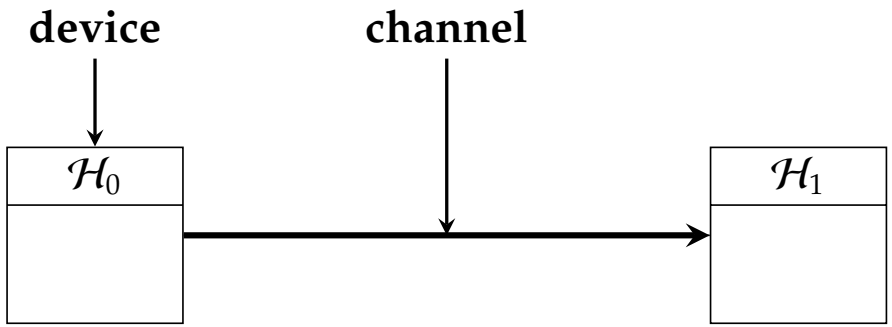
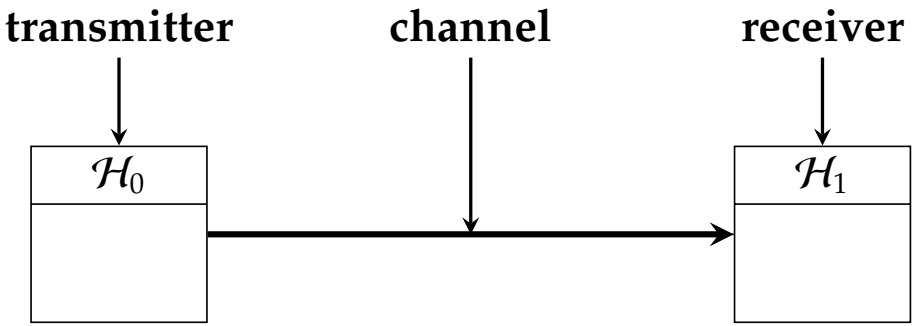
- We can *already* form a simple **point-to-point** communication channel



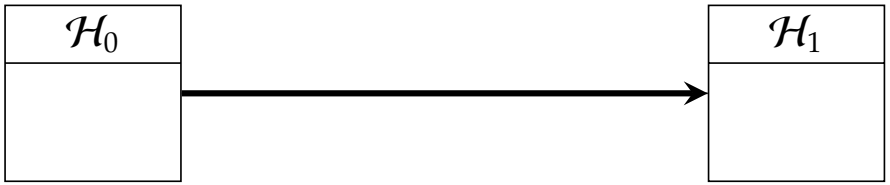
using TIA-232-F.

- We can *already* form a simple **point-to-point** communication channel



using TIA-232-F.

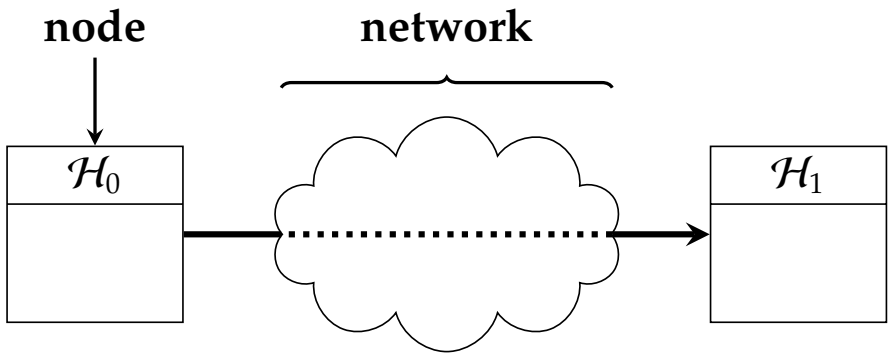
- We can *already* form a simple **point-to-point** communication channel



using TIA-232-F, st.

- Good:
  - modular wrt. communication medium and protocol,
  - uses standardised components,
  - ...
- Bad:
  - well defined, but quite limited functionality,
  - the organisation of components is fixed,
  - there are limits wrt. physical locality,
  - ...

- We can *already* form a simple **point-to-point** communication channel



using TIA-232-F.

- Challenge: expand our remit to use of a **computer network** ...

- ... *how*?

- ... *how*?
  1. Requirements: what do we expect from a network?
     - supports a high degree of connectivity,
     - allows inter-operation between heterogeneous components,
     - uses appropriate level of abstraction to provide useful functionality,
     - satisfies any relevant quality metrics (e.g., efficiency, reliability),
     - can be (dynamically) scaled wrt. components and usage.

- ... *how*?

2. Architecture: how should we approach the design of a network?
   - (physical or logical) **topology**, i.e., the structure of components in the network, plus
   - control i.e., how do we use those components to communicate, and
   - standardisation, i.e., how do we ensure components which communicate can **inter-operate**.

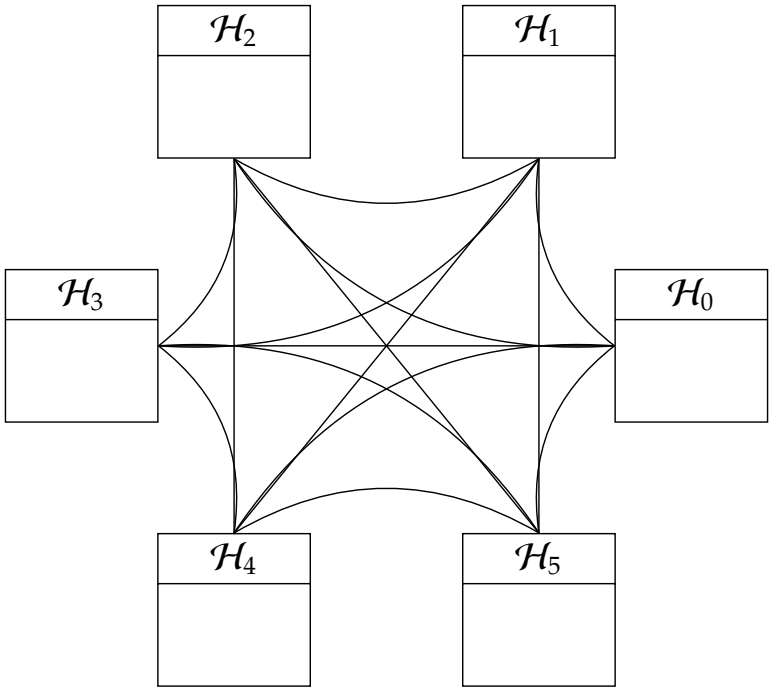
Problem #1: topology (1)

- Idea #1: *fully*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S = \{0, 1, \ldots n-1\} \setminus \{i\}$.

- Note that access to a given channel may be
  1. **dedicated**, or
  2. **shared**, implying a need to control access but also the posibility to **broadcast** to *multiple* receiving nodes.

Problem #1: topology (1)

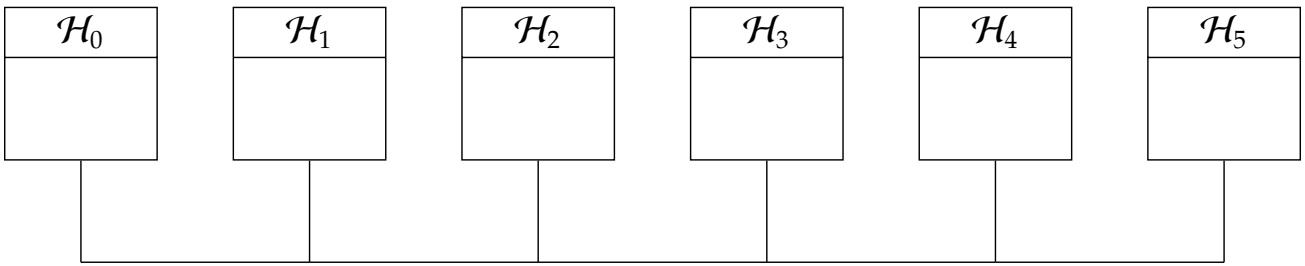
- Idea #1: *fully*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S = \{0, 1, \ldots n-1\} \setminus \{i\}$.
- Example: a **mesh topology**, i.e.,



- Note that access to a given channel may be
  1. **dedicated**, or
  2. **shared**, implying a need to control access but also the posibility to **broadcast** to *multiple* receiving nodes.

Problem #1: topology (1)

- Idea #1: *fully*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S = \{0, 1, \ldots n-1\} \setminus \{i\}$.
- Example: a **bus topology**, i.e.,



- Note that access to a given channel may be
  1. **dedicated**, or
  2. **shared**, implying a need to control access but also the posibility to **broadcast** to *multiple* receiving nodes.

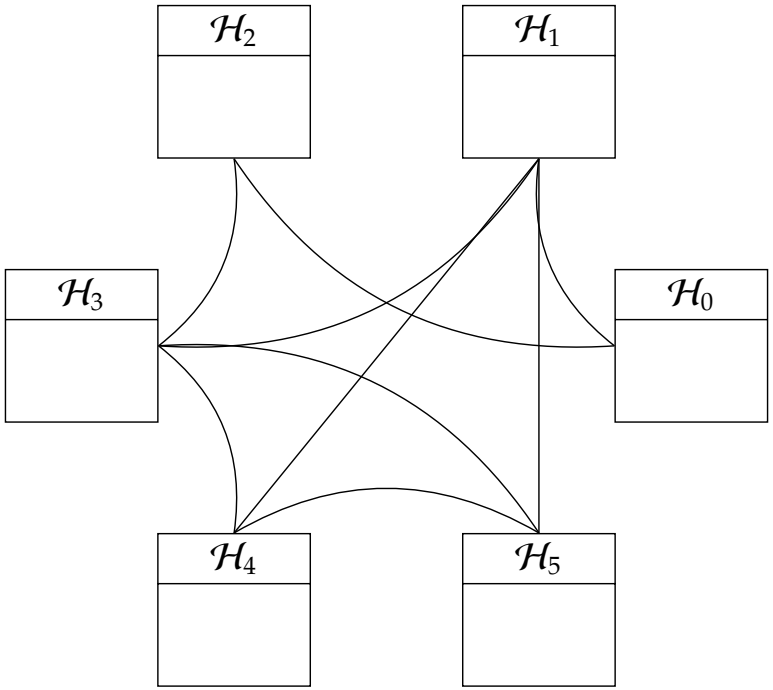
Problem #1: topology (2)

- Idea #2: *partially*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S \subset \{0, 1, \ldots n-1\} \setminus \{i\}$.

- Note that connectivity may now be either
  1. **direct**, or
  2. **indirect**, implying a need for intermediate **hops**, e.g., as realised via **store-and-forward** by (intermediate) switching nodes.

Problem #1: topology (2)

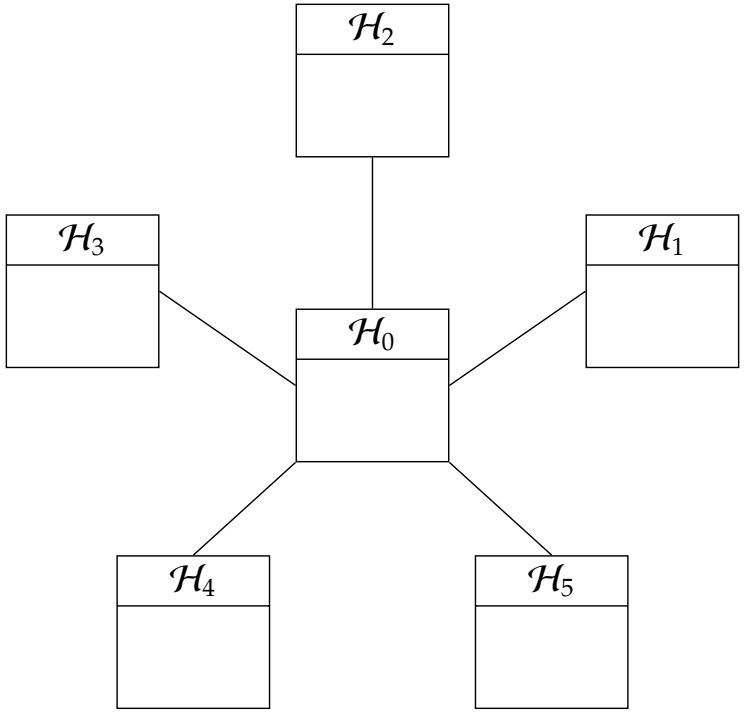
- Idea #2: *partially*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S \subset \{0, 1, \ldots n-1\} \setminus \{i\}$.
- Example: a **mesh topology**, i.e.,



- Note that connectivity may now be either
  1. **direct**, or
  2. **indirect**, implying a need for intermediate **hops**, e.g., as realised via **store-and-forward** by (intermediate) switching nodes.

Problem #1: topology (2)

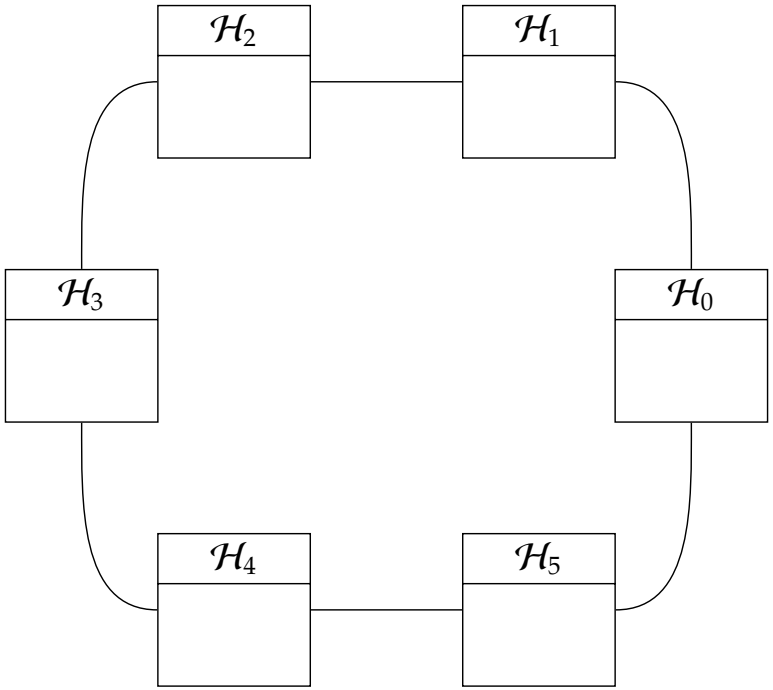
- Idea #2: *partially*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S \subset \{0, 1, \ldots n-1\} \setminus \{i\}$.
- Example: a **star topology**, i.e.,



- Note that connectivity may now be either
  1. **direct**, or
  2. **indirect**, implying a need for intermediate **hops**, e.g., as realised via **store-and-forward** by (intermediate) switching nodes.

Problem #1: topology (2)

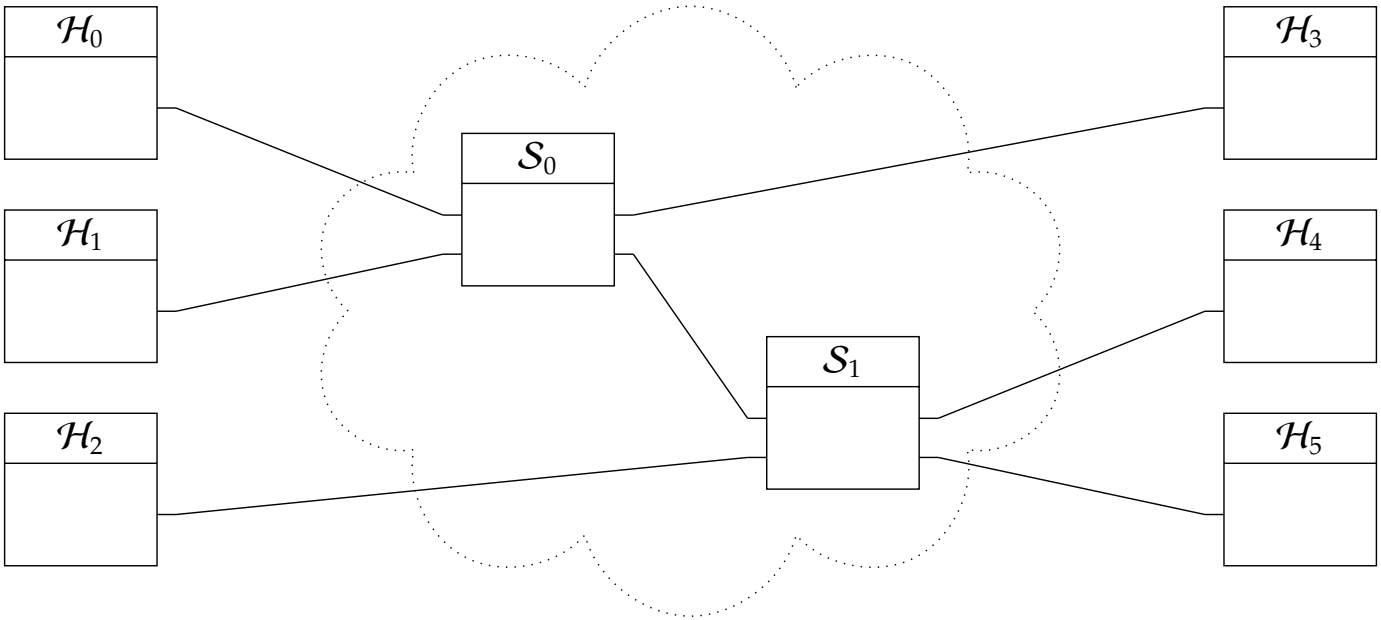
- Idea #2: *partially*-connected, st. $\mathcal{H}_i$ is connected to $\mathcal{H}_j$ for $j \in S \subset \{0, 1, \ldots n-1\} \setminus \{i\}$.
- Example: a **ring topology**, i.e.,



- Note that connectivity may now be either
  1. **direct**, or
  2. **indirect**, implying a need for intermediate **hops**, e.g., as realised via **store-and-forward** by (intermediate) switching nodes.

