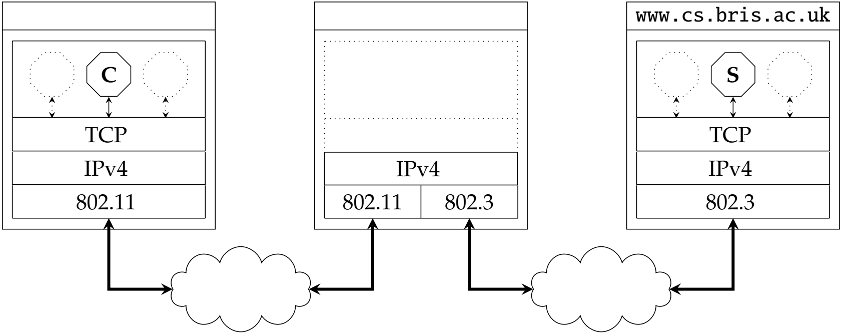
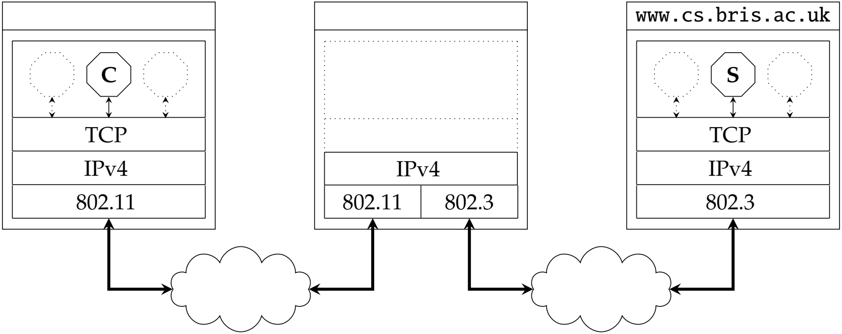
- Recall: we know how to realise



st. hosts can transmit IP packets to each other.

- Recall: we know how to realise



st. hosts can transmit IP packets to each other ...

- ... *but*

1. how does the destination get an IP address in the first place,
2. how does the source find out the router MAC address, and
3. how are communication errors signalled by the router?

Problem #1 $\rightsquigarrow$ DHCP (1)

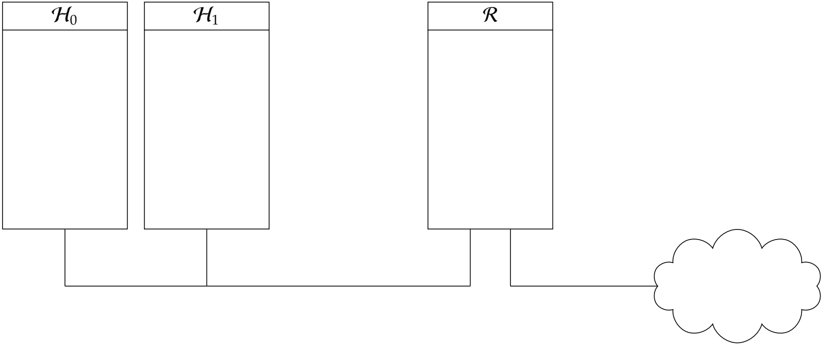
- Problem: how does some host, say $\mathcal{H}_2$, get assigned an IP address?
- Solution(s):
  1. manually assign one, *or*
  2. automatically assign one via
     - **Reverse Address Resolution Protocol (RARP)** [8],
     - **BOOTstrap Protocol (BOOTP)** [9], or
     - **Dynamic Host Configuration Protocol (DHCP)** [7]

  noting that

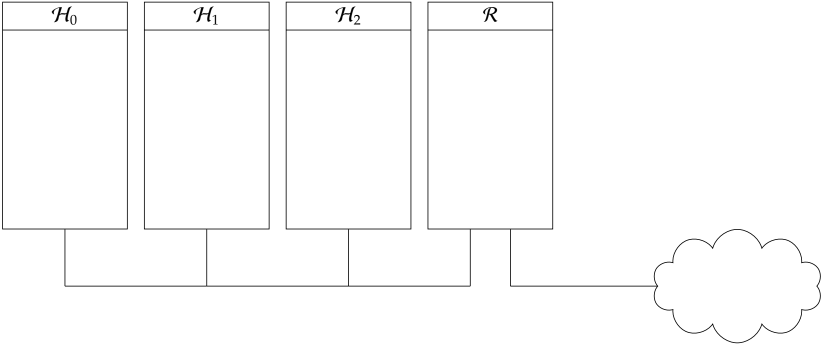
  - maintainability at scale, and
  - support for *re*assignment of addresses

  are important drivers for the latter.

