Computer Science 415.14FC 2000

Spread Spectrum Radio

Introduction

Spread spectrum develops the information theory principle that signals are best protected from noise by making them look like noise. The basic patent was issued during World War II for torpedo guidance and names as co-inventor the Hollywood actress Hedi Lamarr.

Instead of allocating a signal to a narrow channel within a radio spectrum, it spreads the signal over the whole spectrum in a predictable way. The receiver must somehow track the transmitter "spreading function". Spreading the signal gives several advantages

- **Security against against snooping**. An intruder must duplicate the spreading function.
- **Security against most forms of noise**. Noise within a narrow bandwidth tends to be highly correlated and difficult to separate from the desired signal. Using a wide range of frequencies "decouples" the uncorrelated noise from the correlated signal and gives some measure of noise protection.
  Alternatively it is possible to use much lower signal powers for adequate noise performance. This is important for mobile phones.
- **Security against jamming**. Jamming is a form of deliberate noise which is meant to overwhelm the signal. It is very difficult and expensive to overwhelm a wide range of frequencies.

Types of Spread Spectrum

Spread spectrum uses forms of double modulation involving the data, a pseudo-random generator and the RF carrier. The bit-time of the pseudo-random generator is known as a *chip*, and states change at the *chip rate.*

There are two types of spread spectrum

- **Frequency hopping spread spectrum**. This uses the pseudo-random generator (an *N*-ary generator) to frequency modulate the carrier and then uses data to modulate the hopping carrier.
- **Direct sequence spread spectrum** first applies the data to the pseudo-random generator (both binary) and then uses the combination to modulate the rf carrier.

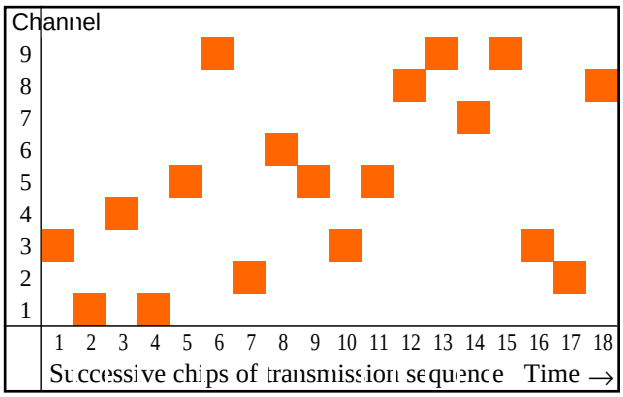
Quite apart from frequency hoping and direct sequence, there are two further classifications –

- **Slow Spread Spectrum** has
  `chip rate < data rate`,
  or has several data bits per chip. It is simpler and cheaper but has poorer performance.
- **Fast Spread Spectrum** has
  `chip rate > data rate`,
  or has several chips per data bit. The equipment is more expensive but gives better noise rejection and security.

Frequency Hopping Spread Spectrum

The spectrum is divided into separate channels (possibly tens or hundreds), usually at a fixed frequency separation.

The transmission frequency hops among the channels according to a pseudo-random sequence and the data is then modulated onto the hopping frequency; the receiver tracks the transmitter



Think of a radio system with 10 preset channels and a random sequence of decimal digits (such as π or √2). The transmitter and receiver agree on the channels and the time. If both enter the next random digit say every second, the transmitter and receiver will always use the same channel and will communicate. However an intruder who does not know the channel assignment or random sequence will hear only 10% of the communication (less if there are more channels).

Direct-Sequence Spread Spectrum

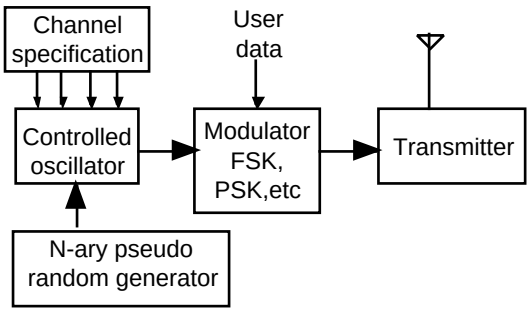
In direct-sequence spread spectrum the data is first exclusive-ORed with a pseudo-random binary sequence (the "spreading code"). This is equivalent to modulating a pseudo-random "carrier" with the data.

