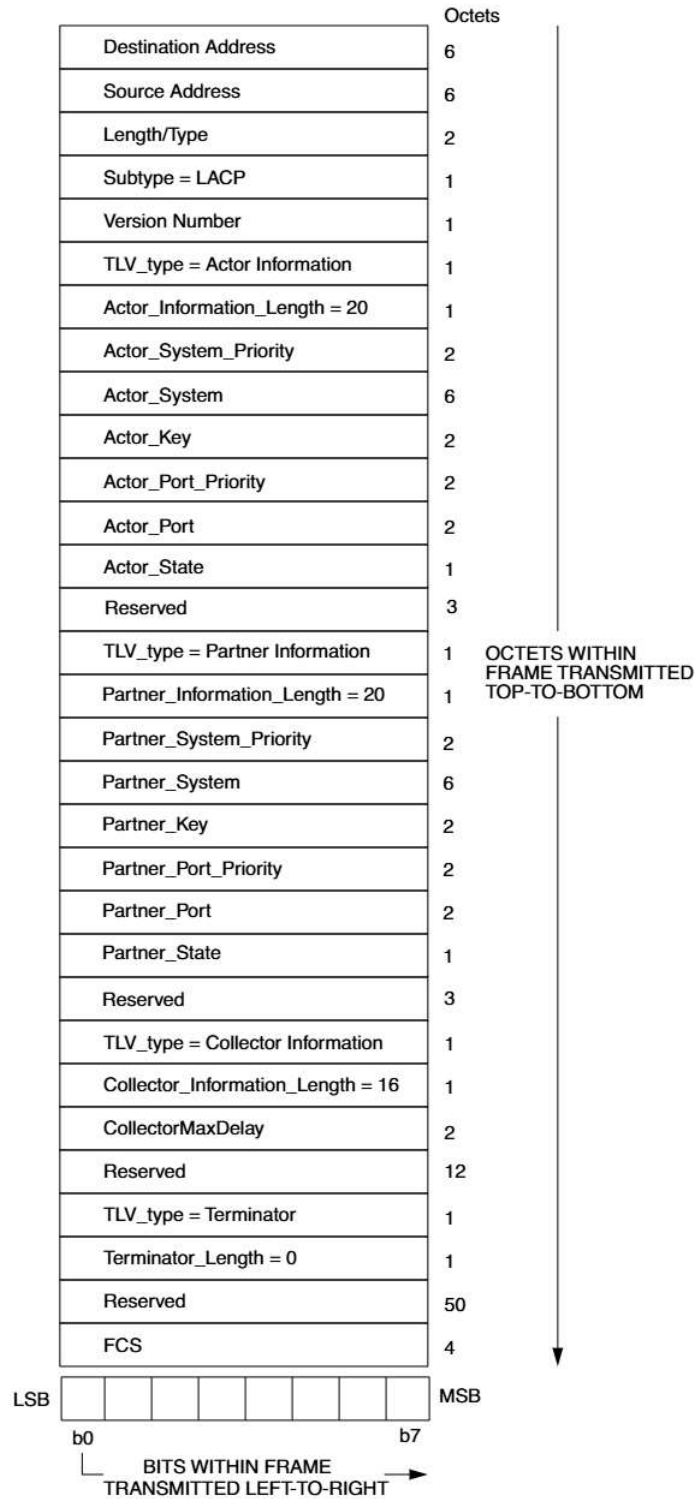


- d) When consecutive octets are used to represent a MAC address, the least significant bit of the first octet is assigned the value of the first bit of the MAC address, the next most significant bit the value of the second bit of the MAC address, and so on through the eighth bit. Similarly the least significant through most significant bits of the second octet are assigned the value of the ninth through seven-teenth bits of the MAC address, and so on for all the octets of the MAC address.

#### 43.4.2.2 LACPDU structure

LACPDU are basic IEEE 802.3® frames; they shall not be tagged (See Clause 3). The LACPDU structure shall be as shown in Figure 43–7 and as further described in the following field definitions:

- a) *Destination Address (DA)*. The DA in LACPDU is the Slow\_Protocols\_Multicast address. Its use and encoding are specified in Annex 43B.
- b) *Source Address (SA)*. The SA in LACPDU carries the individual MAC address associated with the port through which the LACPDU is transmitted.
- c) *Length/Type*. LACPDU are always Type encoded, and carry the Slow\_Protocols\_Type field value. The use and encoding of this type is specified in Annex 43B.
- d) *Subtype*. The Subtype field identifies the specific Slow Protocol being encapsulated. LACPDU carry the Subtype value 0x01.
- e) *Version Number*. This identifies the LACP version; implementations conformant to this version of the standard carry the value 0x01.
- f) *TLV\_type = Actor Information*. This field indicates the nature of the information carried in this TLV-tuple. Actor information is identified by the value 0x01.
- g) *Actor\_Information\_Length*. This field indicates the length (in octets) of this TLV-tuple, Actor information uses a length value of 20 (0x14).
- h) *Actor\_System\_Priority*. The priority assigned to this System (by management or administration policy), encoded as an unsigned integer.
- i) *Actor\_System*. The Actor's System ID, encoded as a MAC address.
- j) *Actor\_Key*. The operational Key value assigned to the port by the Actor, encoded as an unsigned integer.
- k) *Actor\_Port\_Priority*. The priority assigned to this port by the Actor (the System sending the PDU; assigned by management or administration policy), encoded as an unsigned integer.
- l) *Actor\_Port*. The port number assigned to the port by the Actor (the System sending the PDU), encoded as an unsigned integer.
- m) *Actor\_State*. The Actor's state variables for the port, encoded as individual bits within a single octet, as follows and as illustrated in Figure 43–8:
  - 1) *LACP\_Activity* is encoded in bit 0. This flag indicates the Activity control value with regard to this link. Active LACP is encoded as a 1; Passive LACP is encoded as a 0.
  - 2) *LACP\_Timeout* is encoded in bit 1. This flag indicates the Timeout control value with regard to this link. Short Timeout is encoded as a 1; Long Timeout is encoded as a 0.
  - 3) *Aggregation* is encoded in bit 2. If TRUE (encoded as a 1), this flag indicates that the System considers this link to be *Aggregatable*; i.e., a potential candidate for aggregation. If FALSE (encoded as a 0), the link is considered to be *Individual*; i.e., this link can be operated only as an individual link.
  - 4) *Synchronization* is encoded in bit 3. If TRUE (encoded as a 1), the System considers this link to be *IN\_SYNC*; i.e., it has been allocated to the correct Link Aggregation Group, the group has been associated with a compatible Aggregator, and the identity of the Link Aggregation Group is consistent with the System ID and operational Key information transmitted. If FALSE (encoded as a 0), then this link is currently *OUT\_OF\_SYNC*; i.e., it is not in the right Aggregation.

**Figure 43–7—LACPDU structure**

- 5) *Collecting* is encoded in bit 4. TRUE (encoded as a 1) means collection of incoming frames on this link is definitely enabled; i.e., collection is currently enabled and is not expected to be disabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise FALSE (encoded as a 0);
- 6) *Distributing* is encoded in bit 5. FALSE (encoded as a 0) means distribution of outgoing frames on this link is definitely disabled; i.e., distribution is currently disabled and is not expected to be enabled in the absence of administrative changes or changes in received protocol information. Its value is otherwise TRUE (encoded as a 1);
- 7) *Defaulted* is encoded in bit 6. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is using Defaulted operational Partner information, administratively configured for the Partner. If FALSE (encoded as a 0), the operational Partner information in use has been received in a LACPDU;
- 8) *Expired* is encoded in bit 7. If TRUE (encoded as a 1), this flag indicates that the Actor's Receive machine is in the EXPIRED state; if FALSE (encoded as a 0), this flag indicates that the Actor's Receive machine is not in the EXPIRED state.

NOTE—The received values of Defaulted and Expired state are not used by LACP; however, knowing their values can be useful when diagnosing protocol problems.

| BIT | 0             | 1            | 2           | 3               | 4          | 5            | 6         | 7       |
|-----|---------------|--------------|-------------|-----------------|------------|--------------|-----------|---------|
|     | LACP_Activity | LACP_Timeout | Aggregation | Synchronization | Collecting | Distributing | Defaulted | Expired |

NOTE—Bit ordering within this field is as specified in 43.4.2.1.

**Figure 43–8—Bit encoding of the Actor\_State and Partner\_State fields**

- n) *Reserved*. These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.
- o) *TLV\_type = Partner Information*. This field indicates the nature of the information carried in this TLV-tuple. Partner information is identified by the integer value 0x02.
- p) *Partner\_Information\_Length*. This field indicates the length (in octets) of this TLV-tuple. Partner information uses a length value of 20 (0x14).
- q) *Partner\_System\_Priority*. The priority assigned to the Partner System (by management or administration policy), encoded as an unsigned integer.
- r) *Partner\_System*. The Partner's System ID, encoded as a MAC address.
- s) *Partner\_Key*. The operational Key value assigned to the port associated with this link by the Partner, encoded as an unsigned integer.
- t) *Partner\_Port\_Priority*. The priority assigned to this port by the Partner (by management or administration policy), encoded as an unsigned integer.
- u) *Partner\_Port*. The port number associated with this link assigned to the port by the Partner, encoded as an unsigned integer.
- v) *Partner\_State*. The Actor's view of the Partner's state variables, depicted in Figure 43–8 and encoded as individual bits within a single octet, as defined for Actor\_State.
- w) *Reserved*. These 3 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.
- x) *TLV\_type = Collector Information*. This field indicates the nature of the information carried in this TLV-tuple. Collector information is identified by the integer value 0x03.

