Concurrent Computing (Computer Networks)

Daniel Page

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB. UK.
⟨Daniel.Page@bristol.ac.uk⟩

March 14, 2016

Keep in mind there are *two* PDFs available (of which this is the latter):

1. a PDF of examinable material used as lecture slides, and
2. a PDF of non-examinable, extra material:
   - the associated notes page may be pre-populated with extra, written explaination of material covered in lecture(s), plus
   - anything with a "grey'ed out" header/footer represents extra material which is useful and/or interesting but out of scope (and hence not covered).

Notes:

Notes:

- Goal: investigate the **link layer** e.g.,

1. addressing,
2. framing,
3. multiple access protocols, and
4. examples implementations

st. we can transmit (structured) **frames** between two end-points using a (shared) physical connection.

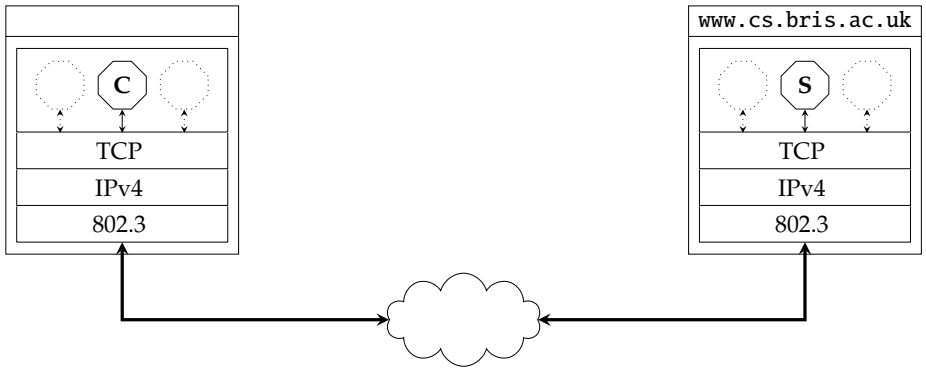
Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes

  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.

COMS20001 lecture: week #20



- Goal: investigate the **link layer** e.g.,

1. addressing,
2. framing,
3. multiple access protocols, and
4. examples implementations

st. we can transmit (structured) **frames** between two end-points using a (shared) physical connection.

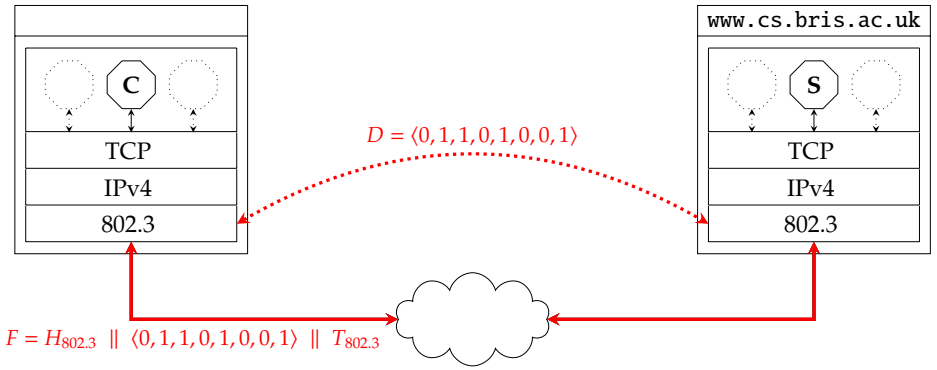
Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes

  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.



- Goal: investigate the **link layer** e.g.,
  1. addressing,
  2. framing,
  3. multiple access protocols, and
  4. examples implementations

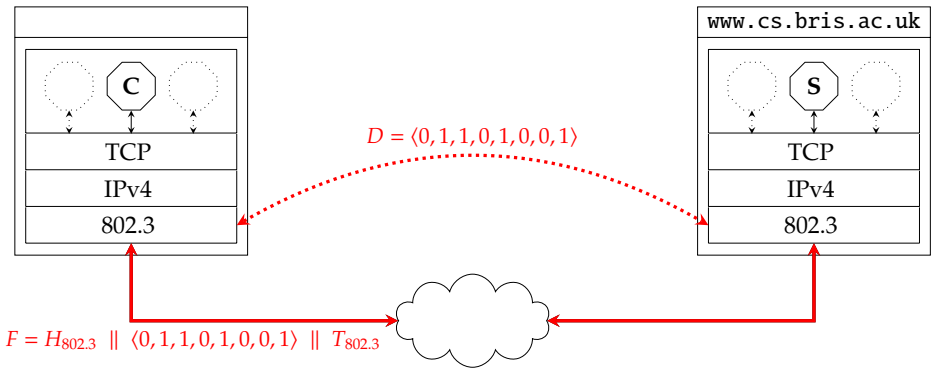
  st. we can transmit (structured) **frames** between two end-points using a (shared) physical connection.

Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes
  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.

COMS20001 lecture: week #20

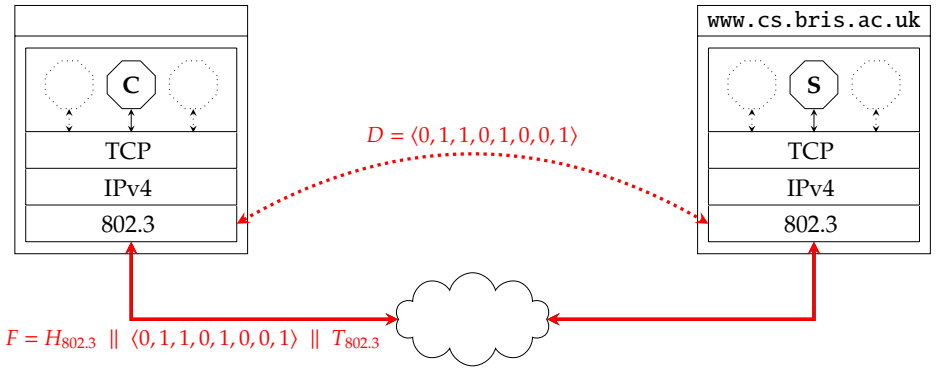


- In theory (per the OSI model): the link layer is composed of
  1. a higher-level **Logical Link Control (LLC)** sub-layer (e.g., 802.2) which interfaces with higher-level protocols, and
  2. a lower-level **Media Access Control (MAC)** sub-layer (e.g., 802.3) which manages the physical communication medium.

Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes
  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.



- In practice (per the Internet model [8, Section 2]):
  - separation of LLC and MAC sub-layers is less prescriptive, plus
  - if we only consider IPv4 and TCP above the link layer, the LLC sub-layer becomes very lightweight

  so we focus on the MAC sub-layer.

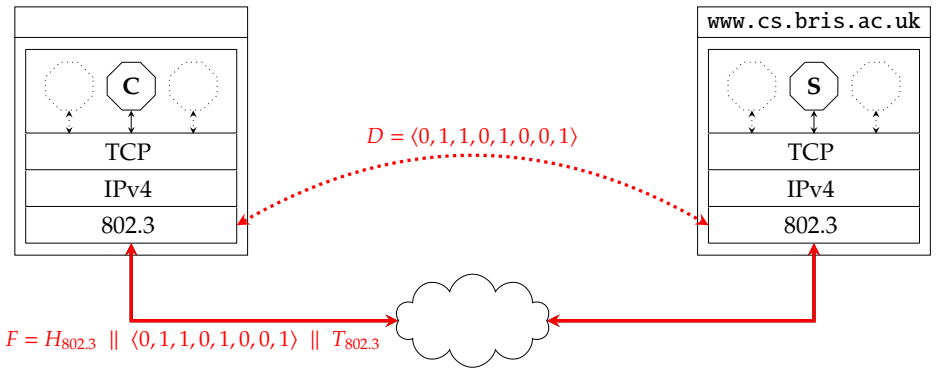
Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes

  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.

COMS20001 lecture: week #20



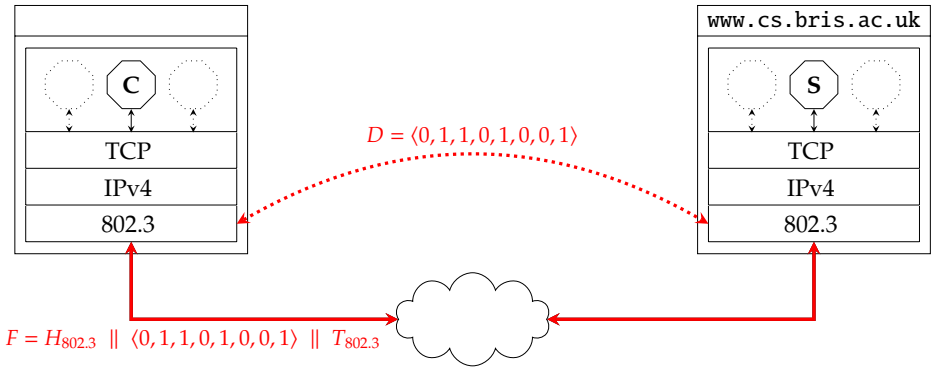
- In theory: the MAC sub-layer supports *either*
  1. point-to-point connections meaning dedicated access, *or*
  2. multi-point connections meaning shared access.

Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes

  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.



- In practice: point-to-point connections
  - prevents contention (i.e., require no management), *but*
  - imply a fixed topology which is hard to scale,
  - cannot support genuine broadcast transmission, plus
  - are likely to be under-utilised

  so we focus on managing **multiple access** to a multi-point connection.

Notes:

- In a rough sense, the LLC sub-layer offers higher-level functionality (e.g., error and flow control) while the MAC sub-layer offers lower-level functionality (e.g., physical addressing and framing, plus implementation of any appropriate multiple access protocols).
- So the LLC sub-layer offers an abstraction of the link layer, as a whole, to the layers above; the MAC layer offers an abstraction of the physical layer. This means a MAC sub-layer for point-to-point connections is more or less redundant, or at least a lot simpler: **Point-to-Point Protocol (PPP)** [12], for example, offers a link layer including only a limited amount of MAC-style functionality.
- The reason for claiming that the LLC sub-layer is lightweight when we use TCP and IP above it, is *they* take on associated work: TCP, for instance, will typically manage error and flow control. As such, the LLC sub-layer is mainly tasked with (de)multiplexing data st. a received frame can be passed to the correct layer above (of which there *could* be many, even though IP is the obvious candidate).
- In a little more detail, the 802.2 LLC sub-layer encapsulates data from the layer above in an **LLC Protocol Data Unit (PDU)** before presentation to the MAC sub-layer; the header includes

  1. a **Source Access Point (SAP)** field,
  2. a **Destination Access Point (DAP)** field, and
  3. a control field.

  The SAP (resp. DAP) field specifies which protocol *above* the LLC sub-layer created (resp. should receive) the payload, thereby allowing correct (de)multiplexing.

An Aside: "802.*wat*?!"

- **IEEE 802** is a family of computer networking standards ...
- ... more specifically,
  1. networks that allow variable-length payloads (cf. cell relay), and
  2. restricted to layer-1 (physical) and layer-2 (data link) of the OSI model.
- Examples:
  - 802.2 is a standard 802-based LLC sub-layer,
  - 802.3 is an **Ethernet** MAC sub-layer,
  - 802.4 is a **Token bus** MAC sub-layer,
  - 802.5 is a **Token ring** MAC sub-layer,
  - 802.11 is a **WiFi** MAC sub-layer

  and so on (there are a *lot*).