The spreading code spreads the radio frequency energy among all of the sidebands, which because of the random code are spread randomly over the available spectrum. The data modulation appears as a disturbance to the spreading code distribution.

The receiver uses a matching spreading code and detects the difference between what is received and what is expected.

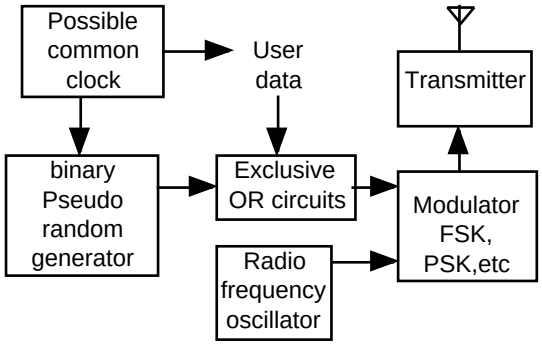
Spread Spectrum Transmitters

A frequency hopping transmitter is based on a controlled frequency oscillator, often using a frequency synthesiser. The channel specification may be just a ROM of frequencies, addressed by the output of the pseudo random generator.



Frequency hopping Spread Spectrum

A direct sequence transmitter applies the user data first to the spreading sequence and then uses that to modulate the RF signal. Here there is much more need to synchronise the user data and the pseudo-random clock.



Direct Sequence Spread Spectrum

Spreading factor

The number of chips per data bit, *N*, is known as the spreading factor but is normally called the *processing gain*, written in decibels –

$$\text{processing gain (in dB)} = 20\log_{10}N$$

If noise is concentrated at a single frequency, it will interfere with only one channel with frequency hopping; when averaged over all channels used for that bit it will be effectively reduced by a factor *N*, or by $20\log_{10}N$ dB.

A similar averaging process applies to direct-sequence spread spectrum, with each data bit averaged over *N* chips; with random noise, spreading a data bit over several chips effectively decorrelates the noise with the same overall effect.

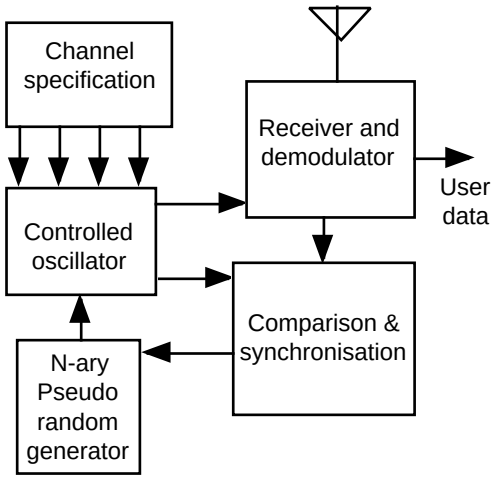
The spreading factor or processing gain is effectively a factor by which the signal/noise ratio is improved by using spread spectrum.

Spread Spectrum Receivers

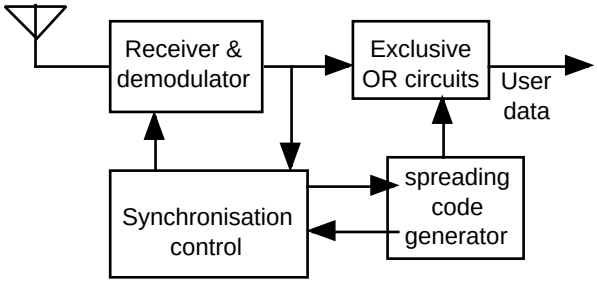
Receivers always work by hunting for a received sequence with the correct hopping or spreading code. They normally rely on very accurate time synchronisation between the receiver.

In one method a frequency hopping receiver might start with the "time of day" a little earlier than expected and run the hopping clock slightly fast. Eventually the received and expected hops will match and the clocks can synchronise.

