IEEE 802 LLC Stack Architecture

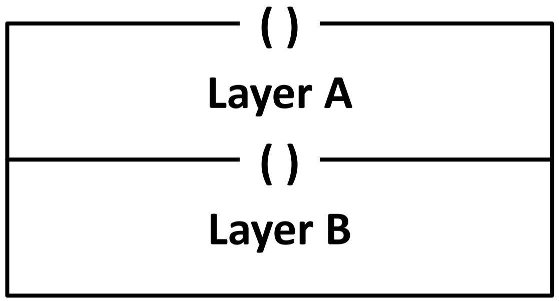
Norman Finn
Huawei Technologies Co. Ltd
nfinn@nfinnconsulting.com

802-finn-LLC-stack-2024-01-v01.pdf

Stacking and parameterization

In a protocol stack, each layer adds protocol information to the data unit on the way down the stack, and removes the protocol information on the way up the stack.

Sometimes the protocol information hangs on as a parameter on the upper Service Access Point. Of course, that makes it more difficult to mix and match layers at the same level, and to invert, stack layer order.



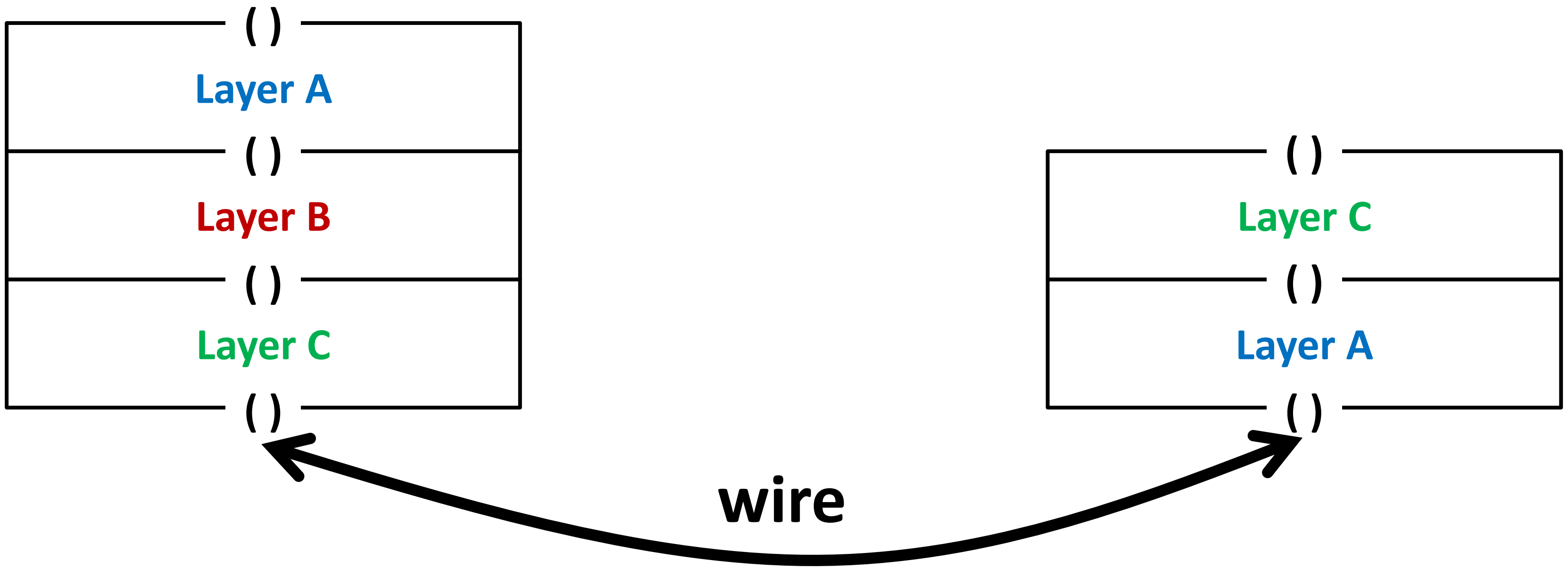
Parameters (x, y, z, a’, data(b, c))

Parameters (x, y, z, data(a, b, c))

Stacking and peers

Protocols must be peered with the same protocol in another system.