Notes:

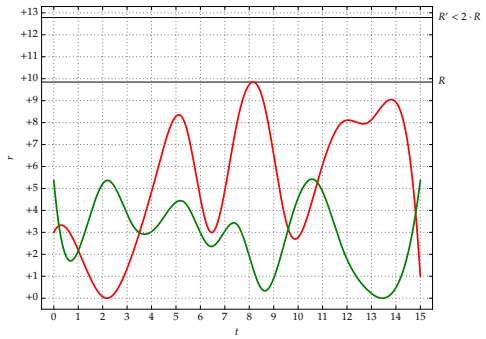
- Since we focus on 802.3, this represents a good example to show how the reality is more complex than this simple overview. Initially we had
  - "experimental" Ethernet (circa 1973) meaning or 2.94Mbit s^{-1} over coaxial cable, then
  - Ethernet-II (circa 1982) meaning or 10Mbit s^{-1} over coaxial cable

  where the latter defined a frame format etc. This was then standardised under IEEE 802, from which was born
  - 802.3 or 10BASE5 (circa 1983) meaning 10Mbit s^{-1} "thick" Ethernet over coaxial cable,
  - 802.3a or 10BASE2 (circa 1985) meaning 10Mbit s^{-1} "thin" Ethernet over coaxial cable,
  - 802.3i or 10BASE-T (circa 1990) meaning 10Mbit s^{-1} Ethernet over twisted pair cable,
  - 802.3u or 100BASE-T (circa 1995) meaning 100Mbit s^{-1} "fast" Ethernet over twisted pair cable.

  plus many others. For example, most of the above are subject to various amendments, and other standards specify auxiliary concepts, techniques or components (e.g., 802.3c deals with 802.3-based repeaters).
  This is, of course, quite confusing. But the main things to take away are a) there is a *lot* of detail, b) someone needs to take care of that detail if hosts are to inter-operate correctly, plus c) if someone says "I use Ethernet" the correct answer is "that's nice, which one of the million incompatible versions of Ethernet do you mean?".
- Nomenclature such as 10BASE5 can be explained by study of the physical layer: the 10 part means 10Mbit s^{-1} bandwidth, the BASE part means "baseband signals are transmitted", and the 5 part relates to the maximum segment (or cable) length (which in this case is 500m).

Concepts (1)

- Consider two hosts $\mathcal{H}_i$ and $\mathcal{H}_j$ communicating at some rate r:



- Problem: how can we support shared access to a connection?
- Solution #1: use **static multiplexing**, i.e.,
  - assume a connection whose available bandwidth is $R > r$ for all t,
  - share that bandwidth, e.g., by using TDM or FDM

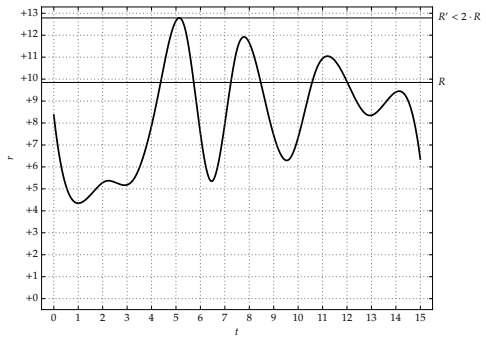
  noting this is ineffective if demand is "bursty".

Notes:

- There are *of course* instances where use of static multiplexing is perfectly effective: if we have a fixed number of hosts (e.g., a fixed number of transmitters each transmitting a TV channel) whose bandwidth requirements are fixed (i.e., they are always transmitting), then there is no under-utilisation to worry about! Network traffic is unlikely to fit this description, however, because both the hosts connected to the network and their use of it vary dynamically.
- With TDM and FDM, the overhead of multiplexing was close to nil: there was no bandwidth wasted as a result of multiplexing itself (bar use of a guard period in TDM). A multiple access protocol, on the other hand, *may* introduce some non-trivial overhead by virtue of having to manage the connection. Typically this overhead is related to, if not directly implied by, complexity of the protocol. That is, enforcing a simpler protocol typically implies lower overhead than a more complex alternative.

Concepts (1)

- Consider two hosts $\mathcal{H}_i$ and $\mathcal{H}_j$ communicating at some rate r:



- Problem: how can we support shared access to a connection?

- Solution #2: use **statistical multiplexing**, i.e.,
  - assume a connection whose available bandwidth is $R' < 2 \cdot R$,
  - enforce a **multiple access protocol** that considers the *combined* rate at a given t, thus
  - *dynamically* sharing the bandwidth based on *demand*.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

Notes:

- There are *of course* instances where use of static multiplexing is perfectly effective: if we have a fixed number of hosts (e.g., a fixed number of transmitters each transmitting a TV channel) whose bandwidth requirements are fixed (i.e., they are always transmitting), then there is no under-utilisation to worry about! Network traffic is unlikely to fit this description, however, because both the hosts connected to the network and their use of it vary dynamically.
- With TDM and FDM, the overhead of multiplexing was close to nil: there was no bandwidth wasted as a result of multiplexing itself (bar use of a guard period in TDM). A multiple access protocol, on the other hand, *may* introduce some non-trivial overhead by virtue of having to manage the connection. Typically this overhead is related to, if not directly implied by, complexity of the protocol. That is, enforcing a simpler protocol typically implies lower overhead than a more complex alternative.

- There are various ways (cf. [11, Figure 1.1]) to classify multiple access protocols, e.g.,
  - channel- vs. packet-mode,
  - centralised vs. decentralised,
  - static- vs. dynamic-allocation,
  - time vs. frequency allocation,
  - contention-based vs. contention-free,
  - deterministic vs. non-deterministic contention resolution
  - ...

  each implying associated advantages, disadvantages and hence use-cases.
- Goal: scalability, in that
  1. we'd like to balance fairness with utilisation, st. for n hosts each one gets $\frac{1}{n}$ of the available bandwidth (or *more* if only $n' \leq n$ are active),
  2. we'd prefer not to have to fix n to get this property, but
  3. we'd like to minimise complexity in order to minimise overhead.

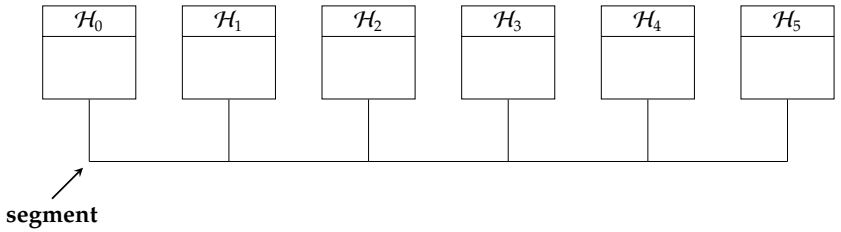
Notes:

- For example, contention-free protocols are effective in high(er)-load scenarios or when guaranteed quality of service is required, whereas randomised contention-based protocols are effective in low(er)-load scenarios.
- The concept of multiple access, and associated protocols, is clearly not unique to computer networks: you can apply the same reasoning, and up to a point the same approaches, to *any* shared resource.

Concepts (3)

Definition (segment)

A network **segment** is a portion of said network, typically (but not necessarily, depending on the network type) understood as a portion on which attached hosts are physically connected.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

MAC/EUI-48	=	3 + 3B	=	6B	=	48bit
EUI-64	=	3 + 5B	=	8B	=	64bit

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

  $$01 : 23 : 45 : 67 : 89 : AB : CD : EF$$

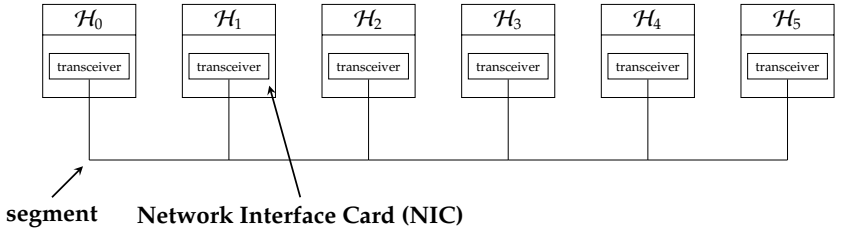
  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

  $$FF : FF : FF : FF : FF : FF$$

  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

Definition (**Network Interface Card (NIC)**)

Each host is connected to a network segment via a transceiver, which we normally term a **Network Interface Card (NIC)**.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

MAC/EUI-48	=	3 + 3B	=	6B	=	48bit
EUI-64	=	3 + 5B	=	8B	=	64bit

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

$$01 : 23 : 45 : 67 : 89 : AB : CD : EF$$

  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

$$FF : FF : FF : FF : FF : FF$$

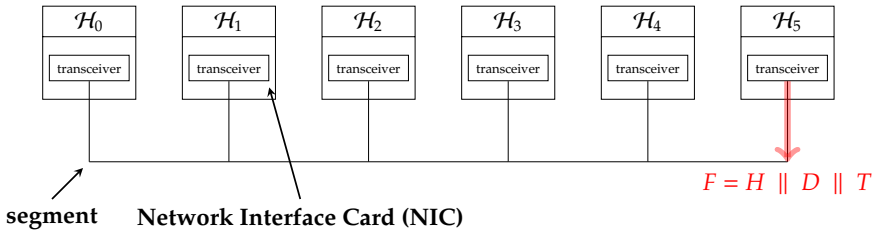
  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

Concepts (3)

Definition (**frame** and **framing**)

When a host wants to transmit some payload, it encapsulates it in a **frame** then presented to the physical layer: the act of **framing**

1. deliniate the start and end of the frame st. it can be parsed from an unstructured bit-sequence, and
2. adds relevant control information.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

MAC/EUI-48	=	3 + 3B	=	6B	=	48bit
EUI-64	=	3 + 5B	=	8B	=	64bit

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

$$01 : 23 : 45 : 67 : 89 : AB : CD : EF$$

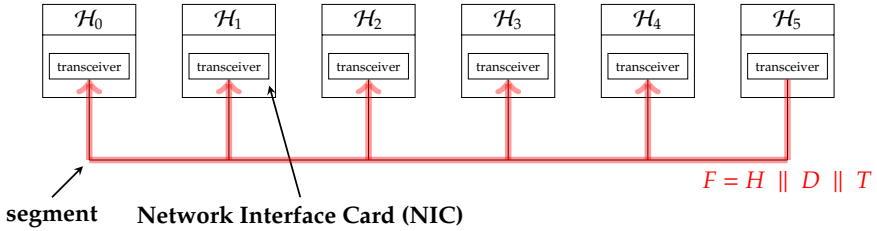
  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

$$FF : FF : FF : FF : FF : FF$$

  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

Definition (**broadcast domain**)

A **broadcast domain** is a logical (sub)division of a network st. all hosts within it can communicate with each other through use of broadcast transmission.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

$$\begin{array}{rcrcrcr} \text{MAC/EUI-48} & = & 3+3\text{B} & = & 6\text{B} & = & 48\text{bit} \\ \text{EUI-64} & = & 3+5\text{B} & = & 8\text{B} & = & 64\text{bit} \end{array}$$

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

$$01:23:45:67:89:AB:CD:EF$$

  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

$$FF:FF:FF:FF:FF:FF$$

  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

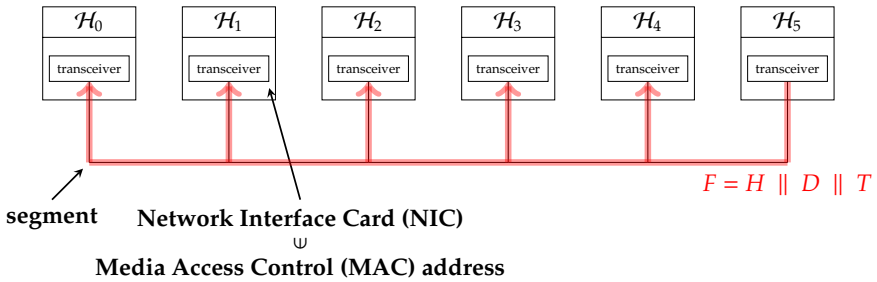
Concepts (3)