A direct sequence transmitter might start a message with a known preamble to which a receiver can synchronise.



Frequency Hopping Spread Spectrum



Direct Sequence Spread Spectrum

Code Division Multiple Access (CDMA)

A traditional transmitter and receiver define a communication channel by the carrier frequency within a frequency band or spectrum (and perhaps the type of modulation). A spread spectrum system defines a communication channel by the pseudo-random spreading sequence which spreads the signal over the whole band.

Different spreading code define different channels. This is the basis of "code division" multiplexing; stations agree on a pseudo random sequence which gives them a private channel. Other "channels" may share the frequency space, each with its own pseudo random code. Thus we have multiplexing not by frequency but by code, hence "Code Division Multiplexing". As many users have simultaneous access to the same frequency spectrum, we have a form of multiple access, hence "Code Division, Multiple Access" or CDMA.

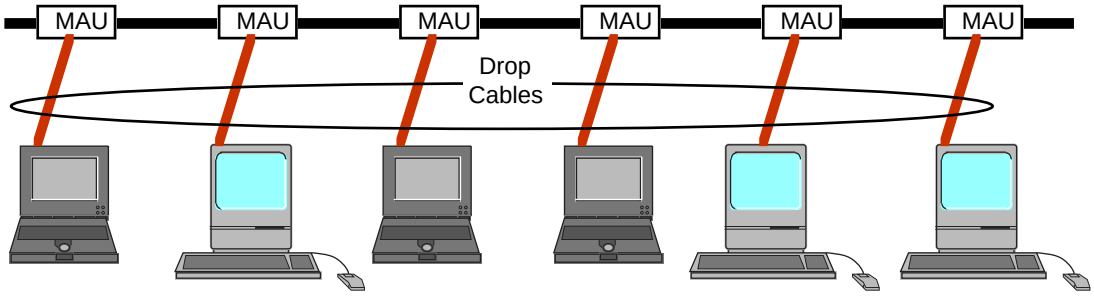
With slow spread-spectrum, collisions appear as bursts of noise, perhaps affecting a few bits.

With fast spread-spectrum a collision affects only part of a bit. Interfering stations just provide a background noise level which increases with more stations.

Direct-Sequence spread-spectrum suffers from a form of capture effect in which strong local transmitters overwhelm a weaker signal. The weaker signal is not just there as interfering background – beyond a certain difference in strength it just disappears!

Development of Ethernet

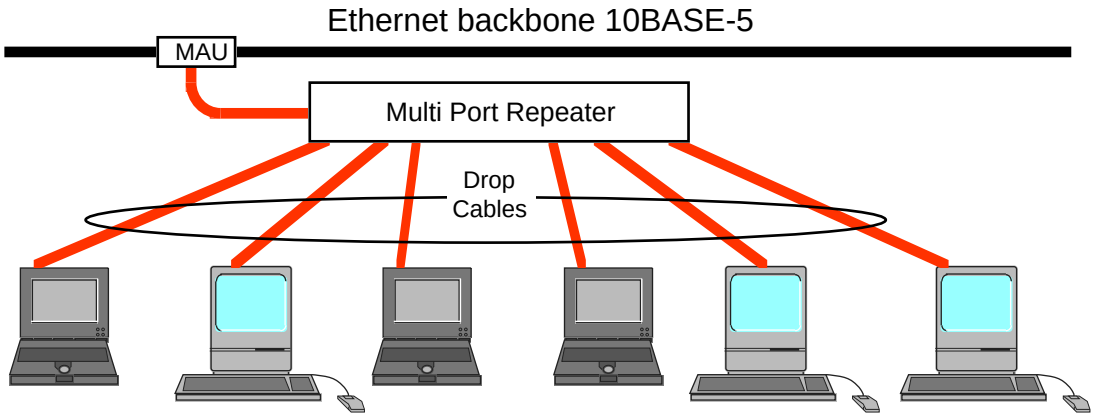
Original Ethernet was based on thick cables, now called “Thickwire” or 10BASE-5. Each device connected to the cable through a “Medium Attachment Unit” (MAU, a hand-sized box which clamped on to the cable) with a drop cable to the device. For electrical reasons, MAUs had to be at regular positions on the cable and no closer than about 1 metre.