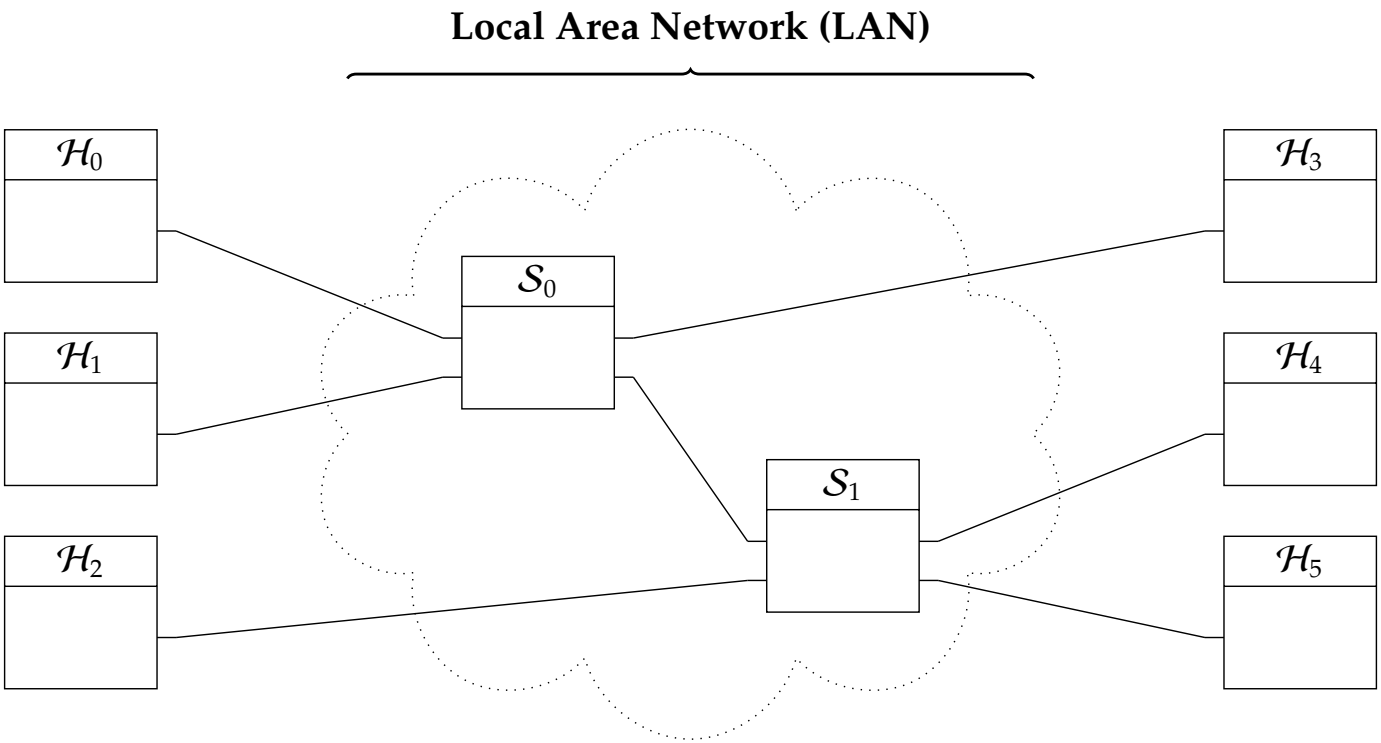
Problem #1: topology (3)

- We can *generalise* partially connected network topologies



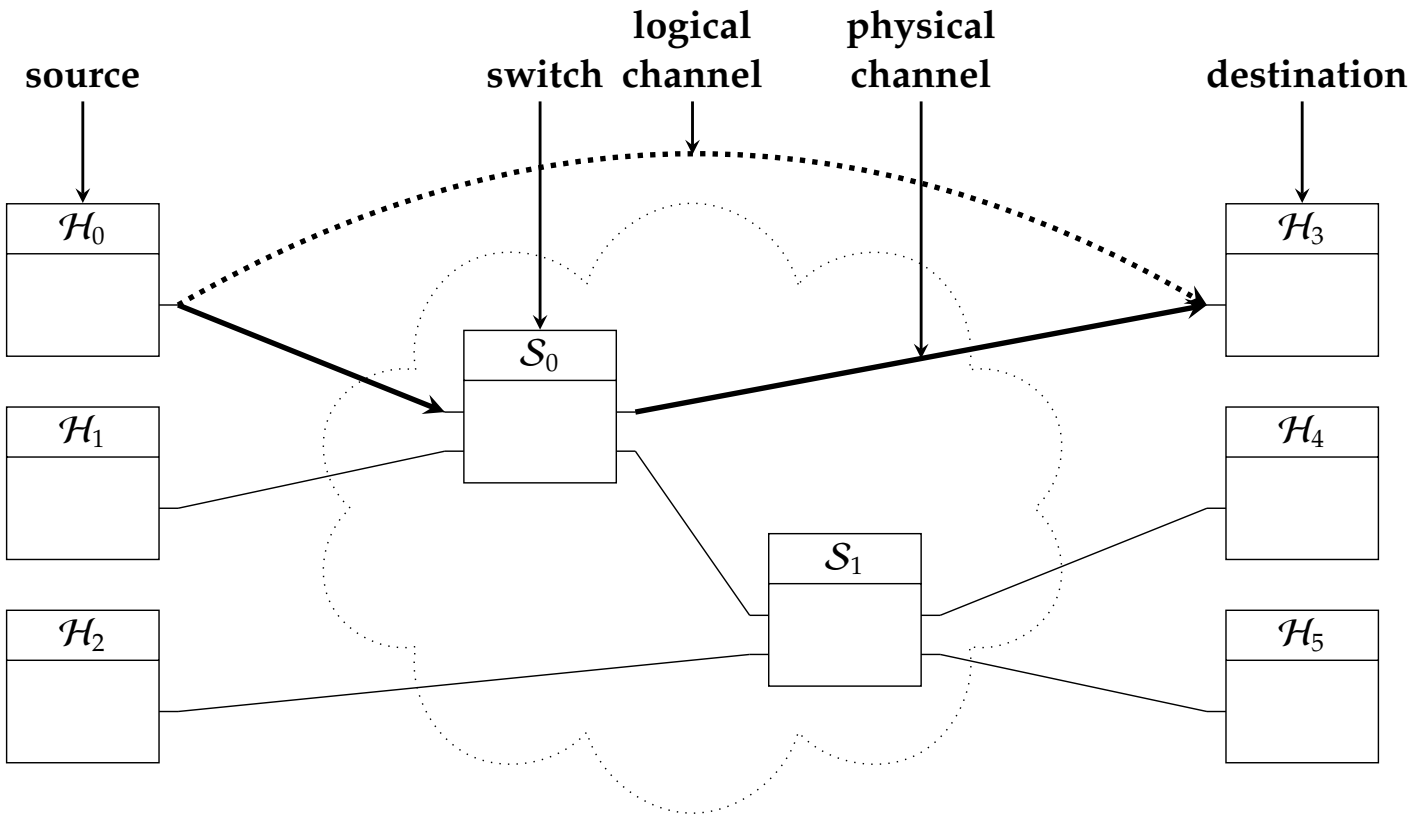
Problem #1: topology (3)

- We can *generalise* partially connected network topologies



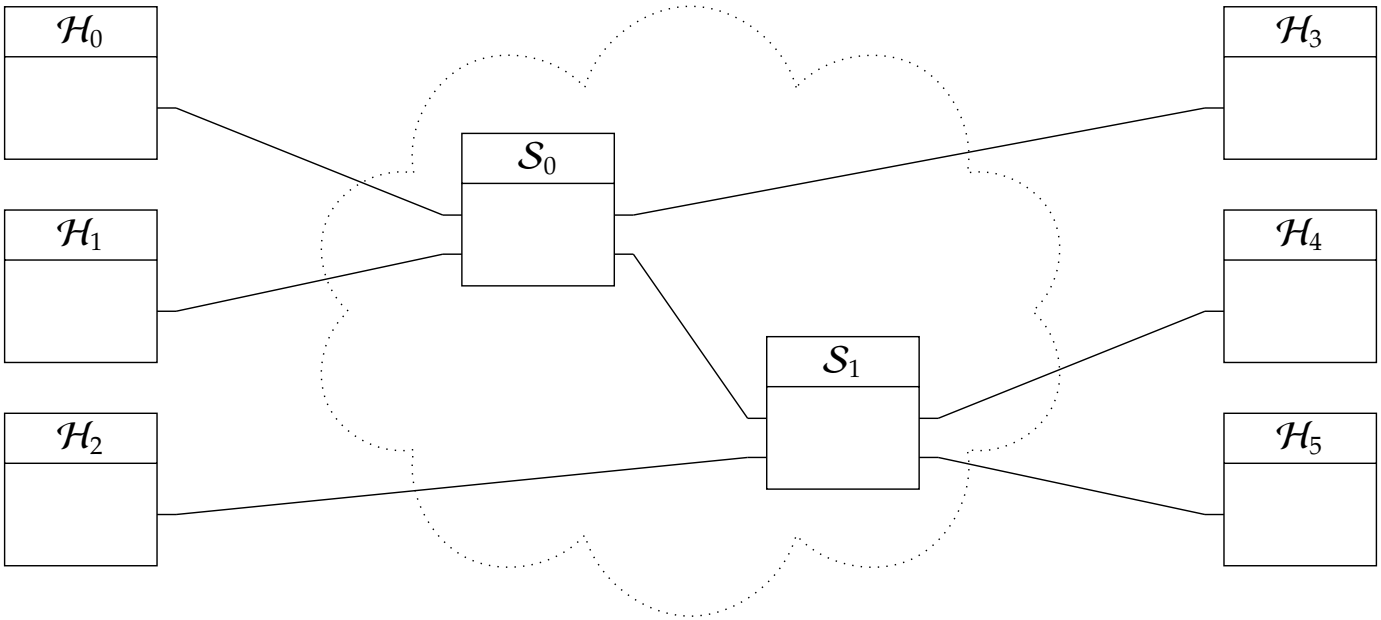
Problem #1: topology (3)

- We can *generalise* partially connected network topologies



Problem #1: topology (3)

- We can *generalise* partially connected network topologies

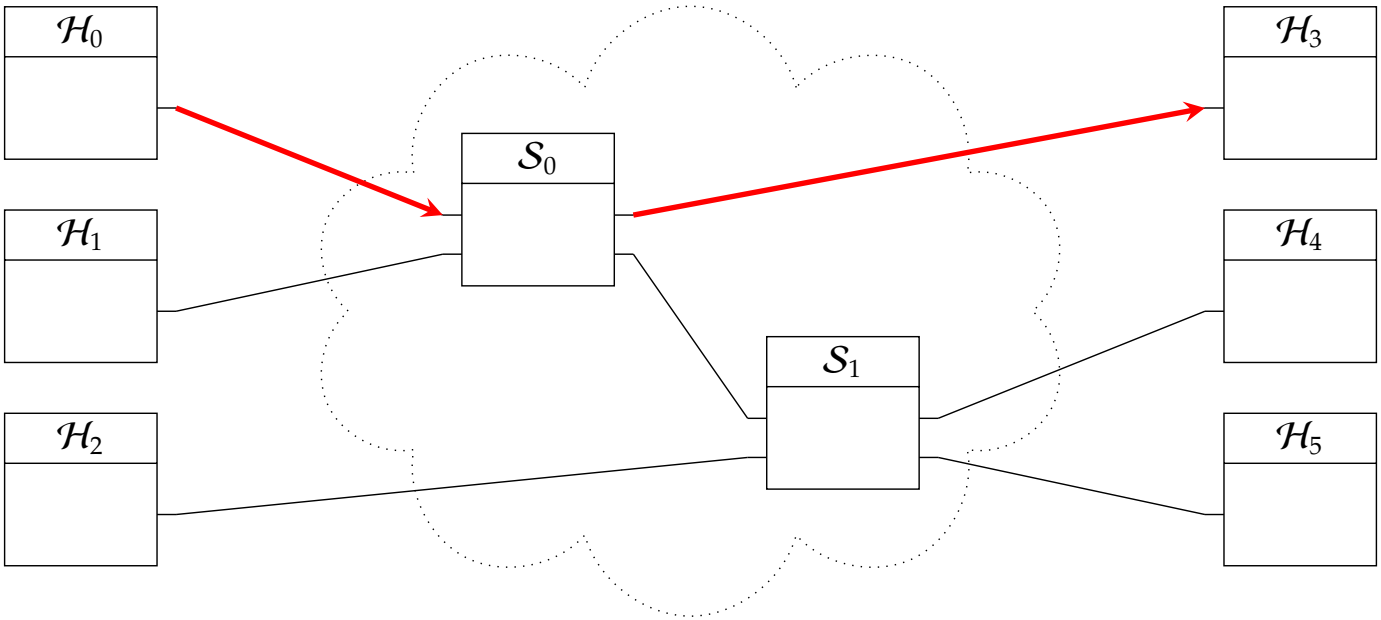


leading to the concepts of

- **circuit switching** and
- **packet switching** (to support connection-based or connection-less channels).

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

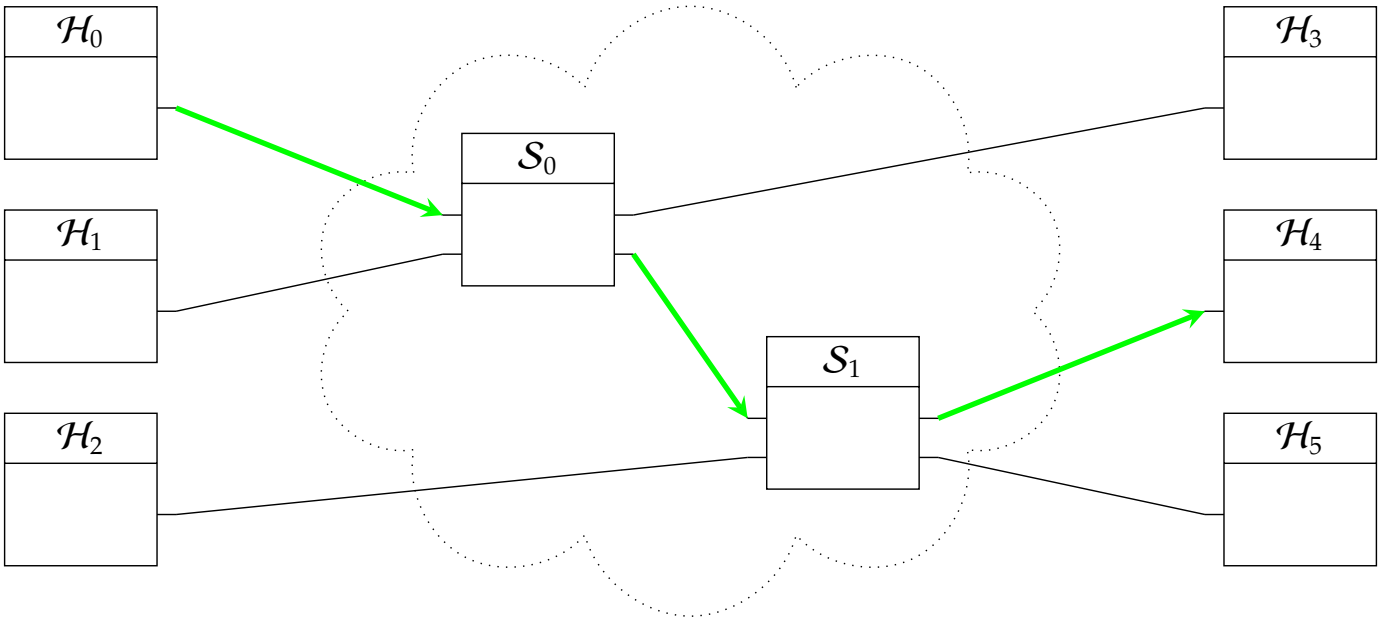


leading to the concepts of

- **circuit switching** and
- **packet switching** (to support connection-based or connection-less channels).