Definition (**Media Access Control (MAC) address**)

To disambiguate broadcast transmissions, each host is identified by a **Media Access Control (MAC) address**:

- typically hard-coded into NIC,
- needs to be unique wrt. broadcast domain,
- formed from an **Organisationally Unique Identifier (OUI)** and a NIC-specific identifier.

which form source/destination fields in the frame header. There is typically a reserved **broadcast address** allowing *all* hosts to receive a transmitted frame.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

$$\begin{array}{rcrcrcr} \text{MAC/EUI-48} & = & 3+3\text{B} & = & 6\text{B} & = & 48\text{bit} \\ \text{EUI-64} & = & 3+5\text{B} & = & 8\text{B} & = & 64\text{bit} \end{array}$$

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

$$01:23:45:67:89:AB:CD:EF$$

  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

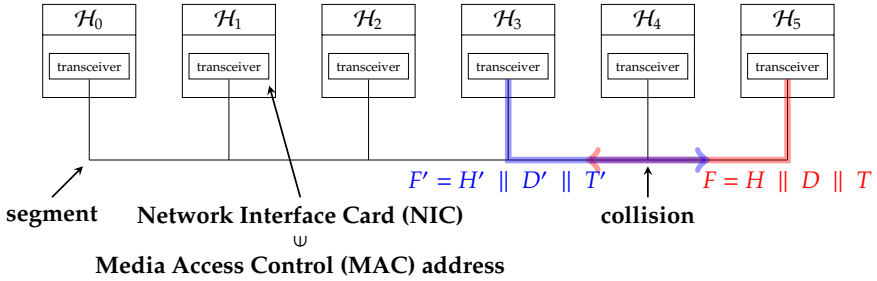
$$FF:FF:FF:FF:FF:FF$$

  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

Concepts (3)

Definition (**collision** and **collision domain**)

A **collision domain** is a logical (sub)division of a multiple access network st. transmission by one host may **collide** (or interfere) with a (simultaneous) transmission by another. In general, collision domains are smaller than (i.e., contained within) an associated broadcast domain.



Notes:

- Many framing mechanisms are possible, which you *could* investigate in some detail. Given the addition of control information is satisfied by simply appending or prepending a header or trailer to the underlying data, the main problem is that of deliniating the frame start and end; example mechanisms include use of a length field, or so-called **byte stuffing** or **bit stuffing**.
- It is common to refer to a transceiver as a physical layer transceiver, or PHY for short: if you read "Ethernet PHY" as part of some product specification, this basically just means "has a 802.3 transceiver".
- Example MAC address formats include

$$\begin{array}{rcccccc} \text{MAC/EUI-48} & = & 3+3\text{B} & = & 6\text{B} & = & 48\text{bit} \\ \text{EUI-64} & = & 3+5\text{B} & = & 8\text{B} & = & 64\text{bit} \end{array}$$

  It is common to see MAC addresses written using a colon-separated list of octets (in transmission order, so normally big-endian). For example, the EUI-64 address

$$01 : 23 : 45 : 67 : 89 : AB : CD : EF$$

  has an OUI of $\langle 45_{(16)}, 23_{(16)}, 01_{(16)} \rangle$, and a NIC-specific identifier of $\langle AB_{(EF)}, CD_{(16)}, AB_{(16)}, 89_{(16)}, 67_{(16)} \rangle$.
  For MAC address formats of this type, the broadcast address is st. all bits of all octets are 1, e.g., for 802.3-based EUI-48

$$FF : FF : FF : FF : FF : FF$$

  would be the broadcast address.
- With a given MAC address, two bits have specific roles:
  1. the 0-th bit of the most-significant OUI octet determines whether uni-cast or multi-cast is used, while
  2. the 1-st bit of the most-significant OUI octet determines whether the address is locally or universally (i.e., globally) administered (i.e., who decides it).
- A given NIC may have a fixed (or **burnt-in**) MAC address, *or* (which is increasingly the case) a reconfigurable address that can be altered; on one hand this ability is useful (e.g., to support virtual machines that share a NIC but need separate MAC addresses, or if/when address clashes occur), but on the other hand can also allow abuse.

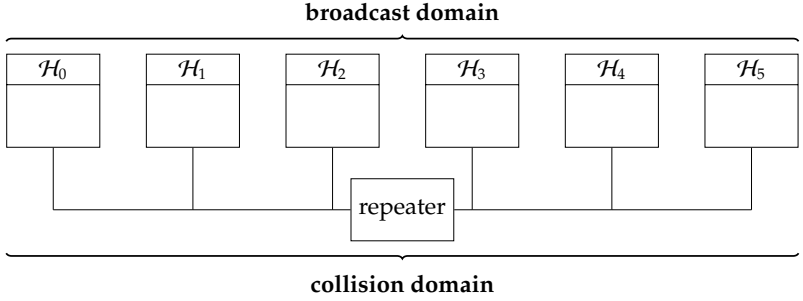
Concepts (4)

Definition (**repeater**, **bridge**, **hub** and **switch**)

Using a suitable component (or **network appliance**), e.g.,

1. a **repeater**,
2. a **bridge**,
3. a **hub**, or
4. a **switch**

one can connect multiple separate networks, or network segments, into a single aggregate network. These components a) operate in different layers, and so b) have somewhat differing capabilities and purposes.



Notes:

- 802.1D specifies the role and operation of bridge-like components within the 802 family of standards.
- One of the main purposes of a repeater, in 802.3 at least, is to deal with the issue of attenuation. That is, the network segment length is limited by virtue of physical characteristics relating to the medium: one can only transmit a signal so far, before attenuation means the strength is too low. Repeaters solve this by "amplifying" signals as they are forwarded from one port to the other.
- While the result is a larger network in some sense, it is important to note how this approach differs from an *inter*-network: the point is that the latter a) can support a mixture of protocols vs. demanding use of the same protocol, and, b) treats the separate networks as isolated wrt. addressing (two hosts on separate networks in an inter-network can have the same MAC address since they never communicate locally, although clearly they require a unique global address to communicate globally).
- Historically, it was common (and still is at least possible) to see bridges that translate between different MAC layers. Such a bridge might enable a connection between 802.3 and 802.5 networks, for example, via translation of frames to and from either format. In theory, this is attractive in the sense it offers similar advantages to an inter-network but on a local scale. In practice, it is often more complex than it seems: it
  1. assumes translating between some frame formats is trivial, which is not always the case (e.g., if one supports feature X but the other does not),
  2. assumes a (roughly) similar data rate (otherwise on network might provide data faster than the other can accept it, implying a need to buffer somehow),
  3. assumes a compatible way to address hosts.

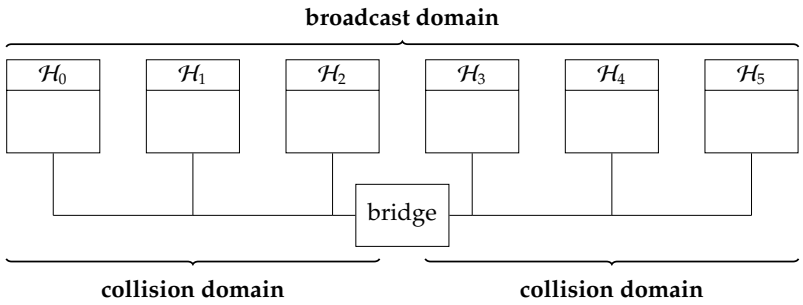
  So, more often, this type of functionality is delivered at the network layer (e.g., via a router) instead.
- There are (at least) two switch types: those based on **store-and-forward**, and **cut-through** switching. The former receives and buffers (or stores) an entire frame from one port, before transmitting (or forwarding) it to another. The latter optimises this process by starting to transmit before the entire frame has been received; this reduces latency, but needs care wrt. error control (e.g., for 802.3 frames, the CRC field cannot be received and checked by the switch before partial transmission of preceding fields).
- The hosts do not need to be altered, either wrt. hardware or software, when such components are used: from their perspective the broadcast and collision domains *still* exist (whether or not their sizes change), so as long as the component correctly connects *between* domains then everything works as normal.

Definition (**repeater**, **bridge**, **hub** and **switch**)

Using a suitable component (or **network appliance**), e.g.,

1. a **repeater**,
2. a **bridge**,
3. a **hub**, or
4. a **switch**

one can connect multiple separate networks, or network segments, into a single aggregate network. These components a) operate in different layers, and so b) have somewhat differing capabilities and purposes.



Notes:

- 802.1D specifies the role and operation of bridge-like components within the 802 family of standards.
- One of the main purposes of a repeater, in 802.3 at least, is to deal with the issue of attenuation. That is, the network segment length is limited by virtue of physical characteristics relating to the medium: one can only transmit a signal so far, before attenuation means the strength is too low. Repeaters solve this by "amplifying" signals as they are forwarded from one port to the other.
- While the result is a larger network in some sense, it is important to note how this approach differs from an *inter*-network: the point is that the latter a) can support a mixture of protocols vs. demanding use of the same protocol, and, b) treats the separate networks as isolated wrt. addressing (two hosts on separate networks in an inter-network can have the same MAC address since they never communicate locally, although clearly they require a unique global address to communicate globally).
- Historically, it was common (and still is at least possible) to see bridges that translate between different MAC layers. Such a bridge might enable a connection between 802.3 and 802.5 networks, for example, via translation of frames to and from either format. In theory, this is attractive in the sense it offers similar advantages to an inter-network but on a local scale. In practice, it is often more complex than it seems: it
  1. assumes translating between some frame formats is trivial, which is not always the case (e.g., if one supports feature X but the other does not),
  2. assumes a (roughly) similar data rate (otherwise on network might provide data faster than the other can accept it, implying a need to buffer somehow),
  3. assumes a compatible way to address hosts.

  So, more often, this type of functionality is delivered at the network layer (e.g., via a router) instead.
- There are (at least) two switch types: those based on **store-and-forward**, and **cut-through** switching. The former receives and buffers (or stores) an entire frame from one port, before transmitting (or forwarding) it to another. The latter optimises this process by starting to transmit before the entire frame has been received; this reduces latency, but needs care wrt. error control (e.g., for 802.3 frames, the CRC field cannot be received and checked by the switch before partial transmission of preceding fields).
- The hosts do not need to be altered, either wrt. hardware or software, when such components are used: from their perspective the broadcast and collision domains *still* exist (whether or not their sizes change), so as long as the component correctly connects *between* domains then everything works as normal.

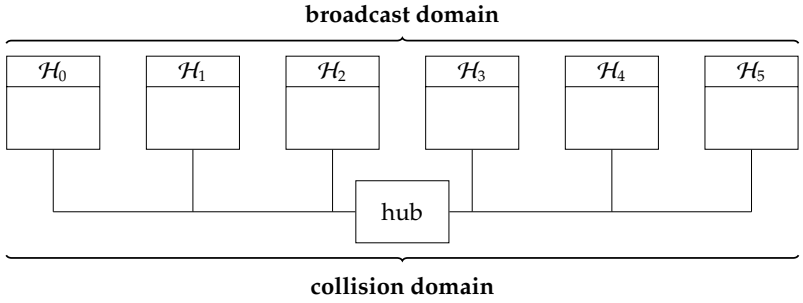
Concepts (4)

Definition (**repeater**, **bridge**, **hub** and **switch**)

Using a suitable component (or **network appliance**), e.g.,

1. a **repeater**,
2. a **bridge**,
3. a **hub**, or
4. a **switch**

one can connect multiple separate networks, or network segments, into a single aggregate network. These components a) operate in different layers, and so b) have somewhat differing capabilities and purposes.



