Protocol entities, service access points, and YANG models

Mick Seaman

This note is a result of discussion with Marc Holness. Errors, omissions, and opinions are mine. At the Budapest 802.1 interim meeting Norm Finn and others asked an important question. Does the IETF interface YANG model manage a service access point (SAP) [which the reference model would consider to be an instance of a service interface] or the protocol entity supporting that SAP? In the simplest cases a protocol entity supports a single SAP using the service provided by another single (lower) SAP and ‘interface’ may be considered equivalent to ‘port’ in IEEE 802.1 standards (802.1AC-2012 7.4): referring quite generally to an entry in a bridging table, the SAP provided by the interface stack, the whole of the stack, protocol entities in the stack, or the media connector. Where multiplexing and/or demultiplexing are provided within the interface stack, greater precision is required. IETF experts have been strong advocates of augmenting their interface model. Augmentation avoids replicating the unicast, multicast, and broadcast statistics that are part of the interface model. This works well in simple stacks provided that the stack order of the augmenting components is obvious and no instance of a component represented by a single model augmentation can appear more than once[1]. Avoiding a YANG development path that cannot manage functionality provided by existing standards or that over constrains future standardization is a concern[2].

This note uses MACsec (802.1AE) and (in the future) Link Aggregation (802.1AX) as real examples, but attempts a general analysis.

One immediate conclusion is that 802.1X PAE instances should be indexed by controlledPortNumber rather than by uncontrolledPortNumber (as currently in the MIB).

1. Terminology

This note uses the term ‘interface’ only when referring to the IETF interface YANG model and the data and control aspects associated with a (possibly augmented) instance of that model. Service access points are referred to as SAPs, and protocol entities as entities.

When describing graphs I use ‘node’ or ‘nodes’ where some might use ‘vertex’ or ‘vertices’. I believe all are agreed on ‘edge’ (‘edges’) as the connection(s) between the nodes (vertices). A principal goal of this note is to render apparent individual details that might be merged and hence confused, so interface stacks are described as bi-partite directed graphs. That is to say as graphs in which two sorts of node alternate, in this case nodes that are SAPs alternating with nodes that are entities with (to pick a convention) edges shown as arrows from each SAP’s user (client) to the SAP, and from the SAP to the supporting entity(ies). Possible representations of these graphs in terms of YANG model instances (some but not necessarily all of which may be ‘interfaces’) are overlayed on these graphs.

[1]Some of the issues raised may already be addressed by IETF documentation with which I am unfamiliar. The discussion in RFC2863 3.1 is still very relevant.
[2]It is not clear that we need the statistics component of the IETF interface model for every interface in a stack that uses more interfaces (each with a distinct if-type, if-index, and higher-layer-if and lower-layer-if references) to provide flexibility. See 6 below. We may better off augmenting something simpler

2. MACsec interface stacks

Figure 1 shows a MACsec protected port in an end station[3]. The pertinent multiplexing issues are illustrated by including two LLDP agents, one (using the Nearest Bridge group address) supports power over ethernet (PoE) negotiation and necessarily uses the Uncontrolled Port, while the other shares and receives protected information[4].

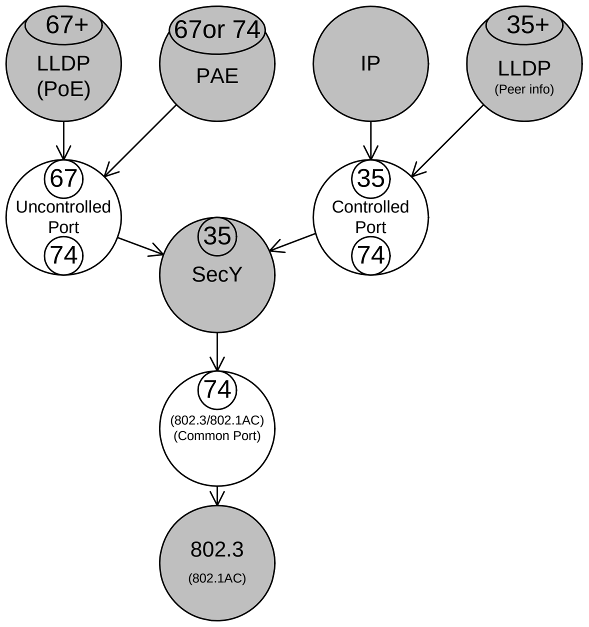


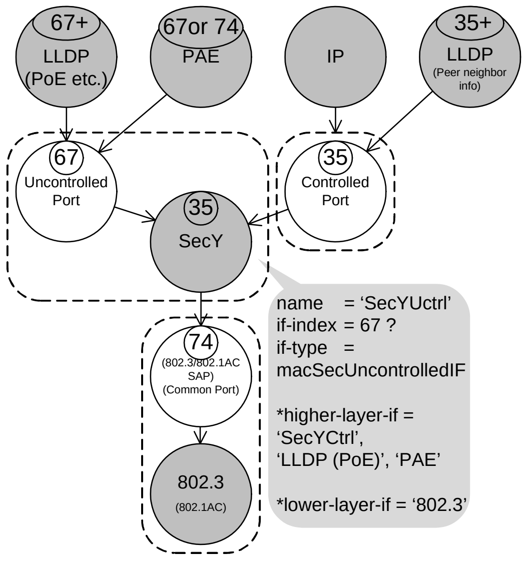
Figure 1—MACsec protected end station

The larger circles depict protocol entities (shaded) and the SAPs that connect them (clear). The (arbitrarily chosen) numbers in the figure represent MIB information. Within each of the SAPs, the upper small circle contains its own ifIndex, while the lower (if present) contains the ifIndex of the supporting sub-layer. So, for example, one of the {higher layer, lower layer} entries in the ifStackTable will be {35, 74} reflecting the relationship between the SecY's Controlled and Common Ports.

The upper circle in (some of) the protocol entities (shaded) shows how that entity is indexed within its own MIB. Each LLDP agent is identified by the combination of the ifIndex of the SAP/interface and the destination MAC address that it uses[5]. The PAE is indexed (in the IEEE8021X-PAE-MIB) by the ifIndex of the Common Port (if it is controlling a real port) and by the ifIndex of the Uncontrolled Port (if controlling a virtual port)[6]. However indexing a real port's PAE by the Common Port doesn't remove the need (in the MIB) to allocate an ifIndex for the UncontrolledPort[7], though the MIB is actually inconsistent on this point. The Common Port and the Uncontrolled Port have different ifTypes and different statistics, though the statistics for the latter can be derived from those for the Controlled and Common Ports[8].

In addition to the indexes shown in Figure 1, the PAE and SecY MIBs both contain MIB specific pointers. If (for example) the PAE shown is indexed by the Common Port's ifIndex (74 in the figure), then its associated SecY can be found without having go down the ifStack table and up the inverted Interfaces Stack Table (ifInvStackTable)[9].

It would be nice if we could pick the simplest representation of Figure 1 in YANG, taking advantage of augmentation. Unfortunately the entities above the SecY are attached to two distinct SAPs with different properties (oper-status in particular)[10]. A game we can play is to try to cover the maximum number of entities and SAPs with the minimum number of augmented interfaces. Figure 2 shows one attempt:



name = 'SecYUctrl'
if-index = 67 ?
if-type = macSecUncontrolledIF

*higher-layer-if = 'SecYCtrl', 'LLDP (PoE)', 'PAE'

*lower-layer-if = '802.3'

Figure 2—A possible YANG mapping

[3]A realistic graph for a MACsec protected Bridge Port interface stack necessarily includes detail irrelevant to the present discussion.

[4]For mapping topology, for example, or that can be relied on to identify mismatched configuration information.

[5]So two LLDP agents that use same MSAP, each using a different group address, can be indexed without requiring additional interfaces/ifIndexes.

[6]See 802.1X-2010 12.9.2. Consult 802.1X (don't try guessing) for what a virtual port is in this context.

[7]See 802.1X 13.3.2. However in the MIB the ieee8021XPaeUncontrolledPortNumber OBJECT-TYPE DESCRIPTION claims that this can have the same index as the Common Port and references 12.9.2 (incorrectly) as its authority. That won't work in the ifStack. It's not clear what implementations do for the usual case of Real Ports (it may be that the macSecUncontrolledIF ifType is not used).

[8]See 802.1X 6.4.3. The Uncontrolled receive stats are identical to those of the Common Port, the transmit stats are Common's minus Controlled's.

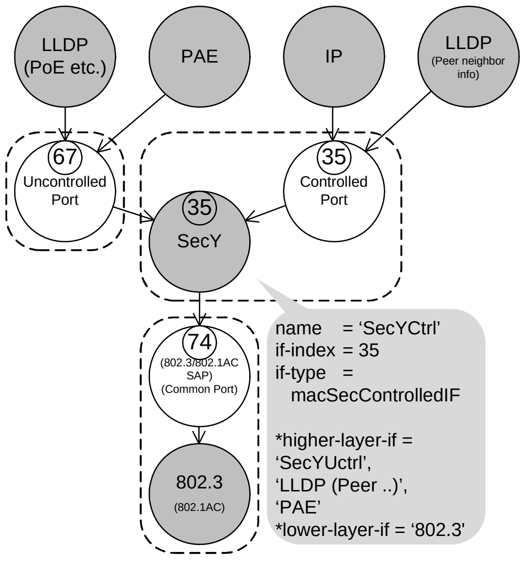
[9]But note that making these direct associations requires knowledge of the specific MIB, which seems an onerous requirement.

[10]The network manager really needs to know this fact, though the attached entities remain the same whether they are or are not using secured service.

This not ideal. While it succeeds in representing the protected service delivered to the IP and LLDP Entity in the interface stack by including an interface of type macSecControlledIF, it leaves the Controlled Port as a separate augmentation of an interface. RFC 7223 says the interfaces 'mapping [of if-index] to ifIndex used by ... SNMP ... must be clear', but the SecY's parameters were indexed by the Controlled Port's ifIndex (35 in the figure) rather than that of the Uncontrolled Port. Worst of all the parameters most closely associated with the Controlled Port are in a different interface.

NOTE—Although RFC 7223 YANG interface model's higher-layer-if and lower-layer-if lists are a substitute for MIB ifStackTable functionality they are list of "name"s (each usually mapped to ifName, at least for interfaces, and uniquely identifying each interface instance). They are data node names, so I have assumed (but 7223 does not say) that the higher-layer-if names at the top of the stack reference the entities making use of the stack (these would not have ifNames, of course). The names in quotes in the figure represent the real names (assigned by the system in some user friendly way when the interfaces are created, I assume) so there would not be two interfaces with the name 'SecYUctrl' but perhaps (for example) one 'SecYUctrlPort1' and one 'SecyUctrlPort2'.

Figure 3 shows another unsatisfactory mapping:



name = 'SecYCtrl'
if-index = 35
if-type = macSecControlledIF

*higher-layer-if = 'SecYUctrl', 'LLDP (Peer ..)', 'PAE'
*lower-layer-if = '802.3'

Figure 3—Another possible mapping

This time the Controlled Port has been included in the SecY interface (associating its management variables with the correct interface) at the cost of throwing out the Uncontrolled Port (into an unsatisfactory augmented interface of its own), since there can be more Uncontrolled Port attached entities that the LLDP (PoE) shown. It also has the strange effect of putting the Uncontrolled Port on top of the Controlled.

