole (inter-)network,
  - goal is to make best use of connectivity and thus bandwidth: it can be viewed as a form of resource allocation.
- Question: how does the forwarding table get populated with entries?
- Answer(s):
  1. static (or fixed) routing, i.e., hard-code routing information by hand,
  2. source routing, i.e., let the source pre-determine routing decisions, or
  3. adaptive routing, e.g., i.e., use a distributed **routing algorithm** (or **routing protocol**).

Concepts (2)

- Good news: we can reason about routing by noting

$$\text{network} \equiv \text{graph} \implies \text{graph theory} \subset \text{data structures and algorithms},$$

  and that

  - a network graph will be weighted to capture the properties of each connection,
  - we could use directed graphs (e.g., to capture uni-directional connection properties) ...
  - ... but for simplicity we'll consider undirected graphs only

  st. our network is modelled by

$$G = (V, E = \langle (u_0, v_0, d_0), (u_1, v_1, d_1), \ldots, (u_{m-1}, v_{m-1}, d_{m-1}) \rangle)$$

  where $|V| = n$ and $|E| = m$.

Concepts (3)

- Bad news: any algorithm (ideally) needs to be

correct	$\Rightarrow$	find paths that provide end-to-end connectivity
efficient	$\Rightarrow$	make good use of resources
fair	$\Rightarrow$	will not "stave" nodes of bandwidth
convergent	$\Rightarrow$	initialises/recovers quickly, e.g., after change to topology
scalable	$\Rightarrow$	remains efficient even with large n and/or m
	$\vdots$	

  and *must* be **decentralised**: the model is that

  1. no (global) controller node exists,
  2. nodes typically start with only local knowledge of topology,
  3. nodes communicate (concurrently) with neighbours only, and
  4. nodes *and* links can fail!

Concepts (4)

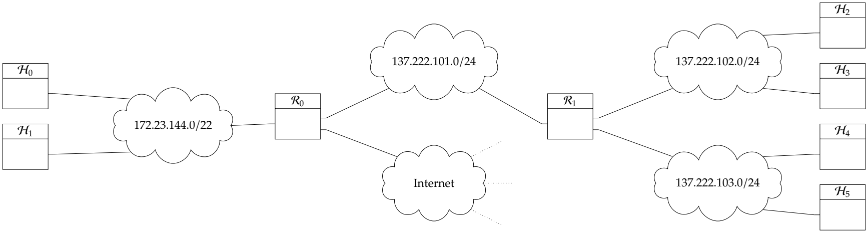
- Recall: we have *already* assumed
  - routers make (global) routing decisions, whereas
  - hosts communicate locally, or forward to nearest router

  so we can route wrt. **routing units** (or regions), *not* per-node destinations.
- The strategy is to address scalablity using hierarchy, so
  1. route *to* region, then
  2. route *in* region

  e.g., by leveraging IP prefixes to coalesce multiple destinations into one region (or block) ...

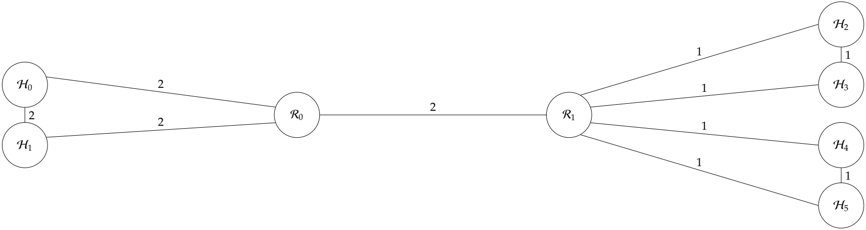
Concepts (5)

- ... so we *significantly* simplify the problem to:



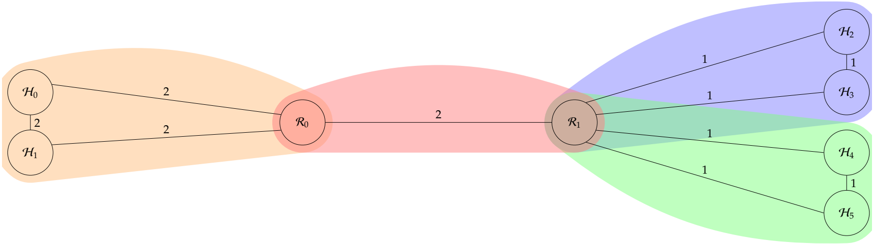
Concepts (5)

- ... so we *significantly* simplify the problem to:



Concepts (5)

- ... so we *significantly* simplify the problem to:



Routing Algorithms (1)

- Idea: in general, we'll have each routing algorithm
  1. maintain some state, i.e., a **routing table**,
  2. periodically transmit, receive and integrate information via (local) communication with neighbouring nodes,
  3. periodically translate the routing table into forwarding table, e.g., via some form of (local) computation

  keeping in mind that

  - periodically means at regular intervals, *plus* when a change is topology occurs, and
  - a cost of ∞ means a link does not exist *or* has failed: either way, avoid it!

- Idea:

$$\textbf{distance vector routing} \simeq \text{distributed Bellman-Ford},$$

  in the sense each node u

  1. maintains a **distance vector**

  $$\langle (v_0, d_0), (v_1, d_1), \ldots, (v_{l-1}, d_{l-1}) \rangle$$

     of next hop and cost tuples, initialised st.

  $$d_i = \begin{cases} 0 & \text{if } v_i = u \\ \infty & \text{otherwise} \end{cases},$$

  2. periodically transmits the distance vector to all neighbours,
  3. periodically updates the distance vector with new information, st.

  $$dist(u, v) = \min_{\forall w,\ (u,w,d) \in E} \left[dist(w, v) + d \right].$$

- Example: **Routing Information Protocol (RIP)** [7].
  - uses hop count as a cost function, assuming $\infty \equiv 16$,
  - transmits distance vectors every ~ 30s with ~ 180s failure time-out.