Notes:

- 802.1D specifies the role and operation of bridge-like components within the 802 family of standards.
- One of the main purposes of a repeater, in 802.3 at least, is to deal with the issue of attenuation. That is, the network segment length is limited by virtue of physical characteristics relating to the medium: one can only transmit a signal so far, before attenuation means the strength is too low. Repeaters solve this by "amplifying" signals as they are forwarded from one port to the other.
- While the result is a larger network in some sense, it is important to note how this approach differs from an *inter*-network: the point is that the latter a) can support a mixture of protocols vs. demanding use of the same protocol, and, b) treats the separate networks as isolated wrt. addressing (two hosts on separate networks in an inter-network can have the same MAC address since they never communicate locally, although clearly they require a unique global address to communicate globally).
- Historically, it was common (and still is at least possible) to see bridges that translate between different MAC layers. Such a bridge might enable a connection between 802.3 and 802.5 networks, for example, via translation of frames to and from either format. In theory, this is attractive in the sense it offers similar advantages to an inter-network but on a local scale. In practice, it is often more complex than it seems: it
  1. assumes translating between some frame formats is trivial, which is not always the case (e.g., if one supports feature X but the other does not),
  2. assumes a (roughly) similar data rate (otherwise on network might provide data faster than the other can accept it, implying a need to buffer somehow),
  3. assumes a compatible way to address hosts.

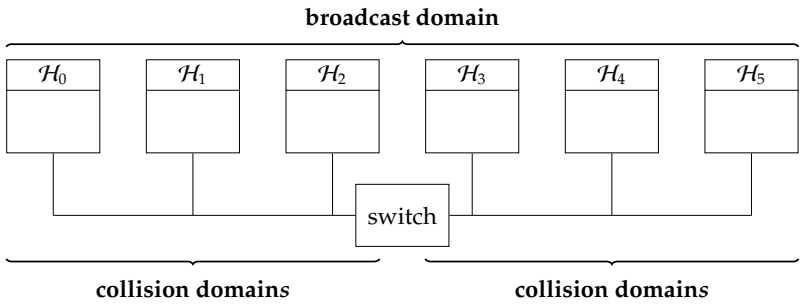
  So, more often, this type of functionality is delivered at the network layer (e.g., via a router) instead.
- There are (at least) two switch types: those based on **store-and-forward**, and **cut-through** switching. The former receives and buffers (or stores) an entire frame from one port, before transmitting (or forwarding) it to another. The latter optimises this process by starting to transmit before the entire frame has been received; this reduces latency, but needs care wrt. error control (e.g., for 802.3 frames, the CRC field cannot be received and checked by the switch before partial transmission of preceding fields).
- The hosts do not need to be altered, either wrt. hardware or software, when such components are used: from their perspective the broadcast and collision domains *still* exist (whether or not their sizes change), so as long as the component correctly connects *between* domains then everything works as normal.

Definition (repeater, bridge, hub and switch)

Using a suitable component (or **network appliance**), e.g.,

1. a **repeater**,
2. a **bridge**,
3. a **hub**, or
4. a **switch**

one can connect multiple separate networks, or network segments, into a single aggregate network. These components a) operate in different layers, and so b) have somewhat differing capabilities and purposes.



Notes:

- 802.1D specifies the role and operation of bridge-like components within the 802 family of standards.
- One of the main purposes of a repeater, in 802.3 at least, is to deal with the issue of attenuation. That is, the network segment length is limited by virtue of physical characteristics relating to the medium: one can only transmit a signal so far, before attenuation means the strength is too low. Repeaters solve this by "amplifying" signals as they are forwarded from one port to the other.
- While the result is a larger network in some sense, it is important to note how this approach differs from an *inter*-network: the point is that the latter a) can support a mixture of protocols vs. demanding use of the same protocol, and, b) treats the separate networks as isolated wrt. addressing (two hosts on separate networks in an inter-network can have the same MAC address since they never communicate locally, although clearly they require a unique global address to communicate globally).
- Historically, it was common (and still is at least possible) to see bridges that translate between different MAC layers. Such a bridge might enable a connection between 802.3 and 802.5 networks, for example, via translation of frames to and from either format. In theory, this is attractive in the sense it offers similar advantages to an inter-network but on a local scale. In practice, it is often more complex than it seems: it
  1. assumes translating between some frame formats is trivial, which is not always the case (e.g., if one supports feature X but the other does not),
  2. assumes a (roughly) similar data rate (otherwise on network might provide data faster than the other can accept it, implying a need to buffer somehow),
  3. assumes a compatible way to address hosts.

  So, more often, this type of functionality is delivered at the network layer (e.g., via a router) instead.
- There are (at least) two switch types: those based on **store-and-forward**, and **cut-through** switching. The former receives and buffers (or stores) an entire frame from one port, before transmitting (or forwarding) it to another. The latter optimises this process by starting to transmit before the entire frame has been received; this reduces latency, but needs care wrt. error control (e.g., for 802.3 frames, the CRC field cannot be received and checked by the switch before partial transmission of preceding fields).
- The hosts do not need to be altered, either wrt. hardware or software, when such components are used: from their perspective the broadcast and collision domains *still* exist (whether or not their sizes change), so as long as the component correctly connects *between* domains then everything works as normal.

- ... or, to summarise, we have something like the following:

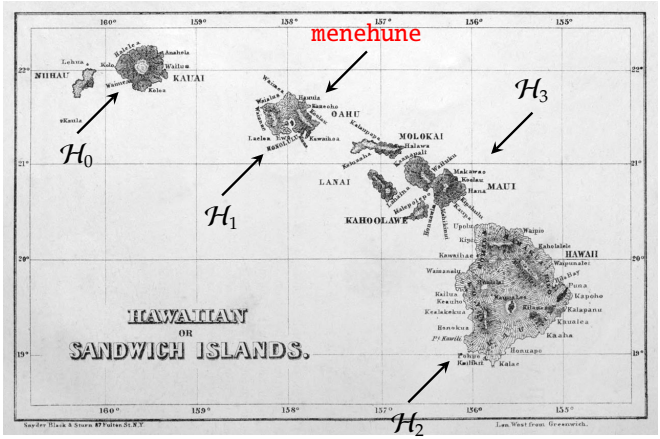
Component	Ports	Total bandwidth	Layer	Buffered?	> broadcast domain?	< collision domain?
repeater	2	r	physical	×	✓	×
hub	n	r	physical	×	✓	×
bridge	2	$2 \cdot r$	MAC	✓	✓	✓
switch	n	$n \cdot r$	MAC	✓	✓	✓

Notes:

- It is convenient to characterise repeaters as "dumb" bridges, and hubs as "dumb" switches: in both cases, the latter components operate at the MAC sub-layer, allowing "smarter" behaviour because of access to information such as MAC addresses (versus the physical layer, where components simply see a bit-sequence). Alternatively, one can think of repeaters (resp. hubs) as 1-port (resp. many-port) hubs (resp. repeaters) and similarly for the case of bridges and switches.
- In a sense, multi-port hubs and switches aid maintainability by allowing centralised management (e.g., a central wiring closet). The disadvantage might be that they also act as a central point of failure; arguably this is manageable, however, in the sense that if the component fails it is at least obvious what has failed (as there will be *no* connectivity. In a bus, for example, it is not as easy to diagnose problems (e.g., a break in the shared medium could be anywhere along it): here, the medium acts as a single point of failure!

ALOHA (1)

- **ALOHAnet** [7] pioneered multiple access protocols



by using

- a star-based topology, with several hosts around one central "hub", and
- wireless communication on two *different* incoming and outgoing (wrt. hub) frequencies.

http://commons.wikimedia.org/wiki/File:Edward_T._Perkins,_Hawaiian_or_Sandwich_Islands,_1854.jpg

Notes:

- The central hub of the system was termed "menehune", the Hawaiian word for elf: this is a nod to the ARPANET Interface Message Processor (IMP).
- The two (incoming and outgoing wrt. the hub) wireless communication channels operated on two (allocated) frequencies of 407.350MHz and 413.475MHz at 24000baud. The fact that there were two distinct channels means outgoing transmission *from* the hub is contention-free by definition; the problem that needs to be resolved is that of contention wrt. incoming transmission (from the hosts).

ALOHA (2)

- The (pure) ALOHA protocol is *very* simple ...

Protocol (hosts)

1. Broadcast payload frame
   $F = H_{ALOHA}[\text{src} = \mathcal{H}_i, \text{dst} = \mathcal{H}_j] \parallel D \parallel T_{ALOHA}$.
2. Receive acknowledgement frame
   $F' = H_{ALOHA}[\text{src} = \texttt{menehune}, \text{dst} = \mathcal{H}_i, \text{ack} = \textbf{true}] \parallel T_{AL}$
   and
   - *if* such an F' is received within a specified time limit then assume transmission of F succeeded, otherwise
   - assume transmission of F failed, so back-off then attempt to retransmit.

Protocol (hub)

1. Receive payload frame
   $F[src = \mathcal{H}_i, dst = \mathcal{H}_j]$,
   and if F is valid wrt. error detection,
   - broadcast acknowledgement frame
     $F'[src = \texttt{menehune}, dst = \mathcal{H}_i, ack = \textbf{true}]$,
   - broadcast payload frame F,

   otherwise drop F.

... or, even *more* simply still, "if you have something to transmit then do so, and if there's a problem just retransmit it sometime later".

Notes:

- To **drop** F means to *actively* decide not to process it further; this contrasts with the **loss** of F, in the sense that the former is intentional whereas the latter is accidental.
- In a generic sense, the term **back-off** means "wait for some period of time". In ALOHA, the specific back-off scheme used is randomised: when a host needs to back-off, it waits for some random period of time (with the aim of preventing a repeated collision).
- ALOHA uses the term packet to describe units of data being communicate; to match our discussion, it is reasonable to read this as frame instead. All frames were
  $$8\text{Bheader} + 80\text{Bpayload} = 704\text{bit}$$
  in size (i.e., had a fixed size, padded if the payload was less): the header included a 32 bits of addressing and control information, and 32 bits of error detecting code. The latter is a vital ingredient in the system, since this is how the hub knows whether or not a collision occurred. That is, a collision is detected iff. the error correction mechanism flags a failure (which is reasonable, since, for example, the interference would yield a corrupted frame).
- Note that the need for acknowledgement frames is implied by the inability to transmit *and* receive at the same time: this is a characteristic of the wireless medium, rather than a limitation per se. Put another way, if hosts could detect when their transmissions overlapped (by detecting periods when they received frames at the same time as transmitting) then *they* could detect collisions rather than relying on the hub to inform them via an acknowledgement.
- The efficiency of ALOHA can be analysed wrt. throughput (i.e., the number of frames successfully transmitted per unit of time, or wrt. latency (i.e., the delay associated with transmission of a given frame); in either case, the probability of collisions has a clear influence. Such an analysis is given in [11, Section 3.1], with the conclusion that (pure) ALOHA yields roughly 18% utililisation.
  In a sense, this is unsurprising since ALOHA is such a simple protocol: it's obvious that the probability of collisions is relatively high. Improvements are possible, however. Slotted ALOHA [11, Section 3.2] is one example, whereby transmission by hosts is restricted (so basically synchronised) to discrete time slots. This prevents partial collisions, whereby transmissions may overlap by only a small amount but still result in a collision, since now transmissions either coincide wrt. a slot and hence collide or not.