The PAE instance could be included within the SecY interface in Figure 2, though not in Figure 3 as it would then be using the service provided by an interface (the Uncontrolled Port) that is one of its own higher interfaces.

Faced with these two unsatisfactory alternatives, what should we do?

The 'Y' function that provides promiscuous receive at the bottom of the SecY and that multiplexes the protected and the unsecured frames was included within the SecY specification to:

a) make the (possibly) promiscuous reception[11] explicit,

(b) avoid any dispute with (possibly numerous) providers (and standardizers) of Common Port services as to whether their interfaces would or would not inherently provide that capability.

If we simply insist on the availability of the Common Port functionality in the management model,[12] we can redraw Figure 2 (for the MIB) as Figure 4.

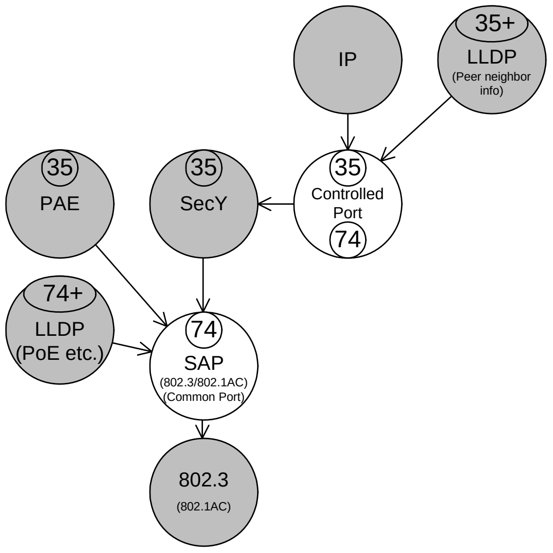


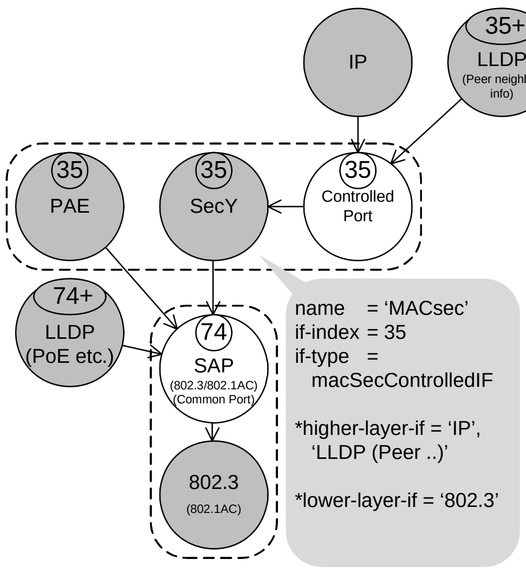
Figure 4—Relocating the SecY 'Y'

Now the PAE and the PoE related instance of LLDP use the Common Port directly. The PAE index has been changed to match that of the Controlled Port and no longer depends on a real/virtual port distinction. It seems that original indexing arose from a series of mis-steps. First, while the Uncontrolled Port/Common

[11]A given frame that has been received from the Common Port may need to be delivered to both an Uncontrolled Port attached entity and to the user of a Controlled Port and cannot necessarily be demultiplexed on the basis of EtherType or even {MAC DA, MAC SA, and EtherType}, even though most frames will, in the most common circumstance, be of interest to only one or the other (or neither). 802.1AE naturally permits implementation of optimization of common cases but there is no reason to bake their potential complexities into management that has to cover all cases.

[12]Possibly as part of the mapping of the ISS to individual media provided by IEEE Std 802.1AC.

Port distinction was recognized it was easy for MIB developers to simply refer to the Common Port. Then virtual ports were introduced and could not be indexed by the Common Port, so an option to use the Controlled Port index was introduced, but the MIB text description of port numbering added to its referenced text (in 12.9.2) so that developers using only the Common Port could ignore the option. While we could decide to get rid of virtual ports entirely it is not clear that the need for them (even if not in their original form using true shared media) will not persist, and aligning the PAE and SecY parameter indexes makes for a clean YANG solution. See Figure 5.



name = ‘MACsec’
if-index = 35
if-type = macSecControlledIF

*higher-layer-if = ‘IP’, ‘LLDP (Peer ..)’

*lower-layer-if = ‘802.3’

Figure 5—YANG model with relocated ‘Y’

This mapping is probably much closer to what the interface augmentation enthusiasts (and the naive user) would expect. It lacks annoying redundant elements. Note however that the PAE and SecY cannot be used to augment the 802.3 ‘interface’ unless every other protocol entity that wants to access the Common Port can also be represented by an augmentation of the same interface[13]. Note also that the management relationship of the PAE to the 802.3 interface may differ from that of the LLDP (PoE) entity, as the frames that the 802.3 interface delivers to and receives from the MACsec as a whole can have any EtherType[14] while the latter may be distinguished by EtherType (as well as by destination MAC address).

We may or may not wish to re-index the PAE MIB so that it matches the YANG, thus circumventing any objection to having the PAE parameters indexed differently in the MIB and in the YANG.