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

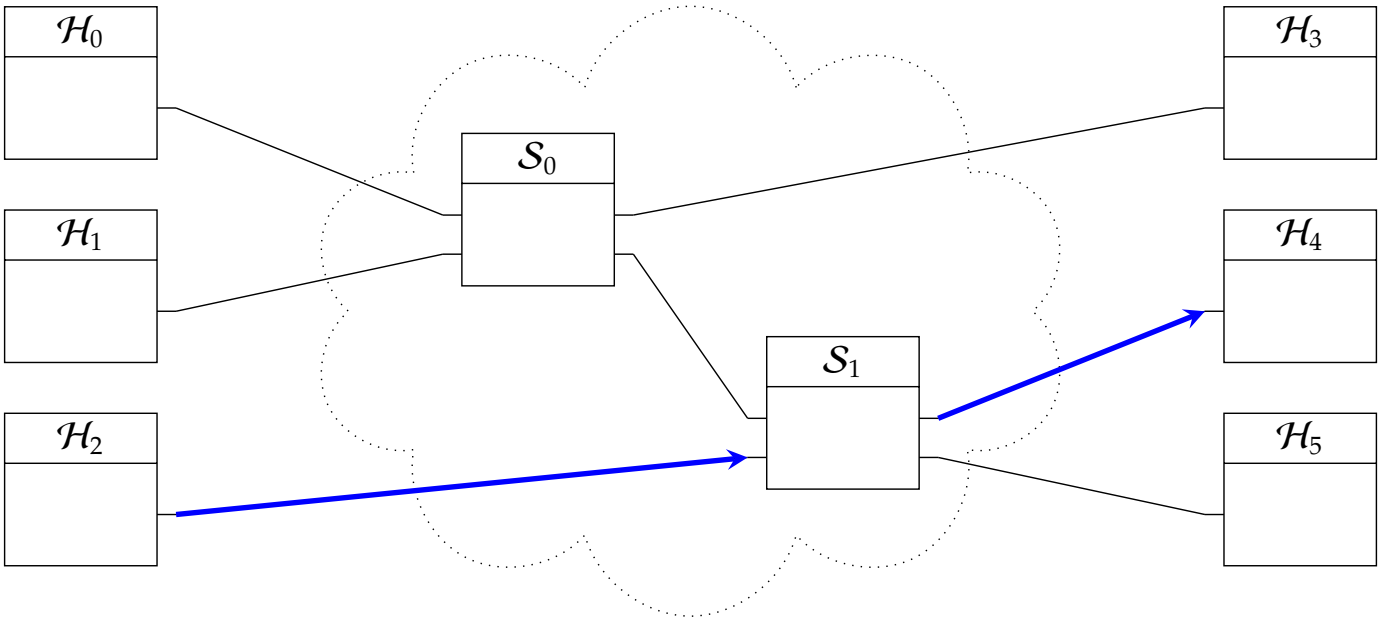


leading to the concepts of

- **circuit switching** and
- **packet switching** (to support connection-based or connection-less channels).

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

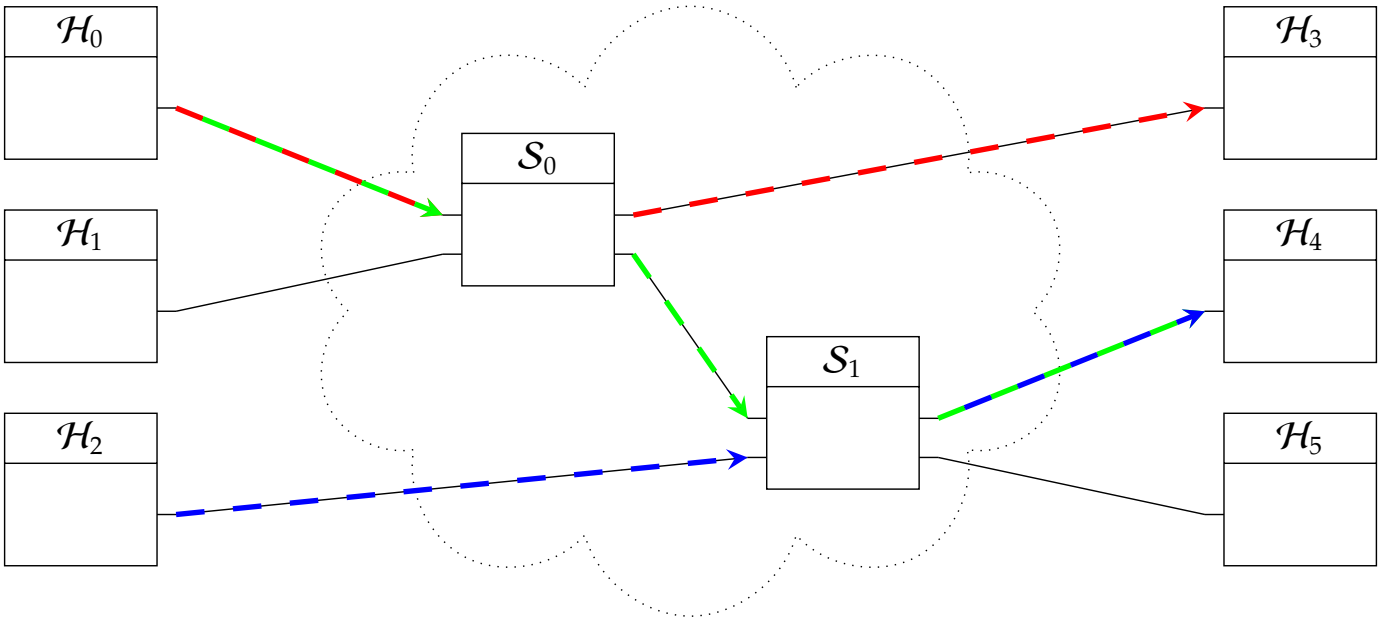


leading to the concepts of

- **circuit switching** and
- **packet switching** (to support connection-based or connection-less channels).

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

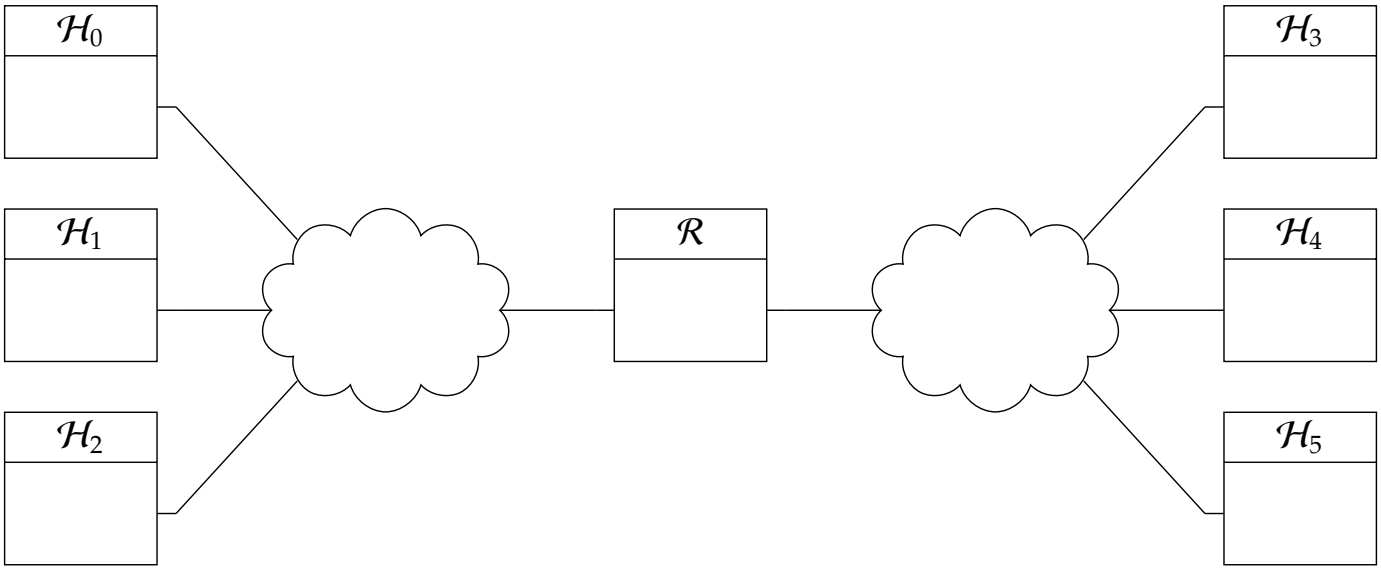


leading to the concepts of

- **circuit switching** and
- **packet switching** (to support connection-based or connection-less channels).

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

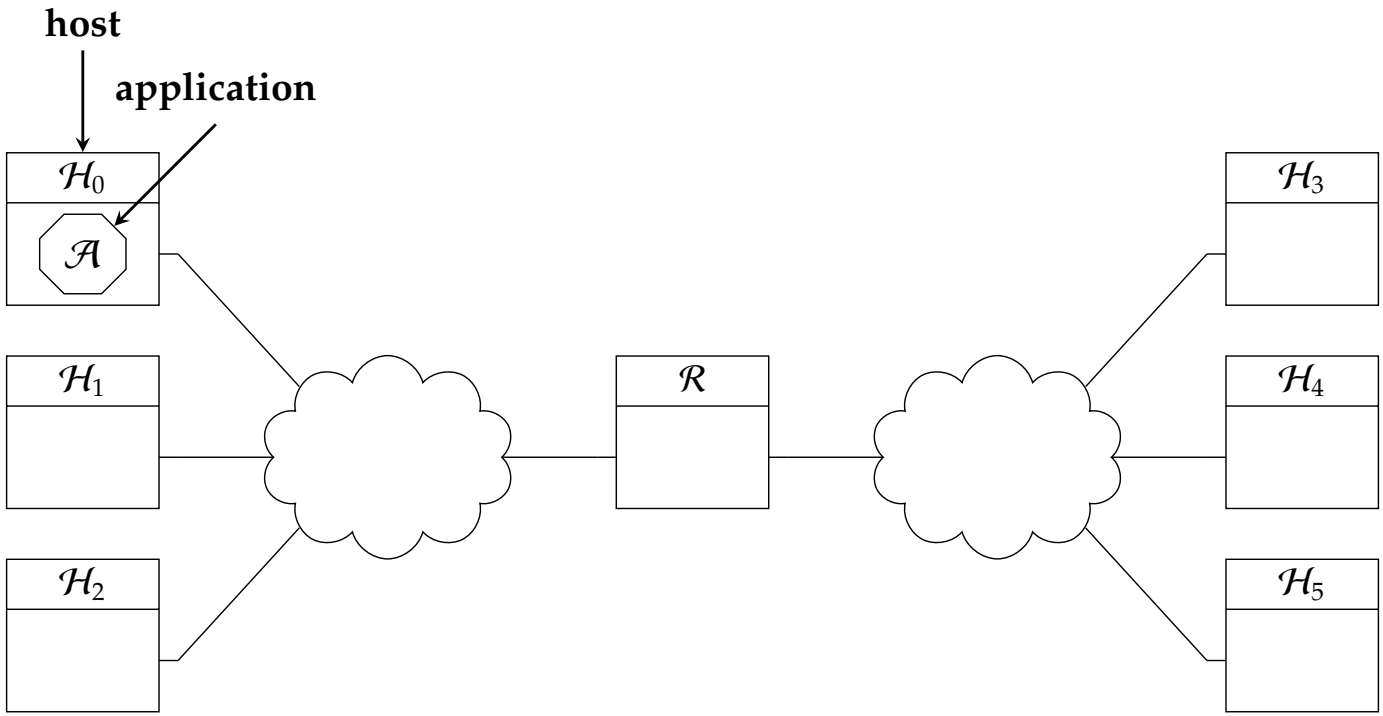


then, finally, noting

- a network provides connectivity between nodes, while
- an *inter*-network connects networks themselves together.

Problem #1: topology (3)

- We can *generalise* partially connected network topologies

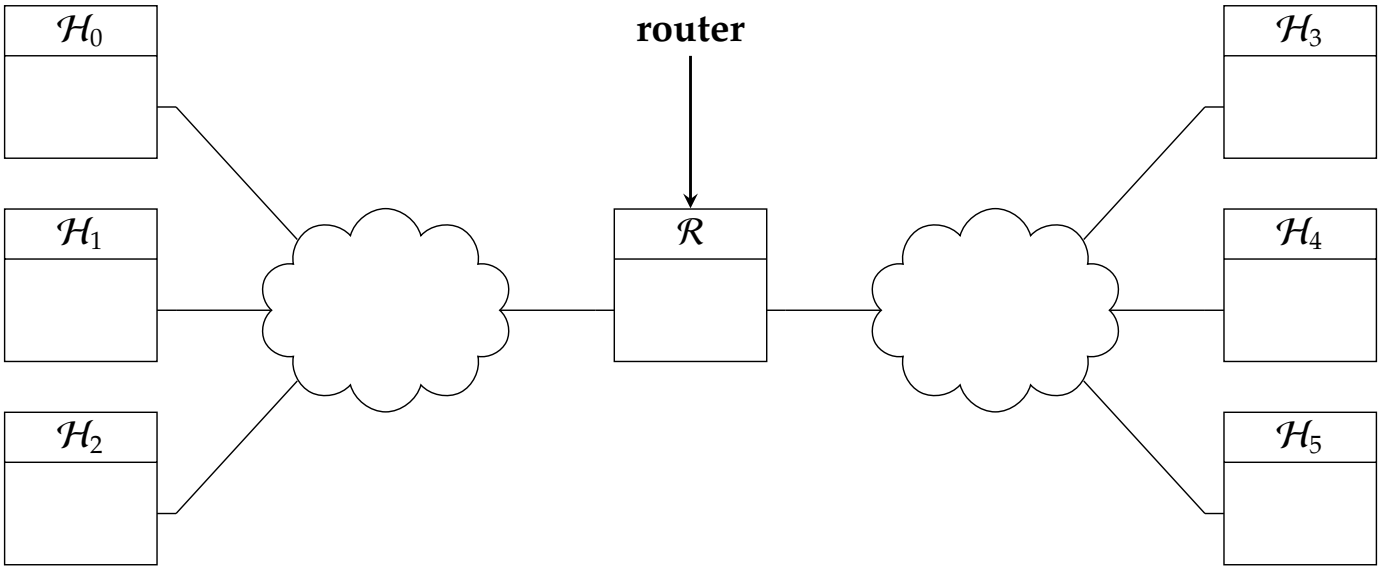


then, finally, noting

- a network provides connectivity between nodes, while
- an *inter*-network connects networks themselves together.

Problem #1: topology (3)

- We can *generalise* partially connected network topologies



then, finally, noting

- a network provides connectivity between nodes, while
- an *inter*-network connects networks themselves together.