- 802.3 is based on a number of more general principles.
- In 802-like terminology:
  1. uses a p-persistent **Carrier Sense Multiple Access (CSMA)** protocol,
  2. reduces the cost of collisions via **Collision Detection (CD)**,
  3. reduces the probability of collision synchronisation via **Binary Exponential Back-off (BEB)**

  which is probably clear as mud!

Notes:

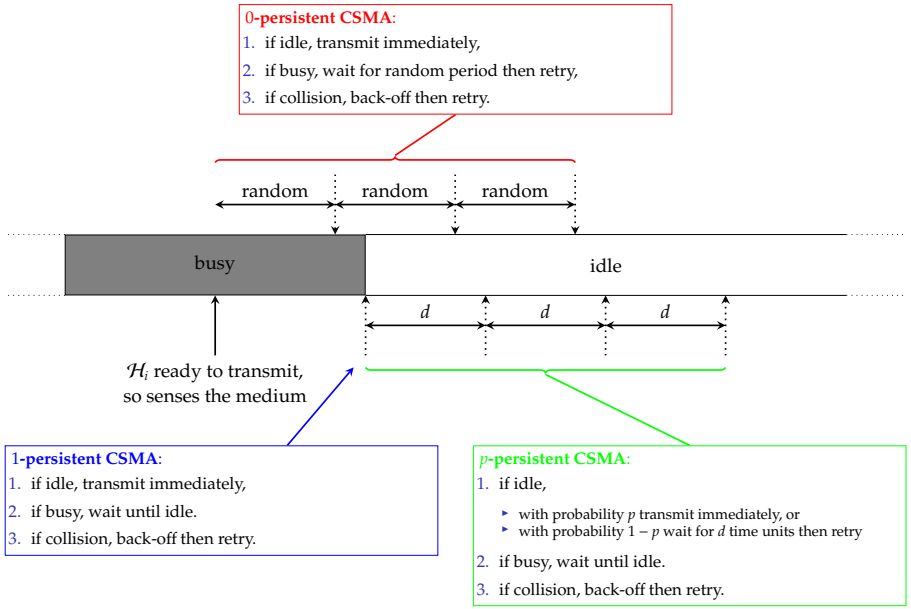
802.3 (1)

- 802.3 is based on a number of more general principles.
- In English:
  1. senses the medium, and transmits with probability p if idle,
  2. stops transmitting when a collision is detected,
  3. waits upto $2^m \cdot d$ time units when the m-th successive collision is detected

  which *hopefully* makes more sense.

Notes:

802.3 (2) – CSMA/CD

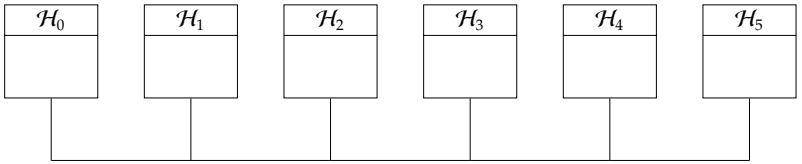


Notes:

- It is hard to cite it from the diagram, but this is essentially the same as [16, Page 503]. Similar ways of visualising activity highlight the fact that the medium can be in one of three states, namely
  1. an **idle period** (i.e., is unused),
  2. a **transmission period** (i.e., is used), or
  3. a **contention period** (i.e., is recovering from a collision).
- In any one of these cases, the number of reattempted transmissions is normally limited by some maximum after which the strategy aborts.
- In reality, there are now *three* components relating to latency of transmission: the transmission delay and propagation delay, which are the same as before, plus a **sensing delay** that relates to how quickly a host can determine whether the medium is busy or idle.
- The case of $p = 0$ is *non*-persistent; you can view this strategy as being deferential, in that it is considerate to other hosts. The clear opposite is $p = 1$ which is selfish: is simply transmits as soon as it can, regardless of other hosts.
- For the p-persistent strategy, one has to select a p: this turns out to be quite difficult. Ideally we want 1 host (on average) to transmit st. there are no collisions; this implies that if there are n host then $p = \frac{1}{n}$, or, put another way, if $n \cdot p > 1$ then (on average) more than 1 host will transmit. Therefore we want $n \cdot p < 1$, but this raises two problems:
  1. we don't necessarily know what n is, and
  2. in reality, we care about the $n' \leq n$ active hosts: if the number of active hosts is lower than n, which is likely, we penalise them by selecting a smaller p than necessary (which means they wait longer than necessary to transmit).

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

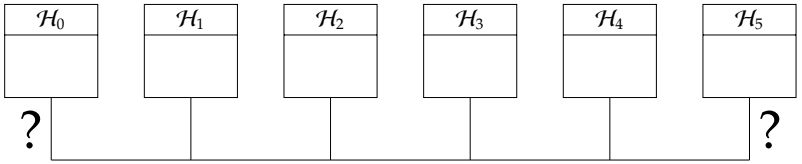


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

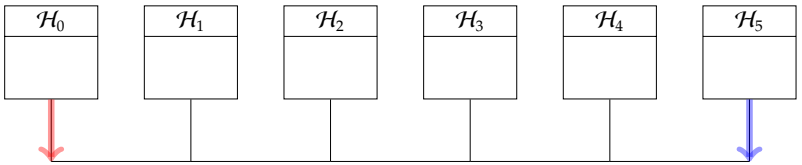


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



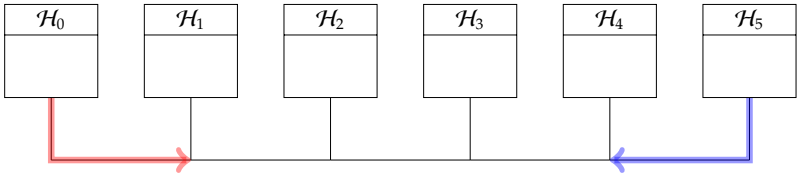
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

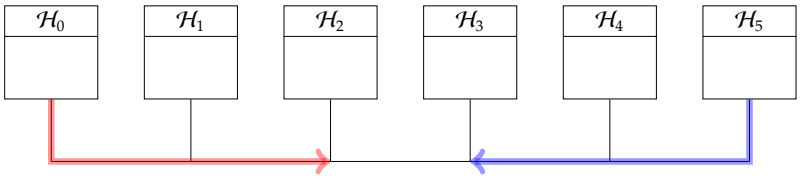


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



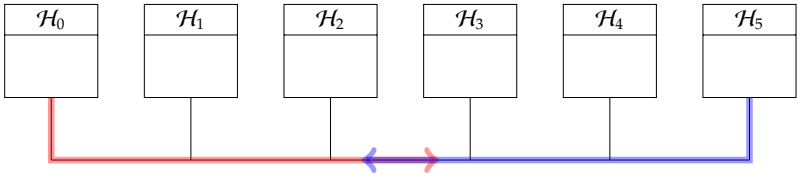
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

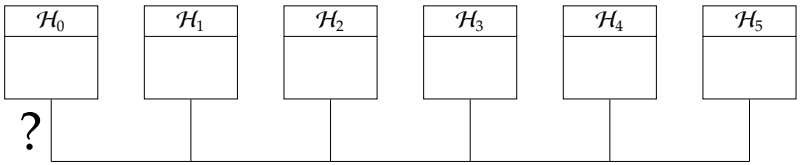


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



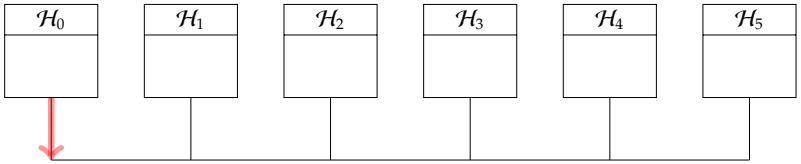
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

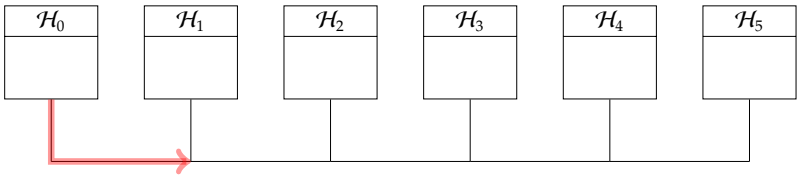


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



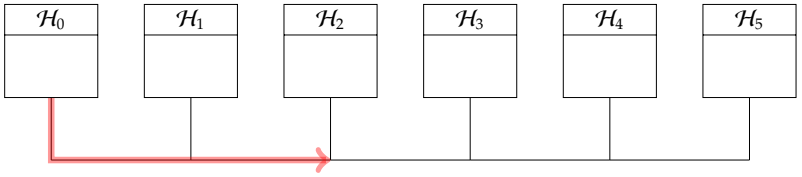
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

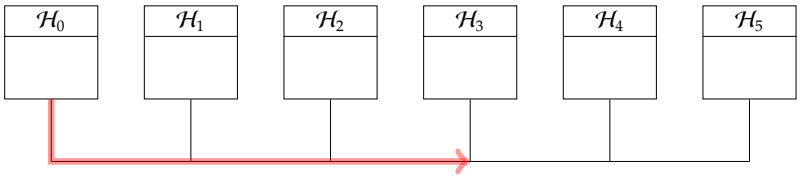


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



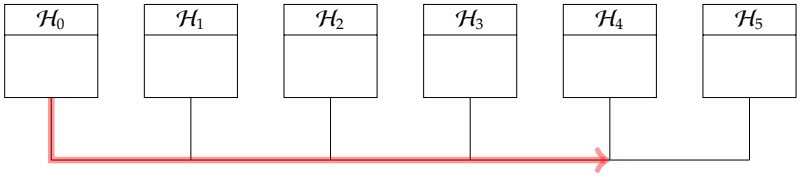
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

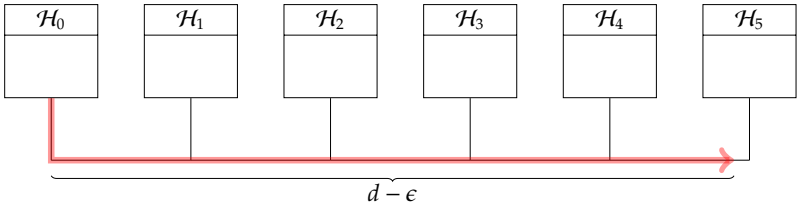


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



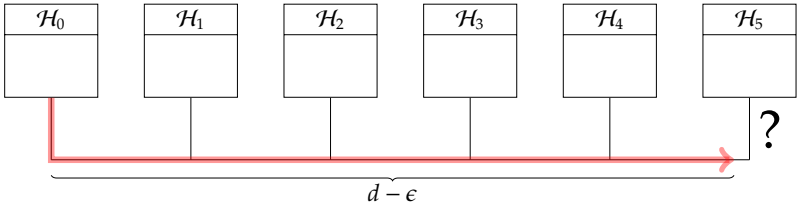
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:

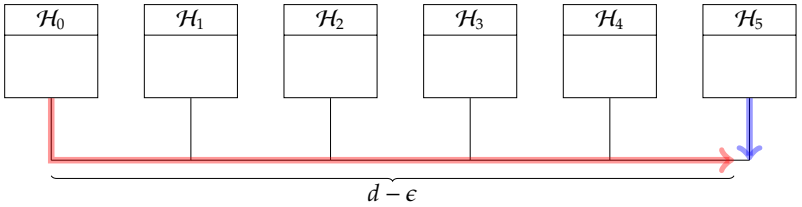


Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



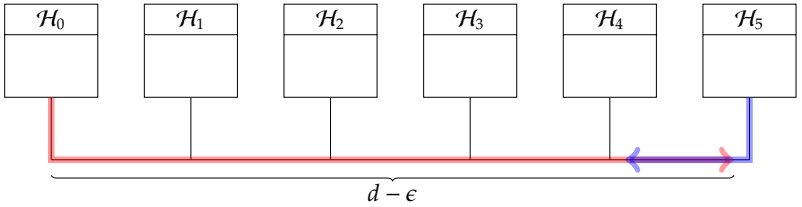
Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions:



Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (3) – CSMA/CD

- CSMA reduces but does not *eliminate* collisions ...
- ... and, even then, the addition of collision detection via CSMA/CD

CSMA/CD:

1. if idle, follow appropriate CSMA persistence strategy,
2. if busy, follow appropriate CSMA persistence strategy,
3. if collision,
   - abort transmission,
   - transmit **jam signal**,
   - back-off then retry.

focuses on optimisation rather than elimination:

1. all hosts are guaranteed to detect each collision, so there is no need for acknowledgements, and
2. hosts can stop transmitting as soon as a collision is detected, so the medium is idle again sooner.