When virtual ports are supported a ‘real port’ PAE protocol entity (instance) may be required even if it is not directly associated with a usable SecY or Controlled Port, though it is unlikely to be worth optimizing for that case as the real port’s SecY parameters serve as the prototype for each of the virtual port’s SecYs[15]. A case that is worth optimizing is when the PAE supports a simple PAC[16] (i.e. when MACsec is not being used). Since the parameters associated with a PAC, beyond those already provided by the basic IETF interface for the Controlled Port, are just those associated with any ISS SAP[17] the suggested augmentation hierarchy (if that is the appropriate term) is that:

a) a PAE model includes PAC parameters, and can augment just the basic interface.

NOTE—802.1X-2010 13.3.2 specifies that the interface’s (the Controlled Port’s) ifType) and thus the YANG model’s if-type is macSecControlledIF even if no MACsec is involved[18].

b) a SecY model can augment an interface that has been augmented by the PAE model (and has not already been incremented by a SecY).

Finally Figure 6 shows how the proposed YANG interface data model would represent a real and two virtual ports, both making use of the same 802.3 interface and both supporting an instance of IP and LLDP (each communicating through paths separated —at least as far as the next bridge—by MACsec.

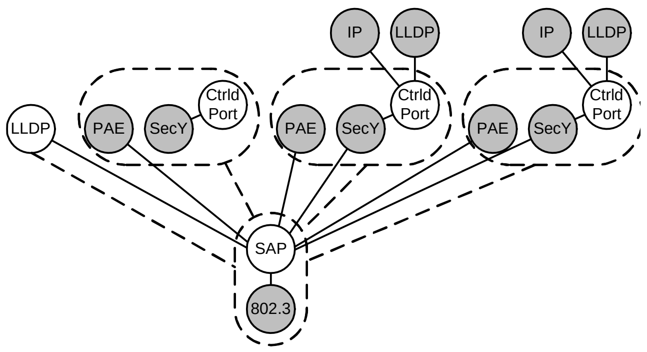


Figure 6—MACsec with virtual ports (example)

[13] In the limit all the possible interface stacks would be accommodated within a single interface (with internal higher and lower sub-layer references?).

[14] It would be an unnecessary complication to reconfigure MACsec’s use of the 802.3 interface if validateFrames (Null, Disabled, Check, or Strict) or protectFrames (True, False) change.

[15] At present, at least, virtual ports (each with a PAE and SecY or PAC) are automatically created (if their creation is enabled) on receipt of an EAPOL frame from a new potential peer. The Controlled Port interface for the real port may be unused.

[16] See 802.1X.

[17] AdminPt2PtMAC and OperPt2PtMAC.

[18] 802.1X-2004 makes no mention of ifType, though it does reference RFC 2863.

While discussing how 802.1X and MACsec can be managed we should not forget that some management parameters are not specific to individual SAPs or protocol entities that might represented by YANG interfaces.

In the case of MACsec/.1AE the only interface independent parameters provide information about Cipher Suites that might be implemented within the managed system. This doesn't present a scoping problem[19] — each port has its own parameters indicating which of these Cipher Suites it supports, and those parameters can be included within each interface.

In the case of 802.1X-2010, the only PAE management information shown[20] as applying to multiple PAEs, are those for NIDs (Network Identities). Their use is controlled per port, so is sufficiently finely grained even if some NIDs should only be used on a subset of the ports (on customer facing ports, or on network facing ports, for example). What 802.1X-2010 does not provide[21] is management of the CAK cache (802.1X 12.6) other than per port. This could be taken as implying that cached CAKs can only be used by the port that originally cached them[22], and makes no provision for pre-shared keys (PSKs) for use by any one of a group of ports. A better structure needs discussion—one idea would be to explicitly tie CAK caching, subsequent use, and pre-provisioning to NIDs, thus avoiding having to introduce another type of grouping. This tie would also allow the existing per-port per-participant view of CAKs, both recently acquired and activated from cache, to be retained, while allowing the creation of an explicit CAK Cache object under PAESystem.

3. Link Aggregation

Figure 7 shows the entities and SAPs called out by 802.1AX's detailed description of link aggregation[23,24].

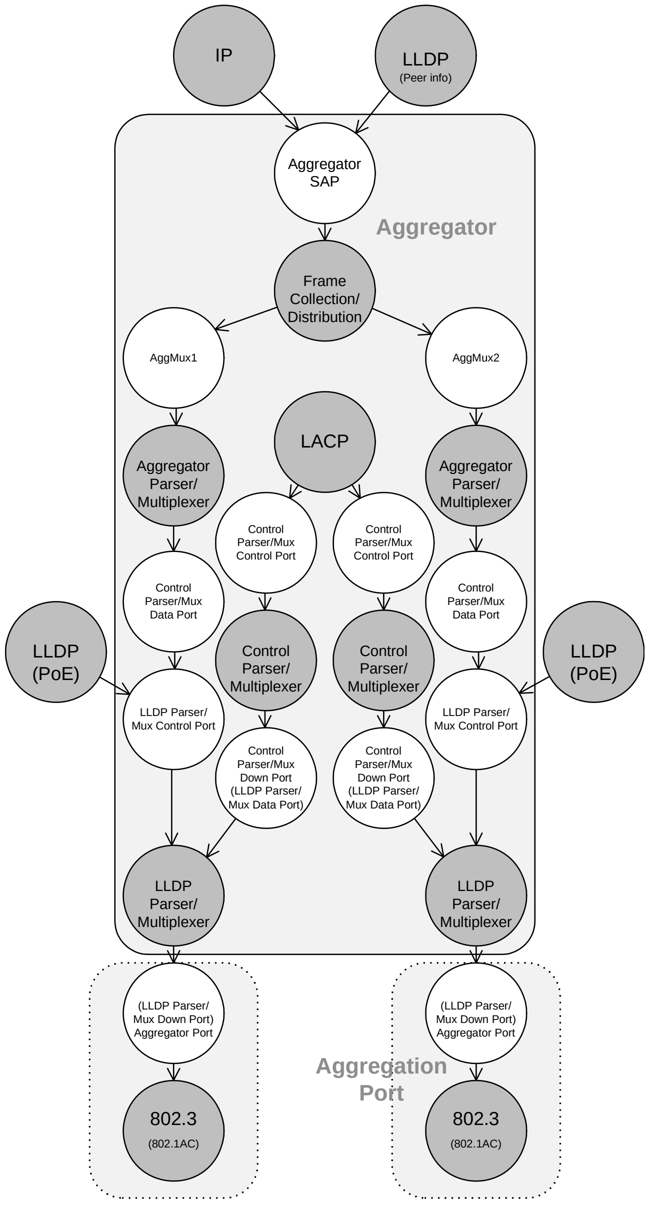

IP
LLDP
(Peer info)
Aggregator SAP
Aggregator
Frame Collection/ Distribution
AggMux1
AggMux2
LACP
Aggregator Parser/ Multiplexer
Aggregator Parser/ Multiplexer
Control Parser/Mux Control Port
Control Parser/Mux Control Port
Control Parser/Mux Data Port
Control Parser/Mux Data Port
LLDP (PoE)
LLDP (PoE)
Control Parser/ Multiplexer
Control Parser/ Multiplexer
LLDP Parser/ Mux Control Port
LLDP Parser/ Mux Control Port
Control Parser/Mux Down Port (LLDP Parser/ Mux Data Port)
Control Parser/Mux Down Port (LLDP Parser/ Mux Data Port)
LLDP Parser/ Multiplexer
LLDP Parser/ Multiplexer
(LLDP Parser/ Mux Down Port) Aggregator Port
(LLDP Parser/ Mux Down Port) Aggregator Port
Aggregation Port
802.3
(802.1AC)
802.3
(802.1AC)

Figure 7—Link Aggregation Entities and SAPs

Note that Figure 7 just shows the interaction of a system wide LACP (Link Aggregation Control Protocol) entity with the two 802.3 Aggregation Ports currently attached to the Aggregator. Figure 8 is a

[19]In the 802.1AEcg Figure 10-6 UML, CipherSuite information is shown as contained by the SecYSystem that also contains each of the Controlled Ports (and thus in the preferred mapping to YANG models, each of the PAE/SecY tuples). Since each port has its own list of supported Cipher Suite information, the SecYSystem can include ports of different capabilities and attributes, there is no need to have different SecYSystem within any managed system.

[20]802.1X Figure 12-3 (PAE management information).

[21]Or discuss in 12.9.2/Figure 12-3 or in the MIB. See Figure 12-2 portnumber>PAE>mka>Kay>participants>Participant for a per port view of the contents of the CAK cache.

[22]This was not the intent in the development of 802.1X-2010, which is why the NID was introduced.

[23]802.1AX-2014 Figure 6-2 with the addition of the LLDP Parse/Multiplexer (one of the possible Protocol Parser/Multiplexer's described in 6.2.7) as described in the last paragraph of 6.2.2.

[24]Omitting details particular to DRNI and DRCP, which are not considered for the present.

UML view of the management parameters. The majority of the management parameters are associated with the individual Aggregation Ports, rather than with each of the Aggregators (though the latter cannot be neglected). Figure 8 also shows that a managed system might contain a number of separate aggregating system, each with its own System ID and System Priority (802.1X-2014 7.3.1.1.4). However the existence of each of these can be inferred from different configuration of those parameters for individual Aggregation Ports[25].

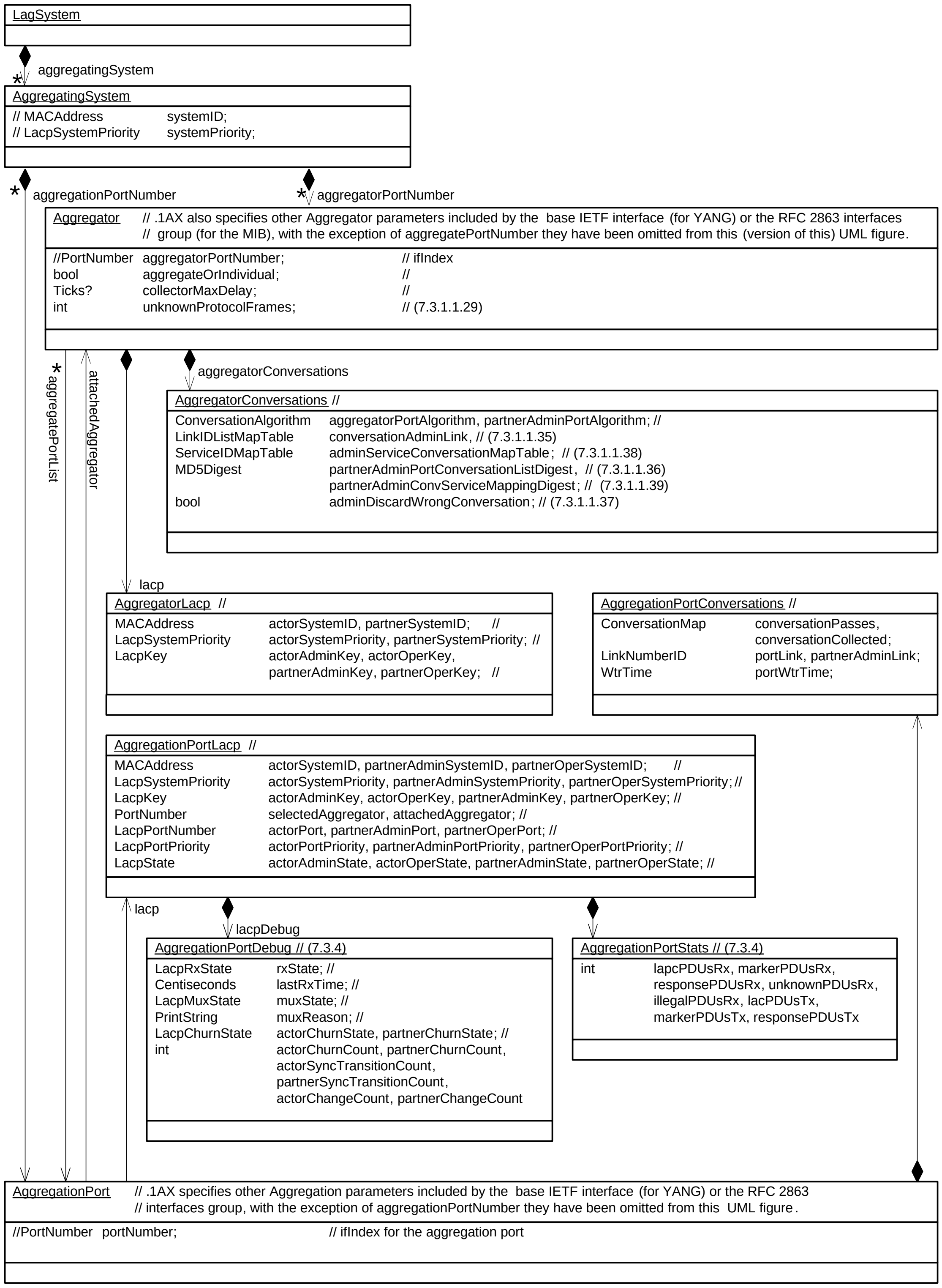

LagSystem
aggregatingSystem
*
AggregatingSystem
// MACAddress systemID;
// LacpSystemPriority systemPriority;
*
aggregationPortNumber
*
aggregatorPortNumber
Aggregator // .1AX also specifies other Aggregator parameters included by the base IETF interface (for YANG) or the RFC 2863 interfaces // group (for the MIB), with the exception of aggregatePortNumber they have been omitted from this (version of this) UML figure.
//PortNumber aggregatorPortNumber; // ifIndex
bool aggregateOrIndividual; //
Ticks? collectorMaxDelay; //
int unknownProtocolFrames; // (7.3.1.1.29)
*
aggregatePortList
attachedAggregator
aggregatorConversations
AggregatorConversations //
ConversationAlgorithm aggregatorPortAlgorithm, partnerAdminPortAlgorithm; //
LinkIDListMapTable conversationAdminLink, // (7.3.1.1.35)
ServiceIDMapTable adminServiceConversationMapTable; // (7.3.1.1.38)
MD5Digest partnerAdminPortConversationListDigest, // (7.3.1.1.36)
partnerAdminConvServiceMappingDigest; // (7.3.1.1.39)
bool adminDiscardWrongConversation; // (7.3.1.1.37)
lacp
AggregatorLacp //
MACAddress actorSystemID, partnerSystemID; //
LacpSystemPriority actorSystemPriority, partnerSystemPriority; //
LacpKey actorAdminKey, actorOperKey, partnerAdminKey, partnerOperKey; //
AggregationPortConversations //
ConversationMap conversationPasses, conversationCollected;
LinkNumberID portLink, partnerAdminLink;
WtrTime portWtrTime;
AggregationPortLacp //
MACAddress actorSystemID, partnerAdminSystemID, partnerOperSystemID; //
LacpSystemPriority actorSystemPriority, partnerAdminSystemPriority, partnerOperSystemPriority; //
LacpKey actorAdminKey, actorOperKey, partnerAdminKey, partnerOperKey; //
PortNumber selectedAggregator, attachedAggregator; //
LacpPortNumber actorPort, partnerAdminPort, partnerOperPort; //
LacpPortPriority actorPortPriority, partnerAdminPortPriority, partnerOperPortPriority; //
LacpState actorAdminState, actorOperState, partnerAdminState, partnerOperState; //
lacp
lacpDebug
AggregationPortDebug // (7.3.4)
LacpRxState rxState; //
Centiseconds lastRxTime; //
LacpMuxState muxState; //
PrintString muxReason; //
LacpChurnState actorChurnState, partnerChurnState; //
int actorChurnCount, partnerChurnCount, actorSyncTransitionCount, partnerSyncTransitionCount, actorChangeCount, partnerChangeCount
AggregationPortStats // (7.3.4)
int lapcPDUsRx, markerPDUsRx, responsePDUsRx, unknownPDUsRx, illegalPDUsRx, lacPDUsTx, markerPDUsTx, responsePDUsTx
AggregationPort // .1AX specifies other Aggregation parameters included by the base IETF interface (for YANG) or the RFC 2863 // interfaces group, with the exception of aggregationPortNumber they have been omitted from this UML figure.
//PortNumber portNumber; // ifIndex for the aggregation port

Figure 8—Link Aggregation management information (rough draft)

Figure 9 provides an appropriate external management view of the complexities of Figure 7, showing both the MIBs ifIndex/ifStack information and the mapping to YANG interfaces.