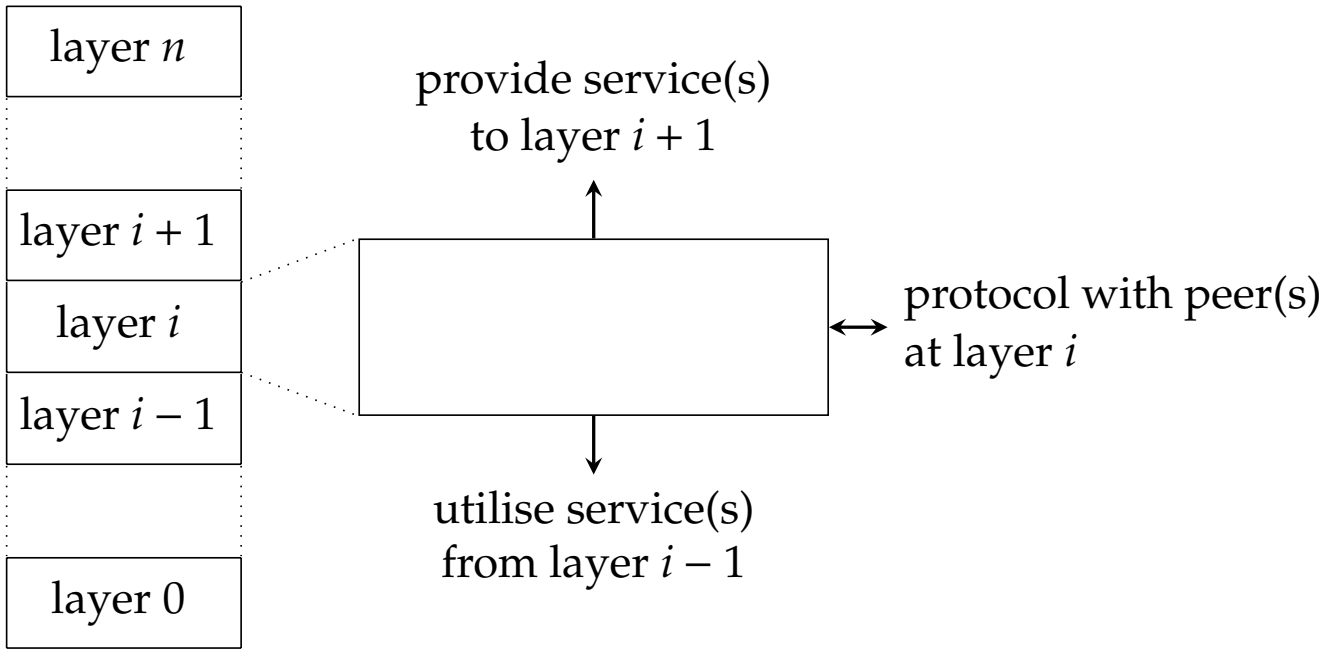
Problem #1: topology (6)

- An inter-network architecture seems a good approach!
  - Good:
    - scalable by virtue of a flexible and modular design, *and*
    - offers resilience against failure.
  - Bad: we need to cope with the fact
    - ideally we support a multiplicity of communication mediums and protocols, and share access to both,
    - each host needs a globally unique **address**, and
    - router(s) need to know or discover how to communicate (or **route**) data from one host to another

    the difficulty of which is enhanced by a need to avoid impact on scalability (e.g., avoid centralised solutions).

Problem #2: control (1)

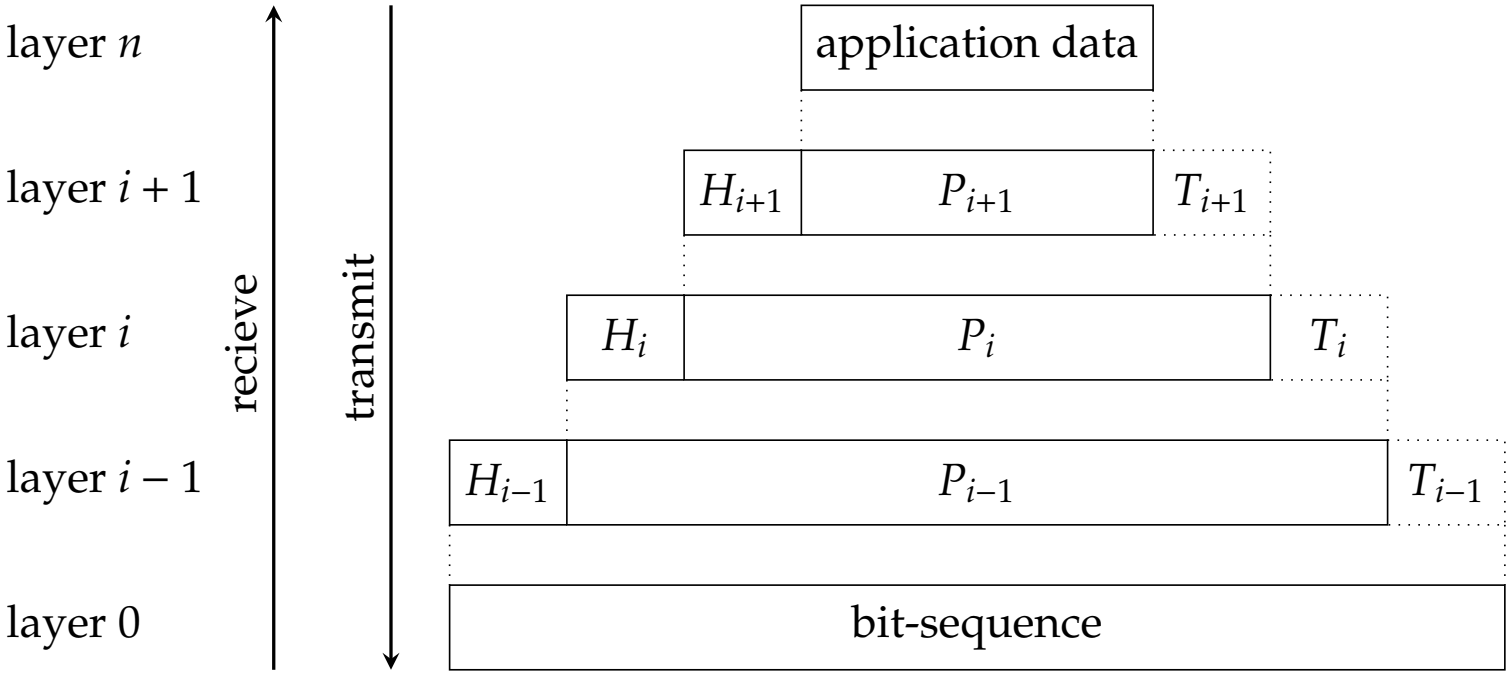
- Step #1: rather than use one monolithic protocol, decompose control, e.g.,



and hence provide various advantages stemming from modularity ...

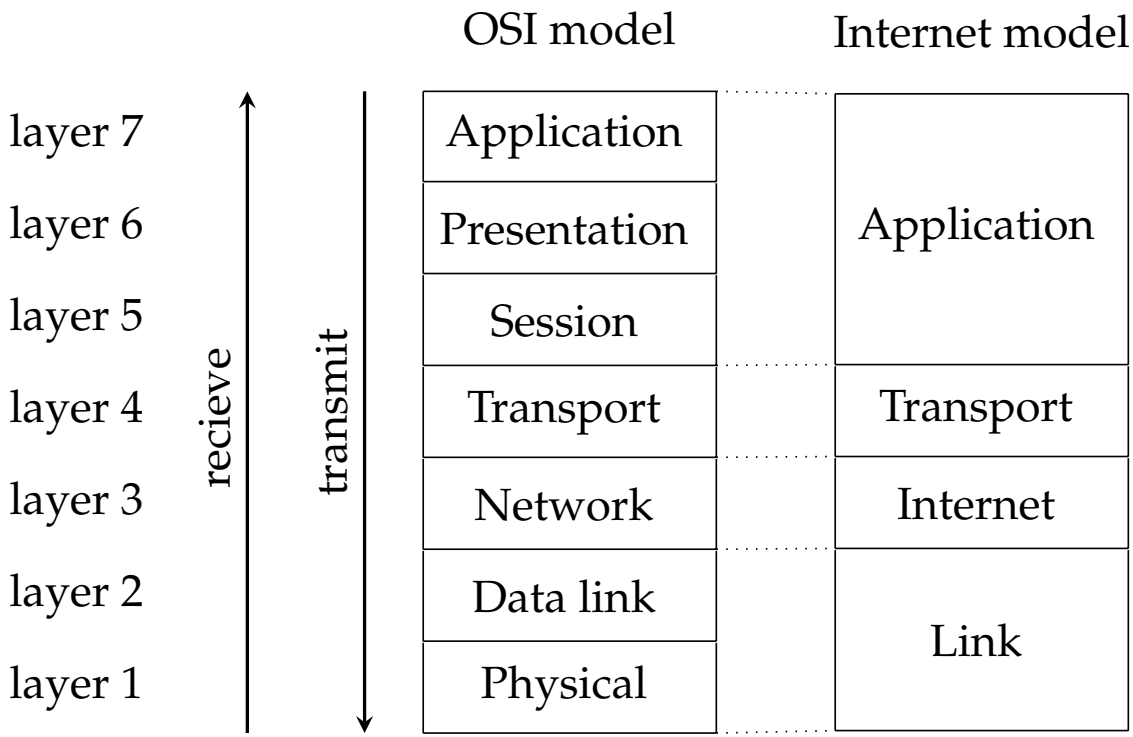
Problem #2: control (2)

- ... then support inter-layer (peer or otherwise) communication using **encapsulation**:



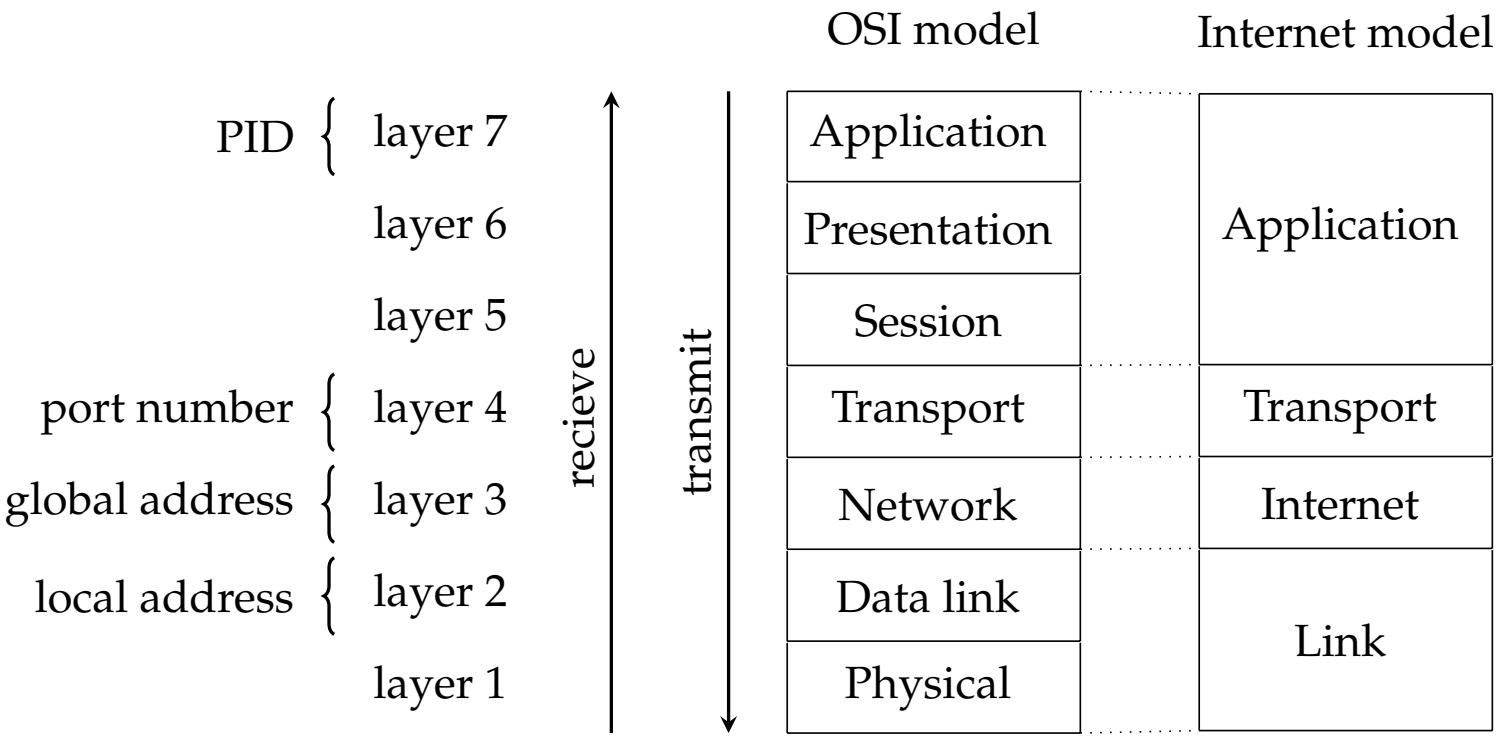
Problem #2: control (3)

- Step #2: specify (abstract) layers we need, e.g.,



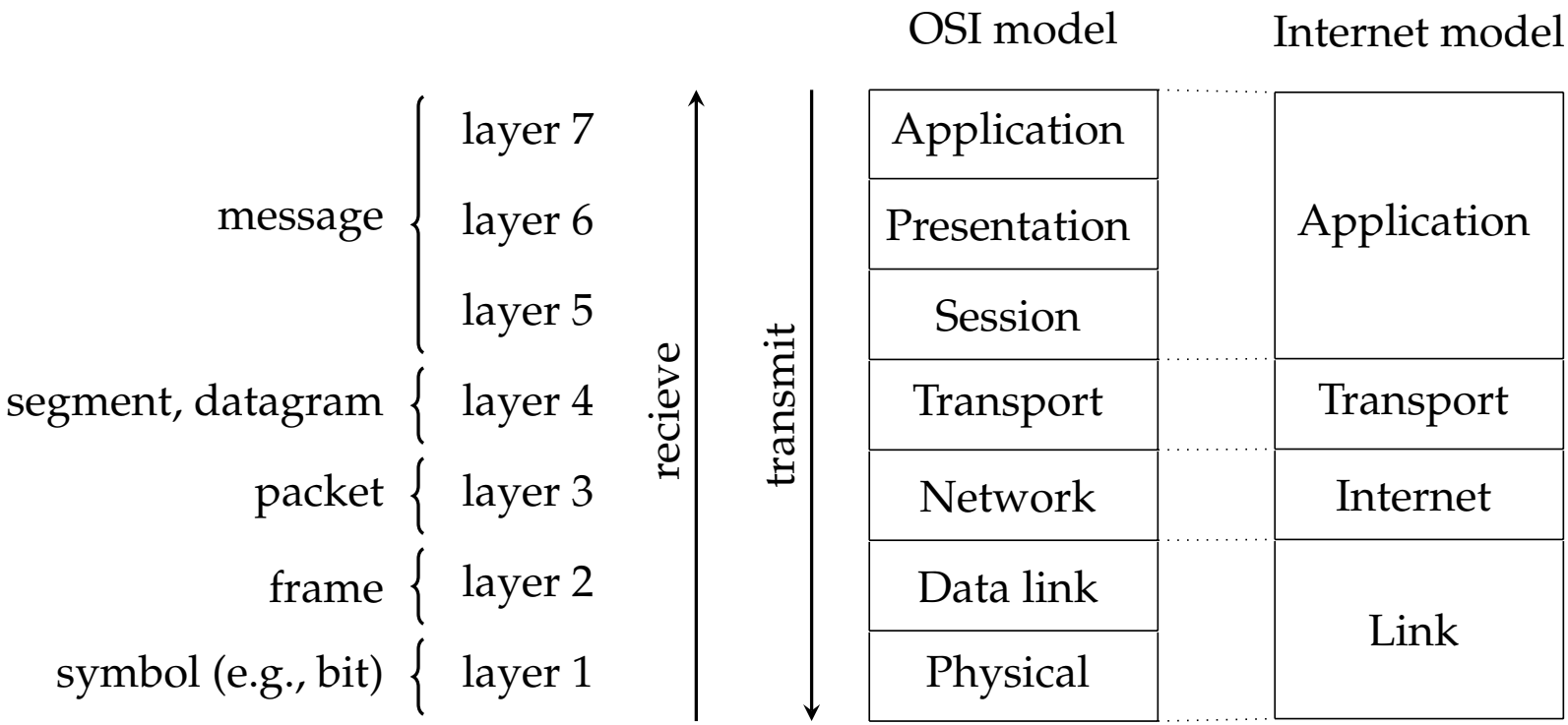
Problem #2: control (3)