Notes:

- The latency between start of transmission and detecting a jam signal is at most $2 \cdot d$, where d is the propagation delay; wlog. this can be the maximum propagation delay, between the hosts furthest apart, say $\mathcal{H}_i$ and $\mathcal{H}_j$. This is obvious when you think about it: it takes d time units for transmission by $\mathcal{H}_i$ to reach $\mathcal{H}_j$ (or vice versa), and *another* d time units for any associated jam signal transmitted by $\mathcal{H}_j$, due to a collision, to get back to $\mathcal{H}_i$. Put another way, $2 \cdot d$ is basically the round-trip time.
- Why is it true that we can *guarantee* all hosts connected to the medium detect a collision? Since we control all aspects of the design, we can simply fix the minimum frame size st. the transmitter can never *finish* before a jam signal resulting from a collision is received. Or, put another way, we ensure the transmission period for each frame is greater than $2 \cdot d$. The ability of early abort has the greatest impact when large frames are used: one could argue this is likely, since use of large frames best amortises the overhead of framing itself.

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.

Notes:

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.

- Eh?! Basically,

$$\begin{aligned} 0 \text{ collisions} &\rightsquigarrow \text{guess } n = 1 \Rightarrow \text{transmit immediately} \\ &\Rightarrow \text{transmission probability is } \tfrac{1}{1} \end{aligned}$$

then give up after ~ 16 attempts.

Notes:

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.
- Eh?! Basically,

$$\text{1 collisions} \rightsquigarrow \text{guess } n = 2 \Rightarrow \text{select } r \xleftarrow{\$} \{0, \ldots, 2^1 - 1\}, \text{ wait for } r \cdot d \text{ time units}$$
$$\Rightarrow \text{transmission probability is } \tfrac{1}{2}$$

then give up after ~ 16 attempts.

Notes:

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.
- Eh?! Basically,

$$\text{2 collisions} \rightsquigarrow \text{guess } n = 4 \Rightarrow \text{select } r \xleftarrow{\$} \{0, \ldots, 2^2 - 1\}, \text{ wait for } r \cdot d \text{ time units}$$
$$\Rightarrow \text{transmission probability is } \tfrac{1}{4}$$

then give up after ~ 16 attempts.

Notes:

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.
- Eh?! Basically,

$$\begin{aligned} 3 \text{ collisions} \quad &\rightsquigarrow \quad \text{guess } n = 8 \quad \Rightarrow \quad \text{select } r \xleftarrow{\$} \{0, \ldots, 2^3 - 1\}, \text{ wait for } r \cdot d \text{ time units} \\ &\Rightarrow \quad \text{transmission probability is } \tfrac{1}{8} \end{aligned}$$

  then give up after ~ 16 attempts.

Notes:

802.3 (4) – BEB

- Problem:
  - n hosts share access to the medium, and we ideally want 1 to transmit,
  - if each host transmits with probability $\frac{1}{n}$, then on average this satisfies the requirement ...
  - ... but we don't necessarily know what n is!
- Solution: BEB
  - estimate n, assuming more collisions implies larger n,
  - cater for low- and high-loads by starting optimistically, but allowing transmission probability to shrink quickly

  by (randomly) waiting upto $2^m \cdot d$ time units after m collisions.
- Eh?! Basically,

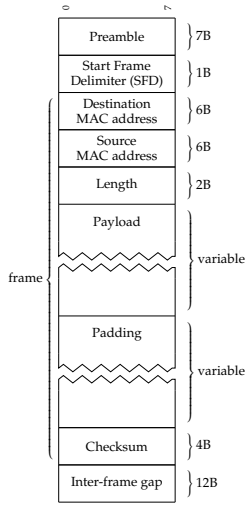
$$\begin{aligned} m \text{ collisions} \quad &\rightsquigarrow \quad \text{guess } n = 2^m \quad \Rightarrow \quad \text{select } r \xleftarrow{\$} \{0, \ldots, 2^m - 1\}, \text{ wait for } r \cdot d \text{ time units} \\ &\Rightarrow \quad \text{transmission probability is } \tfrac{1}{m} \simeq \tfrac{1}{n} \end{aligned}$$

then give up after ~ 16 attempts.

Notes:

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

Data Structure (**802.3 frame**)



The data structure includes:

- A destination MAC address.
- A source MAC address.
- A payload length (measured in 8-bit octets).
- The payload.
- Any padding required to ensure the frame satisfies the minimum length required.
- A 32-bit checksum (on the entire frame) used to detect errors.

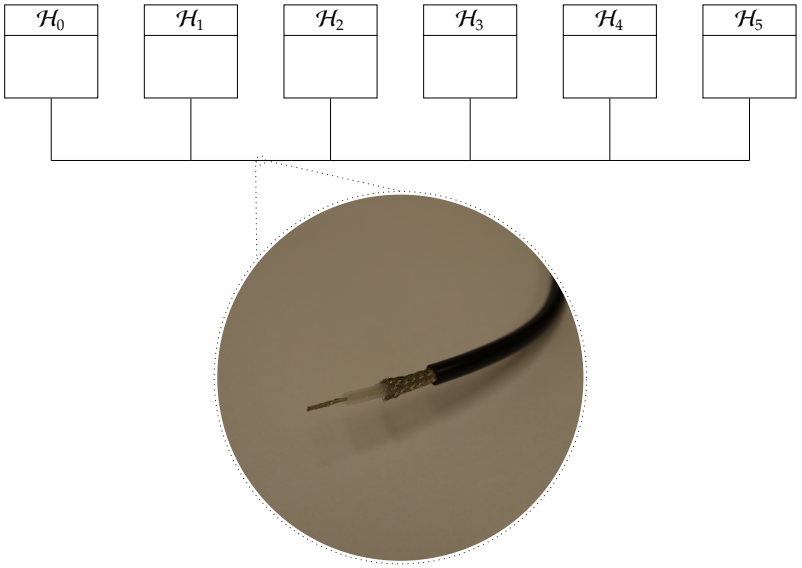
Notes:

- The preamble is a 7-octet sequence where each octet has the binary value $10101010_{(2)}$: as discussed earlier, in combination with the SFD whose value is $10101011_{(2)}$, this delimits the frame. Given the inclusion of a payload length field, there is no need for an end of frame delimiter but 802.3 does enforce a 12-octet inter-frame gap (or guard period).
  Note that, per the diagram, you can either include or discount these fields from the frame definition. In reality, they are more related to the underlying physical layer than link layer (e.g., carry no information) so we discount them.
- The inter-frame gap allows the receiver time to recover, and prepare to receive the next frame.
- The payload length field can be interpreted in two ways:
  - if it is ≤ 1500 then it specifies the payload length, whereas
  - if it is > 1500 then it specifies the frame type.

  The maximum payload length (so-called "jumbo frames" excluded) is 1500B, meaning a 1518B maximum frame length. The minimum frame length is 64 octets (which is selected to align with requirements of collision detection): any shorter payloads are padded.
- Between source MAC address and payload length fields, it is possible to include an optional 4-octet 802.1Q tag: this is optionally used to specify Quality of Service (QoS) requirements for the frame, but effectively requires a different frame type (a so-called Q-tagged frame, indicated via alternative interpretation of the payload length field).
- As is the case is some other MAC formats, 802.3 frames do not have a sequence number or identifier. This limits the type of service they can provide: without a way to group frames together somehow, it can only provide a connection-less service to the layer above.
- The checksum used is a 32-bit CRC; the field name is more formally defined as the **Frame Check Sequence (FCS)**.

802.3 (6) – "classic" Ethernet

- Basically, "classic" Ethernet = 1-persistent CSMA/CD with BEB:



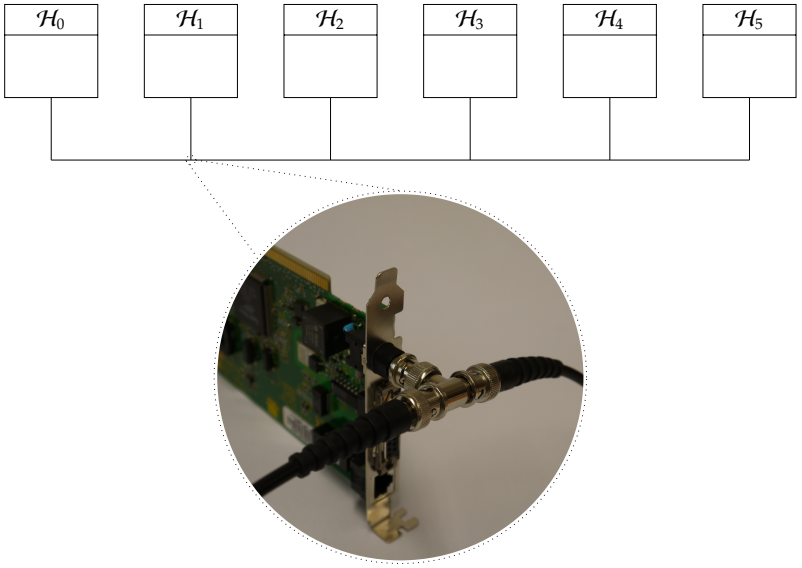
Notes:

- A single **coaxial cable** is used to connect hosts: a T-piece allows each host to "tap into" the medium, thus connecting each NIC, plus a terminator is required (at each end) to prevent reflected signals from (re)propagating along the medium. There are basically four layers in the cable itself: from inside-out, these are
  1. a copper core layer used to conduct the signal itself,
  2. a plastic insulation layer,
  3. a copper shield layer, and
  4. a plastic casing layer used for protection.

  The shielding (which is grounded) controls magnetic/electrical ingress and egress, thus prevents external interference with the signal.
- The example shown relates to 10BASE2, or "thin" Ethernet. Physical properties of the cable type dictate aspects of the topology: there is a maximum segment length of roughly 185m for instance, to which a maximum of 30 hosts can be connected.
- Metcalfe and Boggs [9, Section 3.1] offer a compelling way to categorise Ethernet, describing it as the dual of star-based topology: whereas the latter has a centralised control and distributed connectivity (from each host to the central controller), the former has distributed control and centralised connectivity.