The following ~~is not allowed~~ will not interoperate:

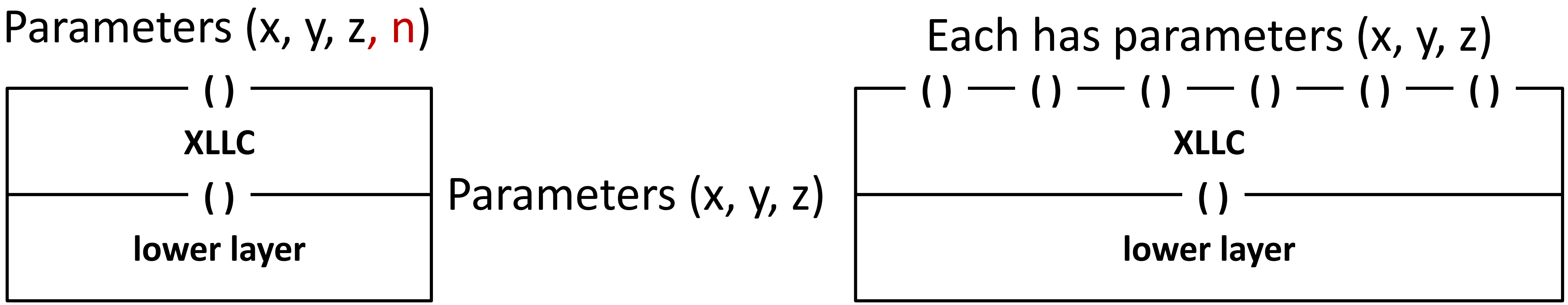


Multiplexing and parameterization

Any sublayer that provides a multiplexing function can be represented as either:

- An array of demultiplexed SAPs above a multiplexed SAP.
- A single upper SAP with an additional parameter identifying the SAP.

Don’t let this fact confuse the basic issues in this presentation.



Parameters (x, y, z, n)

( )

XLLC

( )

lower layer

Parameters (x, y, z)

Each has parameters (x, y, z)

( ) — ( ) — ( ) — ( ) — ( ) — ( )

XLLC

( )

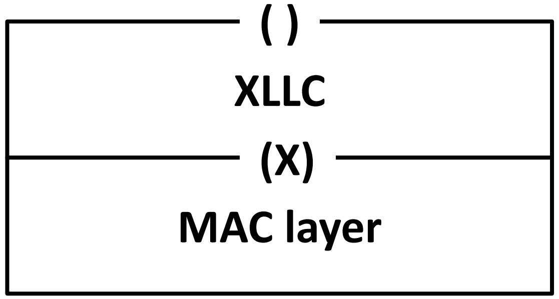
lower layer

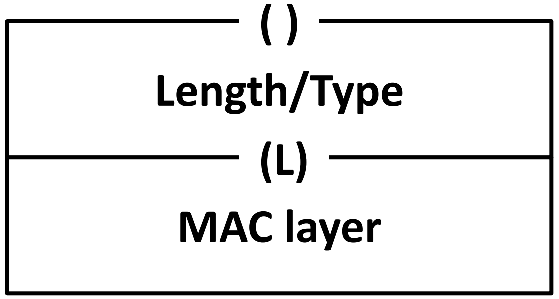
Two kinds of MAC SAPs

There are (at least the following) two kinds of service access points (SAPs) provided by IEEE 802 media.

- XLLC SAPs (I use “XLLC” to prevent confusion with any current or proposed definition of “LLC layer”).
- Length/Type SAPs.

An implementation is expected to attach the appropriate layer, XLLC or Length/Type, on top of the MAC SAP.