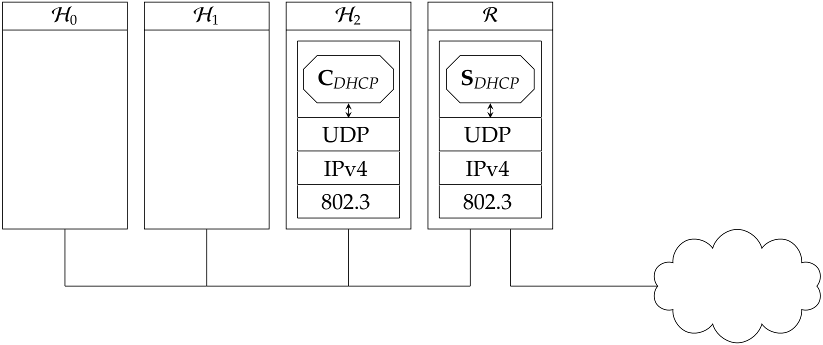
Problem #1 $\rightsquigarrow$ DHCP (2)



Problem #1 $\rightsquigarrow$ DHCP (2)

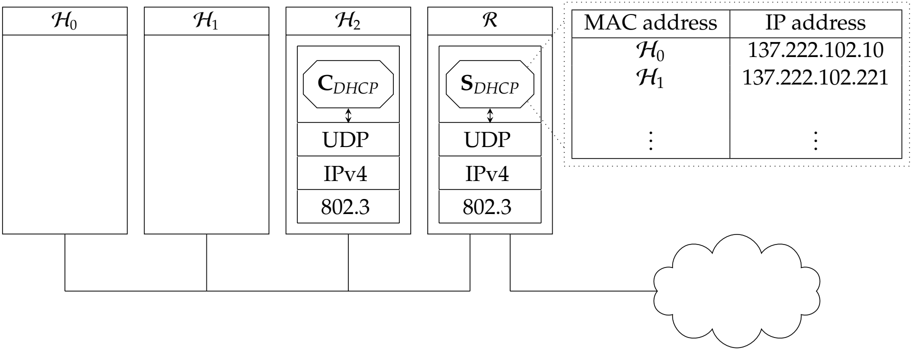


Problem #1 $\rightsquigarrow$ DHCP (2)



- Basic idea:
  - The DHCP protocol operates within the *application* layer, making use of UDP (and hence IP).
  - Each host executes a DHCP client that configures the network stack; a DHCP server executes somewhere on the same sub-network.

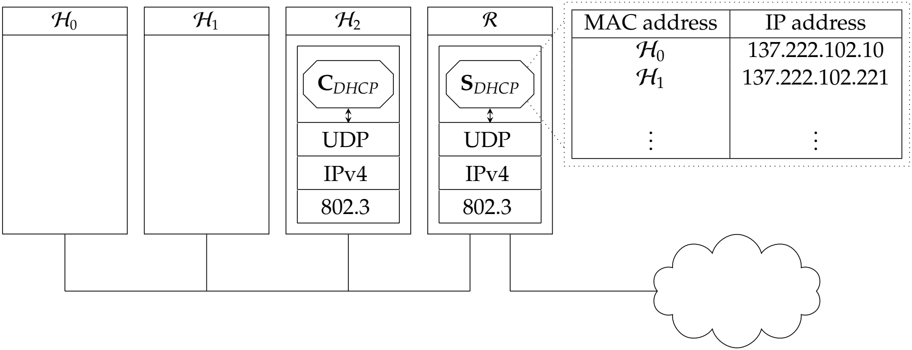
Problem #1 $\leadsto$ DHCP (2)



- **Basic idea**:
  - The server maintains a table (or **pool**) mapping keys (e.g., MAC addresses) to configurations (e.g., IP addresses); a given configuration is **leased** to a client for some period.
  - Several different strategies are possible:

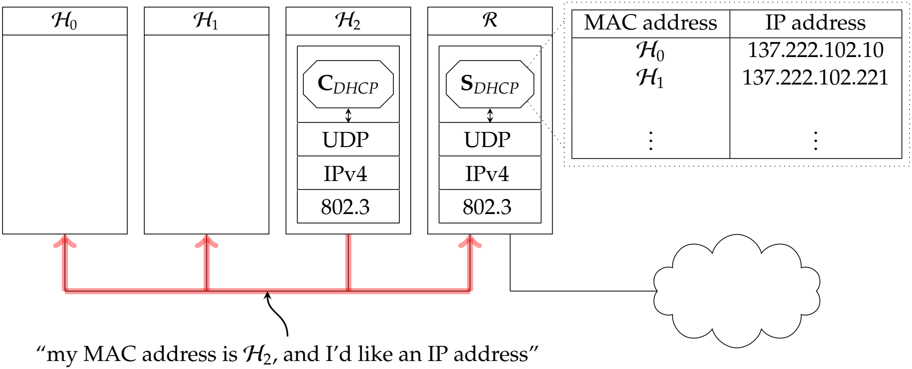
Strategy	Assignment	Lease
Static	Static, pre-defined mapping	Permanent
Automatic	Dynamic, drawn from range(s)	Permanent
Dynamic	Dynamic, drawn from range(s)	Temporary

Problem #1 ⇝ DHCP (2)