802.3 (6) – “classic” Ethernet

- Basically, “classic” Ethernet = 1-persistent CSMA/CD with BEB:



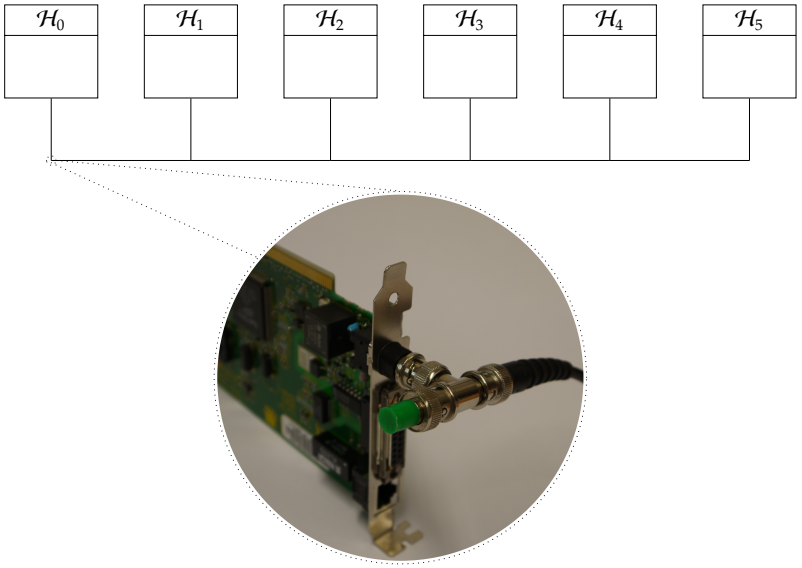
Notes:

- A single **coaxial cable** is used to connect hosts: a T-piece allows each host to “tap into” the medium, thus connecting each NIC, plus a terminator is required (at each end) to prevent reflected signals from (re)propagating along the medium. There are basically four layers in the cable itself: from inside-out, these are
  1. a copper core layer used to conduct the signal itself,
  2. a plastic insulation layer,
  3. a copper shield layer, and
  4. a plastic casing layer used for protection.

  The shielding (which is grounded) controls magnetic/electrical ingress and egress, thus prevents external interference with the signal.
- The example shown relates to 10BASE2, or “thin” Ethernet. Physical properties of the cable type dictate aspects of the topology: there is a maximum segment length of roughly 185m for instance, to which a maximum of 30 hosts can be connected.
- Metcalfe and Boggs [9, Section 3.1] offer a compelling way to categorise Ethernet, describing it as the dual of star-based topology: whereas the latter has a centralised control and distributed connectivity (from each host to the central controller), the former has distributed control and centralised connectivity.

802.3 (6) – “classic” Ethernet

- Basically, “classic” Ethernet = 1-persistent CSMA/CD with BEB:



Notes:

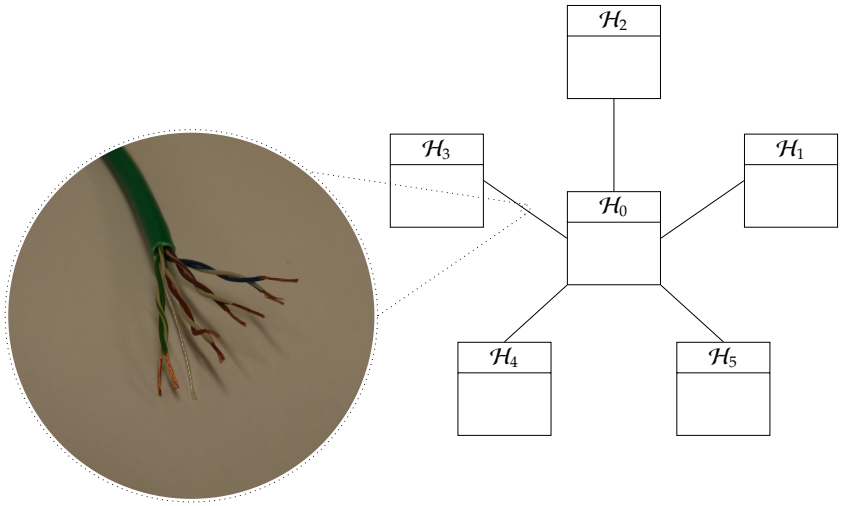
- A single **coaxial cable** is used to connect hosts: a T-piece allows each host to “tap into” the medium, thus connecting each NIC, plus a terminator is required (at each end) to prevent reflected signals from (re)propagating along the medium. There are basically four layers in the cable itself: from inside-out, these are
  1. a copper core layer used to conduct the signal itself,
  2. a plastic insulation layer,
  3. a copper shield layer, and
  4. a plastic casing layer used for protection.

  The shielding (which is grounded) controls magnetic/electrical ingress and egress, thus prevents external interference with the signal.
- The example shown relates to 10BASE2, or “thin” Ethernet. Physical properties of the cable type dictate aspects of the topology: there is a maximum segment length of roughly 185m for instance, to which a maximum of 30 hosts can be connected.
- Metcalfe and Boggs [9, Section 3.1] offer a compelling way to categorise Ethernet, describing it as the dual of star-based topology: whereas the latter has a centralised control and distributed connectivity (from each host to the central controller), the former has distributed control and centralised connectivity.

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (7) – “modern” Ethernet

- Use of “modern” Ethernet differs significantly:

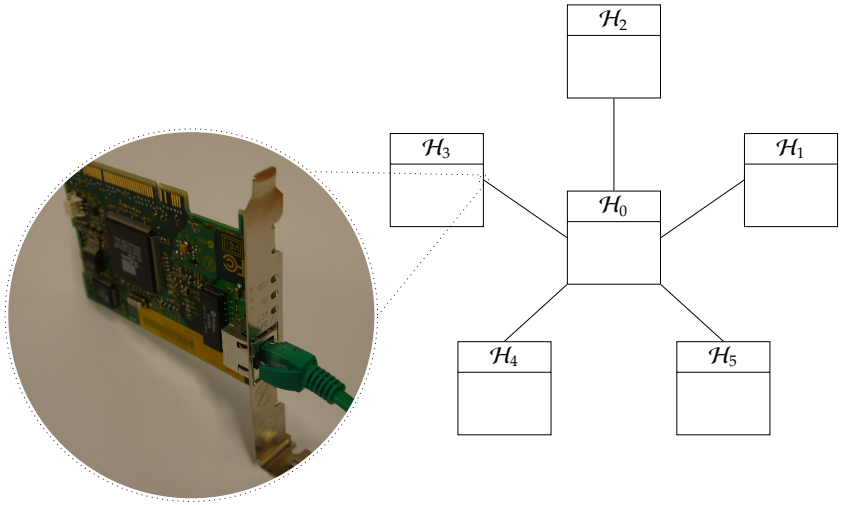


Notes:

- The cable shown is a type of **Unsheilded Twisted Pair (UTP)**: there are 4 pairs of wires, meaning 8 in total, with each wire in a given pair representing half of the associated circuit. Unlike coaxial cable, the wires have no shielding; the fact they are twisted around each other (at a fixed rate) helps resolve interference (any cross-talk between wires in particular) instead.
- The number of pairs, their colours and so on are standardised: TIA-568-A specifies a range of different UTP cable categories, such as CAT5 (which this is). Each of the categories is rated wrt. features such as maximum length (upto 100m for CAT5), and includes a specification of the wire/pair colours and their mapping to wire/pair numbers (plus pin number, on the connector which in this case would be something like RJ45).

802.3 (7) – “modern” Ethernet

- Use of “modern” Ethernet differs significantly:



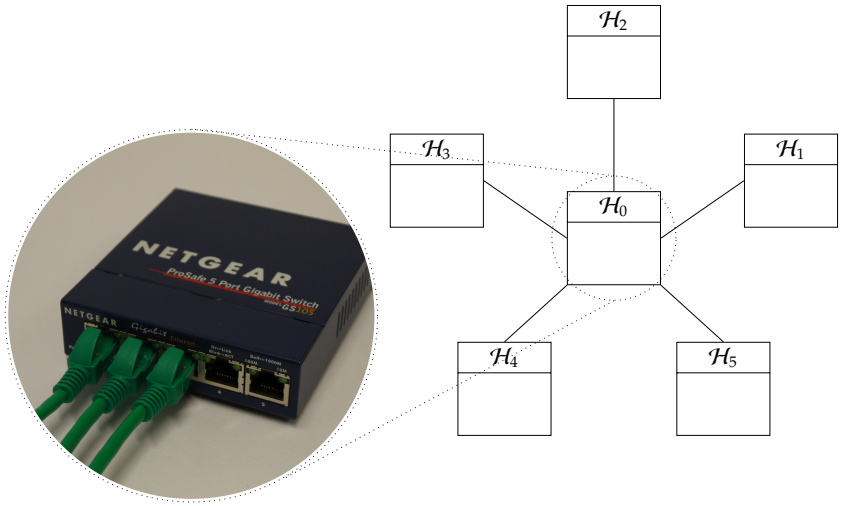
Notes:

- The cable shown is a type of **Unsheilded Twisted Pair (UTP)**: there are 4 pairs of wires, meaning 8 in total, with each wire in a given pair representing half of the associated circuit. Unlike coaxial cable, the wires have no shielding; the fact they are twisted around each other (at a fixed rate) helps resolve interference (any cross-talk between wires in particular) instead.
- The number of pairs, their colours and so on are standardised: TIA-568-A specifies a range of different UTP cable categories, such as CAT5 (which this is). Each of the categories is rated wrt. features such as maximum length (upto 100m for CAT5), and includes a specification of the wire/pair colours and their mapping to wire/pair numbers (plus pin number, on the connector which in this case would be something like RJ45).

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

802.3 (7) – "modern" Ethernet

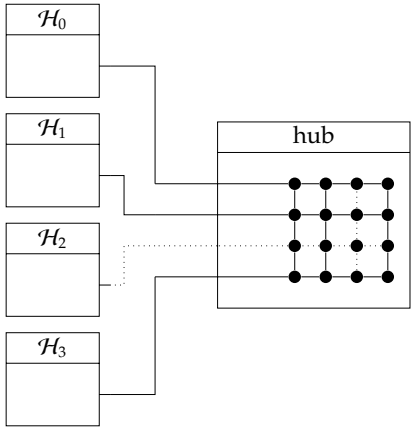
- Use of "modern" Ethernet differs significantly:



Notes:

- The cable shown is a type of **Unsheilded Twisted Pair (UTP)**: there are 4 pairs of wires, meaning 8 in total, with each wire in a given pair representing half of the associated circuit. Unlike coaxial cable, the wires have no shielding; the fact they are twisted around each other (at a fixed rate) helps resolve interference (any cross-talk between wires in particular) instead.
- The number of pairs, their colours and so on are standardised: TIA-568-A specifies a range of different UTP cable categories, such as CAT5 (which this is). Each of the categories is rated wrt. features such as maximum length (upto 100m for CAT5), and includes a specification of the wire/pair colours and their mapping to wire/pair numbers (plus pin number, on the connector which in this case would be something like RJ45).

802.3 (8) – "modern" Ethernet

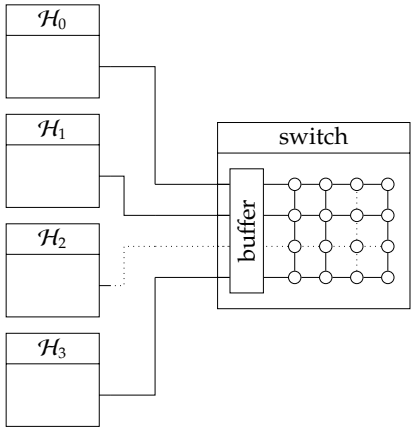


- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



Algorithm (backward learning)

1. Initialise
$$T[i] \leftarrow (\perp, \perp)$$
for each i-th entry.
2. Each time the switch receives a frame
$$F = H_{802.3}[\text{src} = x, \text{dst} = y] \parallel D \parallel T_{802.3}$$
on port p:
   - if $\exists i$ st. $T[i] = (y, p')$ then transmit F via port p' only,
   - otherwise broadcast F via *all* ports

   then
   - if $\exists j$ st. $T[j] = (x, p')$ then select that j-th entry
   - otherwise pick some unused j-th entry

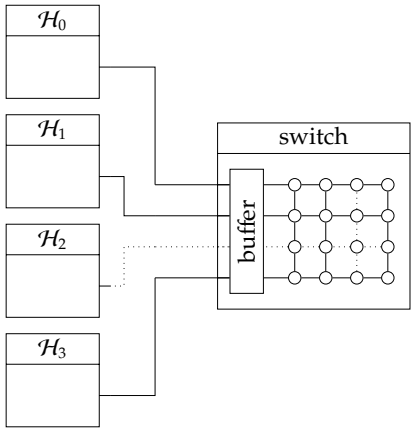
   and update $T[j] \leftarrow (x, p)$.