A problem in a laboratory or other large concentration of computers is that a long cable may be required just to fit the necessary MAUs. The cable is not very flexible and it may be difficult to fit the necessary length into a room. It is also easy to exceed the the cable length and/or the number of physical connections to the cable.

Repeaters and switches

A solution is to connect the devices to the network through a multi-port repeater.



The Repeater provides several ports (often 8) each looking like an MAU, and has a single “device” port to allow connection to an MAU on the cable. The repeater detects collisions on “its” devices and from the cable. It passes appropriate collision signals back to the devices and a jam to the cable. The result is that all devices see standard Ethernet and follow Ethernet protocols, but there is only one connection to the cable.

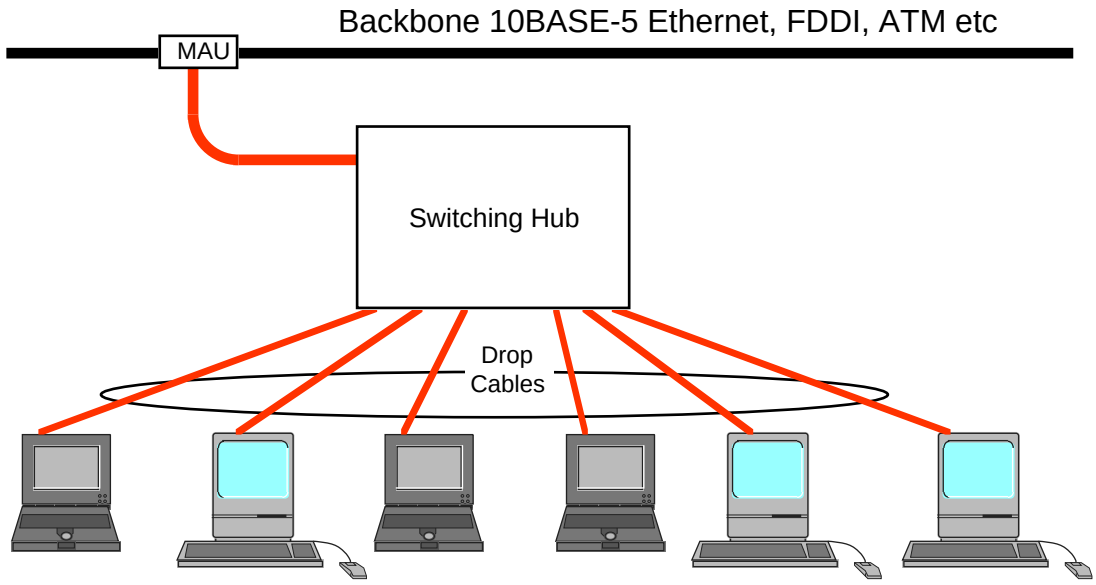
As it stands, the Ethernet Repeater is a relatively “dumb” device. With no change to protocols it can be changed to a switch (or switching hub) which does not rely on a single time-shared bus to connect its ports. This has several consequences –

1. Switches can be programmed to form “virtual LANs” so that even a broadcast message from one node is sent only to nodes on its virtual LAN, rather than to all nodes on the switch (or network).
2. Broadcast or multicast messages must be distributed by the switch to all stations, or to all stations on the virtual LAN.
3. Only stations directly involved in a transfer see the traffic. It is no longer possible for Ethernet adapters to operate in “promiscuous mode”, forwarding all network traffic to their host.
4. While only one of the attached devices can communicate with the backbone at any instant, two devices on the same switch can communicate *independently* of the main Ethernet. The system throughput can then exceed the 10 Mb/s of standard Ethernet. (This parallels the action of a Bridge in isolating Segments.)
5. The backbone need not be Ethernet. It can be 100 Mb/s FDDI, 150 Mb/s ATM, etc. Thus the switch can isolate not only user traffic from the backbone, but also user protocols. More importantly, although each user sees only 10 Mb/s, there may be many users communicating at once and much less contention for the shared backbone.

Changes to Drop Cables

The original drop cables connecting the device to the MAU are quite bulky and inflexible. One solution is to eliminate them completely, building the MAU into the device and using thinner, more bendable backbone cable. This leads to "Thinwire Ethernet", "CheaperNet" or 10BASE-2.

Another solution is to change the drop cables to a much lighter twisted-pair cable, similar to that used for telephone connections (*but not the same*!). With appropriate design these can still operate at Ethernet speeds, but over much longer distances than the original drop cables, giving 10BASE-T Ethernet. It is now convenient to extend the simpler repeater or switch to make a "switching hub" which receives all the 10BASE-T cables for a single floor, entire laboratory, etc. It is with 10Base-T (and later 100Base-T) that switches and hubs as described above become really important.



Traffic to another idle port may be forwarded immediately, but traffic to a busy port or to the backbone may be buffered until it can be sent. A collision may be signalled if no buffers are available or if the destination is unavailable. Even though the switching hub often contains buffers for each port, the ports still operate under standard Ethernet protocols as far as the user is concerned. The "collision" indication just signals that information cannot be accepted just then.

With experience in 10BASE-T connections over twisted pairs, people realised that operation was possible at even faster rates, up to 100 Mb/s still over twisted pair cables. The switch must be correspondingly faster and the backbone should usually be faster again, but the user still sees standard Ethernet but at 10 times the original speed. The "Fast Ethernet" 100BASE-T protocols are designed so that they are compatible with 10BASE-T over the same cable. A 100BASE-T switch or hub can recognise that a device is capable of only 10 Mb/s operation and fall back to 10BASE-T operation.

Thus even though the original Ethernet philosophy of contention access to a shared medium has largely disappeared, the user still sees a system with the same external appearance and the same protocols. With Fast Ethernet and even 10BASE-T, the collision indication has been made much more general; while the Ethernet Collision indication always meant "network overloaded" it was usually interpreted in just the sense by which Ethernet *detected* such overload.

Gigabit Ethernet

With the growing need for faster communications came a desire for even faster Ethernet. Gigabit Ethernet, with a bit rate of 1,000 Mb/s, just extends the experience already gained from Fast Ethernet. Communication with the hub is initially based on an 8B/10B code used with Fibre Channel. This resembles the 4B/5B code of FDDI, but encodes 8-bit units and has a many non-data or control codes.

Most of the parameters of gigabit Ethernet (frame size, attempt limit etc) are the same as for the 10Mb/s and 100 Mb/s, except that the slot time is extended to 512 byte times (4096 bit times). Although the minimum user frame is still 46 bytes, a short frame must be extended beyond the CRC and normal end of frame to force at least 1 slot time of activity. If a gigabit ethernet station has several short frames queued it may send several as a burst, to build the total activity up beyond the minimum 512 bytes to a maximum 8192 bytes .

IEEE 802.2 Logical Link Control

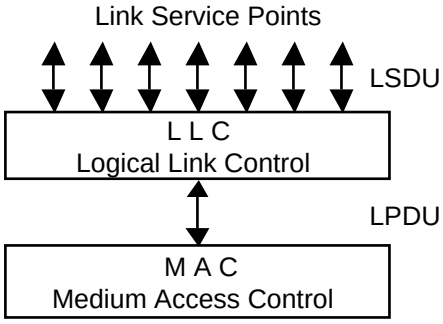
Traditional OSI

tThe IEEE 802.*x* standards define a series of related Local Area Network implementations. The three layers of the 802 specification are –

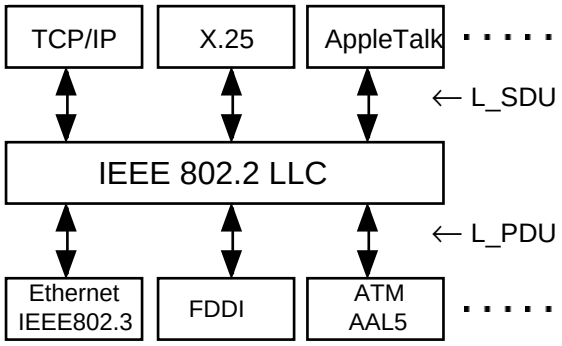
- The Logical Link Control (LLC, 802.2) provides a basic interface for both connectionless and connection-oriented services.
- Medium Access Control (MAC) provides for the transmission of data on the medium and also the defines the protocols which maintain the link operation.
- The Physical Layer provides the actual physical connection for data transfer. The Physical and MAC layers are specific to a network type, but present a common interface to the LLC layer.

IEEE 802.2 – Logical Link Control

This is the upper part of the DataLink layer for the 802.*x* networks and provides a common interface to the Network Layer from the different media. It accepts *Link Service Data Units* from the Network Layer and delivers *Link Protocol Data Units* to the MAC layer (and the several Link Service Access Points are multiplexed on to the basic service and effectively define sub-addresses for the node.



An actual LLC layer might have several systems above it and several physical protocols below it



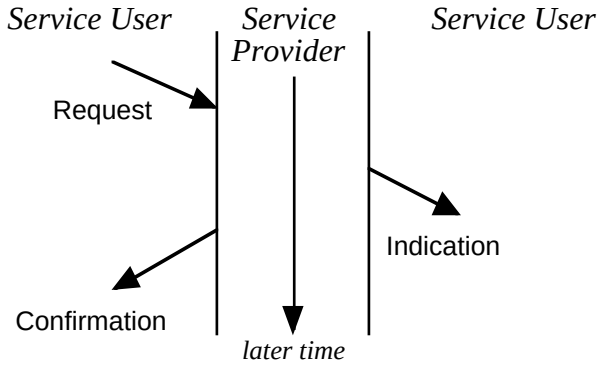
Protocol diagrams

These diagrams indicate the messages exchanged between a service user and a service provider. The basic timing diagram is widely used in data communciations to indicate a time sequence of messages. The basic style of message is well-established in OSI protocols and often carried over into other areas.

The 802.2 LLC protocols use three message types —

- A **Request** is passed down to request a service
- It appears at the "peer" service user as an **Indication**
- A **confirmation** is returned to the requester.

Other protocols may use more message types



Logical Link Control Services

The services are provided by subroutine calls (or equivalent) to the LLC. The most important is the unacknowledged connectionless service, with the two primitives —

L_DATA.request
L_DATA.indication

with basic functions

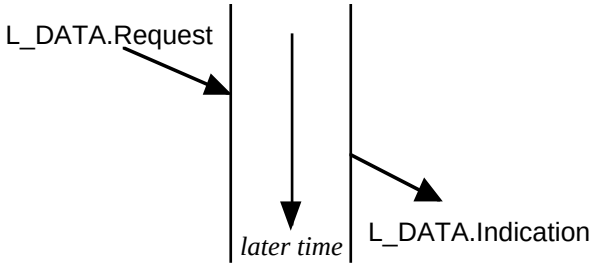
```
L_DATA.request(local_address,
               remote_address,
               l_sdu,
               service_class)
```

and

```
L_DATA.indication(local_address,
               remote_address,
               l_sdu,
               service_class)
```

Thus the LLC layer is called with a request to pass "`l_sdu`" (user data) from "`local_address`" to "`remote_address`", or to receive `l_sdu` from a remote address. There is no acknowledgement (ie confirmation); the `l_sdu` is just sent and appears.

The protocol diagram is trivial —



The address parameters provide at least the concatenation of the MAC address field and the LLC address field (SSAP or DSAP – Source Service Access Point and Destination Service Access Point). The remote_address for the L_DATA.request may be a broadcast address. The L_DATA.indication returns the identical LSDU as was provided to the matching L_DATA.request.

MAC Service Data Units

The Logical Link Control Layer supplies `m_sdu` (MAC service data units) for transmission by the MAC layer. The format of the `m_sdu` is given later, but uses the primitives –

```
MA_DATA.request(
          destination_address,
          m_sdu,
          requested_service_class)
```

and the corresponding –

```
MA_DATA.indication(
          destination_address,
          source_address,
          m_sdu,
          reception_status,
          requested_service_class)
```

Link Protocol Data Units.

The LLC layer receives information (the LLC SDU) through one of its "Service Access Points" (SAPs), and delivers it to its MAC layer, or vice versa. The information transferred through the network must specify the "Source Service Access Point" (SSAP) and "Destination Service Point" (DSAP); this is held in the LLC header prefixed to the SDU.

For simple traffic the LLC header has the form

DSAP address	SSAP address	Control	Information
8 bits	8 bits	8 bits	8*M bits

The Service Access Points are given "well known" addresses for the usual cases, and the Control byte has the value 0x03 for connectionless data. For data transfer the "Information " field is the LLC SDU data.

For routed ISO protocols (X.25 etc) the LLC Header value (in hexadecimal) is 0xFE-FE-03.

For protocols such as IP there is a further level of encapsulation. The LLC Header, with address 0xAA, is followed by a "SubNetwork Attachment Point (SNAP) Header". The 5-byte SNAP Header has a 3 byte (24 bit) code giving an "administering authority" and 16 bits to identify the protocol (00-80 for IP). The full prefix is then

0x AA-AA-03	LLC Header
0x 00-00-00 08-00	SNAP Header

When transferring AppleTalk the SNAP header value is 0x 08-00-07 80-9B.

In total, an 8-byte header is added to the user message (LLC_SDU) in forming the LLC_PDU submitted to the MAC layer.

Many systems use Ethernet conventions, rather than IEEE 802.3. The 802.3 length *field becomes a* protocol type *field, set to denote the traffic type.*

SNAP headers

The subnetwork attachment point headers from the previous section (usually expanded to the 8-octet form) are important because they give a well-defined way of transporting one protocol over another and identifying the transported protocol. Many many protocols and data transfer systems use the SNAP headers in this way.

Other LLC Control values

The control byte (and indeed the whole LLC Header) is based on X.25 conventions. Two other values are used by the connectionless service —

LSB MSB

1	2	3	4	5	6	7	8	
1	1	1	1	P	1	0	1	XID Exchange IDs
1	1	0	0	P	1	1	1	Test
1	1	0	0	P	0	0	0	UI Unnumbered Info.

The Poll/Final bit (P) may be 0 or 1. (A UI field with P=0 has the value 0x03, as noted before.)

The **XID** command is used to exchange information on the supported LLC types and the receive window size (amount of unacknowledged data).

The **test** command is echoed to show that the link is working.

Note that the bit order is reversed from "normal", with the least-significant on the left.

Service Access Point Addresses

The address LSB (leftmost) bit is 0 for unique and 1 for group addresses, 00000000 defines a null address and 11111111 is a broadcast address, usually intended for all destination SAPs.

Connection oriented :

This section is not examinable, but is included for completeness and to give you an idea of a standard set of primitive operations.

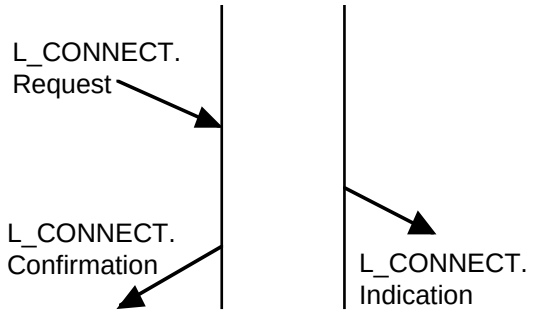
While the 802.2 standard certainly defines a comnection oriented service, this seems to be seldom used. The basic operations are —

L _ CONNECT	*. request*
L _ DATA _ CONNECT	
L _ DISCONNECT	*. indication*
L _ RESET	
L _ CONNECTION _ FLOWCONTROL	*. confirmation*

The services are very similar to those associated with ISO X.25 communication; L_DATA_CONNECT replaces L_DATA, and a new primitive is added — L_CONNECTION_FLOWCONTOL. (In the connectionless protocols, 802.2 replaces L.UNITDATA by L.DATA.)

Each Link Service Primitive specifies addresses which are as `Dest_Adr.LSAP`. Connectionless data and L_CONNECT may specify a “service class” or priority for the transfer. FLOWCONTROL specifies “the amount of data which may be passed” and may be any agreed value, including zero or infinity.

For all of these services the protocol diagram is like —



The **connection** is initially requested by –

```
L_CONNECT.request(
                local_address,
                remote_address,
                service_class)
```

which appears at the destination as –

```
L_CONNECT.indication(
                local_address,
                remote_address,
                status,
                service_class)
```

and at the source station as –

```
L_CONNECT.confirm(
                local_address,
                remote_address,
                status,
                service_class)
```

A **data transfer** is initiated by the command –

```
L_DATA_CONNECT.request(
                local_address,
                remote_address,
                l_sdu)
```

with the corresponding indication –

```
L_DATA_CONNECT.indication(
                local_address,
                remote_address,
                l_sdu,
                service_class)
```

and confirmation –

```
L_DATA_CONNECT.confirm(
                local_address,
                remote_address,
                l_sdu,
                service_class)
```

Service disconnection is provided by –

```
L_DISCONNECT.request(
                local_address,
                remote_address)
```

with the remote indication –

```
L_DISCONNECT.indication(
                local_address,
                remote_address,
                reason)
```

and local confirmation –

```
L_DISCONNECT.confirm(
                local_address,
                remote_address,
                status)
```

The connection may be **reset**, discarding all unacknowledged PDUs, by –

```
L_RESET.request(
                local_address,
                remote_address)
```

with the remote indication –

```
L_RESET.indication(
                local_address,
                remote_address,
                reason)
```

and local confirmation –

```
L_RESET.confirm(
                local_address,
                remote_address,
                status)
```

Finally, the **amount** of traffic may be varied by the **FlowControl** primitive –

```
L_CONNECTION_FLOWCONTROL.request(
                local_address,
                remote_address,
                amount)
```

which has the indication –

```
L_CONNECTION_FLOWCONTROL.indication(
                local_address,
                remote_address,
                amount)
```

The amount may be zero to stop the transfer, or may be set in “implementation specific units”.

The LLC Control field may be 8 or 16 bits, defining three basic control classes.

Information Transfer Command/Response (I-Format PDU)	0	N(S)						P / F	N(R)						
Supervisory Command/Response (S Format PDUs)	1	0	S	S	X	X	X	P / F	N(R)						
Unnumbered Command/Response (U-format PDU)	1	1	M	M	P / F	M	M	M							

The commands and responses are –

Commands	Responses	
All services		
UI		Un-numbered information
XID	XID	Exchange IDs
TEST		Test
Connection oriented		
I	I	Information
RR	RR	Receive Ready
RNR	RNR	Receive Not Ready
REJ	REJ	Reject
SABME		Set Asynch Bal Mode Extended
DISC		Disconnect
	UA	Unnumbered Acknowledge
	DM	Disconnected Mode
	FRMR	Frame Reject

The “Poll/Final” bit (P/F) may be either 0 or 1 in a command (Poll bit). In the response (Final Bit) it must reflect the value of the Command Poll bit.

The bit codings for these messages are –

Supervisory Command/Response

1 0	0 0	X X X X	P/F	N(R)	RR – Receive Ready
1 0	0 1	X X X X	P/F	N(R)	REJ – Reject
1 0	1 0	X X X X	P/F	N(R)	RNR – Receive not Read

Unnumbered Command/Response

1 1	0 0	P	0 0 0	UI
1 1	1 1	P	1 0 1	XID
1 1	0 0	P	1 1 1	TEST
1 1	1 1	F	0 0 0	XID
1 1	0 0	F	1 1 1	TEST
1 1	1 1	P	1 1 0	SABME
1 1	0 0	P	0 1 0	DISC
1 1	0 0	F	1 1 0	UA
1 1	1 1	F	0 0 0	DM
1 1	1 0	F	0 0 1	FRMR

The entries with a “P” are commands, and those with a “F” are responses.