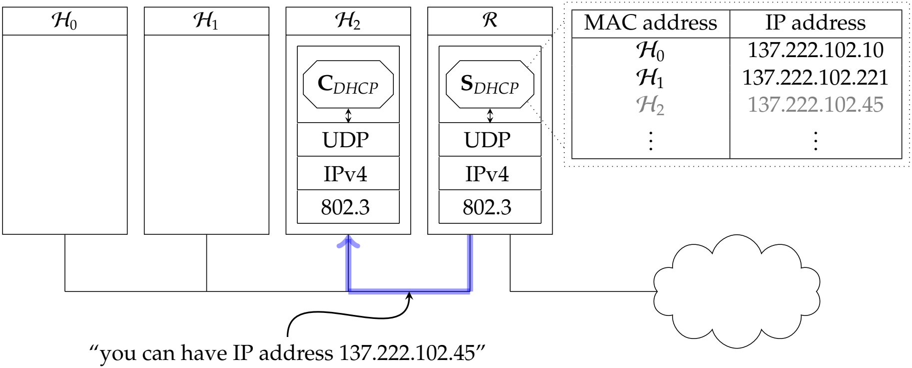
- Basic idea:
  - The end-points engage in a 4-step protocol:
    1. the client broadcasts a **discover** message,
    2. the server transmits an **offer** message,
    3. the client broadcasts a **request** message, and
    4. the server transmits an **acknowledgement** message.

Problem #1 ⇝ DHCP (2)



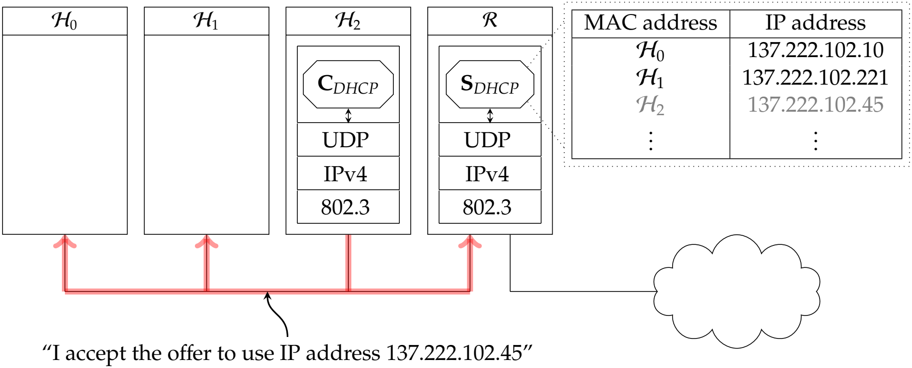
- Basic idea:
  - The end-points engage in a 4-step protocol:
    1. the client broadcasts a **discover** message,
    2. the server transmits an **offer** message,
    3. the client broadcasts a **request** message, and
    4. the server transmits an **acknowledgement** message.

Problem #1 ⇝ DHCP (2)



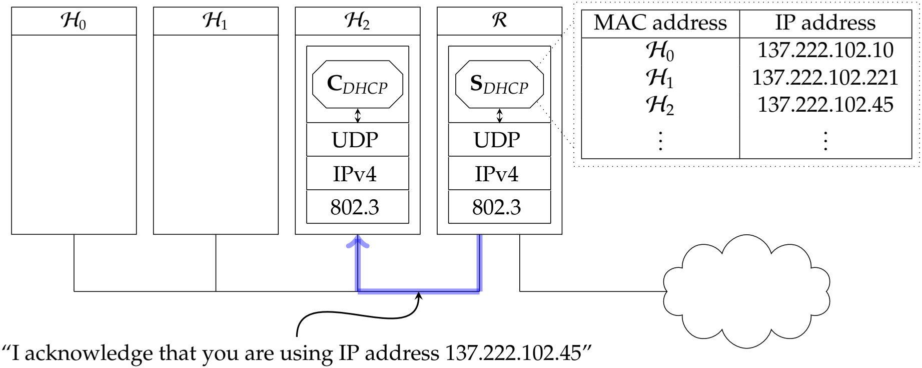
- Basic idea:
  - The end-points engage in a 4-step protocol:
    1. the client broadcasts a **discover** message,
    2. the server transmits an **offer** message,
    3. the client broadcasts a **request** message, and
    4. the server transmits an **acknowledgement** message.

Problem #1 $\rightsquigarrow$ DHCP (2)



- Basic idea:
  - The end-points engage in a 4-step protocol:
    1. the client broadcasts a **discover** message,
    2. the server transmits an **offer** message,
    3. the client broadcasts a **request** message, and
    4. the server transmits an **acknowledgement** message.

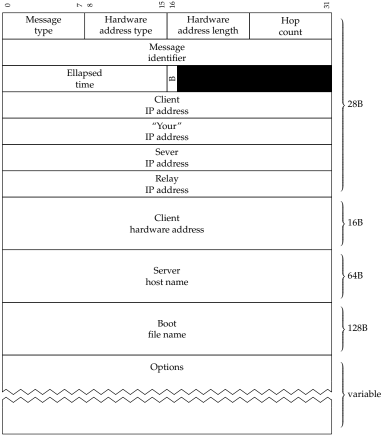
Problem #1 ⇝ DHCP (2)