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



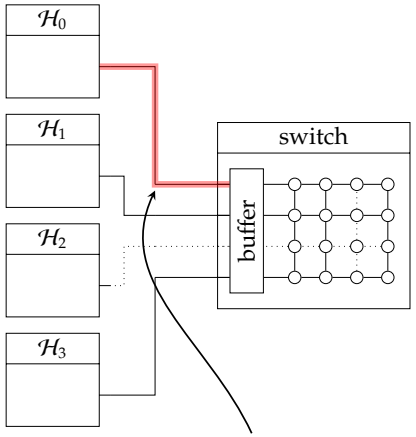
MAC address	port number
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



MAC address	port number
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

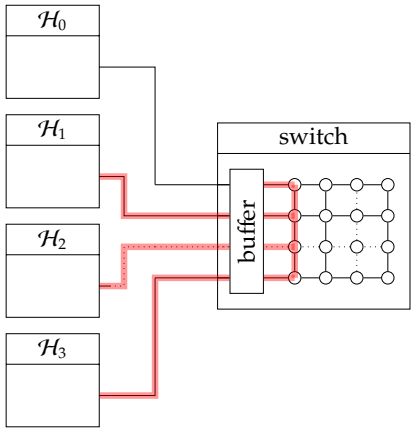
$$F = H_{802.3}[\text{src} = \mathcal{H}_0, \text{dst} = \mathcal{H}_1] \parallel D \parallel T_{802.3}$$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



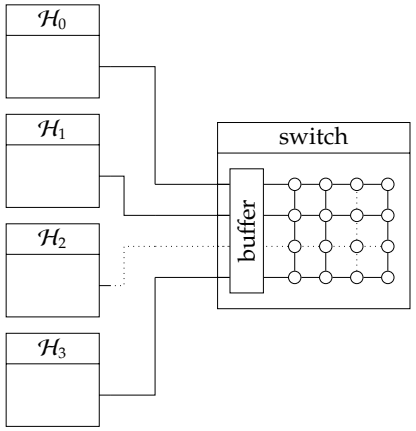
MAC address	port number
$\mathcal{H}_0$	0
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



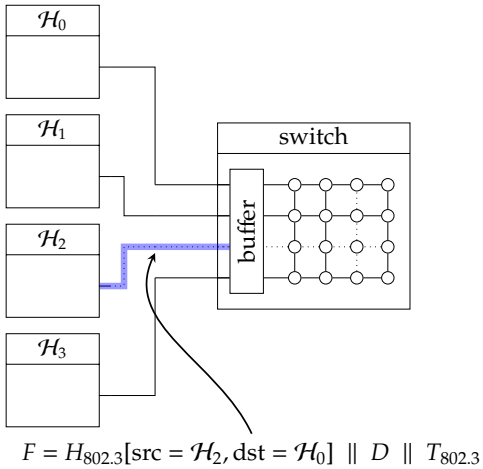
MAC address	port number
$\mathcal{H}_0$	0
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



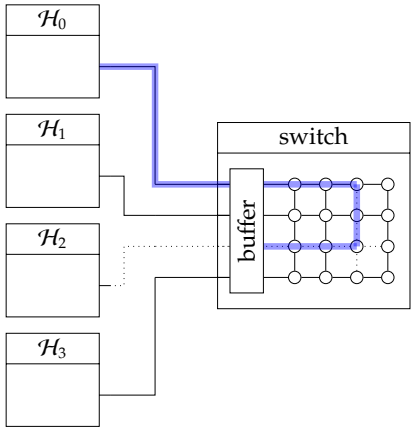
MAC address	port number
$\mathcal{H}_0$	0
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – “modern” Ethernet



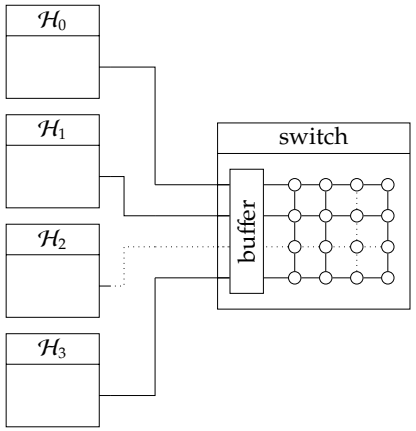
MAC address	port number
$\mathcal{H}_0$	0
$\mathcal{H}_2$	2
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – “modern” Ethernet



MAC address	port number
$\mathcal{H}_0$	0
$\mathcal{H}_2$	2
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

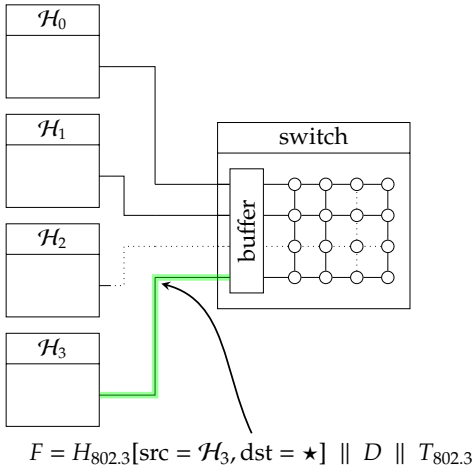
- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Daniel Page ⟨daniel.page@bristol.ac.uk⟩
Concurrent Computing (Computer Networks) git # 3627080 @ 2016-03-11 University of BRISTOL

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



MAC address	port number
$\mathcal{H}_0$	0
$\mathcal{H}_2$	2
$\perp$	$\perp$
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

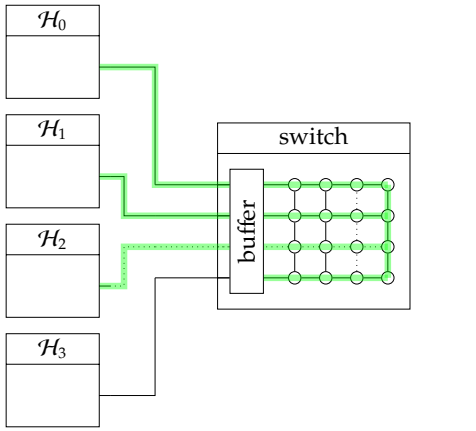
$$F = H_{802.3}[\text{src} = \mathcal{H}_3, \text{dst} = \star] \parallel D \parallel T_{802.3}$$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



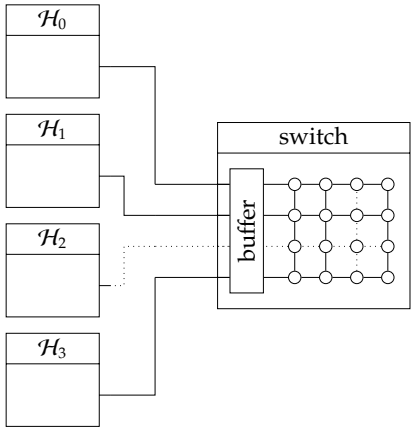
MAC address	port number
$\mathcal{H}_0$	0
$\mathcal{H}_2$	2
$\mathcal{H}_3$	3
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

802.3 (8) – "modern" Ethernet



MAC address	port number
$\mathcal{H}_0$	0
$\mathcal{H}_2$	2
$\mathcal{H}_3$	3
$\perp$	$\perp$
$\perp$	$\perp$
$\vdots$	$\vdots$

- Note that:
  - operation of the switch is totally automatic, thus permitting flexible topology,
  - per port, i.e., per segment, everything uses CSMA/CD as normal,
  - get xMbit s^{-1} per *port* not shared across one segment, but
  - fails where topology includes cyclic connectivity!

Notes:

- Note that use of a hub implies a physical, electrical connection between the ports, whereas on a switch they are buffered (and so best described as being logical): this is required to realise a store-and-forward style mode of transmission between end-points.
- Each port is full-duplex, meaning it can support simultaneous incoming and outgoing communication; although one buffer is shown, in reality this will also be split into incoming and outgoing buffers. The reason that buffers are needed at all is that contention is *still* possible even though it *seems* the switch has removed the shared medium. It is possible, for example, that $\mathcal{H}_i$ and $\mathcal{H}_j$ simultaneously transmit a frame to $\mathcal{H}_k$: in such a case, one or other is buffered since the switch can only transmit one frame at a time via whatever port that $\mathcal{H}_k$ is attached to (i.e., it stores-then-forwards). This also implies that sustained congestion may result in a given buffer becoming full, and, given there is nowhere to store additional frames, they will be lost.
- You might argue that the issue of cyclic connectivity can be resolved by simply disallowing this sort of topology. In reality, however, cycles may be *useful* (e.g., for redundancy) or occur due to configuration error (which is even more likely when the topology changes often). As such, it would be useful if they could be tolerated. The solution is to have switches agree, using a distributed algorithm [10], and then use a spanning tree to guide the forwarding process.

Conclusions

- Take away point: we now have a **Local Area Network (LAN)**.
  - topological complexity is reduced using a shared medium,
  - this limits locality of connections (i.e., *local* area), and
  - we need a protocol to control access to the (shared) medium, but
  - once a host has access, the medium looks like a point-to-point connection.

Notes:

References

[1] Wikipedia: ALOHAnet.
http://en.wikipedia.org/wiki/ALOHAnet.

[2] Wikipedia: Ethernet.
http://en.wikipedia.org/wiki/Ethernet.

[3] Wikipedia: IEEE 802.
http://en.wikipedia.org/wiki/IEEE_802.

[4] Wikipedia: Link layer.
http://en.wikipedia.org/wiki/Link_layer.

[5] Wikipedia: Logical link control.
http://en.wikipedia.org/wiki/Logical_link_control.

[6] Wikipedia: Media access control.
http://en.wikipedia.org/wiki/Media_access_control.

[7] N. Abramson.
The ALOHA system: Another alternative for computer communications.
In *Fall Joint Computer Conference (AFIPS)*, pages 281–285, 1970.

[8] 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.

Notes:

References

[9] R.M. Metcalfe and D.R. Boggs.
Ethernet: distributed packet switching for local computer networks.
Communications of the ACM (CACM), 19:395–404, 1976.

[10] R. Perlman.
An algorithms for distributed computation of a spanning tree in an extended LAN.
ACM SIGCOMM Computer Communications Review, 15(4):44–53, 1985.

[11] R. Rom and M. Sidi.
Multiple Access Protocols: Performance and Analysis.
Springer, 2011.

[12] W. Simpson.
The Point-to-Point Protocol (PPP).
Internet Engineering Task Force (IETF) Request for Comments (RFC) 1548, 1993.
http://tools.ietf.org/html/rfc1548.

[13] W. Stallings.
Chapter 16: Local area network overview.
In *Data and Computer Communications* [16].

[14] W. Stallings.
Chapter 17: Ethernet.
In *Data and Computer Communications* [16].

Notes:

[15] W. Stallings.
Chapter 18: Wireless LANs.
In *Data and Computer Communications* [16].

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

Notes: