Contribution to the November ‘98 Link Aggregation Meeting
TonyJeffree
November 9, 1998

Strawman/D2½: Changes/Additions to 802.3 required in order to specify

Link Aggregation

Prepared by Tony Jeffree as a contribution to the work of the Link Aggregation Study Group at its November 1998 meeting. **Its contents shall not be assumed to reflect any opinions other than those of the author.**

NOTE: This document fixes some typos & minor inconsistencies/bugs in Strawman/D2, circulated on November 4, 1998. Changes are marked by change bars.

<<Comments on this document may be sent to the author:

Tony Jeffree
11A Poplar Grove
Sale
Cheshire
M33 3AX
UK
+44 161 282 3824 (Tel)
+44 161 973 6534 (Fax)
Email: tony@jeffree.co.uk

>>

<< Author’s Notes

This Draft has been generated by the author as a contribution to the work of the Link Aggregation Study Group. The contents of the document reflects some of the material presented during various meetings of the Link Aggregation Study Group and Task Force during the course of 1998, and in particular, is intended to reflect a proposal for the operation of the Link Aggregation Control Protocol, based on a synthesis of the best features of the two protocol proposals made at the April interim meeting (one made by Jeffree, the other made by Wakerly & Fine - see the Link Aggregation website for full details of those proposals - http://199.172.136.47/groups/802/3/trunk_study/april98/index.html). The presentation by Jeffree at the July meeting of the task force gave a description of an early attempt at a synthesis between the two proposals; this draft reflects the further development of those ideas following feedback from the July and September meetings and further work on clarification and simplification. See also Mick Seaman’s paper “Link Aggregation Control Protocol” circulated in late August, and shortly to be recirculated in updated form to reflect the protocol as described here.

Particular changes of note since my presentation of the protocol at the Austin interim meeting:

- Simplification of the state machines still further - in particular, combining the functionality of the Desirable and Nervous machines with some of the functionality of the Transmit machine to form a single Periodic Transmission machine that controls both whether or not protocol is exchanged, and at what rate. This has effectively reduced the number of states involved in this aspect of the protocol to three;

- Addition of a description of a Flush Protocol, along with a service description that defines how and by whom it may be used;

- Addition of the description of a “Churn Detection” mechanism that can provide a means of alerting management to some classes of fault/misconfiguration.

This document contains a complete description of the operation of Link Aggregation Control and its associated protocols. Given that the November meeting is the deadline for protocol proposals, this (along with Mick’s revised paper and the updated presentation material) constitutes a response to that proposal deadline.

The document is presented in the rough format of a standards draft, in order to accelerate the process of structuring the eventual standard, and to get ideas & concepts documented in a form that is reasonably close to that needed for the final text. However, there are still numerous areas of the text that will require clarification, completion or modification in order to produce a complete first draft of the standard; in this respect the document must therefore only be regarded as “Work In Progress”. The intent is to collect, develop and consolidate ideas in order to focus the work & take it forward, and not to present a complete, coherent and polished draft.

Tony Jeffree
November 9, 1998>>

Contents

Figures

Tables

Strawman/D2½: Changes/Additions to 802.3 required in order to specify Link Aggregation

1.1 *Changes to* References

<<Author's Note: References to be added. These would consist of a set of changes/additions to existing 802.3 section 1.3 & Annex A.>>

1.2 *Changes to* Definitions[1]

<<Author's Note: The following Definitions to be added. These would form a set of changes/additions to 802.3 section 1.4.>>

1.2.1 Bridge Port

A point of attachment to a LAN through which a MAC Bridge transmits and receives MAC frames.

NOTE—See ISO/IEC 15802-3 7.2, 7.2.3.

1.2.2 Bridged LAN

A concatenation of individual IEEE 802 Local Area Networks interconnected by MAC Bridges.

NOTE—This is identical to the definition in ISO/IEC 15802-3.

1.2.3 End station

A system attached to a LAN that is an initial source or a final destination of MAC frames transmitted across that LAN.

<<Author's Note: The original definition was for "Host"; however, 802-speak for Host is End station.>>

[1]These definitions are based on the set of definitions presented by Floyd Backes during the Seattle interim meeting, Feb 98. I have included all the definitions from that presentation that might be relevant to the standard, regardless of whether they are actually used in this particular draft. Any that are not needed can easily be excised at a later date. Some have been modified in the light of the discussion that took place in Seattle.

1.2.4 Conversation

A set of MAC frames exchanged between a pair of end stations, where all of the MAC frames form a part of an ordered sequence, and where there exists a requirement for ordering to be maintained among the set of MAC frames exchanged. A conversation may be uni-directional (i.e., a monologue), or bi-directional (i.e., a dialogue).

There may be more than one conversation in progress between a given pair of end stations at any one time; similarly, a given end station may take part in conversations with more than one end station at any one time.

<<Author's Note: The intent of this definition is to encompass the concept of a “flow”, without attempting to tie it to particular layer attributes, such as MAC addresses, IP addresses, protocols...etc. It was noted at the Seattle meeting that using “flow” as the name would have potential of confusion with “flow control”, hence the change of name.>>

1.2.5 Aggregate Conversation

A set of conversations, treated as if they are all part of a single conversation. The particular set of conversations that is aggregated to form a given aggregate conversation is determined by means of a conversation aggregation rule.

NOTE—The terms Conversation and Conversation aggregation rule are defined in 1.2.4 and 1.2.6.

1.2.6 Conversation Aggregation Rule

A rule that specifies how individual conversations (1.2.4) are allocated to aggregate conversations (1.2.5).

NOTE—There are potentially many such aggregation rules; for example, a rule might specify aggregation on the basis of source/destination address hashing, VLAN ID, IP subnet, protocol type, etc. The terms Conversation and Aggregate conversation are defined in 1.2.4 and 1.2.5.

1.2.7 Key

A parameter of each physical Port of a system identifying those Ports that can be aggregated together.

NOTE—Ports in a system that share the same value of this parameter are potentially able to aggregate together.

1.2.8 Link Aggregation Group

A grouping of Link Segments, of the same medium type and speed, that are treated as if they are all part of a single Link Segment. The MDIs associated with each Link Segment in a Link Aggregation Group are associated with the same pair of devices. For the purposes of this definition, a device is a MAC Bridge, an end station or a repeater.

<<Author's Note: Link Segment and MDI are defined 802.3 terms.>>

Traffic is allocated to the individual Link Segments in a Link Aggregation Group on the basis of one or more Conversation Aggregation Rules. One or more Aggregate Conversations are associated with each Link Segment that is part of a Link Aggregation Group. A given Cnversation is associated with a single Link Segment.

1.2.9 LAN Aggregation Group

A grouping of LANs or Bridged LANs, of the same or dissimilar medium access method, medium type or speed, that form parallel paths between a pair of connected end stations, and that are treated as if they form a single LAN between those end stations.

Traffic is allocated to the individual LANs in a LAN Aggregation Group on the basis of one or more Conversation Aggregation Rules; i.e., one or more Aggregate Conversations are associated with each LAN that is part of a LAN Aggregation Group.

<<Author’s Note: This definition is probably only useful insofar as it will allow us to identify stuff that is outside the scope of this standard.>>

1.3 *Changes to* Abbreviations

<<Author’s Note: The following abbreviations to be added. These would form a set of changes/additions to 802.3 section 1.5.>>

LACP	Link Aggregation Control Protocol

91. Link Aggregation

<<Author’s Note: Final chapter numbering to be determined.>>

91.1 Overview

This supplement to ISO/IEC 8802-3 defines an optional Link Aggregation sublayer for use with CSMA/CD MACs. The sublayer allows one or more individual Link Segments to be aggregated together to form a Link Aggregation Group (1.2.8), such that the MAC Client is able to treat the Link Aggregation Group as if it were a single Link Segment.

Figure 91-1[2] shows the positioning of the Link Aggregation sublayer in the CSMA/CD layer architecture, and the relationship of that architecture with the Data Link and Physical layers of the OSI Reference Model. The figure also shows the ability of the Link Aggregation sublayer to aggregate a number of individual Link Segments in order to present a single MAC interface to the MAC Client.

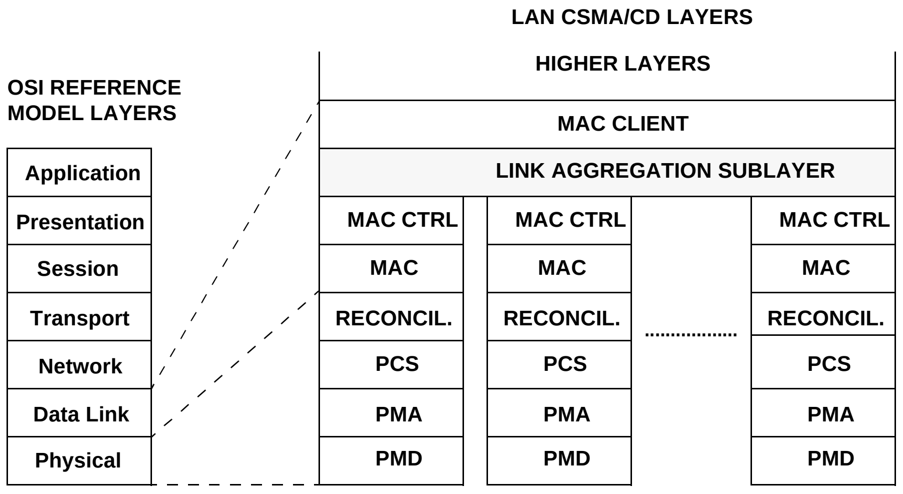

Figure 91-1—Link Aggregation Reference Model

Figure 91-2[3] shows the individual components that form the Link Aggregation Sublayer, and their interrelationships.

91.1.1 Frame Collection

Frame Collection is responsible for receiving incoming frames from the set of individual Link Segments that form the Link Aggregation Group with which the collection function is associated. Frames received are delivered to the MAC Client. Frames received from a given Link Segment are delivered to the MAC Client in the order that they are received by Frame Collection. As the Frame Distribution function is responsible for meeting any frame ordering constraints, there is no requirement for Frame Collection to perform any re-ordering of received frames across multiple Link Segments.

A detailed description of Frame Collection can be found in clause 91.6.

[2]Figure 91-1 is based on Paul Bottorff’s architectural diagram as presented at the Irvine meeting, simplified as agreed in the meeting. Other material from that presentation has been used as the basis for the subsequent description of the components of the sublayer.

[3]Figure 91-2 is based on the internal structure of the Link Aggregation Sublayer shown in Paul Congdon’s presentation at the Irvine meeting. Other parts of that presentation have been used as the basis for the subsequent description of the components of the sublayer.

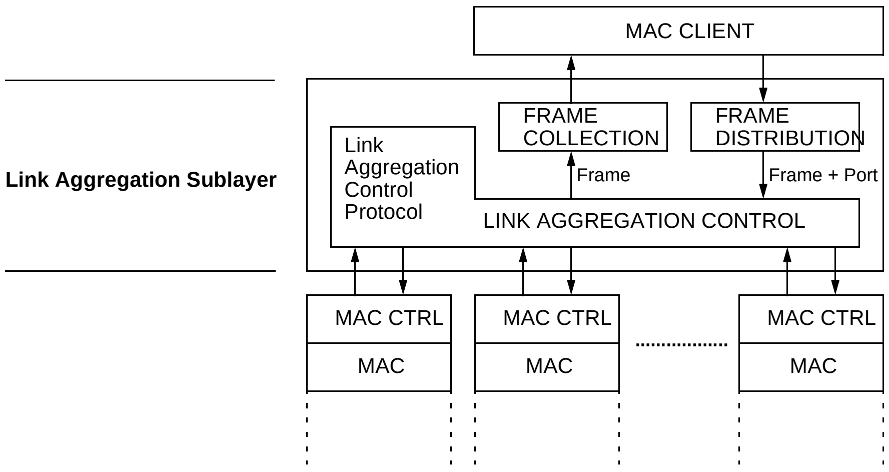

Figure 91-2—Link Aggregation Sublayer

91.1.2 Frame Distribution

Frame Distribution is responsible for receiving outgoing frames from the MAC Client, and for transmitting them on the set of Link Segments that form the Link Aggregation Group with which the distribution function is associated. The distribution function is responsible for making all decisions related to load balancing among the Link Segments in the Link Aggregation Group. This supplement to ISO/IEC 8802-3 does not standardize the details of any load balancing algorithms that may be used to perform this function; however, any load balancing algorithm is required to ensure that:

a) The algorithm does not cause re-ordering of frames that are part of any given *conversation* (1.2.4);
b) The algorithm does not cause duplication of frames.

The former condition is met by ensuring that all frames that are part of a given conversation are transmitted on a single Link Segment, in the order that they are received from the MAC Client. Hence, the requirement not to misorder frames does not involve the addition of (or modification of) any information to the MAC frame, or any processing on the part of the corresponding collection function in order to re-order frames. This approach to the operation of the distribution function permits a wide variety of distribution and load balancing algorithms to be used, while also ensuring interoperability between devices that adopt differing algorithms.

A detailed description of Frame Distribution can be found in clause 91.7.

91.1.3 Link Aggregation Control

Link Aggregation Control is responsible for performing the configuration and control functions of the Link Aggregation Sublayer. These functions are performed on the basis of:

a) Static configuration information, local to the control function;
b) Dynamic configuration information, acquired and exchanged by means of the Link Aggregation Configuration Protocol (LACP).

Link Aggregation Control ensures that configuration (and re-configuration) of Link Aggregation Groups occurs automatically, within any constraints imposed by the static configuration information. In particular, it ensures that:

c) The configuration achieved is deterministic; that is to say, for a given static configuration and physical topology, the allocation of Link Segments to Link Aggregation Groups does not depend upon the order in which those segments are activated;
d) If a given Link Segment can be included in a given Link Aggregation Group, then it is included in that aggregation;
e) A given Link Segment cannot be included in more than one Link Aggregation Group at any one time.

A detailed description of Link Aggregation Control can be found in clause 92.3.

91.1.4 Aggregator

An Aggregate Port or *Aggregator* consists of an instance of the Frame Collection function and an instance of the Frame Distribution function. A single Aggregator is associated with each Link Aggregation Group. An Aggregator offers a MAC service to its associated MAC Client; access to the MAC service by a MAC Client is always achieved via an Aggregator. An Aggregator can therefore be considered to be a *logical MAC*, bound to one or more physical MAC interfaces, and through which the MAC client is provided access to the MAC service.

A detailed description of the Aggregator can be found in clause 91.8.