- Step #2: specify (abstract) layers we need, e.g.,



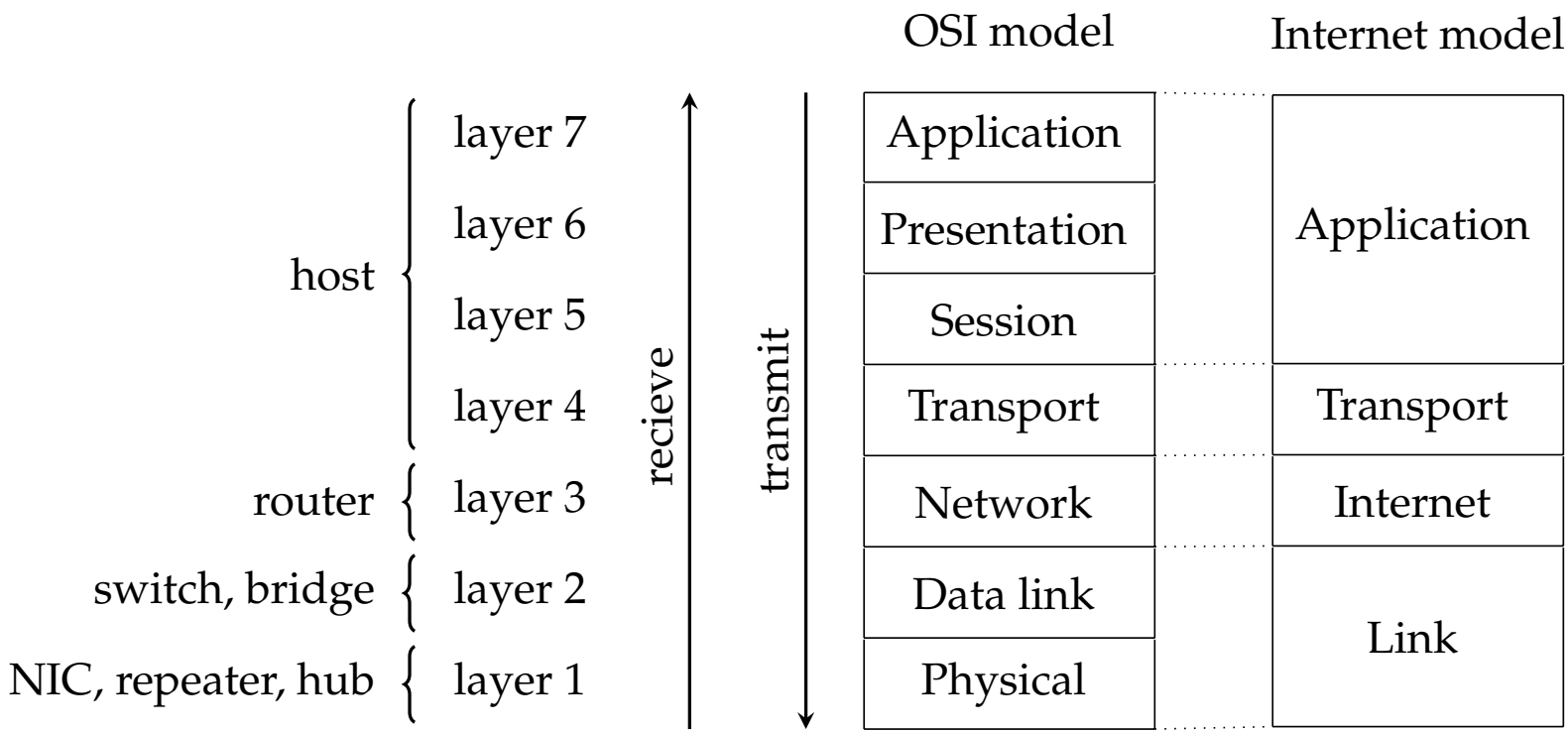
Problem #2: control (3)

- Step #2: specify (abstract) layers we need, e.g.,



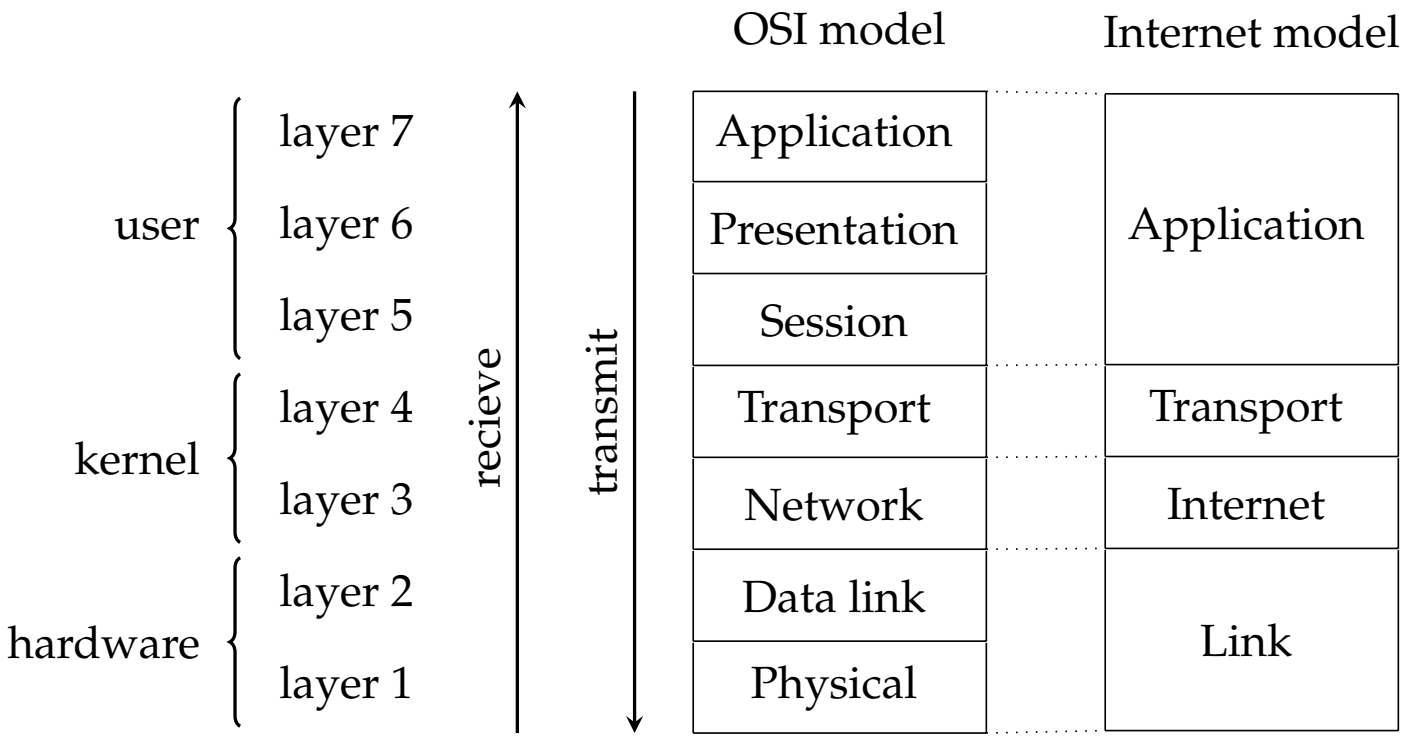
Problem #2: control (3)

- Step #2: specify (abstract) layers we need, e.g.,



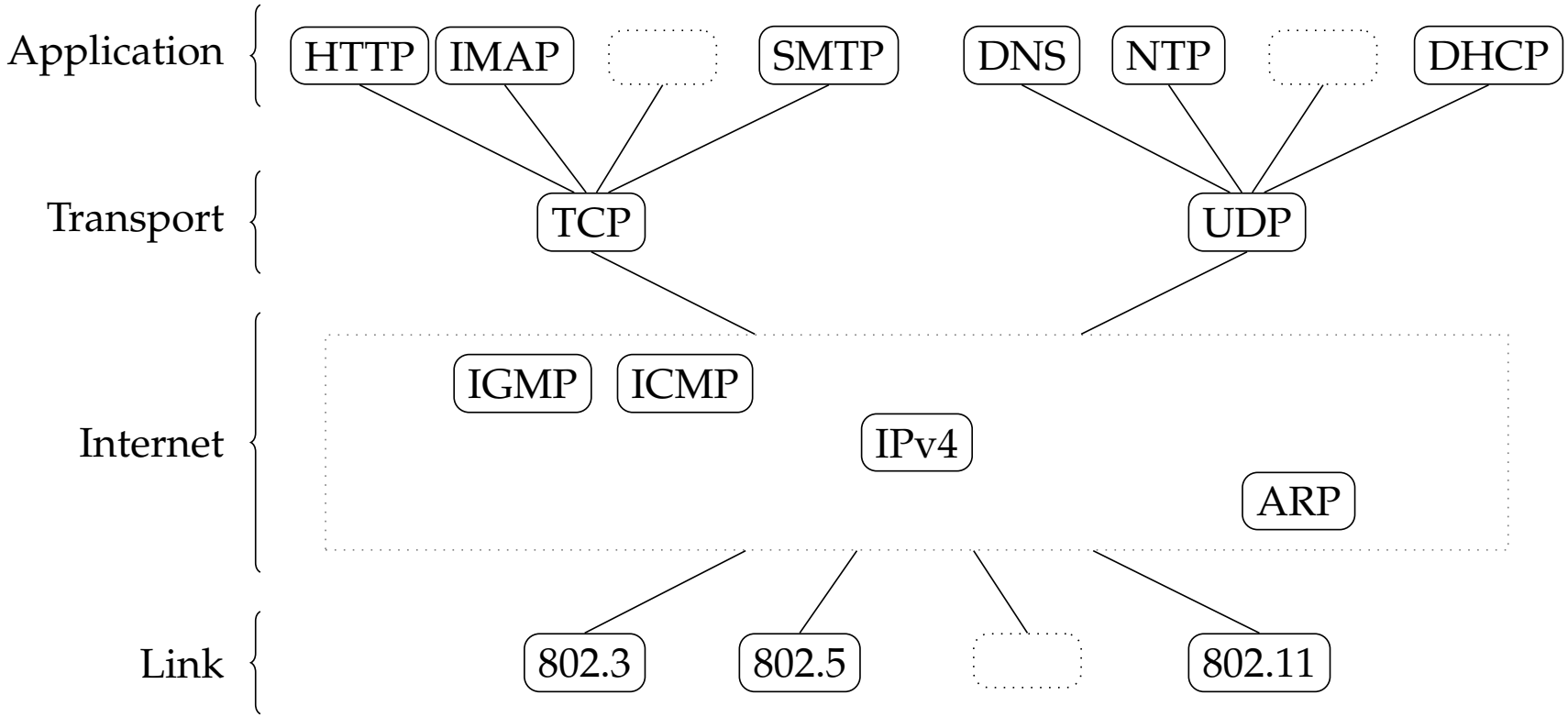
Problem #2: control (3)

- Step #2: specify (abstract) layers we need, e.g.,



Problem #2: control (4)

- Step #3: populate layers with (concrete) protocols, e.g.,

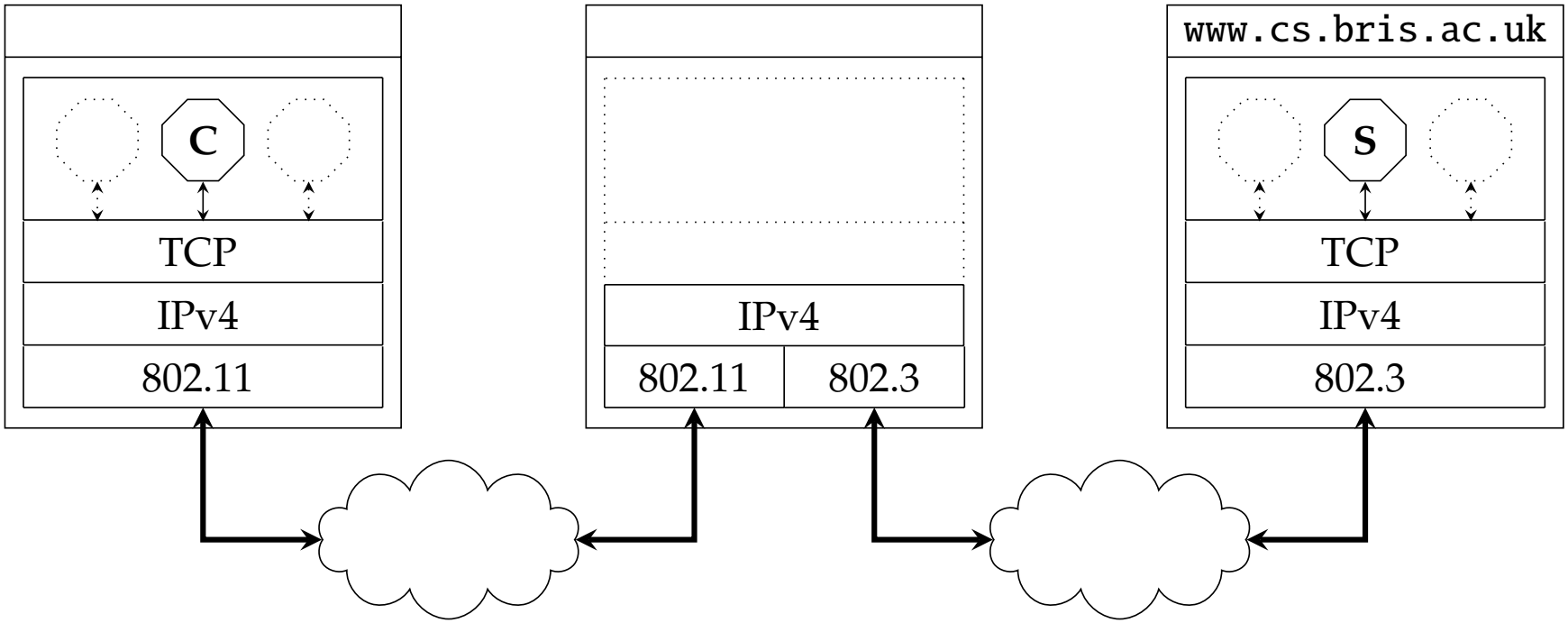


noting that

- a **protocol graph** is an abstract description of how protocols fit into the layered model, whereas
- a **protocol stack** is a concrete implementation of *one* top-to-bottom combination of protocols.

Conclusions

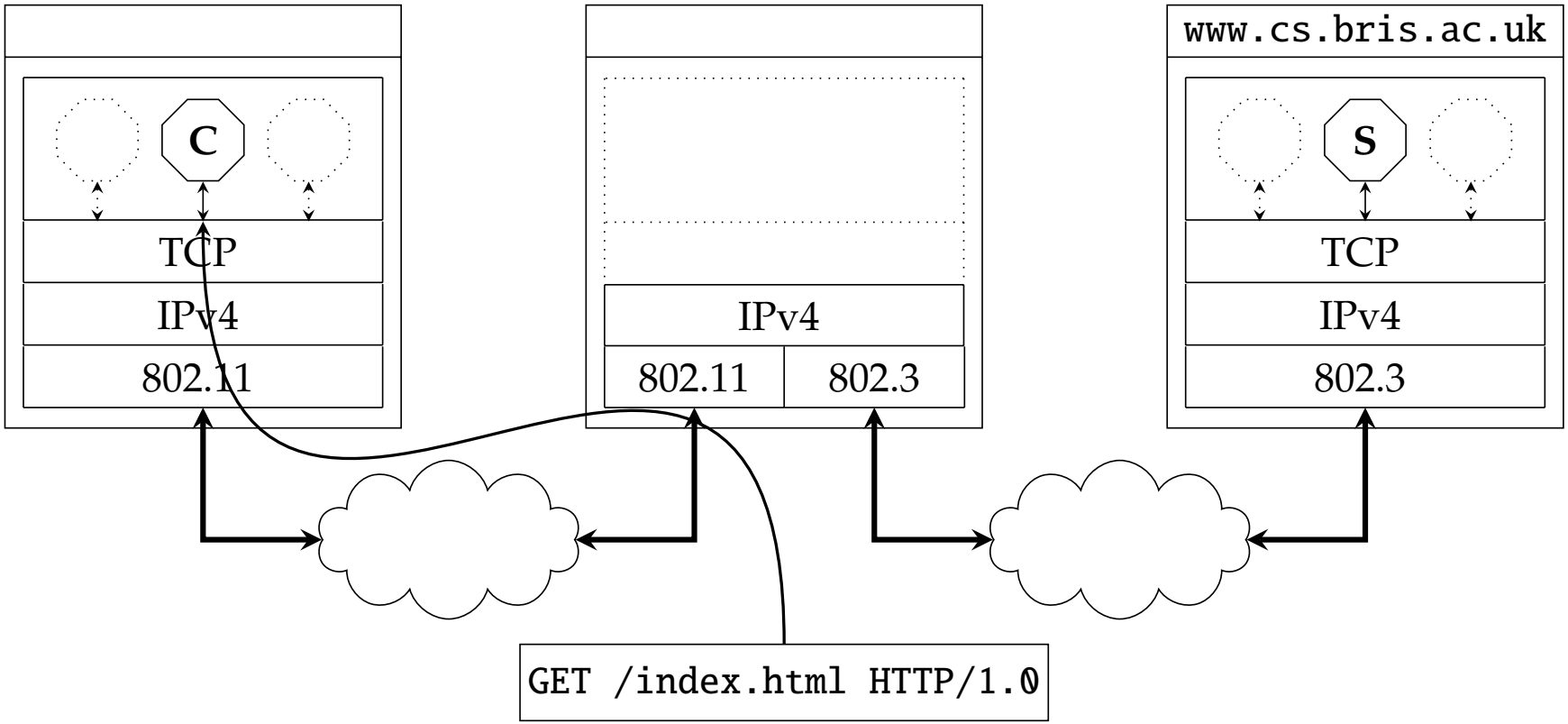
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