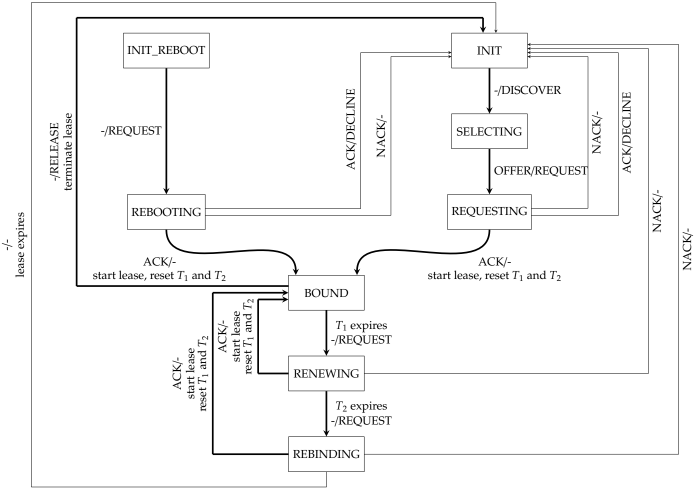
- Basic idea:
  - The end-points engage in a 4-step protocol:
    1. the client broadcasts a **discover** message,
    2. the server transmits an **offer** message,
    3. the client broadcasts a **request** message, and
    4. the server transmits an **acknowledgement** message.

Problem #1 ⇝ DHCP (3)

Data Structure (**DHCP message** [7, Section 2])



Algorithm (**DHCP state machine** [7, Figure 5])



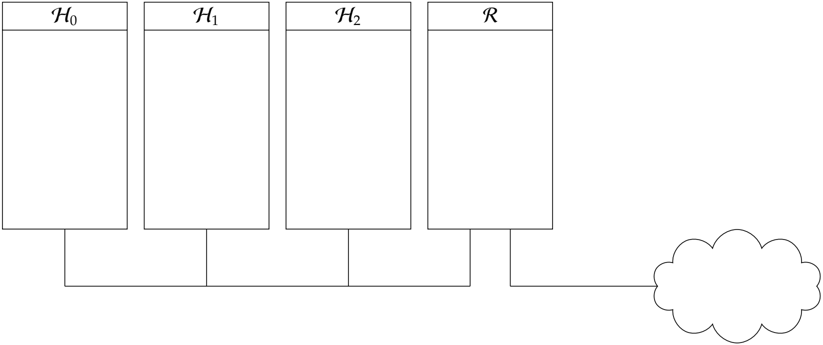
- Problem: imagine some host $\mathcal{H}_2$ wants to transmit an encapsulated IP packet

$$P = H_{802.3}[\text{src} = \alpha, \text{dst} = \beta] \parallel H_{IPv4}[\text{src} = \gamma, \text{dst} = \delta] \parallel D \parallel T_{802.3}$$

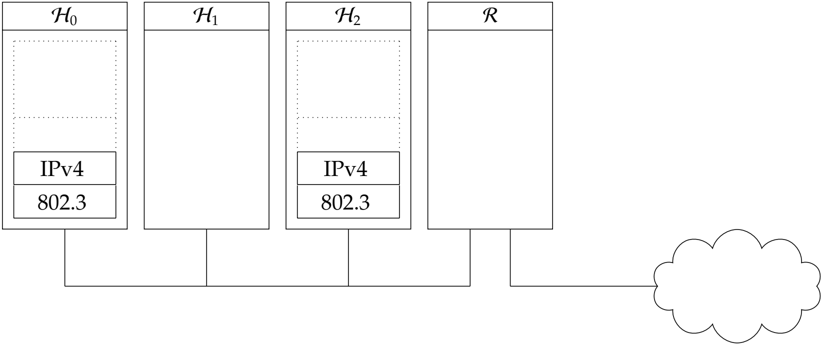
  to $\mathcal{H}_0$ on the same (sub-)network ... given
  - γ is the IP address of $\mathcal{H}_2$,
  - δ is the IP address of $\mathcal{H}_0$,
  - α is the MAC address of $\mathcal{H}_2$, and
  - β is the MAC address of $\mathcal{H}_0$

  how does $\mathcal{H}_2$ know, or how can it find out, β?
- Solution: it uses **Address Resolution Protocol (ARP)** [10].

Problem #2 $\rightsquigarrow$ ARP

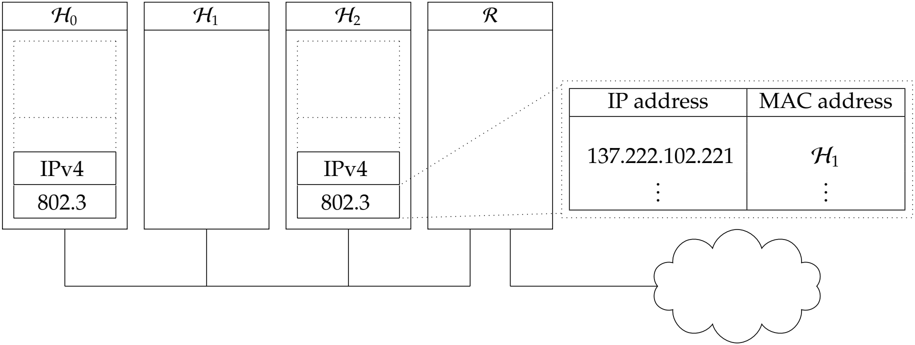


Problem #2 $\rightsquigarrow$ ARP



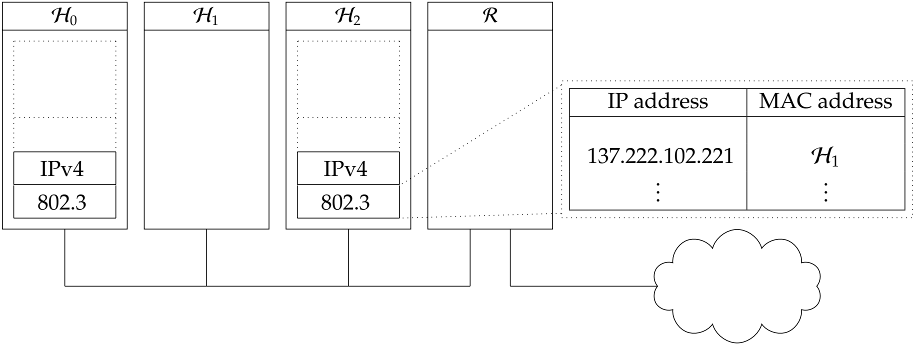
- Basic idea:
  - ARP operates within the *link* layer, servicing requests from the network layer (i.e., IP).
  - In this case, ARP messages will therefore be encapsulated in 802.3 frames and processed by the 802.3 MAC sub-layer.

Problem #2 $\rightsquigarrow$ ARP



- Basic idea:
  - Using ARP for *every* IP address used would represent too high an overhead, so each host maintains a **cache**.
  - Cached entries can of course become stale (i.e., the mapping becomes invalid) over time, so are periodically refreshed.

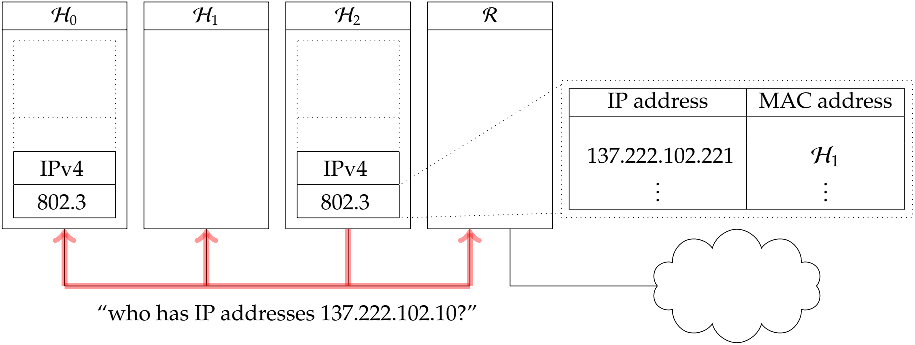
Problem #2 ⇝ ARP



- Basic idea:
  - The end-points engage in a 1- or 3-step protocol:
    1. the requester can use a cached entry if available, otherwise
    2. the requester broadcasts a **request** message, and
    3. the requestee transmits a **response** message

    where a lack of response in step 3 prompts the requester to retry (including a back-off), then eventually give up.

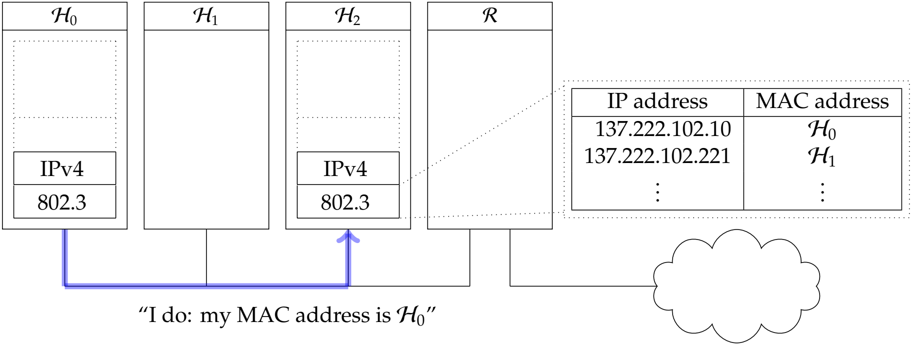
Problem #2 ⇝ ARP



- Basic idea:
  - The end-points engage in a 1- or 3-step protocol:
    1. the requester can use a cached entry if available, otherwise
    2. the requester broadcasts a **request** message, and
    3. the requestee transmits a **response** message

    where a lack of response in step 3 prompts the requester to retry (including a back-off), then eventually give up.

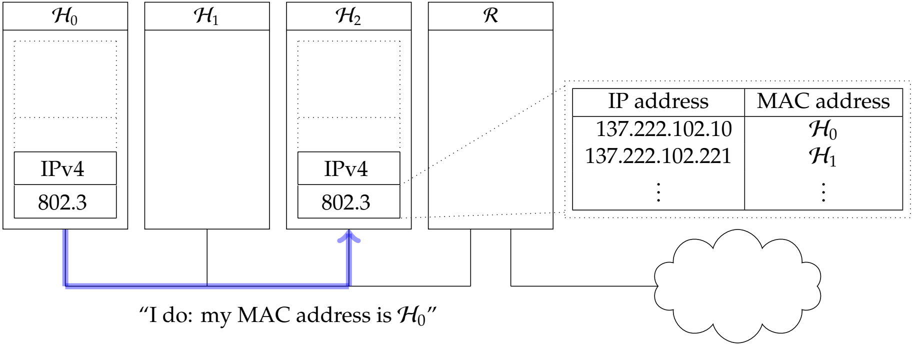
Problem #2 ⇝ ARP



- Basic idea:
  - The end-points engage in a 1- or 3-step protocol:
    1. the requester can use a cached entry if available, otherwise
    2. the requester broadcasts a **request** message, and
    3. the requestee transmits a **response** message

    where a lack of response in step 3 prompts the requester to retry (including a back-off), then eventually give up.

Problem #2 $\rightsquigarrow$ ARP



- Basic idea:
  - The end-points engage in a 1- or 3-step protocol:

    1. the requester can use a cached entry if available, otherwise
    2. the requester broadcasts a **request** message, and
    3. the requestee transmits a **response** message

    where a lack of response in step 3 prompts the requester to retry (including a back-off), then eventually give up.

Problem #3 ⇝ ICMP

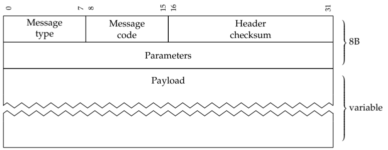
- Problem: how are communication errors signalled?
- Solution: via **Internet Control Message Protocol (ICMP)** [11], which actually covers

  1. error reporting,
  2. network control, *and*
  3. information retrieval

  operations.

Problem #3 ⇝ ICMP

Data Structure (ICMP message [11])

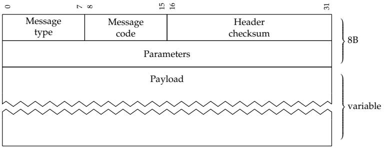


- Each ICMP message is encapsulated in an IP packet:
  - the source is the host/router where the error was detected, while
  - the destination is the host which transmitted the packet

  noting

  - the 16-bit message type and code specify the message type,
  - the payload of some messages captures the header *and* 64 bits of payload relating to the trigger packet.

Data Structure (ICMP message [11])



- (Selected) examples:

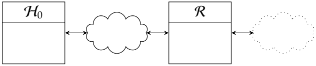
$$
\left.\begin{array}{lcl}
03_{(16)} \parallel 00_{(16)} & \mapsto & \text{destination network unreachable} \\
03_{(16)} \parallel 01_{(16)} & \mapsto & \text{destination host unreachable} \\
03_{(16)} \parallel 06_{(16)} & \mapsto & \text{destination network unknown} \\
03_{(16)} \parallel 07_{(16)} & \mapsto & \text{destination host unknown} \\
03_{(16)} \parallel 04_{(16)} & \mapsto & \text{fragmentation required, but prevented} \\
0B_{(16)} \parallel 00_{(16)} & \mapsto & \text{time exceeded (TTL)} \\
0B_{(16)} \parallel 01_{(16)} & \mapsto & \text{time exceeded (fragmentation reassembly)}
\end{array}\right\} \text{error reporting}
$$

$$
\left.\begin{array}{lcl}
09_{(16)} \parallel 00_{(16)} & \mapsto & \text{router advertisement [6]} \\
10_{(16)} \parallel 00_{(16)} & \mapsto & \text{router solicitation [6]}
\end{array}\right\} \text{network control}
$$

$$
\left.\begin{array}{lcl}
08_{(16)} \parallel 00_{(16)} & \mapsto & \text{echo request} \\
00_{(16)} \parallel 00_{(16)} & \mapsto & \text{echo response}
\end{array}\right\} \text{information retrieval}
$$

Problem #3 $\rightsquigarrow$ ICMP

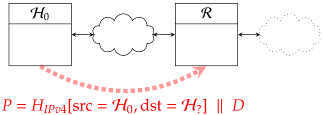
- Example: given some (inter-)connected hosts, e.g.,



what happens if $\mathcal{H}_0$ transmits a packet to an unknown host $\mathcal{H}_?$?

Problem #3 $\rightsquigarrow$ ICMP

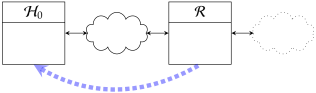
- Example: given some (inter-)connected hosts, e.g.,



what happens if $\mathcal{H}_0$ transmits a packet to an unknown host $\mathcal{H}_?$?

Problem #3 $\rightsquigarrow$ ICMP

- Example: given some (inter-)connected hosts, e.g.,

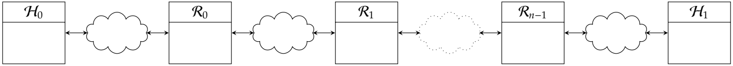


$$P' = H_{IPv4}[\text{src} = \mathcal{R}, \text{dst} = \mathcal{H}_0, \text{protocol} = \text{ICMP}] \parallel H_{ICMP}[\text{type} \parallel \text{code} = \text{host unknown}, \text{payload} = P]$$

what happens if $\mathcal{H}_0$ transmits a packet to an unknown host $\mathcal{H}_?$?

Problem #3 $\rightsquigarrow$ ICMP

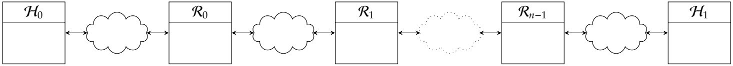
- Challenge: given some (inter-)connected hosts, e.g.,



how can $\mathcal{H}_0$ test whether $\mathcal{H}_1$ is contactable, and the RTT for communication with it?

Problem #3 $\rightsquigarrow$ ICMP

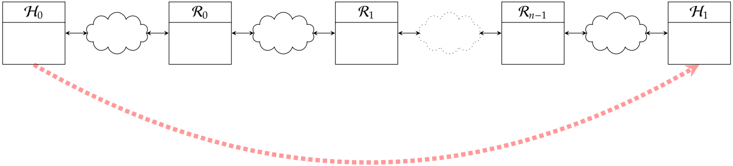
- Challenge: given some (inter-)connected hosts, e.g.,



  how can $\mathcal{H}_0$ test whether $\mathcal{H}_1$ is contactable, and the RTT for communication with it?
- Solution: `ping`.

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

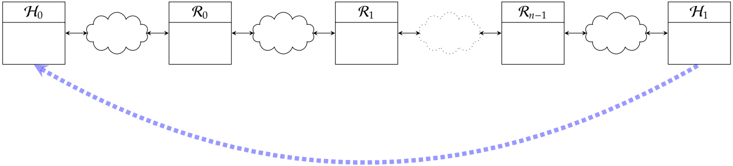


$$P = H_{IPv4}[\text{src} = \mathcal{H}_0, \text{dst} = \mathcal{H}_1, \text{protocol} = \text{ICMP}] \parallel H_{ICMP}[\text{type} \parallel \text{code} = \text{echo request}]$$

  how can $\mathcal{H}_0$ test whether $\mathcal{H}_1$ is contactable, and the RTT for communication with it?
- Solution: `ping`.

Problem #3 $\leadsto$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

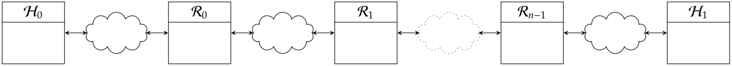


$P = H_{IPv4}[\text{src} = \mathcal{H}_1, \text{dst} = \mathcal{H}_0, \text{protocol} = \text{ICMP}] \parallel H_{ICMP}[\text{type} \parallel \text{code} = \text{echo response}]$

  how can $\mathcal{H}_0$ test whether $\mathcal{H}_1$ is contactable, and the RTT for communication with it?
- Solution: `ping`.

Problem #3 $\rightsquigarrow$ ICMP

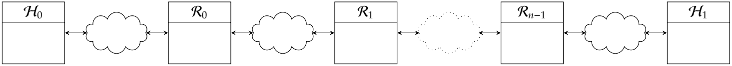
- Challenge: given some (inter-)connected hosts, e.g.,



how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?

Problem #3 ⇝ ICMP

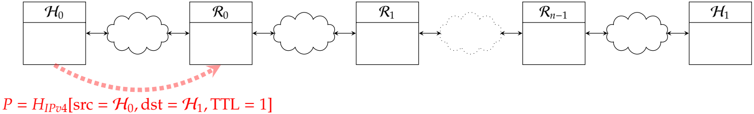
- Challenge: given some (inter-)connected hosts, e.g.,



  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature.

Problem #3 $\rightsquigarrow$ ICMP

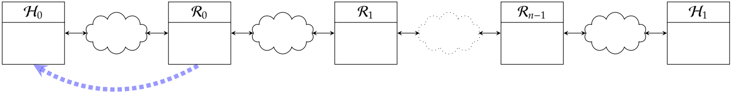
- Challenge: given some (inter-)connected hosts, e.g.,



  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature.

Problem #3 $\rightsquigarrow$ ICMP

- **Challenge**: given some (inter-)connected hosts, e.g.,



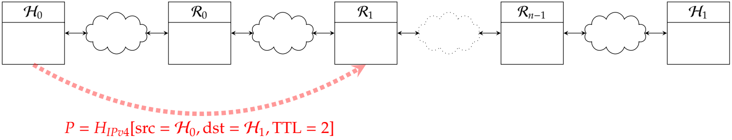
$P = H_{IPv4}[\text{src} = \mathcal{R}_0, \text{dst} = \mathcal{H}_0, \text{protocol} = \text{ICMP}] \parallel H_{ICMP}[\text{type} \parallel \text{code} = \text{TTL expired}]$

  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- **Solution**: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0.$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

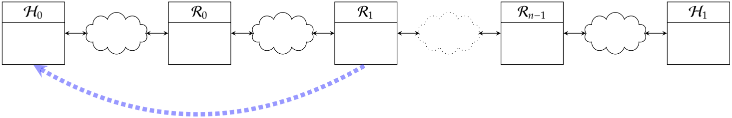


  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0.$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,



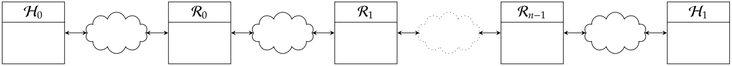
$$P = H_{IPv4}[\text{src} = \mathcal{R}_1, \text{dst} = \mathcal{H}_0, \text{protocol} = \text{ICMP}] \parallel H_{ICMP}[\text{type} \parallel \text{code} = \text{TTL expired}]$$

  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1.$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

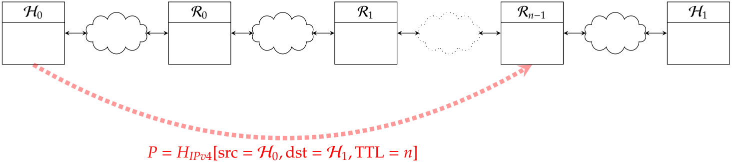


  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1 \rightarrow \cdots .$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

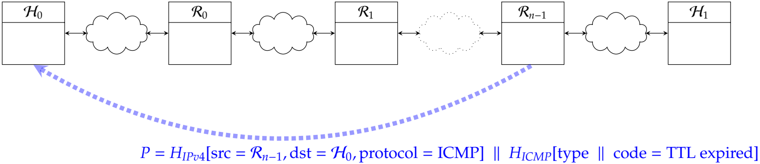


  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1 \rightarrow \cdots.$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

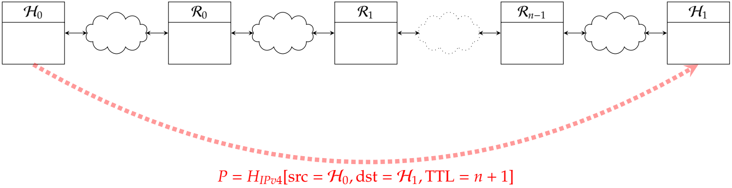


  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1 \rightarrow \cdots \rightarrow \mathcal{R}_{n-1}.$$

Problem #3 $\leadsto$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,

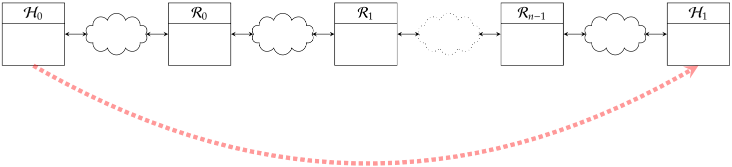


  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1 \rightarrow \cdots \rightarrow \mathcal{R}_{n-1}.$$

Problem #3 $\rightsquigarrow$ ICMP

- Challenge: given some (inter-)connected hosts, e.g.,



  how can $\mathcal{H}_0$ discover the route packets take when transmitted to $\mathcal{H}_1$?
- Solution: `traceroute`, which (ab)uses the TTL feature, st. the path discovered is

$$\mathcal{H}_0 \rightarrow \mathcal{R}_0 \rightarrow \mathcal{R}_1 \rightarrow \cdots \rightarrow \mathcal{R}_{n-1} \rightarrow \mathcal{H}_1.$$

Conclusions

- Take away points:
  - The "glue" protocols outlined here solve issues that stem from practical deployment and use.
  - Applications such as `traceroute` are, in a sense, a by-product of flexibility in protocols such as ICMP.
  - There *are* scenarios, e.g., ARP, where host can disrupt (either maliciously, or by accident) normal operation ...
  - ... a set of requirements [5, 4] mandates what Internet hosts *must* implement.
  - DHCP and ARP are instances of more general **discovery protocols**; in such examples, broadcast capability is often an advantage.

References

[1] Wikipedia: Address Resolution Protocol (ARP).
http://en.wikipedia.org/wiki/Address_Resolution_Protocol.

[2] Wikipedia: Dynamic Host Configuration Protocol (DHCP).
http://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol.

[3] Wikipedia: Internet Control Message Protocol (ICMP).
http://en.wikipedia.org/wiki/Internet_Control_Message_Protocol.

[4] R. Braden.
Requirements for Internet hosts – application and support.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1123, 1989.
http://tools.ietf.org/html/rfc1123.

[5] R. Braden.
Requirements for Internet hosts – communication layers.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1122, 1989.
http://tools.ietf.org/html/rfc1122.

[6] S. Deering.
ICMP router discovery messages.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1256, 1991.
http://tools.ietf.org/html/rfc1256.

References

[7] R. Droms.
Dynamic Host Configuration Protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1541, 1993.
http://tools.ietf.org/html/rfc1541.

[8] R. Finlayson, T. Mann, J. Mogul, and M. Theimer.
A Reverse Address Resolution Protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 903, 1984.
http://tools.ietf.org/html/rfc903.

[9] J. Gilmore.
Bootstrap protocol (BOOTP).
Internet Engineering Task Force (IETF) Request for Comments (RFC) 951, 1985.
http://tools.ietf.org/html/rfc951.

[10] D.C. Plummer.
An Ethernet address resolution protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 826, 1982.
http://tools.ietf.org/html/rfc826.

[11] J. Postel.
Internet Control Message Protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 792, 1981.
http://tools.ietf.org/html/rfc792.