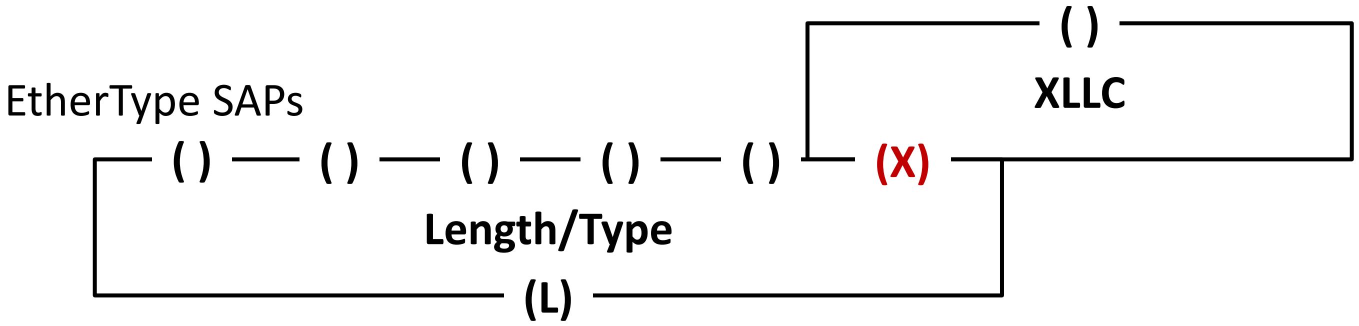




Length/Type SAP

The Length/Type layer sits on a Length/Type SAP. It multiplexes 64,001 ways according to the first two bytes of the MSDU, which it deletes or adds as the frames go up and down the stack. (See note on next page)

- On one SAP (corresponding to values < 0x600) you place the XLLC layer.
- The other 64000 SAPs are best thought of as protocol decode points governed by RAC EtherType assignments.



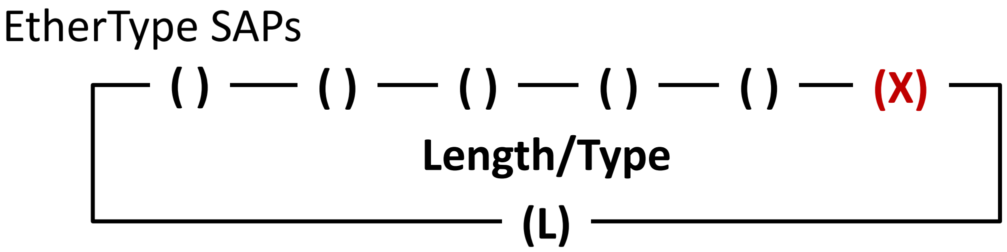
Who adds/removes the EtherType or LLC?

People can and do argue about whether the Length/Type is a layer, and whether the upper layer or the Length/Type layer supplies and/or deletes the Length and type from the beginning of the MAC Service Data Unit (MSDU). But the model on the previous page is consistent, logical, and is a single size that fits all.

If one likes a single upper SAP, the EtherType is supplied either by a parameter or in the MSDU (see page 2) in both directions.

If one likes an array of SAPs, one per EtherType, it is natural to think of the Length/Type layer as adding/removing the EtherType.

Typical implementations use a single down-SAP, and multiple up-SAPs, and the upper layer sees (ingress) and supplies (egress) the EtherType.

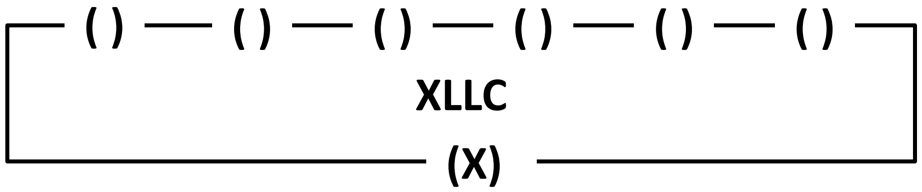


XLLC layer

The XLLC layer performs a number of functions. Among them is the provision of LSAP decodes. These can be conveniently represented as a multiplexing of the lower-layer SAP by SSAP and LSAP. Each SAP address has 8 bits. Whether this results in a 65536-way demultiplex or a 32768 demultiplex and a command/response parameter is not critical (see page 4). It adds/deletes the LLC information as the frames goes down/up the stack.

Most of these multiplexed SAPs (e.g. S=28, L=5C) are best thought of as LSAP/SSAP pair sub-address mux/demux points.

Some of these multiplexed SAPs (e.g. S=AA, L=AA) are best thought of as protocol identifier mux/demux points.

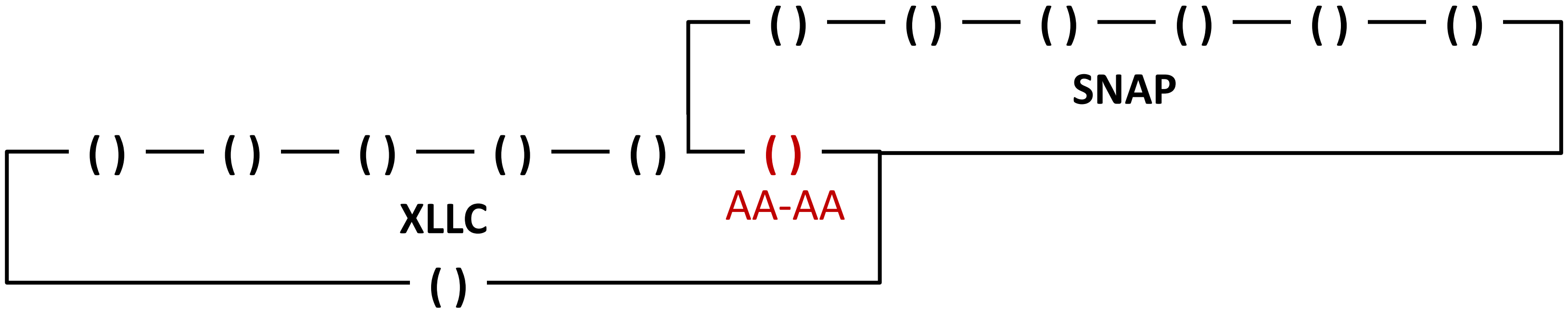


SNAP layer

SNAP provides a multiplexing function. It adds/delete the SNAP data (CID and protocol ID) that follow the AA-AA-0C LLC header as the frames go down/up the stack.

It must attach to the XLLC demux point AA-AA.

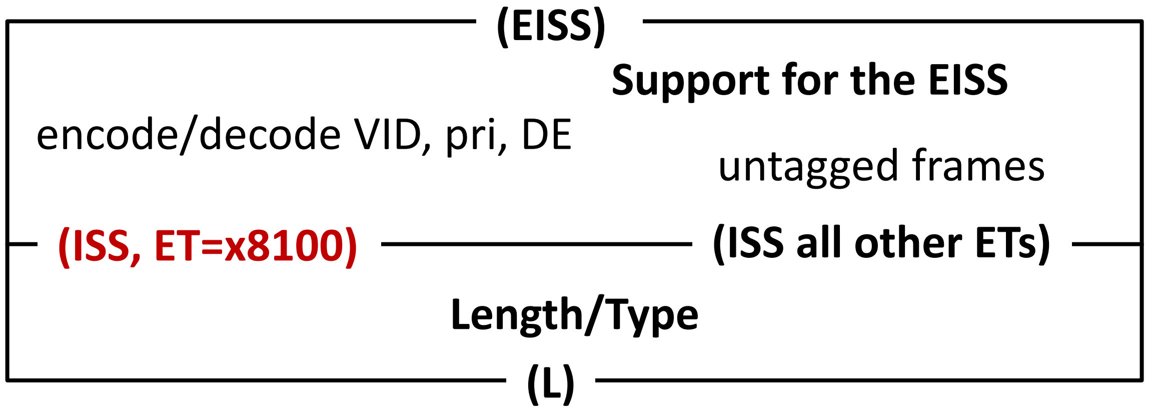
It offers 1,099,511,627,776 demux points (three bytes CID, two bytes protocol ID), usually thought of as protocol decode/encode points.



VLAN layer type 1 (EISS)

The EISS layer, defined in IEEE Std 802.1Q, encodes/decodes VLAN ID, priority, and drop eligible fields. Typically, VLAN ID, priority and DE are parameterized (see page 2). In an end station, the VLAN could be multiplexed(see page 4).

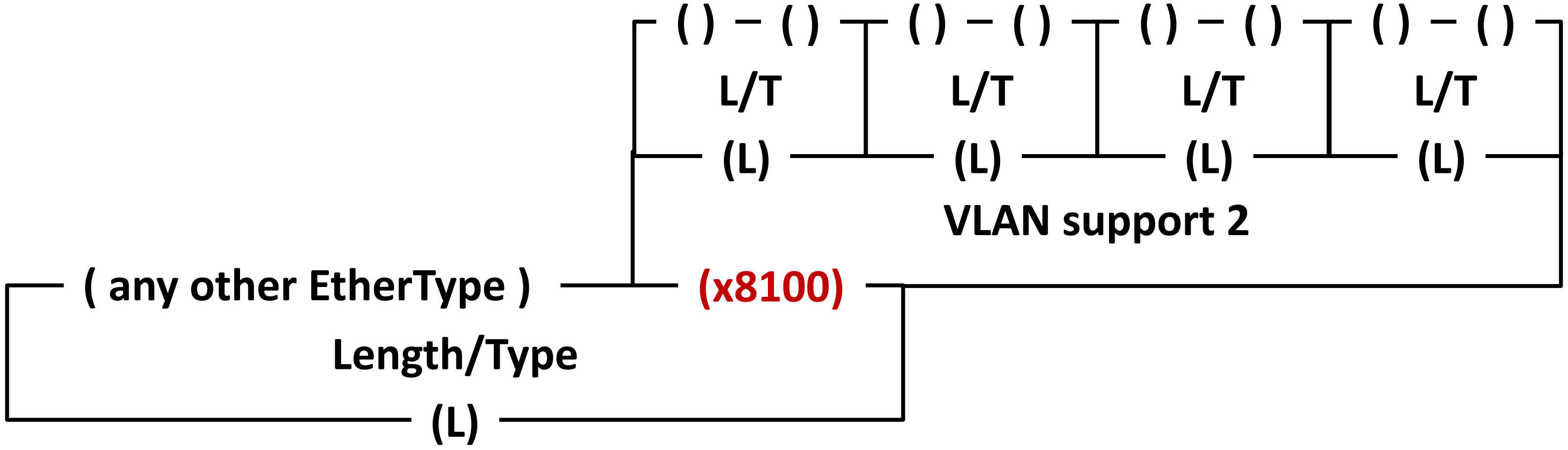
If multiplexed there would be 4094 SAPs; xFFF is not used. x000 and untagged are both the same as the default VLAN (1-0xFFE). The difference between 0x000, untagged, and default VLAN value is not visible above the EISS.



VLAN layer type 2 (common end station, also Bridge "pocket")

The lower Length/Type directs/receives untagged frames on the "any other EtherType" SAP.

VLANs are multiplexed to an array of VLAN SAPs, each of which has a mandatory Length/Tag layer. This is explicitly specified in the definition of the VLAN tag(s), and is typical of most other tags.



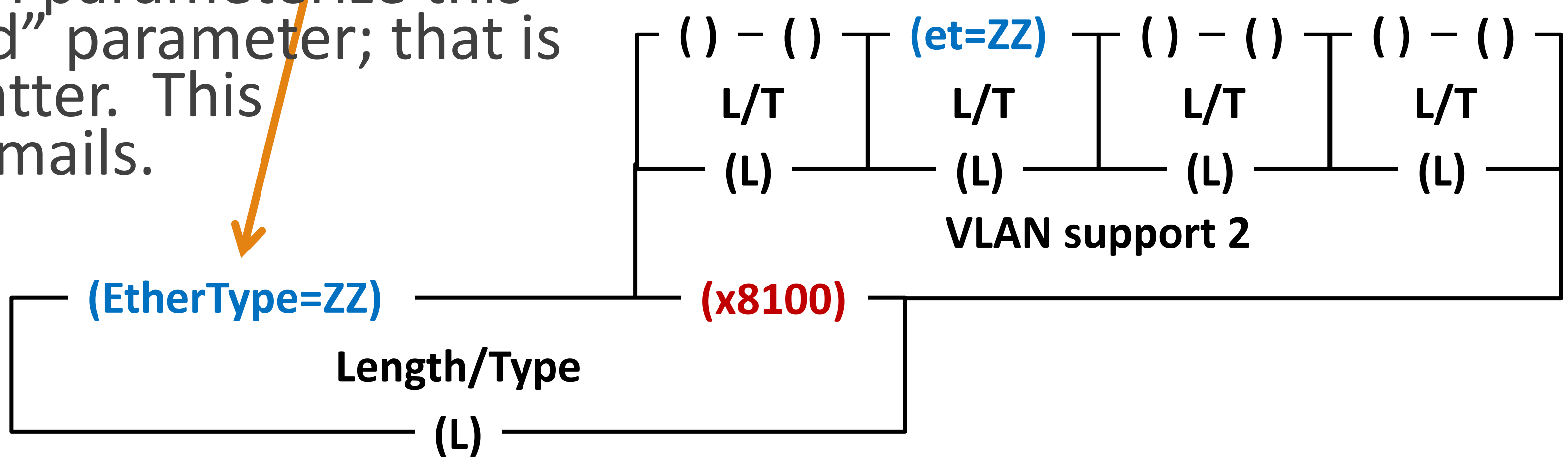
VLAN layer type 2
(common end station, also Bridge “pocket”)

These two SAPs, both for EtherType ZZ, are not equivalent.

One is tagged, the other is not.

Some protocols work differently, depending on whether the frame is tagged.

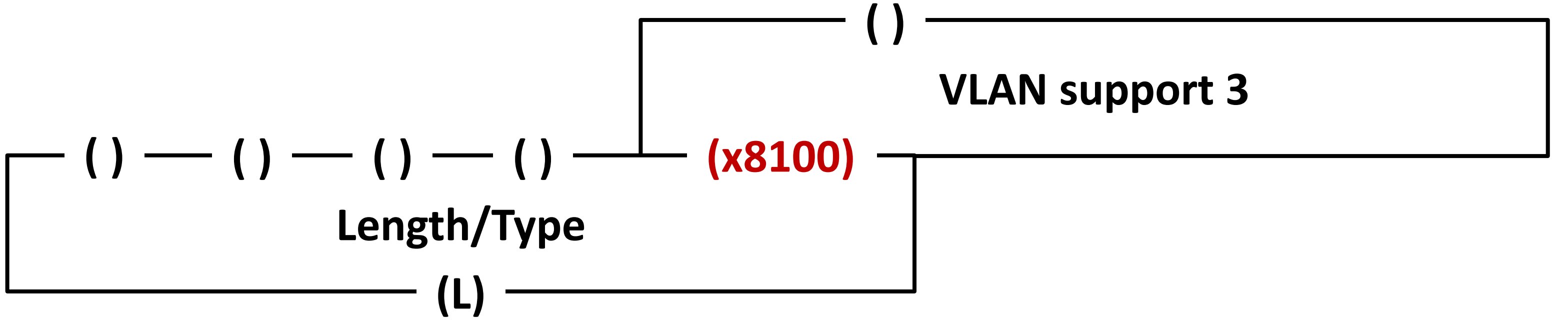
An implementation can parameterize this with a ”frame is tagged” parameter; that is an implementation matter. This architectural model remails.



VLAN layer type 3 (“untagged” end station)

IETF VLAN port, defined in RFC encodes/decodes VLAN ID, priority, and drop eligible fields. Typically, priority and DE are parameterized (see page 2).

VLAN ID is ignored on ingress. VLAN ID 0 is used on egress.

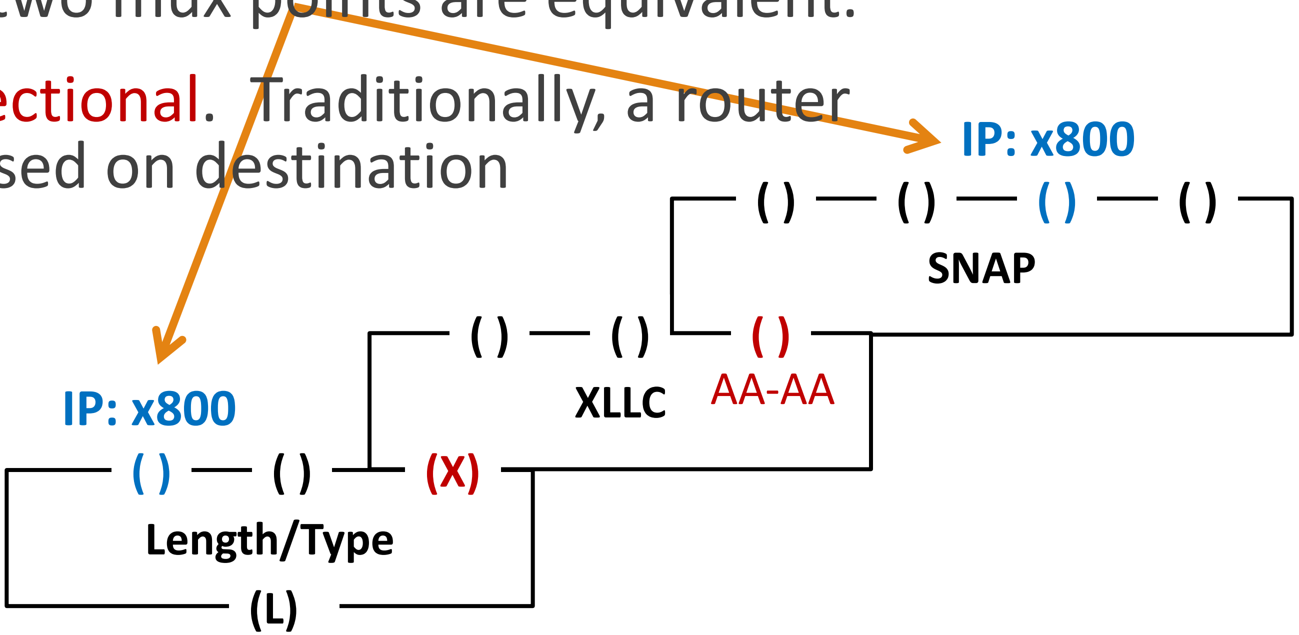


SNAP vs EtherType (1)

Some protocol specification organizations, e.g. the IP protocols from IETF, support both Length/Type encoding and SNAP encoding of IEEE 802.3 Ethernet frames.

That does **NOT** mean that these two mux points are equivalent.

Why? Because the stack is bi-directional. Traditionally, a router chooses one or the other SAP based on destination IP address (see [note on page 2]()).



SNAP vs EtherType (2) a history lesson

The “LLC” as a single demultiplexing layer made sense as long as IEEE 802 maintained the fiction that the IEEE Std 802.2 LLC layer was the “real” IEEE 802 architecture, and Length/Type was an IETF/DEC/Intel/Xerox legacy, soon to be made obsolete.

The LLC/SNAP and EtherType mux points have never been equivalent. This was not a problem in the early days; you used LLC/SNAP on token ring, and EtherType on Ethernet.

When we started bridging between token ring and Ethernet, this was a problem because end stations using different encapsulations couldn’t talk.

One vendor solved this by insisting that Ethernet use LLC/SNAP for all protocols, like token ring.

Another solution was to translate between LLC/SNAP and Length/Type when moving between token ring (or now, many 802.11 media) and Ethernet. But, some protocol defining organizations had other opinions, leading to IEEE Std 802.1H, where exceptions were made on an EtherType-by-EtherType and medium-by-medium basis.

While it is true that those exceptional protocols have largely died out, that does not make the LLC/SNAP and EtherType mux points equivalent. The choice is available, even if not used as often as it once was.

And so on

The reader should now have no trouble at all drawing the “Here is an LLC frame longer than 1500 bytes” EtherType layer.

The definition of a tag is a protocol that requires either the Length/Type or the XLLC layer be present at the next-higher layer. Since 802.1Qbz, tags always require Length/Type, not XLLC. Again, all are trivial to draw in this model.

Conclusion

- Basic layering works. The LLC is in no way a special case.
- In the model, each layer adds/subtracts its information to/from the MSDU (the packet) as the stack is traversed. Implementation can vary.
- The fact that layer information can be parameterized and/or multiplexed must not be allowed obscure the basic architecture.
- Yes, this can create a stack diagram resembling Sleeping Beauty’s Castle. Welcome to layered networking.
- By the way, arguments such as “what are the exact descriptions of EPD and LPD” are either trivial, or become irrelevant (and the terms obsolete), if the model in this document is followed.

THE POINT

- All protocols stack. In particular, tags stack. Protocols, and therefore tags, must be stacked so as to peer properly. A tag can be used multiple times in a stack. Peered layering determines who looks at which instance of a tag. Drawing a box around all tags and calling it a layer is very misleading, as it obscures the peering relationships.
- Protocol SAPs identified by different protocol discrimination stacks are not always equivalent because choices for transmitted encapsulations must be made. Drawing a box around all protocol discrimination and calling it a layer is very misleading, as it obscures this distinction.
- All tagging and protocol discrimination relationships can be described using only the stacking of protocol blocks. Without the block stack, additional parameters and rules are required to express these stacking and peering relationships. Drawing a box around all protocol discrimination and tags and calling it a layer adds complexity unnecessary to the architecture.

OK, Norm. That’s what you want to not do. So, what do you want to do.

- See https://www.ieee802.org/1/files/public/docs2020/maint-seaman-protocol-ids-in-std-802-0520-v01.pdf for 17 pages of text that could well go into P802-rev.
- See https://www.ieee802.org/1/files/public/docs2020/maint-seaman-llc-goodbye-0720-v00.pdf for a presentation showing why that text is a good idea.

Thank you