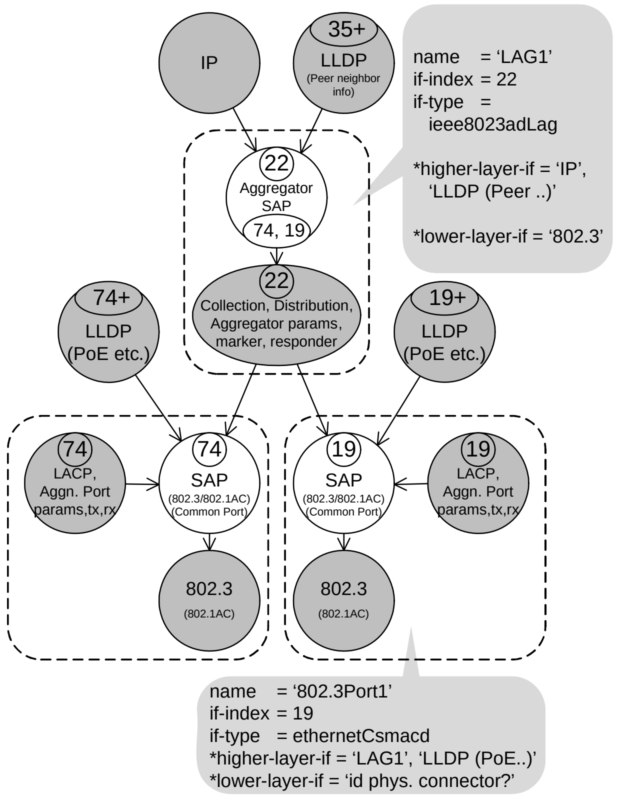

IP
35+
LLDP
(Peer neighbor info)
name = 'LAG1'
if-index = 22
if-type = ieee8023adLag
*higher-layer-if = 'IP', 'LLDP (Peer ..)'
*lower-layer-if = '802.3'
22
Aggregator SAP
74, 19
22
Collection, Distribution, Aggregator params, marker, responder
74+
LLDP (PoE etc.)
19+
LLDP (PoE etc.)
74
LACP, Aggn. Port params,tx,rx
74
SAP
(802.3/802.1AC) (Common Port)
19
SAP
(802.3/802.1AC) (Common Port)
19
LACP, Aggn. Port params,tx,rx
802.3 (802.1AC)
802.3 (802.1AC)
name = '802.3Port1'
if-index = 19
if-type = ethernetCsmacd
*higher-layer-if = 'LAG1', 'LLDP (PoE..)'
*lower-layer-if = 'id phys. connector?'

Figure 9—Link Aggregation: YANG interfaces

In Figure 5, the PAE augmenting the Controlled Port interface uses the service provided by the *lower referenced interface*. In contrast, in Figure 9, the LACP parameters that are associated with each 802.3 SAP use the same provided by *that interface*.

4. Link Aggregation with MACsec

As specified in 802.1AE Clause 11, MACsec secures each of the individual links when used with link aggregation[26]. Figure 10 shows the interface stack. Since LACP uses the secured service provided by MACsec, each of the systems participating in an aggregation can be sure that the LACP System Identifiers and Keys it receives have been sent by a trusted partner. It is possible, of course, that a trusted system simply unaware of its incorrect or conflicting system identifier—which is more likely if that is based on a randomly chosen or other local MAC address[27] for privacy.

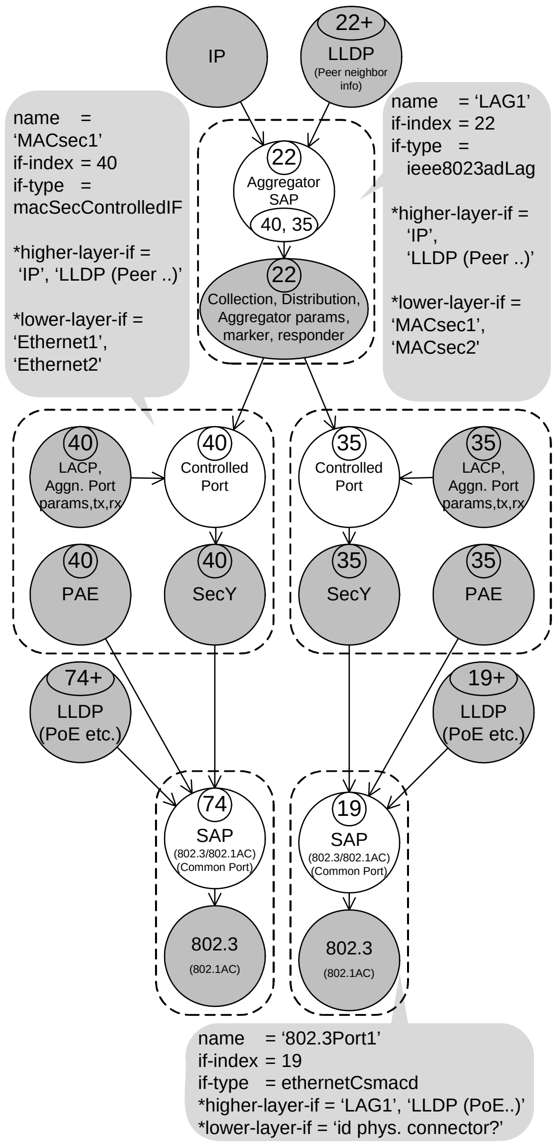

IP
22+
LLDP
(Peer neighbor info)
name = 'MACsec1'
if-index = 40
if-type = macSecControlledIF
*higher-layer-if = 'IP', 'LLDP (Peer ..)'
*lower-layer-if = 'Ethernet1', 'Ethernet2'
22
Aggregator SAP
40, 35
22
Collection, Distribution, Aggregator params, marker, responder
name = 'LAG1'
if-index = 22
if-type = ieee8023adLag
*higher-layer-if = 'IP', 'LLDP (Peer ..)'
*lower-layer-if = 'MACsec1', 'MACsec2'
40
LACP, Aggn. Port params,tx,rx
40
Controlled Port
35
Controlled Port
35
LACP, Aggn. Port params,tx,rx
40
PAE
40
SecY
35
SecY
35
PAE
74+
LLDP (PoE etc.)
19+
LLDP (PoE etc.)
74
SAP
(802.3/802.1AC) (Common Port)
19
SAP
(802.3/802.1AC) (Common Port)
802.3 (802.1AC)
802.3 (802.1AC)
name = '802.3Port1'
if-index = 19
if-type = ethernetCsmacd
*higher-layer-if = 'LAG1', 'LLDP (PoE..)'
*lower-layer-if = 'id phys. connector?'