Note—To simplify the modeling and description of the operation of Link Aggregation, it has been assumed that a given physical MAC is always bound to an Aggregator; there are therefore as many Aggregators as there are physical MACs in a given device. Aggregation of two or more MACs consists of changing the bindings between MACs and Aggregators in order that more than one MAC is bound to a single Aggregator. The operation of aggregation will therefore result in one or more Aggregators that are bound to more than one MAC, and one or more Aggregators that are not bound to any MAC. An Aggregator that is not bound to any MAC appears to the MAC Client to be a MAC interface to an inactive link. It is not a requirement of this standard that a given implementation maintains as many Aggregators as physical Ports; however, if fewer Aggregators are maintained, this may have consequences in some configurations with regard to the deterministic characteristics of the implementation.

91.1.5 Addressing

Associated with each Aggregator is a single, individual MAC address.

<<Author's Note: As observed in the Irvine minutes, there is much discussion to be had on addressing, the implications with respect to re-configuration, and the implications with respect to Bridge operation, before we're done. The above deliberately makes no pre-judgement as to how these addresses get allocated.>>

A detailed description of Addressing can be found in clause 91.9.

91.2 Scope[4]

The purpose of this supplement to ISO/IEC 8802-3 is to increase link availability and bandwidth between DTEs by specifying the necessary mechanisms for parallel Link Segment aggregation. To this end, it specifies the establishment of DTE to DTE logical links, which consist of N parallel instances of an 802.3 Link Segment, all of which are full duplex point-to-point links of the same speed. A logical link so established will support existing ISO/IEC 8802.3 MAC Clients.

In particular, the following are specified:

[4]This Scope is based on the text contained in Scope and Purpose in the proposed Link Aggregation PAR.

a) The architectural model that establishes the relationship between Link Aggregation and the existing ISO/IEC 8802-3 architecture and standards;
b) The procedures involved in the establishment, configuration and removal of logical links;
c) The management functionality provided in order to allow static configuration of logical links;
d) The protocols required in order to allow dynamic configuration of logical links.

91.3 Conformance

91.3.1 Static conformance requirements

<<Author's Note: Static conformance requirements to be added.>>

91.3.2 Options

<<Author's Note: Options to be added.>>

91.4 Recommendations

<<Author's Note: To be added, if needed.>>

91.5 Relationship of Link Aggregation to other standards

<<Author's Note: To be added.>>

91.6 Frame Collection

<<Author's Note: Detailed description/definition of Frame Collection to be added.>>

91.7 Frame Distribution

<<Author's Note: Detailed description/definition of Frame Distribution to be added.>>

91.8 Aggregator

<<Author's Note: Detailed description/definition of Aggregator to be added.>>

91.9 Addressing

<<Author's Note: Detailed description/definition of Addressing to be added.>>

91.10 Protocol Implementation Conformance Statement

The supplier of an implementation that is claimed to conform to clause 91. of this standard shall complete a copy of the PICS proforma provided below and shall provide the information necessary to identify both the supplier and the implementation.

<<Author's Note: PICS Proforma to be added.>>

92. Link Aggregation Control

This section describes the operation of Link Aggregation Control, and a protocol that is capable of automatically exchanging the information necessary in order for Link Aggregation Control to operate in the manner described.

92.1 Conformance

92.1.1 Static conformance requirements

<<Author’s Note: Static conformance requirements to be added.>>

92.1.2 Options

<<Author’s Note: Options to be added.>>

92.2 Recommendations

<<Author’s Note: To be added, if needed.>>

92.3 Link Aggregation Control

92.3.1 Scope of Link Aggregation Control

The scope of Link Aggregation Control includes:

a) Maintenance of configuration information for Link Segments and Link Aggregation Groups;
b) Exchange of configuration information with other systems, in order to determine the requirements for (re-)configuration of Link Aggregation Groups;
c) Addition and removal of links from Link Aggregation Groups;
d) Assigning links to appropriate Aggregators;
e) Communication of link state information to the Frame Collection and Frame Distribution functions.

92.3.2 Objectives of Link Aggregation Control

The operation of the Link Aggregation Control function meets the following objectives:

a) **Automatic configuration.** In the absence of manual override controls, an appropriate set of Link Aggregation Groups is automatically configured, and individual Link Segments are allocated to those groups. In other words, if a set of links can aggregate, they will aggregate.
b) **Continuous operation.** Manual intervention, or initialization events, are not a requirement for correct operation. The configuration mechanism continuously monitors for changes in state that require re-configuration.
c) **Low protocol overhead.** The overhead involved in external communication of configuration information between devices will be small.
d) **Very low probability of misdelivery.** The operation of the (re-)configuration functions minimizes the risk of frame being delivered to the wrong Aggregator.
e) **Low risk of duplication or re-ordering.** The operation of the (re-)configuration functions minimizes the risk of frame duplication and frame re-ordering.
f) **Rapid convergence.** The configuration will resolve rapidly to a stable configuration, in the face of conflicting demands from each end of a link. Convergence will be achieved within at most a very few seconds, and will allow for more rapid convergence where supported.

g) **Deterministic convergence.** The configuration will resolve a deterministic configuration; i.e., the configuration achieved will not be dependent upon the order in which events occur, but will be completely determined by the combination of the capabilities of the individual links and their physical connectivity.
h) **Low failover delay.** Re-configuration on link failure occurs rapidly.
i) **Low risk of mis-configuration.** The configuration functions detect and correct mis-configurations, by performing re-configuration and/or by taking mis-configured links out of service.
j) **Integration of Aggregation-unaware devices.** Link Segments that cannot take part in link aggregation, either because of their inherent capabilities or of the capabilities of the devices to which they attach, operate as normal 802.3 links.
k) **Accommodate differing capabilities/constraints.** The configuration capabilities will allow devices with differing hardware and software constraints on link aggregation to be accommodated.

92.3.3 Overview

Link Aggregation Control is responsible for controlling the creation and maintenance of Link Aggregation Groups and associating them with Aggregators. Its operation makes use of information from the following sources:

a) The inherent properties of the set of individual Link Segments that are visible to Link Aggregation Control;
b) Statically configured parameter values associated with those links;
c) Dynamic information exchanged with other Link Aggregation Controllers reachable via those links, exchanged by means of the Link Aggregation Control Protocol;
d) The properties associated with any existing Link Aggregation Groups.

The operation of Link Aggregation Control involves the following activities:

e) Identification of links that are candidates for aggregation (92.3.3.1);
f) Checking that candidate links can actually be aggregated (92.3.3.2);
g) Controlling the addition of a link to a Link Aggregation Group, and the creation of the group and associated Aggregator if necessary (92.3.3.3);
h) Monitoring the status of aggregated links to ensure that the aggregation is still valid (92.3.3.4);
i) Removal of a link from a Link Aggregation Group if its membership is no longer valid, and removal of the group if it no longer has any member Links (92.3.3.5).

92.3.3.1 Identifying links that are candidates for aggregation

The operation of Link Aggregation Control is such that, if a given link is a suitable candidate for aggregation, then that link will be included in a suitable Link Aggregation Group and associated with an Aggregator. A link is a candidate for aggregation if the following are all true:

a) It is a point-to-point link; and
b) It is a full duplex link; and
c) It connects the same pair of systems; and
d) Both systems are capable of performing Link Aggregation; and
e) Its static configuration permits aggregation; and
f) It is active.

<<Author’s Note: The term “active” is used in the sense that the Link is operable; i.e., the physical MAC can transmit & receive. Is there a suitable 802.3 term for this condition?>>

92.3.3.2 Checking that a candidate link can be added to a Link Aggregation Group

Before a link can be added to a Link Aggregation Group, it is necessary to check that the information on which Link Aggregation Control decided that the link is a candidate for aggregation is still valid, and that all necessary parameters are known. The Link Aggregation Control Protocol is used to validate any existing knowledge related to the link, and to determine the characteristics of the link as understood by the Link Aggregation Control entity attached to its far end.

The result of this checking process is that Link Aggregation Control now understands:

a) The identity of the pair of systems that the link is connected between;
b) The Key that each of those systems has associated with that link;
c) Whether that set of characteristics means that the link can be aggregated;
d) Whether both systems understand the same information regarding the state of the link.

LACP does not detect the presence of multiple Aggregation-aware devices on the same link. Hence, it is assumed that shared medium links will in general be statically configured to indicate that they are unable to aggregate.

LACP allows an Aggregation-aware device to signal to its potential aggregation partners that it considers a particular link to be non-aggregatable. This results in some optimization of the aggregation process in such cases, as there is no possibility of any negotiation between the two devices resulting in aggregating that link with any other link.

92.3.3.3 Adding a link to a Link Aggregation Group

If a link is both a candidate for aggregation, and the link parameters have been successfully checked, then Link Aggregation Control will add it to an existing, compatible, Link Aggregation Group. If no compatible group exists, then Link Aggregation Control will create a new Link Aggregation Group and associate that group with a suitable Aggregator. As part of the process of establishing a new Group, the distribution algorithm that will be employed for that group is also determined.

NOTE—A consequence of the approach described here is that a Link Aggregation Group can exist with only a single active Link Segment in the Group; in fact, all groups (and therefore, all aggregations) start out in life with a single link. This seems to be a simpler approach than treating the presence of 2 compatible links as a special case. Hence, the creation of a group of 2 links involves the necessary pre-cursor step of creating a group containing one of the links, and then adding the second.

Definition of the distribution algorithms themselves is outside the scope of this standard. However, in order to ensure that distribution algorithms are defined such that they operate in a manner that ensures preservation of frame ordering, and prevents frame duplication, the following conditions must be met when adding a link to Link Aggregation Group:

a) If the link concerned is active, then it is de-activated before it is added to the group, and any frames that are in transit are flushed;
b) Activation of the link as part of the group involves signalling to Frame Collection that the link is active, and then ensuring that the corresponding Frame Collection function at the other end of the link is active before signalling to Frame Distribution that the link is active.

The operation of Link Aggregation also takes account of the possibility that more than one link may become active within a short period of time, leading to the possibility that two or more links may need to be added to the same Link Aggregation Group. In order to avoid such events causing repeated configuration changes as individual links become enabled, the algorithm employed applies some hysteresis to the aggregation process, allowing multiple links to be added to an aggregation at the same time.

Link Segments that are not successful candidates for aggregation (e.g., links that are attached to other devices that cannot perform aggregation, links that are connected to shared LAN media, or links that have been manually configured to be non-aggregatable) are enabled to operate as individual 802.3 links. For consistency of modeling, such a link is regarded as belonging to a Link Aggregation Group that contains a single link, and all such links are accessible only via a compatible Aggregator.

92.3.3.4 Monitoring the membership of a Link Aggregation Group

Each link is monitored in order to confirm that the Link Aggregation Control functions at each end of the link still agree on the configuration information for that link. If the monitoring process detects a change in configuration that materially affects the link’s membership of its current Link Aggregation Group, then it will be necessary to remove the link from its current group and to move it to a new group.

92.3.3.5 Removal of a link from a Link Aggregation Group

Removal of a link from a Link Aggregation Group is achieved in a manner that ensures preservation of frame ordering, and prevents frame duplication. The Frame Distribution function is informed that the link is no longer part of the group, the changed configuration information is communicated to the other end of the link, and then the Frame Collection function is informed that the link is no longer part of the group. The link can then be moved to a new Link Aggregation Group.

92.3.3.6 Configuration and administrative control of link aggregation.

Administrative configuration facilities allow a degree of control to be exerted over the way that links may be aggregated. In particular, administrative configuration allows:

a) The Key values (1.2.7) associated with a link to be identified or modified;
b) The Key values associated with an Aggregator to be identified or modified;
c) Links to be identified as being incapable of aggregation;
d) Link Aggregation Control Protocol parameters to be identified or modified.

92.3.4 Interfaces

Figure 92-1 illustrates the components involved in the operation of Link Aggregation, and the interfaces between them.

92.3.4.1 Interface between Link Aggregation Control and each Link

Each link that is part of a Link Aggregation Group presents an 802.3 MAC interface to Link Aggregation Control; this interface is used:

a) To allow Link Aggregation Control to exchange Link Aggregation Control Protocol Data Units (LACPDUs) with any other Link Aggregation Control instance(s) attached to the links;
b) To allow the Aggregator to transmit frames from, and receive frames destined for, its associated MAC Client.

The Link Aggregation Control function maintains the following information with respect to each physical link:

c) The identifier of the Link Aggregation Group to which it currently belongs;
d) The identifier of the Aggregator associated with that Link Aggregation Group;
e) The status of interaction between the Frame Collection function of the Aggregator and the link (Collection Enabled, or Collection Disabled). Collection Enabled indicates that the receive function of

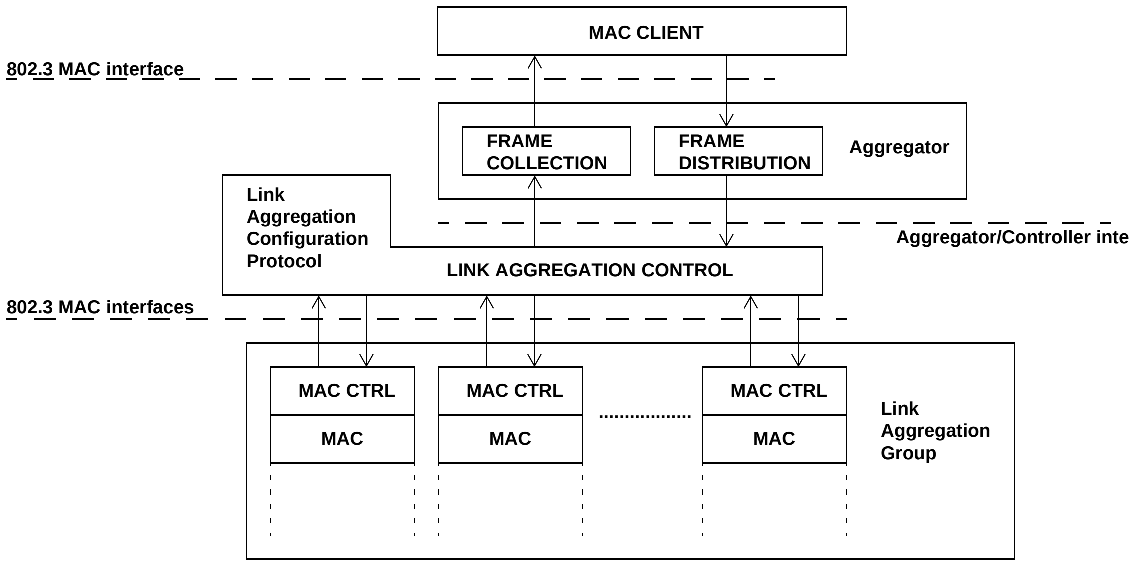

Figure 92-1—Link Aggregation components and interfaces

this physical link is enabled with respect to its participation in an aggregation; i.e., received frames will be passed up to the Aggregator for collection;

f) The status of interaction between the Frame Distribution function of the Aggregator and the link (Distribution Enabled, or Distribution Disabled). Distribution Enabled indicates that the transmit function of this physical link is enabled with respect to its participation in an aggregation; i.e., frames may be passed down from the Aggregator’s distribution function for transmission.