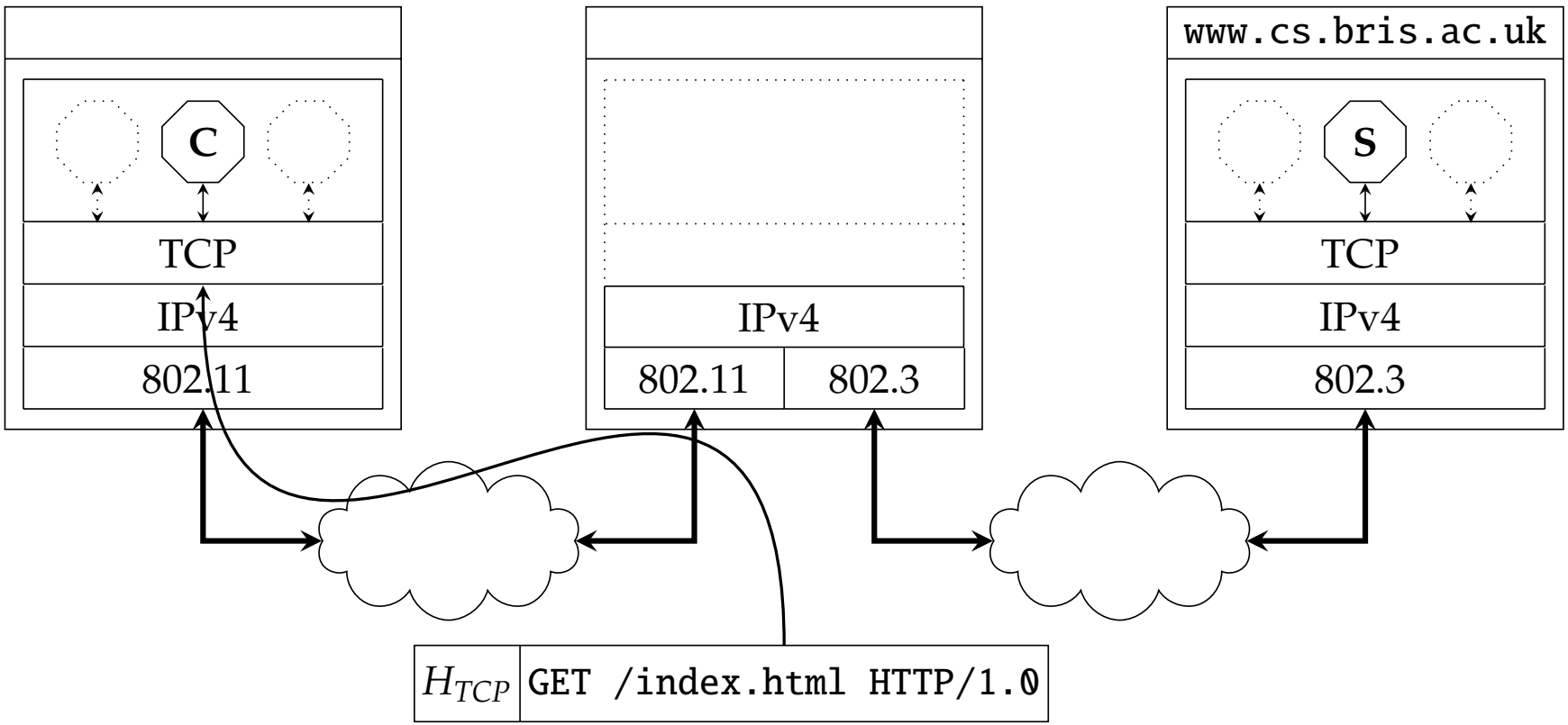
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

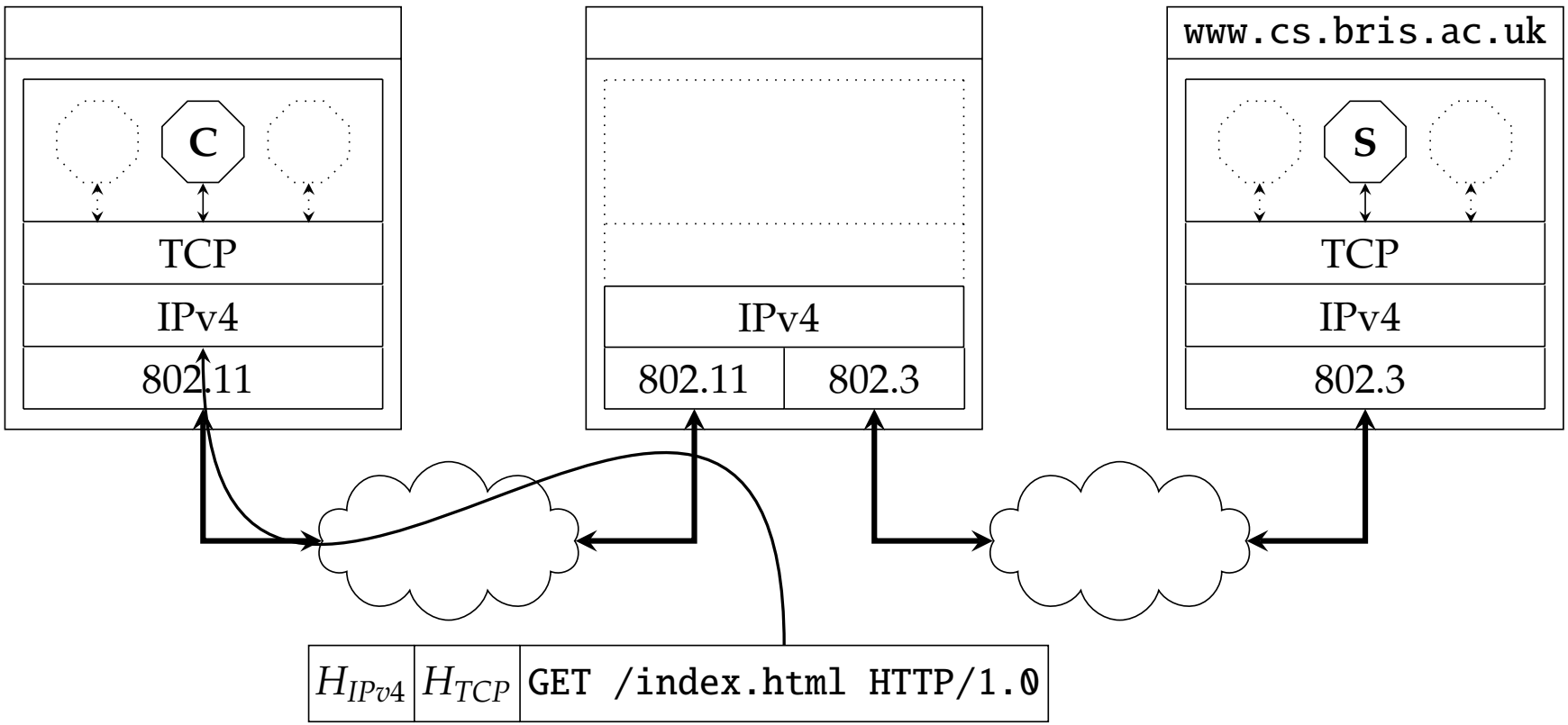
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

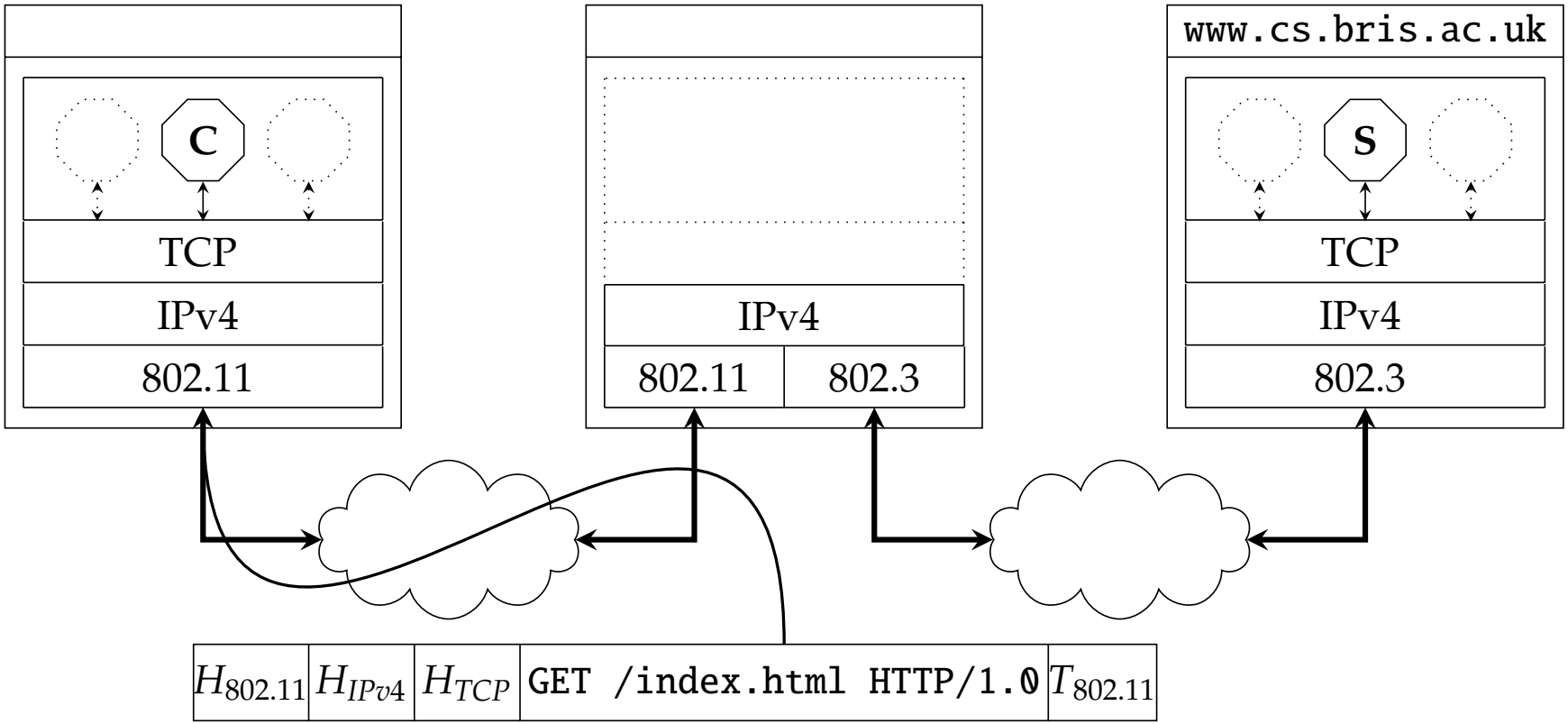
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

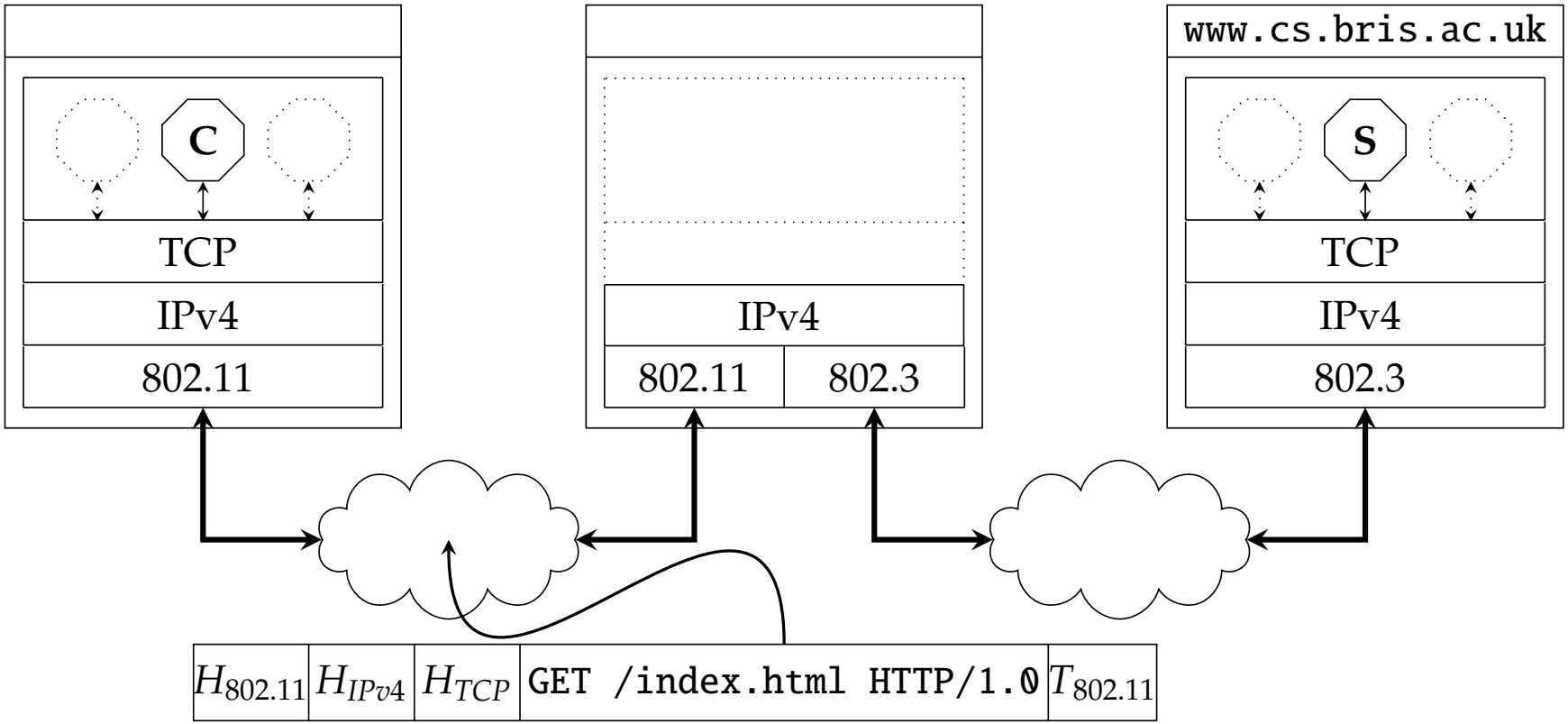
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

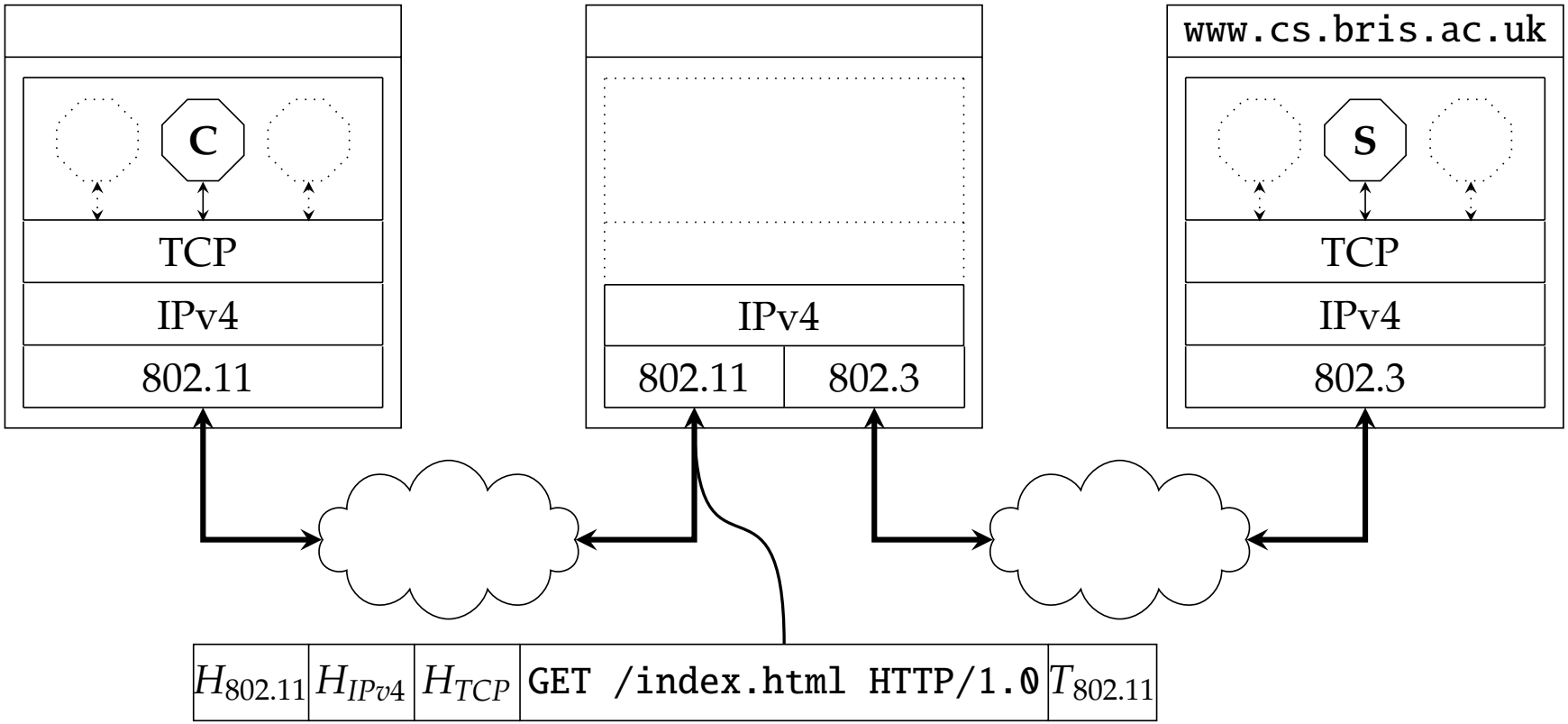
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

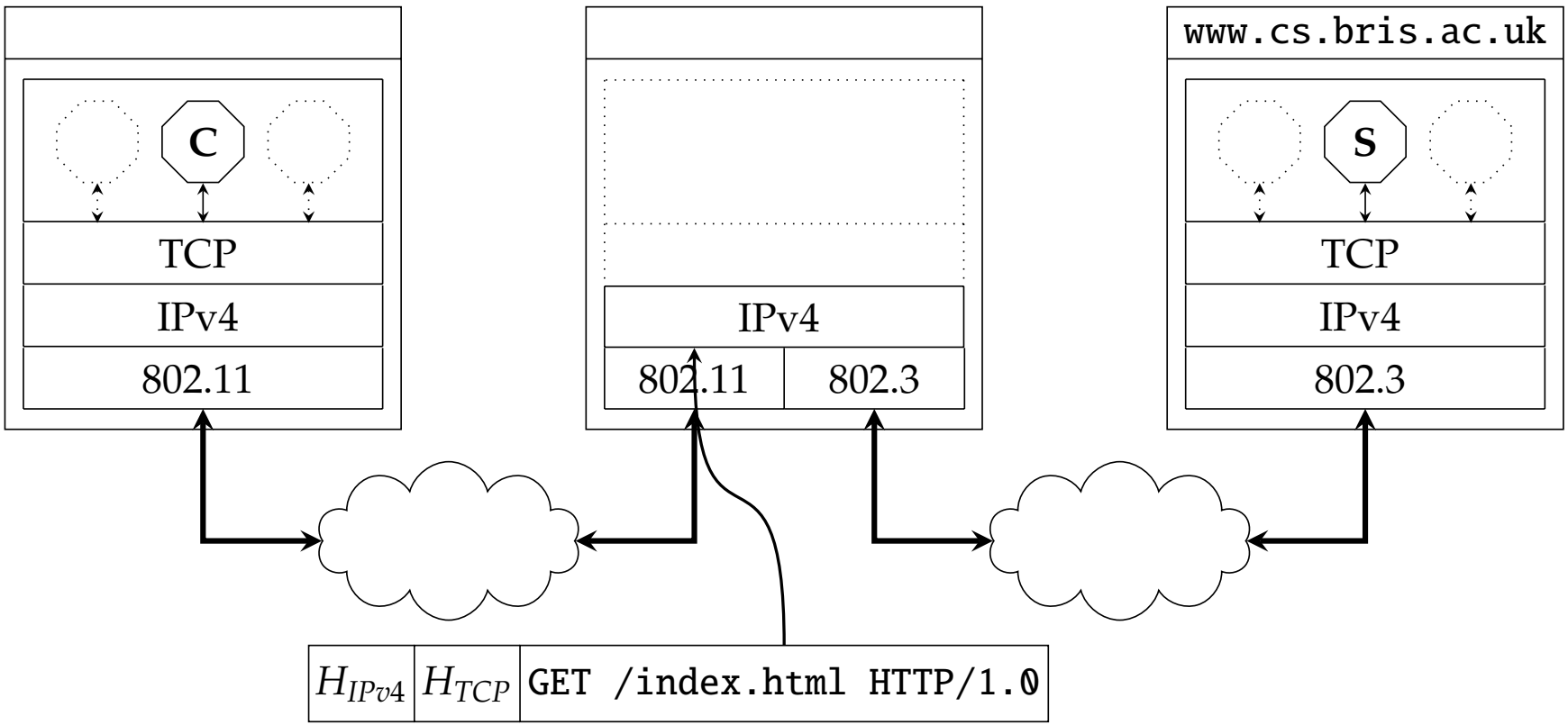
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

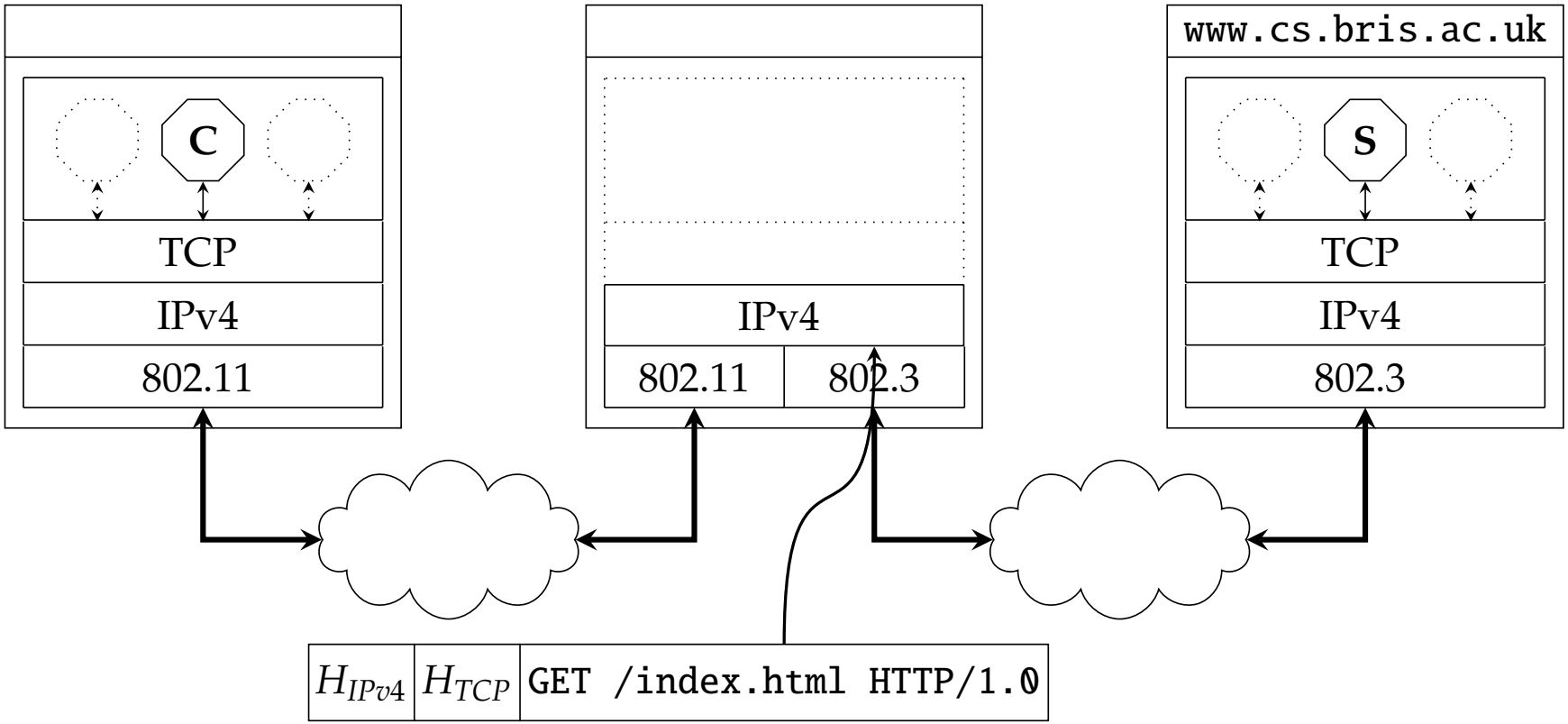
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

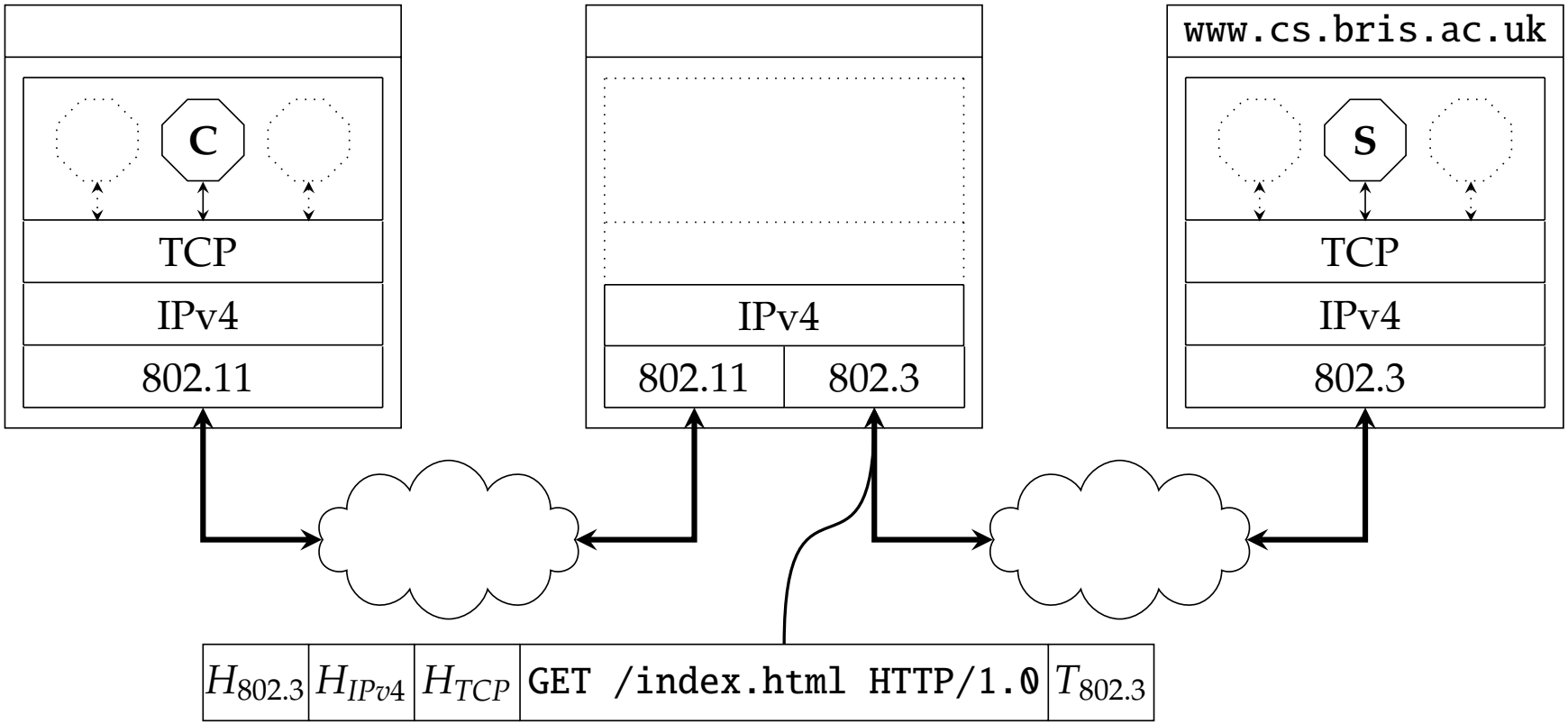
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

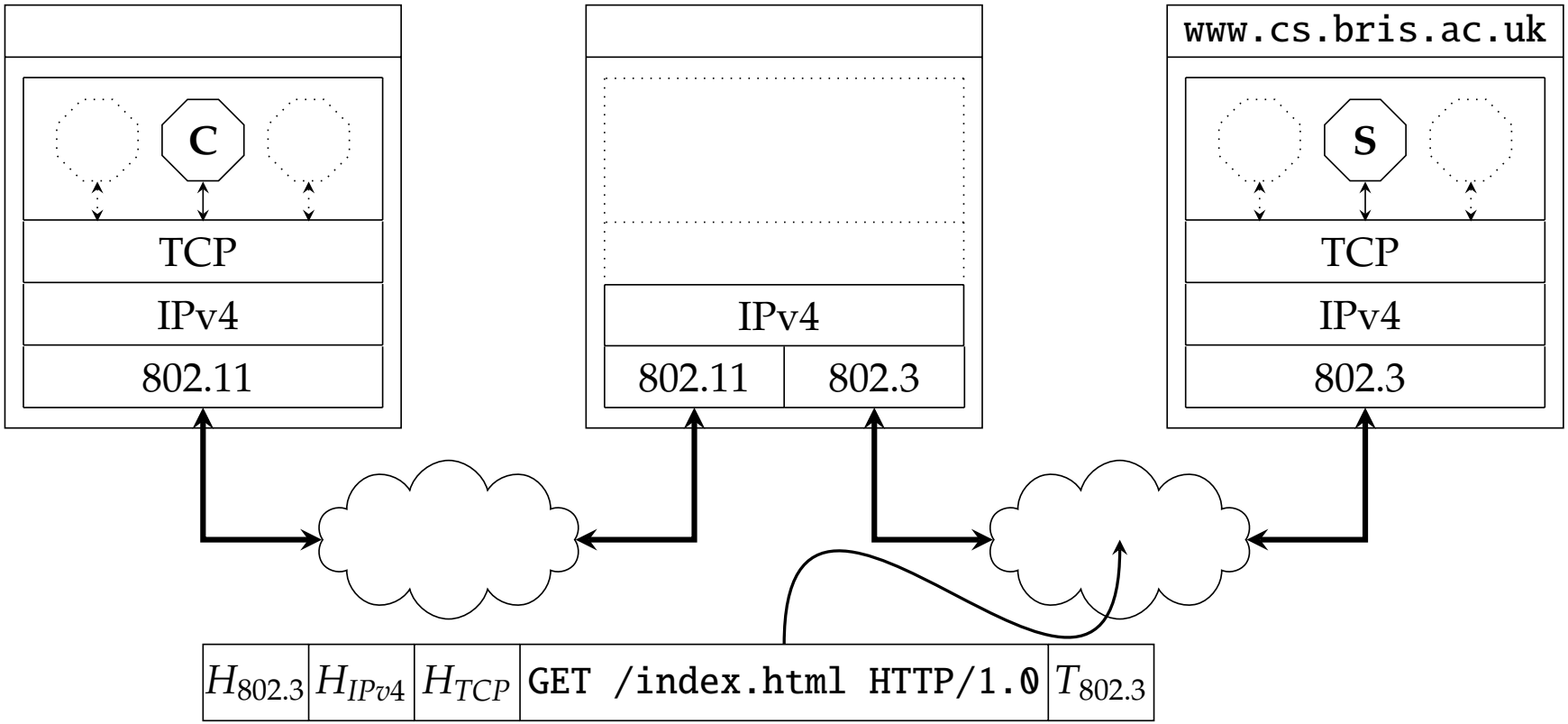
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

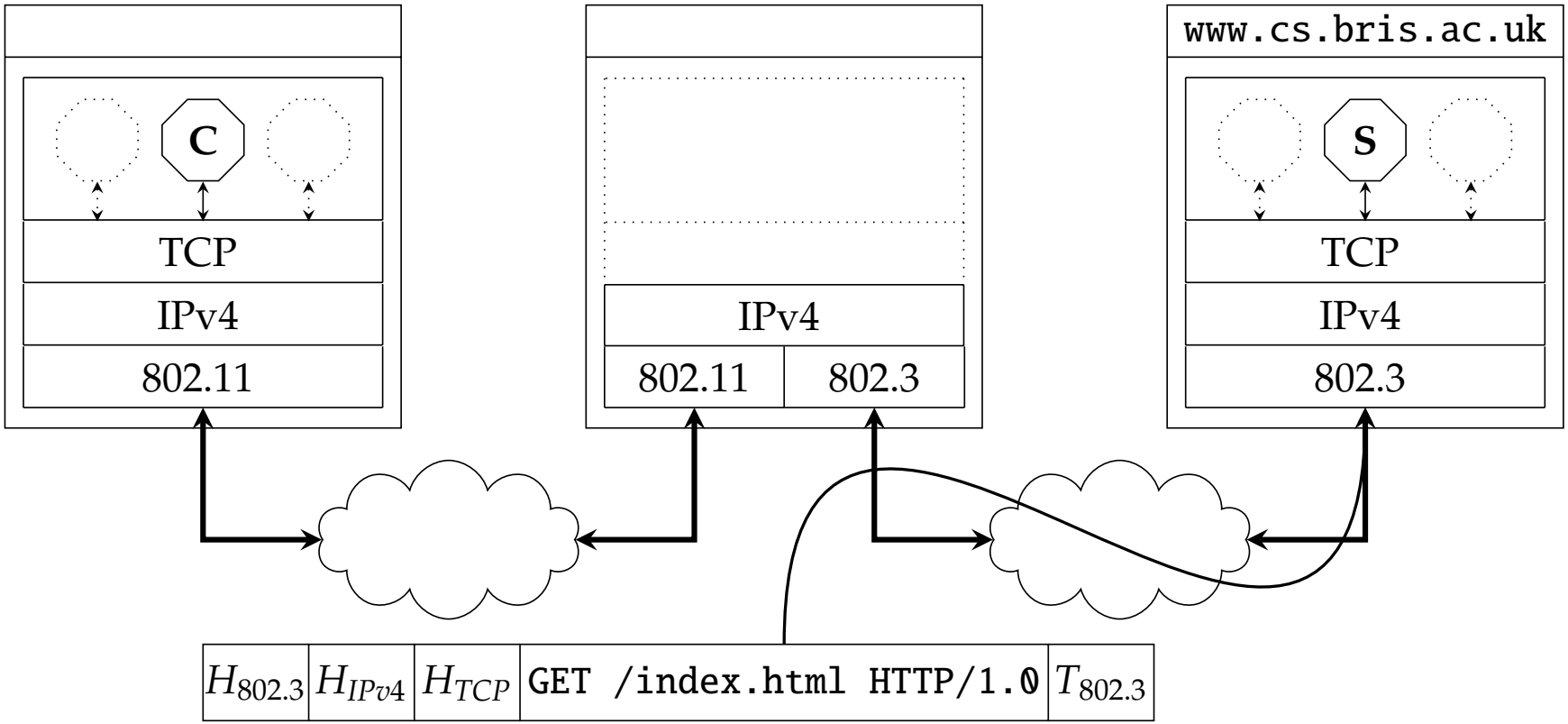
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

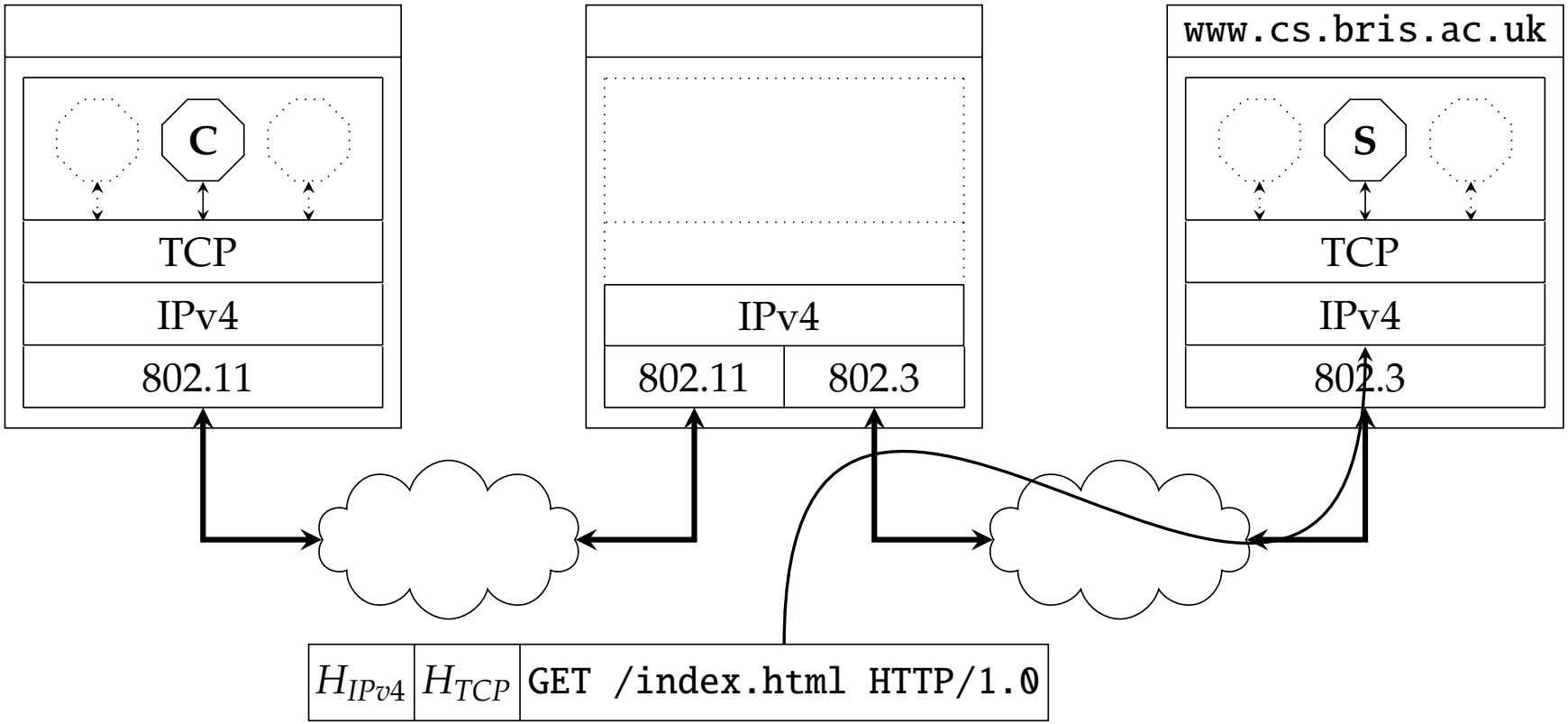
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

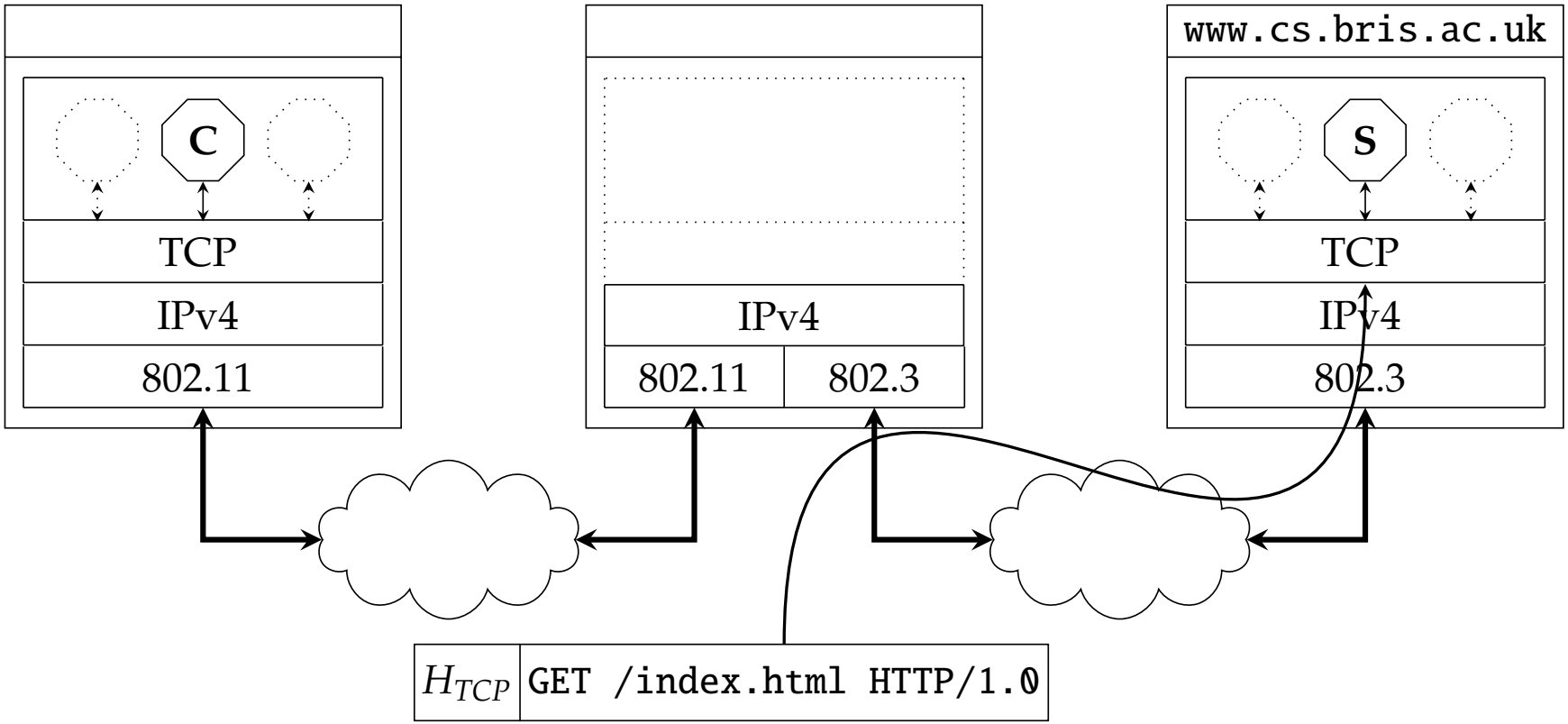
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

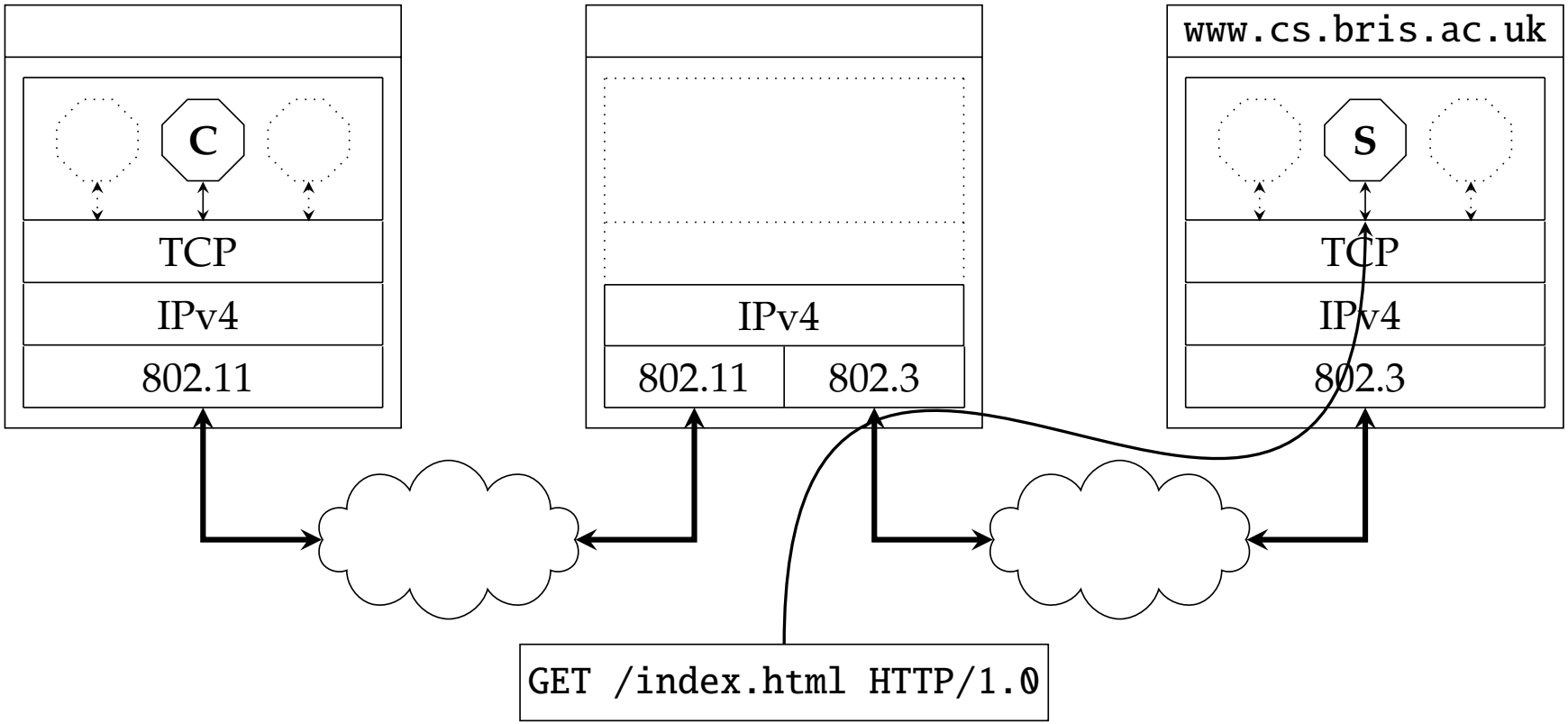
- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

- As a final step, we can add some detail to our running example, e.g.,



with the rest of the unit aiming to further explain each layer.

Conclusions

- Take away points:
  - Although a lot of this content might seem very abstract, the point is that we now have a sensible high-level design.
  - We'll fill in missing detail using a bottom-up approach by covering the
    1. link layer,
    2. internet layer,
    3. transport layer, and
    4. application layer

    while emphasising the *general* concepts as applied in specific technologies ...
  - ... keeping in mind the running, motivating example of HTTP.
  - A central challenge throughout is design of solutions that work efficiently in the general-case, but guarentee correctness in (many) special- or corner-cases.

References

[1] Wikipedia: Circuit switching.
http://en.wikipedia.org/wiki/Circuit_switching.

[2] Wikipedia: Internet protocol suite.
http://en.wikipedia.org/wiki/Internet_protocol_suite.

[3] Wikipedia: Network topology.
http://en.wikipedia.org/wiki/Network_topology.

[4] Wikipedia: OSI model.
http://en.wikipedia.org/wiki/OSI_model.

[5] Wikipedia: Packet switching.
http://en.wikipedia.org/wiki/Packet_switching.

[6] Wikipedia: Protocol stack.
http://en.wikipedia.org/wiki/Protocol_stack.

[7] D. Cohen.
On holy wars and a plea for peace.
IEEE Computer Magazine, 14:48–54, 1981.

[8] J. Postel.
Transmision Control Protocol.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 793, 1981.
http://tools.ietf.org/html/rfc793.

References

[9] J. Reynolds and J. Postel.
Assigned numbers.
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1060, 1990.
http://tools.ietf.org/html/rfc1060.

[10] W. Stallings.
Chapter 11: Circuit switching and packet switching.
In *Data and Computer Communications* [13].

[11] W. Stallings.
Chapter 2: Data communications, data networks and the Internet.
In *Data and Computer Communications* [13].

[12] W. Stallings.
Chapter 3: Protocol architecture, TCP/IP, and Internet-based applications.
In *Data and Computer Communications* [13].

[13] W. Stallings.
Data and Computer Communications.
Pearson, 9th edition, 2010.