Figure 10—Link Aggregation and MACsec

One way for 802.1X to help here would be to compare authentication credentials for the links, checking for a match, though that might involve more knowledge than the systems have with more constraints on the credentials and their authentication than is desirable. A better way would be to start an MKA instance for the same {CAK, CKN} on each of the links, ensuring that they are all attached to the same partner [28,29].

[25]Question. What does it mean if different Aggregation Ports share the same System ID but have different System Priority.

[26]An attacker would not be able to successfully inject or decipher secure data if MACsec was used over the aggregated links, but a large percentage of frames might be lost as an untrustworthy link might not provide connectivity to the correct peer. Applying MACsec before distributing the data across the links would also defeat attempts at conversation balancing.

[27]802.1AX currently mandates use of a globally unique address.

[28]Provided of course that the partner hasn't shared that over a common service, but there is no excuse for the partner being unaware of such sharing.

[29]No new MKA behavior/checking required for correctness, so much simpler than attempting to use one link to distribute SAKs that are used by both.

5. CFM, MACsec, and Link aggregation

While CFM uses a set of shims that can, in principle, be positioned anywhere in the interface stack, 802.1Q is quite clear where they should be placed.

Up MEPs and Down MEPs that monitor individual service instance (distinguished by VID or other service instance selector) and their associated MIPs are at the top of each Bridge Port's interface stack[30], immediately below the MAC Relay Entity's Forwarding Process and above the support of the EISS (802.1Q 6.9), with a precise ordering between the individual shims for Up, Down, MHF, and MD Level.

The only MEP that can be placed elsewhere in a Customer Bridge Port's[31] stack is a Down MEP for all data (irrespective of service selector)[32], below Bridge Port Transmit and Receive (802.1Q 8.5). When Link Aggregation is being used such a Down MEP should be configured to protect the connectivity provided by individual LANs—there is no implementation independent way to constrain frames sent and receive from higher MEPs to a particular link[33]. That Down MEP should be positioned above MACsec[34]. However that positioning, as a user of the Controlled Port, does not preclude managing it as an augmentation of the same interface—just as the Aggregation Port components of LACP augment the Controlled Port interface in Figure 10.

However, while MEPs (and MIPs) can be said to be instantiated in an interface stack, they are created and used as a result of operations on Maintenance Association managed objects that span the bridge[35] (see 802.1Q Figure 12-1). Only a small part of their functionality is primarily related to an interface stack through the CFM Stack managed object (12.14.2, supporting a read operation to discover what MEPs or MHFs have been instantiated as a result other configured rules) and the Configuration Error List managed object (12.14.4, supporting a read operation that returns detected configuration and resource exhaustion errors). Operations for both of these take service instance selectors (e.g. a list of VIDs), and are only available to the owner of the bridge (as opposed to maintenance domain administrators making use of configured services). Since each Maintenance Association managed object is also identified by a service instance selector it is questionable whether organizing the CFM Stack and Configuration List primarily by augmented interface would be useful.

6. Interface counters

The IETF YANG Interface management model maintains separate counts for unicast, multicast, and broadcast packets, for both receive and transmit.

Of course an 802.3/Ethernet interface doesn't distinguish between these so far as its own operation is concerned, these management statistics are being collected to support the (management of) the interfaces client/user.

Of course, if the 802.3 interface's client is a MACsec SecY or an Aggregation Port, or more specifically an augmented interface including one or both of these, as in Figure 10, then the operation of that client is also independent of the distinction—both a SecY and an Aggregation Port will apply exactly the same processing to unicast, multicast, and broadcast frames. However this client interface will also collect separate counts for unicast, multicast, and broadcast—for the benefit of its client.

Of course, if that client is actually a separate augmented interface—which seems likely in the case of a Bridge Port—its operation is also unaffected by the distinction between the unicast, multicast, or broadcast nature of frames at its lower interface[36]. The operation of the Bridge Port's client(s) is dependent on the distinction. Collecting separate statistics for that upper interface does serve a purpose, although one bundled set of statistics is not a great help when considering a protocol entity locally attached to a Bridge Port—any counts relevant to its operation are likely to be swamped by those for relayed frames.

If interface statistics were associated with the clients/users of interfaces, rather than with the provider/supporter of each interface's services then they could be omitted where irrelevant and strengthened where useful (e.g. distinguishing counts for the separate types of multicast their controlling entities produce/consume where relevant.

[30] 802.1Q-2014 Figures 22-1 and 22-4 (though I believe the latter has significant problems).

[31] Further detailed considerations apply to CFM in Provider and Backbone Bridges, but do not invalidate these general conclusions.

[32] Shown as associated with MD Level 1 in 802.1Q Figure 22-1 and MD Level 0 in Figure 22-4.

[33] See 802.1Q 22.1.8 4th Para for discussion.

[34] See 802.1Q 22.1.8 2nd para and following bullet items.

[35] Or other enclosing system.

[36] All received frames go, from a modeling point of view, to the MAC Relay Entity (where they may be filtered) even if they are to be received by a local entity attached to the Bridge Port.