- Idea:

$$\textbf{link state routing} \simeq \text{flooding} + \text{Dijkstra},$$

in the sense each node u

1. floods network with **link state packets** (i.e., neighbouring nodes plus link costs) yielding global topology, then
2. use Dijkstra to compute shortest paths.

- Examples:
  - **Open Shortest Path First (OSPF)** [10], and
  - **Intermediate System to Intermediate System (IS-IS)** [12].

Routing Algorithms (6)

- To summarise the two approaches covered, we can say

Metric	Distance Vector	Link State
correct	distributed Bellman-Ford	replicated Dijkstra
efficient	approximately	approximately
fair	approximately	approximately
convergent	slow: many exchanges	fast: flood then recompute
scalable	excellent	reasonable

  i.e., selection is basically a trade-off between convergence and scalablity ...

- ... *plus* we need to cater for various corner cases:
  - distance vector routing can fail if
    - if network is partitioned (cf. graph cut, meaning "count to infinity" problem),

    while
  - link state routing can fail if
    - flooding sequence numbers can overflow or be corrupted,
    - nodes can fail and reset flooding sequence number,
    - if network is partitioned and then re-joined.

Conclusions

- Take away points:
  - IP is a central, and hence important protocol; how it deals with the challenges of

    1. addressing,
    2. forwarding, and
    3. routing,

    interacts with other components of, and *explains* a lot in an Internet model network stack.

Conclusions

- Take away points:
  - Routing in particular is a broader topic
    - \+ using graph theory, for example, to study routing algorithms gives a formal, theoretical basis
    - – translating theory into practice is *still* a significant challenge wrt.
      - functionality, e.g., interior vs. exterior routing (which then includes **Border Gateway Protocol (BGP)** [9]),
      - efficiency, e.g., scalability, decentralisation,
      - ...
  - Routing protocols are instances of more general **consensus protocols** whereby (distributed) parties need to agree on a shared state.

Additional Reading

- *Wikipedia: Network layer*. URL: http://en.wikipedia.org/wiki/Network_layer.
- *Wikipedia: Internet Protocol (IP)*. . URL: http://en.wikipedia.org/wiki/Internet_Protocol.
- W. Stallings. "Chapter 19: Internet protocols". In: *Data and Computer Communications*. 9th ed. Pearson, 2010.
- W. Stallings. "Chapter 20: Internetwork operation". In: *Data and Computer Communications*. 9th ed. Pearson, 2010.
- W. Stallings. "Chapter 13: Routing in switched data networks". In: *Data and Computer Communications*. 9th ed. Pearson, 2010.

References

[1] *Wikipedia: Internet Protocol (IP)*. URL: http://en.wikipedia.org/wiki/Internet_Protocol (see p. 53).

[2] *Wikipedia: Network layer*. URL: http://en.wikipedia.org/wiki/Network_layer (see p. 53).

[3] W. Stallings. "Chapter 13: Routing in switched data networks". In: *Data and Computer Communications*. 9th ed. Pearson, 2010 (see p. 53).

[4] W. Stallings. "Chapter 19: Internet protocols". In: *Data and Computer Communications*. 9th ed. Pearson, 2010 (see p. 53).

[5] W. Stallings. "Chapter 20: Internetwork operation". In: *Data and Computer Communications*. 9th ed. Pearson, 2010 (see p. 53).

[6] R. Braden. *Requirements for Internet hosts – Communication Layers*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1122. 1989. URL: http://tools.ietf.org/html/rfc1122 (see pp. 33–37).

[7] C. Hedrick. *Routing Information Protocol*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1058. 1988. URL: http://tools.ietf.org/html/rfc1058 (see p. 48).

[8] IANA. *Special-use IPv4 addresses*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 3330. 2002. URL: http://tools.ietf.org/html/rfc3330 (see pp. 21–23).

[9] K. Loughheed and Y. Rekhter. *A Border Gateway Protocol (BGP)*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1105. 1989. URL: http://tools.ietf.org/html/rfc1105 (see pp. 51, 52).

[10] J. Moy. *OSPF specification*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1131. 1989. URL: http://tools.ietf.org/html/rfc1131 (see p. 49).

[11] K. Nichols et al. *Definition of the Differentiated Services field (or DS Field) in the IPv4 and IPv6 headers*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 2474. 1998. URL: http://tools.ietf.org/html/rfc2474 (see p. 12).

[12] D. Oran. *OSI IS-IS Intra-domain Routing Protocol*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1142. 1990. URL: http://tools.ietf.org/html/rfc1142 (see p. 49).

[13] J. Postel. *Internet Protocol*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 791. 1981. URL: http://tools.ietf.org/html/rfc791 (see pp. 9, 11–16).

[14] K. Ramakrishnan, S. Floyd, and D. Black. *The addition of Explicit Congestion Notification (ECN) to IP*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 3168. 2001. URL: http://tools.ietf.org/html/rfc3168 (see p. 12).

[15] Y. Rekhter and T. Li. *An architecture for IP address alloation with CIDR*. Internet Engineering Task Force (IETF) Request for Comments (RFC) 1518. 1989. URL: http://tools.ietf.org/html/rfc1518 (see pp. 18–20).