92.3.4.2 Interface between Link Aggregation Control and the Collector and Distributor

This interface is used by Link Aggregation Control to:

a) Inform the Collector and Distributor of the identity of its associated Link Aggregation Group;
b) Inform the Collector and Distributor of the Collection and Distribution status of each link in the Link Aggregation Group.

Data transfer interactions between the Aggregator and the individual links that form its Link Aggregation Group are controlled by the status information communicated to the Aggregator by Link Aggregation Control.

<<Author’s Note: Although the currently agreed architecture diagram implies that data transfer between the Aggregator and the links passes via Link Aggregation Control, the above description seems to be rather more rational; i.e., the link Aggregation Control establishes the state information that determines which Aggregator talks to which links, and the actual data exchange takes place directly between the Aggregator & the designated links in its aggregation group, under control of the state information maintained by the controller. Perhaps a little re-drawing might help here.>>

92.3.4.3 Interface between Aggregator and MAC Client

The Aggregator presents a logical 802.3 MAC interface to the MAC Client. Link Aggregation Control maintains the following information with respect to the interface:

a) The status of interaction between the Frame Collection function of the Aggregator and the MAC Client (Receive Enabled, or Receive Disabled);

b) The status of interaction between the Frame Distribution function of the Aggregator and the MAC Client (Transmit Enabled, or Transmit Disabled).

These status values are exactly equivalent to the logical OR of the status of the Collection and Distribution status of the individual links; in other words, if one or more links in the Link Aggregation Group are Collection Enabled, then the Aggregator is Receive Enabled, and if one or more links are Distribution Enabled, then the Aggregator is Transmit Enabled.

The Transmit/Receive status of the Aggregator effectively governs the point at which the Aggregator becomes available for use by the MAC Client, or conversely, the point at which it ceases to be available.

92.3.5 System, aggregation, link and compatibility identification

In order to allow Link Aggregation Control to determine whether a set of links connect to the same system, and to determine whether those links are compatible from the point of view of aggregation, it is necessary to be able to establish:

a) A globally unique identifier for each system that is to participate in Link Aggregation;
b) A means of identifying the set of capabilities that are associated with each link, and with each Aggregator, as understood by a given system;
c) A means of identifying a Link Aggregation Group and its associated Aggregator.

<<Author’s Note: May also prove necessary to use global labels for the ends of individual links; if so, their individual MAC addresses would be used - see 92.3.6.>>

92.3.5.1 System identification

The globally unique identifier used to identify a system will be an individual MAC address.

NOTE—The MAC address chosen to identify a system may be the individual MAC address associated with one of its links.

92.3.5.2 Capability identification

A number of factors can determine the capabilities of a given link with respect to its ability to aggregate:

d) Its physical characteristics as determined by the ISO/IEC 8802.3 standard, such as speed, whether full duplex or not, point-to-point or shared medium, etc.;
e) Configuration constraints established by the network administrator;
f) Factors related to higher layer use of the link;
g) The characteristics or limitations of the implementation itself.

Some of these factors will be subject to standardization; others are potentially open for definition external to the scope of the standard.

In order to make it possible for link capabilities to be compared within a given system, and for capability information to be exchanged between systems, a *Key* value is associated with each link. A given Key value is meaningful in the context of the system that allocates it. Hence, if a System S labels a set of links with Key K, say, then it can be assumed that any subset of that set of links can potentially be aggregated together, should it prove to be the case that the subset all terminate in System T and that System T has labelled all of the links in the subset with Key L. The set of links in a given system that share the same Key value are said to be members of the same *Key Group*.

A Key value is also associated with each Aggregator. Links that are members of a given Key Group can only be bound to Aggregators that share the same Key value. This allows administrative configuration to determine which Aggregator(s) a given link can be bound to.

NOTE 1—This allows for two convenient initial configurations. The first is achieved by assigning each Port an initial Key value identical to its Port number, and the same Port numbers are assigned to the corresponding Aggregators. A device with this initial configuration will bring up all links as individual, non-aggregated links. The second is achieved by assigning the same Key value to all Ports. A device with this initial configuration will attempt to aggregate together any set of Links that have the same remote System ID and Key, and for which the remote system is prepared to allow aggregation.

The Key is a simple 16-bit identifier; i.e., there is no sub-structuring within the value allocated that has any significance. All values of Key are freely available for allocation, with locally significant meanings.

<<Author’s Note: There was a suggestion at the Austin meeting that there might be value in reserving the Null Key value, 0 to be used as a means of simply configuring/signalling links that cannot be aggregated.>>

NOTE 2—The above assumes a single Key value is assigned to each link and to each Aggregator. Consequently, given links A, B, C, is it never the case that aggregating A+B is acceptable, aggregating B+C is acceptable, but aggregating A+C is not acceptable.

92.3.5.3 Link Aggregation Group identification

A Link Aggregation Group consists of a set of links that all terminate in the same pair of systems, and that, for each system, the links are members of the same Key Group. A Link Aggregation Group Identifier (LAG ID) is therefore a compound identifier, consisting of:

a) The System Identifier associated with one end of the set of links, and the Key assigned to the set of links by that system; and
b) The System Identifier associated with the other end of the set of links, and the Key assigned to the set of links by that system.

Hence, if System S has allocated Key K to a given Link Aggregation Group, and the same Link Aggregation Group terminates in System T with Key L, the (globally unique) identifier of that Link Aggregation Group is {SK, TL}.

NOTE 1—There is no significance to the ordering of these system/Key pairs; hence {SK, TL} is the same as {TL, SK}, but {SK, TL} is not the same as {SL, TK}. A consequence of this formulation for the ID of a Link Aggregation Group is that, for a given {SK, TL} combination, only a single Link Aggregation Group can exist - see 92.3.6.

NOTE 2—It may also prove to be convenient for some purposes to represent the {SK,TL} pair by a locally significant identifier.

92.3.5.4 Aggregator identification

An Aggregator Identifier (Aggregator ID) is a globally unique identifier consisting of an individual MAC address.

NOTE—This identifier may be the MAC address of one of the links in the associated Link Aggregation Group, or may be a distinct MAC address. The manner in which such addresses are chosen is not constrained by this standard.

92.3.6 Configuration capabilities and restrictions

The formulation chosen for the Link Aggregation Group identifier (92.3.5.3) has the consequence that it is not possible to represent two or more Link Aggregation Groups that share the same combination of {SK, TL}. Hence, placing configuration restrictions on the size of an aggregation (e.g., for a Key Group containing N members, restricting the size of any aggregation to subsets of N of no greater than M members) is only possible if it is also acceptable that only one Link Aggregation Group can be constructed from that Key Group for a given {SK, TL}. In practice, this restriction can be somewhat alleviated by sub-dividing Key Groups and allocating different Keys to each subdivision; however, this is in general only useful if the form of the size restriction is closely bound to physical subdivisions in the implementation - for example, it might be possible to aggregate only those links that are on the same interface card.

If restrictions on the size of Link Aggregation Groups are permitted by the standard, then, in order to maintain the objective of deterministic convergence of the configuration, it will be necessary to communicate link identifiers between participating systems, and for the configuration to substitute links in the Group as they become available or unavailable.

<<Author's Note: Definitely desirable not to have such size restrictions if at all possible, as it complicates matters for the protocol; achieving a deterministic result with such restrictions would involve the establishment of a master/slave relationship between the systems in order to decide whose deterministic configuration to use. However, it is probably smart from a debugging perspective to have a link ID carried in the protocol anyway. The detailed operation and protocol sections below assume no such size restrictions; the mechanisms proposed therefore allow a peer relationship to exist between pairs of protocol partners.>>

92.3.7 Operation of Link Aggregation Control

92.3.7.1 Allocating a Link to a Link Aggregation Group

The operation of Link Aggregation Control will result in each Link either:

a) being allocated to a Link Aggregation Group that can potentially contain multiple links, or
b) being allocated to a Link Aggregation Group that can only contain a single link,

depending upon the configuration at either end of the link and the ability/inability of the remote system to engage in Link Aggregation behavior. If the local configuration indicates that the link cannot be aggregated (e.g., the link is the only member of its Key Group), or if information received from the remote system indicates that the remote system considers the link to be non-aggregatable, then the link can only operate as part of a Link Aggregation Group with a single member (itself); in other words, it will operate as an individual link. If the local and remote configuration indicates that the link can be aggregated, then the membership of the Link Aggregation Group may include other links that share the same LAG ID.

The allocation, and re-allocation, of links to Link Aggregation Groups is determined by the current values of the LAG ID parameters held for each link; changes in these values also affect the behavior of a link with respect to whether or not Collection and Distribution are enabled. A key factor in this process is whether the Local and Remote systems agree on the value of LAG ID, or whether they disagree. The following possibilities exist:

c) Local LAG ID and Remote LAG ID differ, either because the local system has not received up-to-date information from the remote system, or vice versa. Attempts will be made (by means of the Link Aggregation Control Protocol, or by other means) to reach a state where this information no longer differs; in which case, the situation becomes one of d) or e) below. However, if this situation persists regardless of attempts to update the LAG ID information, it can be assumed that the remote system cannot take part in Link Aggregation, and the link is therefore not aggregatable; i.e., it can only be operated as a non-aggregated 802.3 link.
d) Local LAG ID and Remote LAG ID are the same, but the Link is not aggregatable; i.e., both systems can potentially take part in Link Aggregation, however, one or other system regards this link as not suitable for aggregation. The link can only be operated as a non-aggregated 802.3 link.
e) Local LAG ID and Remote LAG ID are the same, and the information exchanged between the systems indicates that the Link is aggregatable; i.e., both systems can take part in Link Aggregation,

and both systems regard this link as suitable for aggregation. The link can be enabled as a member of the LAG associated with that LAG ID; i.e., it can be aggregated with other links that share the same LAG ID, and the LAG can be associated with an Aggregator that shares the same (local) Key as the links in the LAG.

NOTE—In a properly configured system, there should always be a suitable Aggregator available, with the same Key assigned, to serve a newly created Link Aggregation Group. However, given the flexibility of the Key scheme, and given that there may not be enough Aggregators to go round in some implementations, it is possible to create configurations in which there is no Aggregator available to service a newly identified LAG, in which case, the LAG cannot become an active aggregation until such a time as the configuration is changed to free up an appropriate Aggregator.

In case d), the link is simply enabled as a normal 802.3 link.

In case e), the link is added to the Link Aggregation Group identified by the LAG ID; if the Link Aggregation Group does not exist, Link Aggregation Control will create the group.

Once the link has been added to a Link Aggregation Group, its Local Collector state can be switched to Enabled, thus preparing the link for reception of frames from the remote Frame Distribution function, and that information communicated to the remote Link Aggregation Controller. If at least one link in the Link Aggregation Group has its Local Collector state Enabled, then the Receive state of the corresponding Aggregator will also be Enabled. Once the state information held for the Link also indicates that the Remote Collector state is enabled, Link Aggregation Control can set the Local Distributor state to Enabled, thus allowing the link to be used by the Frame Distributor function. If at least one link in the Link Aggregation Group has its Local Distributor state Enabled, then the Transmit state of the corresponding Aggregator will also be Enabled.

NOTE—This description assumes that the implementation is capable of controlling the state of the transmit and receive functions of the MAC independently. In implementations where this is not possible, the transmit and receive functions are enabled or disabled together.

92.3.7.2 Moving a Link to a new Link Aggregation Group

If the LAG ID information for a link changes, due to re-configuration of either end of the link, then it will be necessary for Link Aggregation Control to move the link from its existing Link Aggregation Group to a new Link Aggregation Group. At the point where the change is detected, the Local Frame Collector and Local Frame Distributor states are set to Disabled. The link can then be removed from its current Link Aggregation Group, once it is certain that there are no more frames that are in transit on the link. This may involve the use of an explicit *flush protocol* that ensures that no frames remain to be received at either end of the link before re-configuration takes place. The initiation and operation of the flush protocol is described in 92.5; the decision as to when, or if, the flush protocol is used is entirely dependent upon the nature of the distribution algorithm that is employed on a given link or set of links.

Once the link has been removed from its Link Aggregation Group, the situation has effectively been reduced to the one described in 92.3.7.1; the link can be allocated to its new Link Aggregation Group and re-enabled once agreement has been reached between the Local and Remote LAG IDs.

92.4 Link Aggregation Control Protocol (LACP) and state machines

The Link Aggregation Control Protocol provides a means whereby the necessary information can be exchanged between the two ends of a Link in order to allow two interacting Link Aggregation Control instances to reach agreement on the identity of the Link Aggregation Group to which the Link belongs, move the link to that Link Aggregation Group, and enable its transmission and reception functions in an orderly manner.

NOTE—This is only one means whereby such agreement could be reached. Others include static configuration of the necessary parameters, or the use of information gleaned from other protocol mechanisms.

92.4.1 Protocol design principles

The following design principles were used in developing the protocol described in this section:

a) The protocol depends upon the transmission of *information* and *state*, rather than the transmission of *commands*. In other words, LACPDUs sent by the first party (the *Actor*) convey to the second party (the Actor’s protocol *Partner*) what the Actor knows, both about its own state and that of its Partner;
b) The information conveyed in the protocol is sufficient to allow the Partner to determine what action to take next.
c) Active or passive participation in LACP is controlled by *Actor’s Activity*, an administrative control associated with each Port. Actor’s Activity can take two values; *Active LACP* or *Passive LACP*. Passive LACP indicates the Port’s willingness to participate in the protocol, but only if its Partner’s control value is Active LACP. Active LACP indicates the Port’s desire to participate in the protocol regardless of the Partner’s control value.
d) Periodic transmission of LACPDUs occurs if either the Activity control of the Actor or the Partner is Active LACP. These transmissions are based on slow or fast transmission rate depending upon the expressed *LACP_Timeout* preference (*Short Timeout* or *Long Timeout*) of the Partner system.
e) In addition to periodic LACPDU transmissions, the protocol transmits LACPDUs when there is a *Need To Tell* something to the Partner; i.e., when the Actor’s state changes, or when it is apparent from the Partner’s LACPDUs that the Partner does not know the Actor’s current state;
f) The protocol assumes that the rate of LACPDU loss is very low.

NOTE—There is no explicit frame loss detection/retry mechanism defined; however, if information is received from the Partner indicating that it does not have up to date information on the Actor’s state, or if the next periodic transmission is due, then the Actor will transmit an LACPDU that will correctly update the Partner.

92.4.2 State machine overview

The operation of the protocol is controlled by a number of state machines, each of which performs a distinct function. These state machines are for the most part described on a per-Port basis; any deviations from per-Port description are highlighted in the text. The events that cause transitions in these state machines are “ticks” generated at a regular time interval, and incoming PDUs that are processed in order to extract significant information. These events may cause state transitions and also cause actions to be taken; those actions may include the need for transmission of an LACPDU containing repeated or new information. Irregular transmissions such as these are controlled by the state of the Need To Transmit signal (NTT - see 92.4.3.4), which can be sent by any of the state machines as necessary.

The operation of these state machines assumes that any information contained in an incoming LACPDU is processed by each state machine sequentially, in the order that the state machines are described here. The state machines are as follows:

a) *Receive Machine (RX - 92.4.5).* This state machine receives LACPDUs from the Partner, records the information contained therein and times it out using either Short Timeouts or Long Timeouts, according to the mode to which it has been set.
b) *Periodic Transmission Machine (92.4.6)*. This state machine determines whether the Actor and its Partner will exchange LACPDUs periodically in order to maintain an aggregation (either or both are configured for Active LACP), or whether no attempt will be made to maintain an aggregation (both configured for Passive LACP).
c) *Match Logic (92.4.7)*. This state machine determines if the Actor and Partner have both agreed upon the protocol information exchanged to the extent that the physical port can now be safely used in an aggregate, either aggregated with other links or as an individual port.

d) *Selection Logic and Machine (92.4.8)*. This state machine is responsible for selecting the Aggregator to be associated with this physical port.
e) *Mux Control and Logic Machine (MUX - 92.4.9)*. This state machine turns the distributor and the collector for the physical port on or off as required by the current protocol information.
f) *Transmit Machine (TX - 92.4.10)*. This state machine handles the transmission of LACPDUs, both on demand from the other state machines, and on a periodic basis.
g) *Churn Detection Machine (92.4.11)*. This state machine makes use of the In Sync and Out Of Sync signals generated by the Mux state machine in order to detect the situation where the state machines are unable to resolve the state of a given link; for example, because the Partner system repeatedly sends conflicting information in its LACPDUs. As this situation should not occur in a normally functioning link, this state machine simply detects the presence of such a condition and signals its existence to management.

Figure 92-2 illustrates the relationships between these state machines, and the flow of information between them. The set of arrows labelled New Info represent the information contained in an incoming LACPDU being fed to each state machine in turn. The set of arrows labelled Outgoing PDU represents the collection of current state information for transmission, as a result of the need to transmit a periodic LACPDU, or as a result of one or more state machines asserting NTT. The remaining arrows represent signals that allow one state machine to cause events to occur in one or more other state machines.

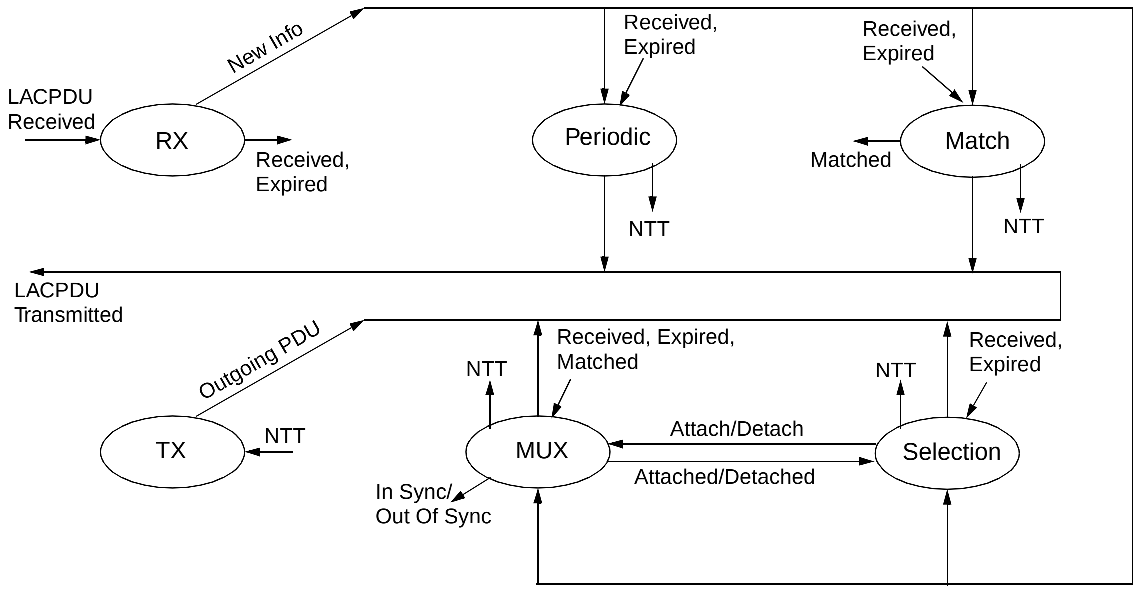

Figure 92-2—Interrelationships between state machines

92.4.3 Parameters

92.4.3.1 Information carried in LACPDUs

The Link Aggregation Control Protocol operates by each of the two Link Aggregation Control Protocol Entities attached to a Link declaring to the other what it currently knows of the state of that Link. The information exchanged consists of the following parameters:

a) *Actor_Port*. The identifier assigned to the physical Port by the local system (the system sending the PDU). This identifier is not used directly by the protocol, but is included for debugging use.
b) *Actor_System*. The System ID (92.3.5.1), S, of the local system.

c) *Actor_Key.* The Key (92.3.5.2), K, that the local system has assigned to the Link.
d) *Actor_State*, comprising the following flags:
    1) *LACP_Activity.* This flag indicates the Actor’s Activity control value with regard to this link. True indicates a value of Active LACP. False indicates Passive LACP.
    2) *LACP_Timeout*. This flag indicates the Actor’s Timeout control value with regard to this link. True indicates a value of Short Timeout; i.e., the Actor requires its Partner to use short timeouts for periodic PDU transmission. False indicates Long Timeout; its Partner is free to use either short or long timeouts for periodic PDU transmission.
    3) *Aggregability.* If True, this flag indicates that the Actor considers this link to be *Aggregatable; i.e.,* a potential candidate for aggregation. If False, the Actor considers this link to be *Individual*; i.e., this link can be operated only as an individual link.
    4) *Synchronization.* If True, then the Actor considers this link to be *In_Sync*; i.e., it is in the right Aggregation, so it has been allocated to the correct Link Aggregation Group, the group has been associated with a suitable Aggregator, and the identity of the Link Aggregation Group is consistent with the System ID and Key information transmitted. If False, then this link is currently *Out_Of_Sync;* i.e., it not in the right Aggregation.
    5) *Collecting.* True if the Actor has enabled collection of incoming frames on this link.
    6) *Distributing.* True if the Actor has enabled distribution of outgoing frames on this link.
e) *Partner_Port*. The identifier assigned to the physical Port by the Partner system, as understood by the Actor. This identifier is not used directly by the protocol, but is included for debugging use.
f) *Partner_System*. The System ID (92.3.5.1), T, of the Partner system, as understood by the Actor.
g) *Partner_Key.* The Key (92.3.5.2), L, that the Partner system has assigned to the Link, as understood by the Actor.
h) *Partner_State.* This conveys the Actor’s current knowledge of its Partner’s state, comprising the following flags:
    1) *LACP_Activity.* This flag indicates the Partner’s Activity control value with regard to this link. True indicates a value of Active LACP. False indicates Passive LACP.
    2) *LACP_Timeout*. This flag indicates the Partner’s Timeout control value with regard to this link. True indicates a value of Short Timeout; i.e., the Partner requires its Partner (the Actor) to use short timeouts for periodic PDU transmission. False indicates Long Timeout; its Partner is free to use either short or long timeouts for periodic PDU transmission.
    3) *Aggregability.* If True, this flag indicates that the Partner considers this link to be *Aggregatable; i.e.,* a potential candidate for aggregation. If False, the Partner considers this link to be *Individual*; i.e., this link can be operated only as an individual link.
    4) *Synchronization.* If True, the Partner considers this link to be *In_Sync*; i.e., it is in the right Aggregation, so it has been allocated to the correct Link Aggregation Group, the group has been associated with a suitable Aggregator, and the identity of the Link Aggregation Group is consistent with the System ID and Key information transmitted. If False, then this link is currently *Out_Of_Sync;* i.e., it not in the right Aggregation.
    5) *Collecting.* True if the Partner has enabled collection of incoming frames on this link.
    6) *Distributing.* True if the Partner has enabled distribution of outgoing frames on this link.

Note that in the above, the “Actor” and “Partner” qualifiers are interpreted from the perspective of the sender of the PDU, the sender being the Actor.

92.4.3.2 Aggregate Port (Aggregator) parameters

a) *MAC_address*. The MAC address assigned to this logical MAC interface.
b) *Port*. The Port number of this Aggregator.
c) *Aggregate*. The Link Aggregation Group associated with this Aggregator.
d) *Individual_Port.* True if the LAG associated with this Port contains one physical Port & that Port is not capable of aggregation with any other Port.
e) *My_Key.* The Key value associated with this Aggregator that defines which links can aggregate via this Aggregator.

f) *Partner_System*. The System ID of the remote system to which this Aggregator is connected - zero if unknown.
g) *Partner_Key*. The Key assigned to this aggregation by the remote system to which this Aggregator is connected - zero if unknown.

NOTE—The above assumes that there are the same number of Aggregators as there are physical MAC interfaces, and that Aggregators and Ports are paired as an initial state. As mentioned earlier, this allows a convenient default case, where an Aggregator is initialized with Actor_Key and Port set equal to the Port number of its associated Physical Port, and the Physical Port’s Actor_Key parameter also set to the same Port number. The result is that all links will operate as individual links “out of the box”. Re-configuration of keys on either the Port or the Aggregator allows aggregation to take place appropriately, resulting in some Ports aggregating via a different Aggregator, and some Aggregators having no active Ports associated with them. If management of the Key values always changes both the Agport & its associated Phyport to the same value, then it is always the case that a Port can find a suitable Agport, regardless of what else has happened to the configuration.

92.4.3.3 Link Aggregation Group parameters

a) *LAG_ID*. The identifier of the Link Aggregation Group (Concatenated values SC,TD - see 92.3.5.3).
b) *Ports*. The set of Ports that belong to this Link Aggregation Group.
c) *Aggregator*. The identifier of the Aggregator associated with this LAG.
d) *Wait_While*. The number of Ticks yet to elapse before performing an aggregation change (e.g., to allow all links that will join this Aggregation to do so).
e) *Collector_State*. The state of the Collector for this Link Aggregation Group (Enabled or Disabled).
f) *Distributor_State*. The state of the Distributor for this Link Aggregation Group (Enabled or Disabled).

92.4.3.4 Physical Port instance parameters

The following parameters are maintained for each physical Port:

a) *Port*. The Port number.
b) *LAG_ID*. The LAG of which this Port is a member.
c) *Aggregator*. The identifier of the Aggregator associated with this Port.
d) *NTT*. Need To Transmit flag. If True, this flag indicates that there is new information that should be transmitted on the link, or that the remote system(s) need to be reminded of the old information.
e) *Transmit_When*. The number of Ticks yet to elapse before the next regular protocol transmission is due.
f) *My_Key*. The Key value assigned to this Port by the Actor.
g) *My_Port*. The Port number assigned to this link by the Actor.
h) *My_State*. The current values of the Actor’s state parameters, as described in 92.4.3.1.
i) *Your_System*. The System ID of the Partner.
j) *Your_Key*. The Key value assigned to this link by the Partner.
k) *Your_Port*. The Port number assigned to this link by the Partner.
l) *Your_State*. The Actor’s view of the current values of the Partner’s state parameters, as described in 92.4.3.1
m) *Current_While*. The number of Ticks yet to elapse before timing out the values of all “*Your_XX*” parameters.

92.4.3.5 Tick counter initialization values

The basic unit if timing in this protocol is the *Tick*, which occurs at intervals of 1 s. Longer timers and delays are derived from the Tick by means of Tick counters associated with some of the state machines. The Tick event applied to each state machine causes these counters to be decremented.

The default values used to initialize these tick counters are as follows:

Fast_Transmission_Ticks = 1

Slow_Transmission_Ticks = 30

Slow-Expiry_Ticks = 90.

Fast_Expiry_Ticks = 3.

Aggregate_Delay_Ticks = 5.

92.4.4 State Machine Notation

The state diagrams use the following notation.

a) Circles represent *states;* the name of the state is written inside the circle. The state machine will occupy its current state until an *event* causes a transition to the next state.
b) Arcs represent possible transition paths from one state to another state; the arrowhead on the arc indicates the direction of transition.
c) Associated with each arc are one or more events that can cause the transition to take place. The list of events is terminated with a colon, and multiple events are separated by commas. If more than one event is associated with an arc, then this indicates the fact that any of the events listed can cause the transition. The definition of each event accompanies the state diagram in the following sub-clauses.
d) Following the list of events is a list of *actions* that are executed and *signals* that are sent as part of the transition between the two states. Action names are surrounded by brackets, [thusly]; signal names are preceded by a bullet point •thusly. The definitions of the actions and signals follow the event definitions for the state machine.

The state tables use a tabular form of this notation. A given row of the table specifies a single event that may be received by the state machine. The cells in the body of the table indicate the next state reached for the stated event. An X in a cell indicates that the event concerned does not occur in this state. A dash indicates that the event causes no state transition to occur; however, there may be action(s) or signal(s) involved with that event. Otherwise, the actions and signals that are associated with the transition are listed, followed by the name of the next state.

92.4.5 Receive Machine

The Receive Machine state diagram is shown in Figure 92-3. The initial state is EXPIRED.

On receipt of an LACPDU, the state machine enters the CURRENT state, having recorded the information contained in the PDU, and having started the current_while timer is started, and the infoReceived signal is generated to indicate to the other state machines that new information is available for processing.If no PDU is received before the current_while timer expires, or if a reinitialisation event occurs, the state machine transits to the EXPIRED state, having cleared out the current PDU information and generated the infoExpired signal to indicate to the other state machines that the Partner information has timed out.

The Receive machine has a single timer, current_while, that runs in the CURRENT state. The timer is started or restarted on receipt of a valid LACPDU. Its starting value is determined by the current administrative setting of the Actor’s Timeout parameter.

The state table for Receive Processing is shown in Table 92-1.

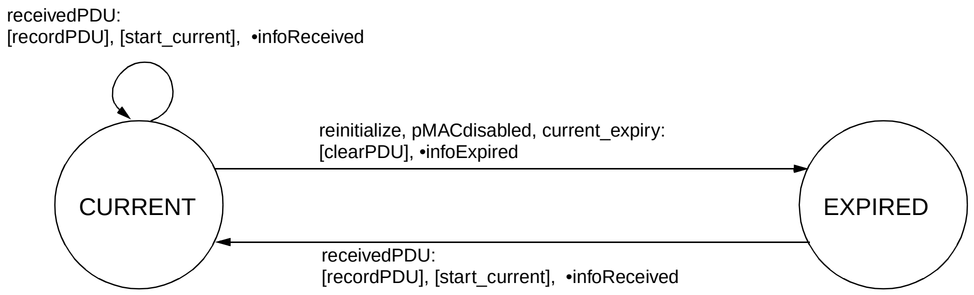
receivedPDU:
[recordPDU], [start_current], •infoReceived
reinitialize, pMACdisabled, current_expiry:
[clearPDU], •infoExpired
CURRENT
EXPIRED
receivedPDU:
[recordPDU], [start_current], •infoReceived

Figure 92-3—Receive Machine state diagram

Table 92-1—Receive machine state table

EVENT	**STATE**	
	EXPIRED	**CURRENT**
create	-	X
reinitialize	-	[clearPDU] •infoExpired EXPIRED
receivedPDU	[recordPDU] [start_current] •infoReceived CURRENT	[recordPDU] [start_current] •infoReceived -
pMACdisabled	-	[clearPDU] •infoExpired EXPIRED
current_expiry	-	[clearPDU] •infoExpired EXPIRED

92.4.5.1 recordPDU

This action records the protocol parameters received in the LACPDU and makes them available to the other state machines. The information recorded from incoming PDUs consists of:

a) The Partner’s System ID;
b) The Partner’s Key;
c) The Partner’s Partner System ID;
d) The Partner’s Partner Key;
e) The Partner’s State;
f) The Partner’s View.

NOTE—The Partner’s Partner System ID and Key are the ID and key values that the Partner has recorded for its partner, i.e., what it thinks the Actor’s ID and Key values currently are. Similarly, the Partner’s View is simply what the Partner currently considers the Actor’s state to be. These values could be in their initial state if the Partner has just started operation, or could be old information relating to a previous partner if the link configuration has changed.

92.4.5.2 clearPDU

This action clears the recorded PDU information held from the last LACPDU received.

92.4.5.3 start_current

This action starts the current_while timer, using the value Slow_Expiry_Ticks (92.4.3.5) if the Actor’s Timeout parameter is set to Slow Timeouts, or Fast_Expiry_Ticks if the Actor’s Timeout parameter is set to Fast Timeouts.

92.4.5.4 infoReceived

This signal indicates to the other state machines that new protocol information has been received.

92.4.5.5 infoExpired

This signal indicates to the other state machines that the most recently received protocol information has expired.

92.4.5.6 create

This event is caused by the initial creation of the state machine.

92.4.5.7 reinitialize

This event is caused by protocol entity being initialized or reinitialized.

92.4.5.8 receivedPDU

This event is caused by the reception of a valid LACPDU, formatted as defined in 92.4.12.

92.4.5.9 pMACdisabled

This event is caused by the MAC associated with the physical port becoming inoperable.

92.4.5.10 current_expiry

This event is caused by the expiry of the current_while timer; i.e., a period of time equivalent to Slow_Expiry_Ticks or Fast_Expiry_Ticks has elapsed since the timer was last started.

92.4.6 Periodic Transmission Machine

The Periodic Transmission Machine state diagram is shown in Figure 92-4. This machine establishes the desire of the participants to exchange periodic LACPDUs on the link in order to maintain the aggregate, and how often those periodic transmissions should occur. Periodic transmissions will take place if either participant so wishes. Transmissions occur at a rate determined by the receiving participant; this rate is linked to the speed at which that participant will time out received information.

The state machine has three states:

a) NO_PERIODIC. While in this state, periodic transmissions are disabled; however, if there are LACPDUs still to be exchanged for the purposes of other state machines in order to effect graceful close-down of an aggregate, then these transmissions will still take place;

b) FAST_PERIODIC. While in this state, periodic transmissions are enabled at the fast transmission rate;
c) SLOW_PERIODIC. While in this state, periodic transmissions are enabled at the slow transmission rate.

The values of the administratively settable parameter Actor’s Activity at each end of the link determines whether periodic transmissions will take place. If either or both parameters are set to LACP Active, then periodic transmissions occur; if both are set to LACP Passive, then periodic transmissions do not occur. If periodic transmissions are enabled, the rate at which they take place is determined by the value of the Partner’s Timeout parameter.

Two timers are maintained by the Periodic Transmission Machine; the *periodic_transmission* timer and thr *fast_while* timer.

The periodic_transmission timer stimulates periodic transmissions to ensure that the Actor’s information is not timed out byits Partner, and, if the Actor is Active LACP, to discover a new Partner. In the FAST_PERIODIC state, and in the SLOW_PERIODIC state, it is restarted with a value of Slow_Transmission_Ticks.

If the Receive machineis in the CURRENT state, the Periodic Transmission Machine state is selected by the received value of the Partner’s LACP_Timeout flag - FAST_PERIODIC for Short Timeout, SLOW_PERIODIC for Fast Timeout.

If the Receive nachine is EXPIRED (including no information having been received from creation, initialization or enabling the physical MAC), the Periodic Transmission Machine will be NO_PERIODIC if the Actor is Passive LACP. Otherwise, if the Actor is Active LACP, the state will be FAST_PERIODIC if the fast_while timer is running, or SLOW_PERIODIC if the fast_while timer has expired. The fast_while timer is started or restatred with an initial value of Fast_Expiry_Ticks when the machine is reinitialized or when the Receive machine signals Expired. The fast_while timer thus operates to ensure that the Partner is discovered rapidly while allowing for a slow periodic rate as the steady state.

The state table for the Periodic Transmission machine is shown in Table 92-2.

92.4.6.1 init_active

This event occurs if the Actor’s Activity parameter is set to Active LACP and any of the following occur:

a) The physical MAC is enabled;
b) The state machine is created;
c) The state machine is re-initialized.

NOTE—The effect of the treatment of initialization events, coupled with the operation of the timers, is that an active participant will send a few LACPDUs at the fast rate before settling down to slow rate transmissions (assuming that its partner does not require fast transmissions). Passive participants remain quiet until spoken to.

92.4.6.2 init_passive

This event occurs if the Actor’s Activity parameter is set to Passive LACP and any of the following occur:

a) The physical MAC is enabled;
b) The state machine is created;
c) The state machine is re-initialized.

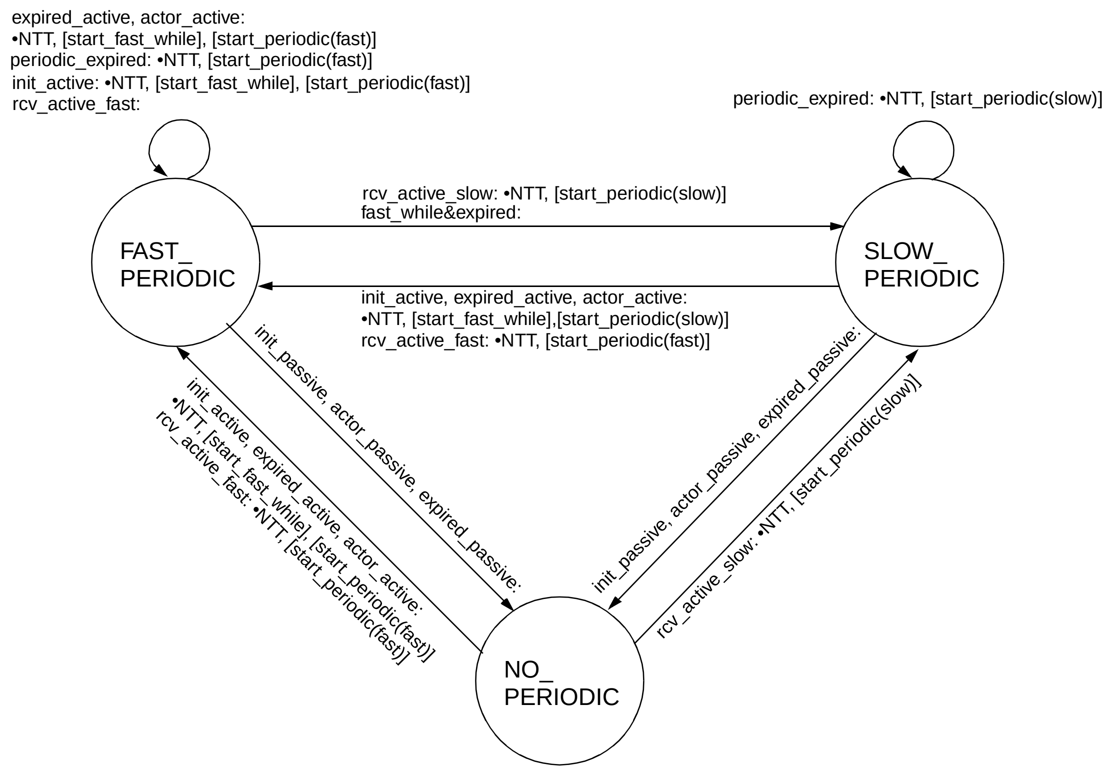
expired_active, actor_active:
•NTT, [start_fast_while], [start_periodic(fast)]
periodic_expired: •NTT, [start_periodic(fast)]
init_active: •NTT, [start_fast_while], [start_periodic(fast)]
rcv_active_fast:
periodic_expired: •NTT, [start_periodic(slow)]
rcv_active_slow: •NTT, [start_periodic(slow)]
fast_while&expired:
FAST_ PERIODIC
SLOW_ PERIODIC
init_active, expired_active, actor_active:
•NTT, [start_fast_while],[start_periodic(slow)]
rcv_active_fast: •NTT, [start_periodic(fast)]
init_passive, actor_passive, expired_passive:
init_active, expired_active, actor_active:
•NTT, [start_fast_while], [start_periodic(fast)]
rcv_active_fast: •NTT, [start_periodic(fast)]
init_passive, actor_passive, expired_passive:
rcv_active_slow: •NTT, [start_periodic(slow)]
NO_ PERIODIC

Figure 92-4—Periodic Transmission Machine state diagram

92.4.6.3 rcv_active_fast

This event occurs if an LACPDU is received that indicates that the Partner's Timeout parameter is set to Fast Timeouts, and the Activity parameter of the Actor, the Partner or both are set to Active LACP.

92.4.6.4 rcv_active_slow

This event occurs if an LACPDU is received that indicates that the Partner's Timeout parameter is set to Slow Timeouts, and the Activity parameter of the Actor, the Partner or both are set to Active LACP.

92.4.6.5 rcv_passive

This event occurs if an LACPDU is received, and the Activity parameter of the Actor and the Partner are both set to Passive LACP.

92.4.6.6 expired_active

This event occurs if the Receive machine signals that information from the Partner has expired and the Activity parameter of the Actor is set to Active LACP.

92.4.6.7 expired_passive

This event occurs if the Receive machine signals that information from the Partner has expired and the Activity parameter of the Actor is set to Passive LACP.

Table 92-2—Periodic Transmission Machine state table

EVENT	STATE		
	NO_PERIODIC	FAST_PERIODIC	SLOW_PERIODIC
init_active	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC	• NTT [start_fast_while] [start_periodic(fast)] -	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC
init_passive	-	NO_PERIODIC	NO_PERIODIC
rcv_active_fast	• NTT [start_periodic(fast)] FAST_PERIODIC	-	• NTT [start_periodic(fast)] FAST_PERIODIC
rcv_active_slow	• NTT [start_periodic(slow)] SLOW_PERIODIC	• NTT [start_periodic(slow)] SLOW_PERIODIC	-
rcv_passive	-	NO_PERIODIC	NO_PERIODIC
expired_active	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC	• NTT [start_fast_while] [start_periodic(fast)] -	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC
expired_passive	-	NO_PERIODIC	NO_PERIODIC
actor_active	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC	• NTT [start_fast_while] [start_periodic(fast)] -	• NTT [start_fast_while] [start_periodic(fast)] FAST_PERIODIC
actor_passive	-	NO_PERIODIC	NO_PERIODIC
fast_while&expire d	-	SLOW_PERIODIC	-
fast_while&current	-	-	-
periodic_expired	-	• NTT [start_periodic(fast)] -	• NTT [start_periodic(slow)] -

92.4.6.8 actor_active

This event occurs if the Activity parameter of the Actor is set to Active LACP by administrative action.

92.4.6.9 actor_passive

This event occurs if the Activity parameter of the Actor is set to Passive LACP by administrative action.

NOTE—The action upon receiving this event could be improved upon by taking account of information from the Receive machine, indicating that the Partner is active. However, as management events such as this are infrequent, and the status quo will in any case be restored in such cases by the Partner’s next transmission, this is not an issue.

92.4.6.10 periodic_expired

This event occurs if the periodic_transmission timer expires.

92.4.6.11 fast_while&expired

This event occurs if the fast_while timer expires and the Receive Machine is in the EXPIRED state.

92.4.6.12 fast_while&current

This event occurs if the fast_while timer expires and the Receive Machine is in the CURRENT state.

92.4.6.13 •NTT

The Need To Transmit (NTT) signal communicates to the Transmit machine the need to transmit an LACPDU.

92.4.6.14 start_fast_while

The fast_while timer is started or restarted. The value used to restart the timer is Fast_Expiry_Ticks (92.4.3.5).

92.4.6.15 start_periodic(fast)

The periodic_transmission timer is started or restarted, using the value fast_transmission_ticks (92.4.3.5).

92.4.6.16 start_periodic(slow)

The periodic_transmission timer is started or restarted, using the value slow_transmission_ticks (92.4.3.5).

92.4.7 Match Logic

The Match Logic determines whether the participants have both agreed upon the protocol information that they have exchanged, to the extent that the physical Port can safely be used in an aggregate. This agreement may result in an agreement that the Port can only be used as an individual; i.e., an aggregate consisting of a single link, or that it can be aggregated with other Ports.

The protocol information is *matched* if the physical MAC is enabled and:

a) The Actor has *no Partner*; i.e., the Receive Machine (92.4.5) is in the expired state, and the Selection Logic (92.4.8) has recorded a null System ID as the selected Partner ID;

NOTE 1—This means that either there has never been a Partner seen on this Port since the last re-initialization, or the Receive Machine has timed out the last LACPDU received on this Port. This can be viewed as a special case of the *matched individual* below; the Actor has only itself to agree with, and is therefore *matched*, but must conclude that the Port is individual, as there is no basis upon which to aggregate it with other Ports.

b) Or, the Actor has identified this Port as a *matched individual*, i.e., there is information available from a current Partner, and:
   1) The Partner’s Aggregability state is Individual; or
   2) The Actor’s own Aggregability state is Individual and the received Partner’s View is Individual; or
   3) The Actor has detected that a loopback condition exists on this Port; i.e., the Partner’s System ID and Key are identical to those of the Actor.

NOTE 2—In other words, the link is Individual if the Partner says that it is Individual (the Actor cannot argue with that decision), or if the Port is connected in loopback, or if the Actor thinks it is Individual and the Partner has seen that the Actor thinks that it is Individual. In the latter case, the Partner cannot disagree with the Actor’s conclusion; however, the Actor needs to know that the Partner knows this before the Actor can signal that a Match exists.

<<Author's Note: The loopback detection deals with two cases of loopback:

- "true" loopback resulting from plugging a Port's output into its own input;

- loopback that results from plugging one Port into another Port of the same system, and giving the same Key to both Ports.

In the second case, if the normal logic for aggregation was applied, the result would be an Aggregator that looked like a physical Port with TX plugged into RX.

In either case, an operable Aggregator connected to this kind of plumbing might give some higher layer protocols some interesting times; hence, the above logic treats such Ports as individual. However, this does not fully deal with the "true" loopback case, as this still looks like a "port on a stick" to the higher layers - the only safe thing to do here would seem to be to force such physical Ports to be permanently disabled as far as the operation of Link Aggregation & higher layers are concerned. Note that Bridges that correctly implement the BPDU reception rules should cope well with Ports in loopback.

Loopback where port A is connected to port B on the same system, but A and B have different Keys, is fine, as the result is two Aggregators connected together - Bridges do not fail with this, and other systems should be able to handle it too.

Is this the appropriate treatment for these conditions? Feedback is needed here.>>

c) Or, the Actor has identified this Port as a *matched aggregate*, i.e., there is current information from a Partner, the Partner's Partner ID and Partner Key match those of the Actor, and the Selection Logic has not identified the selected Aggregator as individual;
d) Or, the Actor has detected that a loopback condition exists; i.e., the Partner's Partner ID, Key and port number are identical to those of the Actor.

Otherwise, the protocol information is *not matched*. The output, matched or not matched, of the match logic is used by the Mux Control and Logic (92.4.9).

92.4.8 Selection Logic and Machine

The Selection Logic selects the Aggregator to be associated with the physical Port. The Selection Machine controls the process of detaching the Port from its existing Aggregator, and attaching it to the selected Aggregator, if the selected Aggregator differs from the one that the Port is currently attached to.

NOTE—The description of the Selection Logic that follows describes a selection mechanism that provides an element of determinism (that is to say, history independence) in the assignment of physical Ports to Aggregators. They also have the characteristic that no additional MAC addresses, over and above those already assigned to the set of physical MACs, are needed. However, as the mechanism described is not required for the proper operation of the protocol, the description also points to variants of the stated rules that might be more appropriate to some implementations or goals.

92.4.8.1 Selection Logic

Each physical MAC has both a physical Port and an Aggregator associated with it (i.e., there are the same number of Aggregators as physical Ports); both the physical Port and the Aggregator are allocated the same Key and Port number.

Aggregation is represented by a physical Port *selecting* an appropriate Aggregator, and then *attaching* to that Aggregator. When there are multiple physical Ports in an aggregation, the Aggregator that the set of Ports selects is the Aggregator with the same Port number as the lowest numbered physical Port. The lowest numbered physical Port may not be in a state that allows data to be transferred on its physical link; however, it has selected that Aggregator. This is illustrated in Figure 92-5.

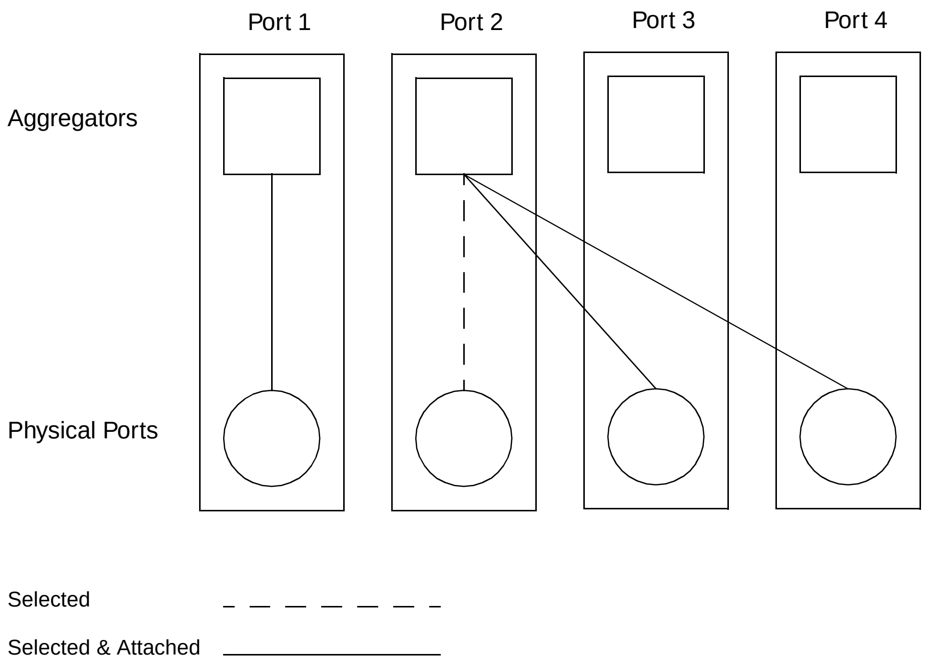

Figure 92-5—Selection of Aggregators

The selection logic operates upon the information recorded by the receive process, along with knowledge of the Actor's own configuration and state. The logic determines:

a) The Partner's System ID and Key;
b) Whether the Port has been identified as an individual link, or whether it may participate in an aggregate with other Ports;
c) Whether a new Partner has been selected, due to a change in the Partner's System ID or Key.

The Partner's System ID and Key, along with the individual/aggregatable state, are recorded.

NOTE—This information is recorded separately from any information that the Receive machine is maintaining as Current; this allows the selection logic to hold a link in its current state of selection when the Receive machine determines that received information has expired. Consequently, expiry of information due to failure of the link (for example, as would happen if a link was unplugged temporarily) can allow an aggregation to remain undisturbed until such a time as the link is restored. If, in restoring the link, the selection state is found to be inappropriate (for example, as could happen if a link was unplugged and then reconnected to a different remote Port), then the selection state changes appropriately. When the Receive state machine is Current and the Mux machine is In Sync, there is no difference between this recorded information and the information already recorded by the Receive machine.

The Partner's System ID and Key, and the individual/aggregatable state, are updated when:

d) New information is signaled by the Receive machine; or
e) The wait_while timer used by the Selection Machine (92.4.8.2) expires; or
f) Any of the Actor's parameters that contribute to the selection process are changed by management.

Selected as Individual is True for the physical Port if the Receive state machine is Expired, if one or more of the Actor's state, the Partner's state or the Partner's View indicate Individual, or if both the Actor's Activity and the Partner's Activity are Passive LACP. In other words, if either the Actor or the Partner suspect the Port should be Selected as Individual, then it will be.

Selected as Individual is also True if the Partner's System ID and Key are identical to those of the Actor; i.e., the Port is connected in loopback (see 92.4.7).

If the Port is Selected as Individual, then the Aggregator selected is always the Port's own Aggregator. Otherwise, the Aggregator selected is the lowest numbered Aggregator with selection parameters that match those of the physical Port. A successful match requires the following parameters to be the same for the Aggregator and the physical Port:

g) The Actor's System ID
h) The Actor's Key;
i) The Partner's System ID;
j) The Partner's Key;
k) The Selected as Individual state (which must be False).

The selection of a new Aggregator by a physical Port, as a result of changes to the selection parameters, can result in other Ports in the system being required to re-select their Aggregators in turn.

92.4.8.2 Selection Machine

The Selection Machine attaches the physical Port to the selected Aggregator. After a physical Port changes its selection of Aggregator, the Selection Machine detaches it from its current Aggregator, then waits for a short period, defined by the wait_while timer, to allow for the possibility that other physical Ports may be reconfiguring at the same time.

NOTE—This waiting time reduces the disturbance that will be visible to higher layers, for example on startup events.

At the end of the waiting period, the selection status is checked to ensure that the selected Aggregator is still valid, and that no other physical Port that has selected the same Aggregator is still waiting. The Selection Machine then attaches the physical Port to the selected Aggregator.

The Selection Machine state diagram is shown in Figure 92-6. The machine has four states; DETACHED, ATTACHING, ATTACHED and DETACHING, and one timer, wait_while.

NOTE—For implementations where there is no delay between requesting that a Port be detached or attached, and the action taking place, the state machine effectively collapses to two states, DETACHED and ATTACHED; however, it is expressed as shown in order to allow for a wider range of implementation possibilities.

The state table for the Selection Machine is shown in Table 92-2.

92.4.8.2.1 change

The change event occurs when the Selection Logic requests a change to the selected Aggregator.

92.4.8.2.2 ready

The ready event occurs when this Port, and any other Ports that have selected the same Aggregator, are not in the process of detaching from another Aggregator, and their wait_while timers have all expired.

92.4.8.2.3 attached

This event is caused by receipt of the attached signal generated by the Mux Control and Logic (92.4.9).

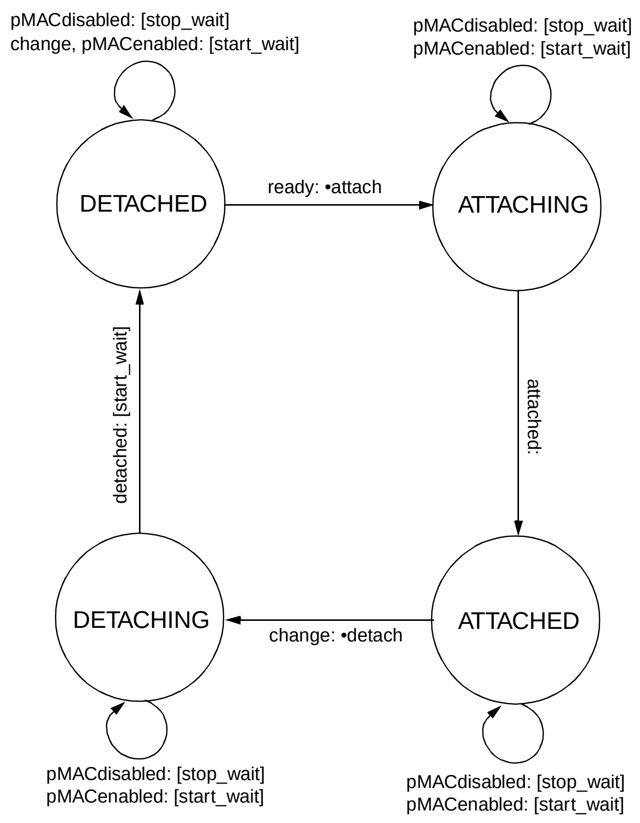
pMACdisabled: [stop_wait]
change, pMACenabled: [start_wait]
pMACdisabled: [stop_wait]
pMACenabled: [start_wait]
DETACHED
ready: •attach
ATTACHING
detached: [start_wait]
attached:
DETACHING
change: •detach
ATTACHED
pMACdisabled: [stop_wait]
pMACenabled: [start_wait]
pMACdisabled: [stop_wait]
pMACenabled: [start_wait]

Figure 92-6—Selection Machine state diagram

Table 92-3—Selection Machine state table

EVENT	**STATE**			
	DETACHED	**ATTACHING**	**ATTACHED**	**DETACHING**
change	[start_wait][a] -	-[b]	• detach DETACHING	-
ready	• attach ATTACHING	-	-	X
attached	X	ATTACHED	X	X
detached	X	X	X	[start_wait][a] DETACHED
pMACenabled	[start_wait] -	[start_wait] -	[start_wait] -	[start_wait] -
pMACdisabled	[stop_wait] -	[stop_wait] -	[stop_wait] -	[stop_wait] -

[a]The wait_while timer is only started if the change is to a possible aggregate. If the Port is selected as individual at any time, the wait_while timer is stopped.
[b]Change will be checked for on transition to the ATTACHED state.

92.4.8.2.4 attached

This event is caused by receipt of the detached signal generated by the Mux Control and Logic (92.4.9).

92.4.8.2.5 pMACenabled

The physical MAC is enabled.

92.4.8.2.6 pMACdisabled

The physical MAC is disabled.

92.4.8.2.7 •attach

This signal instructs the Mux Control and Logic (92.4.9) to attach this physical Port to a specified Aggregator.

92.4.8.2.8 •detach

This signal instructs the Mux Control and Logic (92.4.9) to detach this physical Port from its current Aggregator.

92.4.8.2.9 start_wait

Starts or restarts the wait_while timer, by setting it to the value Aggregate_Delay_Ticks (92.4.3.5).

92.4.8.2.10 stop_wait

Stops the wait_while timer.

92.4.8.3 Selection Logic variants

Two variants of the Selection Logic rules are described:

a) The first accommodates implementations that may wish to operate in a manner that minimizes needless disturbance of existing aggregates, at the expense of the deterministic characteristics of the logic described above.;
b) The second accommodates implementations that may wish to limit the number of Aggregators that are available for use to fewer than the number of physical Ports

92.4.8.3.1 Reduced reconfiguration

Removing the constraint that the Aggregator that is chosen is always the lowest numbered Aggregator associated with the set of Ports in an aggregation, i.e., allowing a set of Ports to choose any suitable and available Aggregator to attach to (even when they are Individual), would allow an implementation to minimize the degree to which changes in the membership of a given aggregation result in changes of connectivity at higher layers. However, as there would still be the same number of Aggregators and physical Ports with a given value of Key, any physical Port will always be able to find an appropriate Aggregator to attach to. Clearly, the configuration reached over time with this relaxation of the rules would not necessarily match the configuration reached after all systems involved were reset.

NOTE—It is implicit in the above that a system reset would cause reconfiguration to the state that would be achieved by the rules stated in 92.4.8.2.

92.4.8.3.2 Limited Aggregator availability

Removing the constraint that there are always as many Aggregators as physical Ports would allow an implementation to limit the number of Ports available to higher layers while maintaining the ability to provide each Aggregator with multiple physical Ports. This has the same effect as removing the restriction that Aggregators and their associated physical Ports have the same Key value; some of the Aggregators can be effectively disabled (and therefore ignored) by configuring their Keys to be different to any Key value allocated to any of the physical Ports.

Any physical Port(s) that cannot find a suitable Aggregator to attach to will simply wait in the DETACHED state until an Aggregator becomes available, and with a Mux state of OUT_OF_SYNC.

92.4.9 Mux Control and Logic

The Mux Control and Logic turn that portion of the Aggregator’s the distributor and collector that is associated with the physical Port (as opposed to the portion that is associated with any other physical Ports that may be part of the aggregate) on or off as required by the state of the Selection Machine and the current protocol information.

Switching on and off the collector and distributor functions can be modeled using administrative and operational states, allowing modeling of implementations where the request to change the operational state cannot be actioned immediately. Leaving these states aside, the Mux has two states: IN_SYNC and OUT_OF_SYNC.

The operation of the Mux is best specified in terms of the goals for the collector and distributor operational states given the above states and the current state of received information from the Partner, in particular, whether the Partner is in sync and whether the Partner’s distributor and/or collector are turned on, as follows:

NOTE 1—If there is no Partner, the Mux behaves as if a Partner is in sync and has both collector and distributor enabled.

a) The Mux is IN_SYNC if the Selection Machine has Attached the physical Port to the correct Aggregator, and the Match logic indicates *matched*.
b) If the Mux is OUT_OF_SYNC or the Partner’s state is signaling out of sync, then both the collector and distributor should be turned off.

NOTE 2—The Actor signals out of sync in the protocol in this case.

c) If the Mux is IN_SYNC and the Partner’s state is in sync, then the collector should be turned on.
d) If the Mux is IN_SYNC, the Partner’s state is in sync, and the Partner’s collector is turned on, then the distributor (and collector) should be turned on.
e) If the Mux hardware is *coupled*, i.e., it can only turn collector and distributor on or off simultaneously, then the above rules also apply.
f) If the Mux hardware is *independent*, i.e., the collector and distributor states can be controlled independently, then if the Partner’s collector is turned off, then the distributor should be turned off.

92.4.10 Transmit Machine

The Transmit Machine maintains the following information for transmission in outgoing LACPDUs (see 92.4.3.1):

a) Actor_Port;
b) Actor_System;
c) Actor_Key;
d) Actor_State.

In addition, the following parameters, maintained by the Receive Machine, are transmitted in outgoing LACPDUs:

e) Partner_Port;
f) Partner_System;
g) Partner_Key;
h) Partner_State.

When an NTT (92.4.6.13) signal is sent by any of the other state machines, the Transmit machine ensures that properly formatted LACPDU (92.4.12) is transmitted, subject to the restriction that no more than 3 LACPDUs may be transmitted in any interval equal to Fast_Transmission_Ticks. If an NTT signal is received when this limit is in force, the transmission shall be delayed until such a time as the restriction is no longer in force.

NOTE—Multiple NTT signals occurring during a restriction period result in a single LACPDU transmission once the restriction no longer applies, with the information set to the Actor's state at that time. In other words, the LACPDU transmission model is based upon the transmission of state information that is current at the time an opportunity to transmit occurs, as opposed to queuing messages for transmission.

92.4.11 Churn Detection Machine

<<Author's Note - This state machine is entirely new, is not central to the operation of the protocol as it is essentially a diagnostic/fault report mechanism, and can therefore be removed with impunity (or made optional?) as desired.

Its function is to detect three classes of problem that might occur on a link, and signal them to Management:

1) If there are not enough Aggregators to service the needs of the physical configuration, some Ports will end up being homeless & therefore unable to be "in synch". The likelihood is that this represents a configuration problem;

2) There might be a Partner on the link that has failed in a manner that results in apparently inconclusive LACP exchanges taking place, preventing convergence upon an agreed state of play for the link;

2) With the existence of non-standard devices such as buffered repeaters, it is possible for a link that looks like a point-to-point full duplex link to have multiple Partners attached; the consequence of this is rather similar to 1) from the point of view of the Actor.

The Churn machine does not attempt to distinguish between these cases, or to fix them; it simply flags up the existence of a non-converging link, leaving management action to fix the problem.>>

The Churn Detection machine detects a situation where the Actor and Partner are unable to reach agreement upon the desired state of a link. Under normal operation of the protocol, such a resolution would be reached very rapidly; continued failure to reach agreement can be symptomatic of component failure, of the presence of non-standard devices on the link concerned, or of mis-configuration. Hence, detection of such failures is signalled by the Churn Detection machine to management in order to prompt administrative action to further diagnose and correct the fault.

NOTE—One of the classes of problem that will be detected by this machinery is the one where the implementation has been designed to support a limited number of Aggregators (fewer than the number of physical Ports - see 92.4.8.3.2) and the physical topology is such that one or more Ports end up with no Aggregator to connect to. This will almost certainly be the result either of a wiring error or an error in the allocation of Key values to the physical Ports and Aggregators.

The symptoms that this state machine detects are that the Actor's Mux logic has determined that it is out of sync (i.e., the Match logic indicates not matched, and/or the Selection machine has determined that the Port is attached to the wrong Aggregator), and that condition has not resolved itself within a short period of time, equal to Fast_Expiry_Ticks (92.4.3.5). Under normal conditions, this is ample time for convergence to take place.

The Churn Detection state machine is shown in Figure 92-7. The machine has three states; CHURN, NO_CHURN and CURN_MONITOR. The initial state is CHURN_MONITOR, with the churn_timer running; this is also the state that is entered on receipt of an out_of_sync signal from the Mux logic. Receipt of an in_sync signal from the Mux logic causes a transition to the NO_CHURN state. Expiry of the timer causes entry to the CHURN state, and causes a signal to be sent to management, indicating that the Port configuration has failed to resolve.

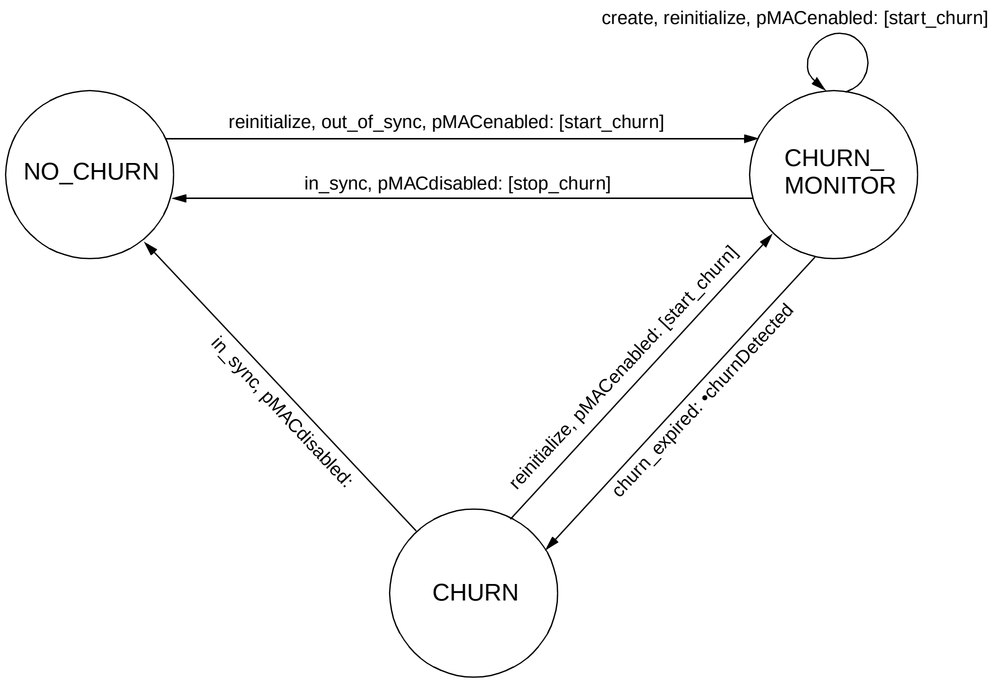
create, reinitialize, pMACenabled: [start_churn]
reinitialize, out_of_sync, pMACenabled: [start_churn]
NO_CHURN
in_sync, pMACdisabled: [stop_churn]
CHURN_ MONITOR
in_sync, pMACdisabled:
reinitialize, pMACenabled: [start_churn]
churn_expired: •churnDetected
CHURN

Figure 92-7—Churn Detection Machine state diagram

The state table for Receive Processing is shown in Table 92-4.

92.4.11.1 create

This event is caused by the initial creation of the state machine.

92.4.11.2 reinitialize

This event is caused by protocol entity being initialized or reinitialized.

92.4.11.3 out_of_sync

This event is caused by the Mux logic determining that it is in the OUT_OF_SYNC state.

92.4.11.4 in_sync

This event is caused by the Mux logic determining that it is in the IN_SYNC state.

Table 92-4—Churn Detection Machine state table

EVENT	STATE		
	CHURN_MONITOR	CHURN	NO_CHURN
create	[start_churn] -	X	X
reinitialize	[start_churn] -	[start_churn] CHURN_MONITOR	[start_churn] CHURN_MONITOR
out_of_sync	-	-	[start_churn] CHURN_MONITOR
in_sync	[stop_churn] NO_CHURN	NO_CHURN	-
churn_expired	•churnDetected CHURN	X	X
pMACdisabled	[stop_churn] NO_CHURN	NO_CHURN	-
pMACenabled	[start_churn] -	[start_churn] CHURN_MONITOR	[start_churn] CHURN_MONITOR

92.4.11.5 churn_expired

This event is caused by the expiry of the churn_timer.

92.4.11.6 pMACdisabled

This event is caused by the MAC associated with the physical port becoming inoperable.

92.4.11.7 pMACenabled

This event is caused by the MAC associated with the physical port becoming operable.

92.4.11.8 churn_detected

This signal is generated when the Churn Detection machine detects that the Port configuration has failed to converge. The signal indicates to management that remedial action is necessary.

92.4.11.9 start_churn

This action starts or restarts the churn_timer, using a value of Fast_Expiry_Ticks (92.4.3.5).

92.4.11.10 stop_churn

This action stops the churn_timer.

92.4.12 LACPDU structure and encoding

<<Author’s Note: The description of the proposed PDU structure shown here will need further work/fleshing out, following agreement within the Task Force on the overall approach and structure. In particular, clarification is required in order to fully specify the encoding rules for these PDUs, and do define the rules for PDU reception.

The intent of the proposed structure is to allow the same MAC address and Ethertype to be re-used for additional “slow” protocols - this is facilitated by the use of the Protocol Subtype and Protocol Version. The first instance of this is the Flush Protocol, which makes use of the second subtype value.

The remaining structure has been designed such that it can be easily extended, without changing version 1 operation, and it can be treated as a fixed length structure by those implementations that wish to do so.>>

The LACPDU structure is shown in 92-8. The various elements of the PDU are as described below (see 92.4.3.1 for definitions of the Actor and Partner information elements):

“Slow Protocols” Ethertype (2 octets)
LACP Subtype (1 octet)
Version number (1 octet)
Type = Actor Information (1 octet)
Length = 16 (1 octet)
Actor_Port (2 octets)
Actor_System (6 octets)
Actor_Key (2 octets)
Actor_State (1 octet)
Reserved (3 octets)
Type = Partner Information (1 octet)
Length = 16 (1 octet)
Partner_Port (2 octets)
Partner_System (6 octets)
Partner_Key (2 octets)
Partner_State (1 octet)
Reserved (3 octets)
Type = Terminator (1 octet - 0)
Length = 0 (1 octet)

Figure 92-8—LACPDU structure

a) *“Slow Protocols” Ethertype.* This Ethertype identifies the PDU as carrying protocol information related to one of the class of 802.3-defined “slow protocols”, of which LACP and the Flush protocol (92.5) are examples. This ethertype is encoded as the value XX-XX {To be allocated at some future date}.
b) *LACP subtype.* This identifies the PDU as a Link Aggregation Control PDU. The LACP subtype is encoded as the integer value 1.
c) *Version number.* This identifies the LACP version; implementations conformant to this version of the standard encode the version number as the integer value 1.
d) *Type = Actor Information.* This indicates that this portion of the PDU contains the Actor’s protocol information. This is encoded as the integer value 1.

e) *Length = 16*. This indicates the number of octets of this portion of the PDU, encoded as an integer.
f) *Actor_Port*. The Actor's Port number, encoded as an integer.
g) *Actor_System*. The Actor's System ID, encoded as a MAC address.
h) *Actor_Key*. The Actor's Key, encoded as an integer.
i) *Actor_State*. The Actor's state variables, encoded as individual bits within a single octet, as follows. Bit 1 is the least significant bit, bit 8 is the most significant bit:
    1) *LACP_Activity* is encoded in bit 1. Active LACP is encoded as a 1; Passive LACP as a 0.
    2) *LACP_Timeout* is encoded in bit 2. Short Timeout is encoded as a 1; Long Timeout as a 0.
    3) *Aggregability* is encoded in bit 3. Aggregatable is encoded as a 1; Individual is encoded as a 0.
    4) *Synchronization* is encoded in bit 4. In_Sync is encoded as a 1; Out_Of_Sync is encoded as a 0.
    5) *Collecting* is encoded in bit 4. True is encoded as a 1; False is encoded as a 0.
    6) *Distributing* is encoded in bit 4. True is encoded as a 1; False is encoded as a 0.
    7) The remaining bits of this octet, 7 and 8, are reserved; these are ignored on receipt and transmitted as zero. However, the received value of these bits shall be recorded on receipt in order to accurately reflect the Actor's view of the Partner's state in outgoing PDUs (see below).
j) *Reserved*. These octets are ignored on receipt and transmitted as zero.
k) *Type = Partner Information*. This indicates that this portion of the PDU contains the Actor's protocol information. This is encoded as the integer value 2.
l) *Length = 16*. This indicates the number of octets of this portion of the PDU, encoded as an integer.
m) *Partner_Port*. The Actor's Port number, encoded as an integer.
n) *Partner_System*. The Partner's System ID, encoded as a MAC address.
o) *Partner_Key*. The Partner's Key, encoded as an integer.
p) *Partner_State*. The Actor's view of the Partner's state variables, encoded as individual bits within a single octet, as follows. Bit 1 is the least significant bit, bit 8 is the most significant bit:
    1) *LACP_Activity* is encoded in bit 1. Active LACP is encoded as a 1; Passive LACP as a 0.
    2) *LACP_Timeout* is encoded in bit 2. Short Timeout is encoded as a 1; Long Timeout as a 0.
    3) *Aggregability* is encoded in bit 3. Aggregatable is encoded as a 1; Individual is encoded as a 0.
    4) *Synchronization* is encoded in bit 4. In_Sync is encoded as a 1; Out_Of_Sync is encoded as a 0.
    5) *Collecting* is encoded in bit 4. True is encoded as a 1; False is encoded as a 0.
    6) *Distributing* is encoded in bit 4. True is encoded as a 1; False is encoded as a 0.
    7) The remaining bits of this octet, 7 and 8, are ignored on receipt; when transmitting, they contain any values received in the incoming LACPDU's Actor's state field.
q) *Reserved*. These octets are ignored on receipt and transmitted as zero.
r) *Terminator*. The PDU is terminated by a 2-octet field consisting of a Type and Length of zero.

All LACPDUs are addressed to the group MAC address allocated for use by the 802.3 "Slow Protocols"; this is defined as:

XX-XX-XX-XX-XX-XX {To be allocated at some future date}.

92.5 Flush protocol

92.5.1 Introduction

The Flush protocol allows the distribution function of the Actor's Link Aggregation Control sublayer to request the transmission of a Marker PDU on a given physical link. The marker PDU is received by the collection function of the Partner's Link Aggregation Control sublayer, and a Marker Received PDU is returned on the same physical link to the initiating Actor's distribution function. Marker and Marker Received PDUs are treated by the MAC function at each end of the link as normal Mac Client PDUs; i.e., there is no prioritization of flush PDUs relative to normal user data frames, and flush PDUs are subject to the operation of flow control, where supported on the link. Hence, if the distribution function requests transmission of a Marker PDU on a given link and does not transmit any further MAC Client PDUs that relate to a given set of conversations until the corresponding Marker Received PDU is received on that link, then it can be certain that there are no MAC Client data PDUs related to those conversations still to be received by the Partner's collection function. The use of the flush protocol can therefore allow the Distribution function a means of determining the point at which a given set of conversations can safely be reallocated from one link to another without the danger of causing frames in those conversations to be re-ordered at the collector.

NOTE—The above discussion does not mention multiple priorities/queues, as this is a function of a MAC Bridge, not the 802.3 MAC itself. At the level we are discussing here, there is only a single priority available; that provided by 802.3 to provide the MA_UNITDATA service. Requiring flush PDUs to be prioritized no differently to other MAC Client frames ensures that flush PDUs stay in the right place in the data stream.

The use of the Flush protocol is optional; some distribution algorithms may not require the use of a flush, and other mechanisms, such as the use of timeouts, may be used as an alternative. As the specification of distribution algorithms is outside the scope of this standard, no attempt is made to specify how, when, or if, this protocol is used. However, a conformant implementation of Link Aggregation Control shall respond to all received Marker PDUs as specified in the description of the protocol operation (see 92.5.3), thus ensuring that implementations that need to make use of the protocol can do so.

The Flush protocol does not provide a guarantee of a response from the Partner system; no provision is made for the consequences of frame loss or for the failure of the Partner system to respond correctly. Implementations that make use of this protocol must therefore make their own provision for handling such errors.

92.5.2 Flush service primitives

The following subclauses define the service primitives associated with the operation of the Flush service.

92.5.2.1 MARKER.request

This service primitive is used only by the Distribution function of the Link Aggregation sublayer. The parameters of this service primitive are as follows:

```
MARKER.request        (
                      system_id
                      port_id
                      transaction_id
                      )
```

The **system_id** parameter carries the MAC address used as the System Identifier by the Link Aggregation Sublayer of the system issuing the MARKER.request.

The **port_id** parameter carries the local identifier used by the Link Aggregation Sublayer of the system issuing the request to identify the physical Port that will be used to convey the MARKER.request.

The **transaction_id** is a locally significant identifier allocated to this MARKER.request by the service user that issued the request.

92.5.2.2 MARKER.indication

This service primitive is used only by the Collection function of the Link Aggregation sublayer. The parameters of this service primitive are as follows:

```
MARKER.indication     (
                      system_id
                      port_id
                      transaction_id
```

```
                                        )
```

The **system_id** parameter carries the MAC address used as the System Identifier by the Link Aggregation Sublayer that issued the request.

The **port_id** parameter carries the local identifier used by the Link Aggregation Sublayer that issued the request to identify the physical Port that was used to convey the MARKER.request.

The **transaction_id** is a locally significant identifier allocated to this MARKER.request by the service user that issued the request.

92.5.2.3 MARKER_RECEIVED.request

This service primitive is used only by the Collection function of the Link Aggregation sublayer. The parameters of this service primitive are as follows:

```
MARKER_RECEIVED.request(
                  system_id
                  port_id
                  transaction_id
                  responder_system_id
                  responder_port_id
                  )
```

The **system_id, port_id** and **transaction_id** parameters are as defined for the corresponding parameters in the MARKER.indication primitive.

The **responder_system_id** parameter carries the MAC address used as the System Identifier by the Link Aggregation Sublayer of the system issuing the request.

The **responder_port_id** parameter carries the local identifier used by the Link Aggregation Sublayer of the system issuing the request to identify the physical Port that will be used to convey the request.

92.5.2.4 MARKER_RECEIVED.indication

This service primitive is used only by the Distribution function of the Link Aggregation sublayer. The parameters of this service primitive are as follows:

```
MARKER_RECEIVED.indication(
                  system_id
                  port_id
                  transaction_id
                  responder_system_id
                  responder_port_id
                  )
```

The **system_id, port_id** and **transaction_id** parameters are as defined for the corresponding parameters of MARKER.indication primitive.

The **responder_system_id** parameter carries the MAC address used as the System Identifier by the Link Aggregation Sublayer of the system issuing the MARKER_RECEIVED.request.

The **responder_port_id** parameter carries the local identifier used by the Link Aggregation Sublayer of the system issuing the request to identify the physical Port that will be used to convey the MARKER_RECEIVED.request.

92.5.2.5 Sequence of service primitives

Figure 92-9 illustrates the time sequence of the Flush service primitives, between an initiating and responding system. Time is assumed to flow from the top of the diagram to the bottom.

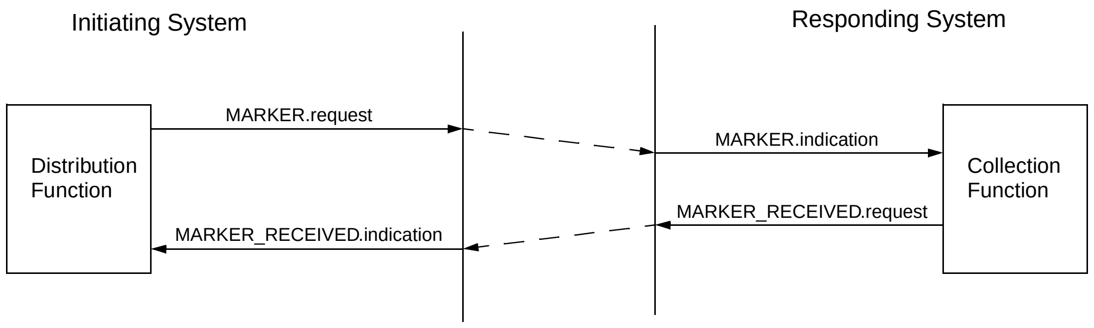

Figure 92-9—Flush service time sequence diagram

92.5.3 Protocol definition

92.5.3.1 Provision and support of the flush service

The MARKER.request service primitive may be used by the Distribution function of the Link Aggregation Sublayer to request the transmission of a Marker PDU. On receipt of a MARKER.request service primitive from the Distribution function, an MA_UNITDATA.request is issued by the sublayer. The destination_address parameter of the request carries the Group MAC address defined for use by the Link Aggregation Control Protocol (see 92.4.12). The m_sdu parameter carries a Marker PDU, formatted as defined in 92.5.3.3, using the parameters received in the MARKER.request primitive.

Received MA_UNITDATA.indication primitives that are destined for the Group MAC address defined for use by the Link Aggregation Control Protocol (see 92.4.12) and in which the m_sdu parameter contains a Marker PDU, formatted as defined in 92.5.3.3, shall cause a MARKER.indication primitive to be issued to the Collection function of the Link Aggregation Control sublayer, using the parameter values contained in the Marker PDU.

On receipt of a MARKER.indication primitive, the Collection function shall issue a MARKER_RECEIVED.request primitive. The **system_id, port_id** and **transaction_id** parameters carry the values of the corresponding parameters received in the corresponding MARKER.indication primitive. The responder_system_id and responder_port_id parameters are set to the System ID and Port ID values used by the Link Aggregation Sublayer in the system issuing the request. On receipt of a MARKER_RECEIVED.request service primitive from the Collection function, an MA_UNITDATA.request is issued by the Link Aggregation Control sublayer. The destination_address parameter of the request carries the Group MAC address defined for use by the Link Aggregation Control Protocol (see 92.4.12). The m_sdu parameter carries a Marker Received PDU, formatted as defined in 92.5.3.3, using the parameters received in the MARKER_RESPONSE.request primitive.

Received MA_UNITDATA.indication primitives that are destined for the Group MAC address defined for use by the Link Aggregation Control Protocol (see 92.4.12) and in which the m_sdu parameter contains a Marker Received PDU, formatted as defined in 92.5.3.3, shall cause a MARKER_RESPONSE.indication primitive to be issued to the Distribution function of the Link Aggregation Control sublayer, using the parameter values contained in the Marker PDU.

92.5.3.2 Timing constraints

MARKER.request service primitives may be issued by the Distribution function at any time, subject to the constraint that there shall be less than five such requests issued on any given physical link during any one second period.

NOTE—This is deliberately rather different to constraining the rate to 1 per 1/4 second. The intent here is to limit the overall loading caused by the protocol, while still allowing for rapid response to changing distribution requirements.

The Collection function shall issue a MARKER_RECEIVED.indication within a maximum of one second of the receipt of a Marker PDU.

NOTE—This correlates with the maximum transit delay permitted in MAC Bridges, which is also one second. Any frames received more than a second ago must either have been forwarded by a Bridge, or have been discarded.

92.5.3.3 PDU structure and encoding

<<Author’s Note: The description of the proposed PDU structure shown here is based on the same structural approach as for the LACPDU.>>

The Marker PDU and Marker Received PDU structure is shown in 92-10. The various elements of the PDU are as described below:

Marker PDU structure
“Slow Protocols” Ethertype (2 octets)
Flush Subtype (1 octet)
Version number (1 octet)
Type = Marker Info (1 octet)
Length = 16 (1 octet)
Requester_Port (2 octets)
Requester_System (6 octets)
Requester_Transaction_ID (2 octets)
Reserved (4 octets)
Type = Terminator (1 octet - 0)
Length = 0 (1 octet)

Marker Received PDU structure
“Slow Protocols” Ethertype (2 octets)
Flush Subtype (1 octet)
Version number (1 octet)
Type = Marker Received Info (1 octet)
Length = 32 (1 octet)
Requester_Port (2 octets)
Requester_System (6 octets)
Requester_Transaction_ID (2 octets)
Responder_System (6 octets)
Responder_Port (2 octets)
Reserved (12 octets)
Type = Terminator (1 octet - 0)
Length = 0 (1 octet)

Figure 92-10—Marker PDU and Marker Received PDU structure

a) *“Slow Protocols” Ethertype.* This Ethertype identifies the PDU as carrying protocol information related to one of the class of 802.3-defined “slow protocols”, of which LACP and the Flush protocol

(92.5) are examples. This ethertype is encoded as the value XX-XX {To be allocated at some future date}.

b) *Flush subtype.* An integer value of 3 identifies the PDU as a Flush PDU.
c) *Version number.* This identifies the Flush protocol version; implementations conformant to this version of the standard encode the version number as the integer value 1.
d) *Type = Marker Info/Marker Received Info.* This indicates that this portion of the PDU contains the Marker or Marker Received information. This is encoded as the integer value 1 for Marker Info, 2 for Marker Received Info.
e) *Length = 16 (request) or 32 (response).* This indicates the number of octets of this portion of the PDU, encoded as an integer.
f) *Requester_Port.* The Requester’s Port number, encoded as an integer.
g) *Requester_System.* The Requester’s System ID, encoded as a MAC address.
h) *Requester_Transaction_ID.* The transaction ID allocated to this request by the requester, encoded as an integer.
i) *Responder_Port.* The Responder’s Port number, encoded as an integer. This field appears only in Marker Received Info.
j) *Responder_System.* The Responder’s System ID, encoded as a MAC address. This field appears only in Marker Received Info.
k) *Reserved.* These octets are reserved; they are ignored upon receipt and transmitted as zero.
l) *Terminator.* Request and response PDUs are terminated by a 2-octet field consisting of a Type and Length of zero.

92.6 Management

<<Author’s Note: This section eventually needs to specify:

- The management functionality that is available for control/monitoring of Link Aggregation;

- The MIB definition that realizes that functionality, using appropriate notation (SNMP MIB, GDMO defs, etc.) to fit in with the format of other 802.3 MIB definitions.

Management is mostly achieved by managing the Key values associated with individual Links. May also be desirable to allow the configuration to be “forced”, i.e., allow total manual control of allocation of Links to LAGs.

In addition to straight configuration Key, management “hooks” to allow requests for resynchronization, in order to enable the use of likely “hints” from other protocols that the configuration may be changing. This resync Key should be available both for individual links and for Link Aggregation Groups.

Some initial proposals follow with regard to the specific controls needed in order to manage the protocol described in this document...>>

92.6.1 Proposed Management Controls for LACP

The Link Aggregation Control Protocol essentially has the following elements that are obvious candidates for management control:

a) The Key value associated with each Aggregator;
b) The Key value associated with each physical Port;
c) The Activity control for each Port - Active LACP or Passive LACP;
d) The Timeout control for each Port - Short Timeout or Long Timeout.

These parameters should be represented as read/write attributes in the Link Aggregation MIB.

In addition to the above controls, which allow the basic operation of the protocol to be managed, it should be possible to specify a default, “hard-wired” configuration that would allow aggregations to be formed even when there is no LACP-aware device on the remote end of the link (but there is a device that can aggregate links on the basis of existing proprietary mechanisms). This has the potential to allow some level of integration between current proprietary schemes and the standardized approach.

The hard-wiring would be achieved by allowing management to specify a Partner’s Key value for each Port; this Key value, along with a Partner’s System ID of all zeroes, would be used only in the absence of a protocol-aware Partner, and would be overridden if any protocol activity is detected from the remote system.

This approach maintains the “plug and play” characteristics of the protocol as described, while at the same time allowing some level of hard configuration as a backup.

92.7 Protocol Implementation Conformance Statement

The supplier of an implementation that is claimed to conform to clause 92. of this standard shall complete a copy of the PICS proforma provided below and shall provide the information necessary to identify both the supplier and the implementation.

<<Author’s Note: PICS Proforma to be added.>>