- y) *Collector\_Information\_Length*. This field indicates the length (in octets) of this TLV-tuple. Collector information uses a length value of 16 (0x10).
- z) *CollectorMaxDelay*. This field contains the value of CollectorMaxDelay (43.2.3.1.1) of the station transmitting the LACPDU, encoded as an unsigned integer number of tens of microseconds. The range of values for this parameter is 0 to 65 535 tens of microseconds (0.65535 seconds).
- aa) *Reserved*. These 12 octets are reserved for use in future extensions to the protocol. They shall be ignored on receipt and shall be transmitted as zeroes to claim compliance with Version 1 of this protocol.
- ab) *TLV\_type = Terminator*. This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information is identified by the integer value 0x00.
- ac) *Terminator\_Length*. This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).  
 NOTE—The use of a Terminator\_Length of 0 is intentional. In TLV encoding schemes it is common practice for the terminator encoding to be 0 both for the type and the length.
- ad) *Reserved*. These 50 octets are reserved for use in future extensions to the protocol. They are ignored on receipt and are transmitted as zeroes to claim compliance with Version 1 of this protocol.  
 NOTE—The Reserved octets are included in all valid LACPDUs in order to force the TLV lengths to multiples of 4 octets, and to force a fixed PDU size of 128 octets, regardless of the version of the protocol. Hence, a Version 1 implementation is guaranteed to be able to receive version N PDUs successfully, although version N PDUs may contain additional information that cannot be interpreted (and will be ignored) by the Version 1 implementation. A crucial factor in ensuring backwards compatibility is that any future version of the protocol is required not to re-define the structure or semantics of information defined for the previous version; it may only add new information elements to the previous set. Hence, in a version N PDU, a Version 1 implementation can expect to find the Version 1 information in exactly the same places as in a Version 1 PDU, and can expect to interpret that information as defined for Version 1. Future versions of this protocol expect to have the Reserved octets available for their use.
- ae) *FCS*. This field is the Frame Check Sequence, typically generated by the underlying MAC.

#### 43.4.3 LACP 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. Events (such as expiration of a timer or received LACPDUs) may cause state transitions and also cause actions to be taken; those actions may include the need for transmission of a LACPDU containing repeated or new information. Periodic and event-driven transmissions are controlled by the state of a Need-To-Transmit (NTT) variable (see 43.4.7), generated by the state machines as necessary.

The state machines are as follows:

- a) *Receive machine (RX—43.4.12)*. This state machine receives LACPDUs from the Partner, records the information contained, and times it out using either Short Timeouts or Long Timeouts, according to the setting of LACP\_Timeout. It evaluates the incoming information from the Partner to determine whether the Actor and Partner have both agreed upon the protocol information exchanged to the extent that the port can now be safely used, either in an aggregation with other ports or as an individual port; if not, it asserts NTT in order to transmit fresh protocol information to the Partner. If the protocol information from the Partner times out, the Receive machine installs default parameter values for use by the other state machines.
- b) *Periodic Transmission machine (43.4.13)*. This state machine determines whether the Actor and its Partner will exchange LACPDUs periodically in order to maintain an aggregation (periodic LACPDU exchanges occur if either or both are configured for Active LACP).
- c) *Selection Logic (43.4.14)*. The Selection Logic is responsible for selecting the Aggregator to be associated with this port.

- d) *Mux machine (MUX—43.4.15)*. This state machine is responsible for attaching the port to a selected Aggregator, detaching the port from a de-selected Aggregator, and for turning collecting and distributing at the port on or off as required by the current protocol information.
- e) *Transmit machine (TX—43.4.16)*. This state machine handles the transmission of LACPDU, both on demand from the other state machines, and on a periodic basis.

Figure 43–9 illustrates the relationships among these state machines and the flow of information between them. The set of arrows labelled Partner State Information represents new Partner information, contained in an incoming LACPDU or supplied by administrative default values, being fed to each state machine by the Receive machine. The set of arrows labelled Actor State Information represents the flow of updated Actor state information between the state machines. Transmission of LACPDU occurs either as a result of the Periodic machine determining the need to transmit a periodic LACPDU, or as a result of changes to the Actor's state information that need to be communicated to the Partner. The need to transmit a LACPDU is signalled to the Transmit machine by asserting NTT. The remaining arrows represent shared variables in the state machine description that allow a state machine to cause events to occur in another state machine.

NOTE—The arrows marked Ready\_N show that information derived from the operation of another port or ports can affect the operation of a port's state machine. See the definition of the Ready and Ready\_N variables in 43.4.8.

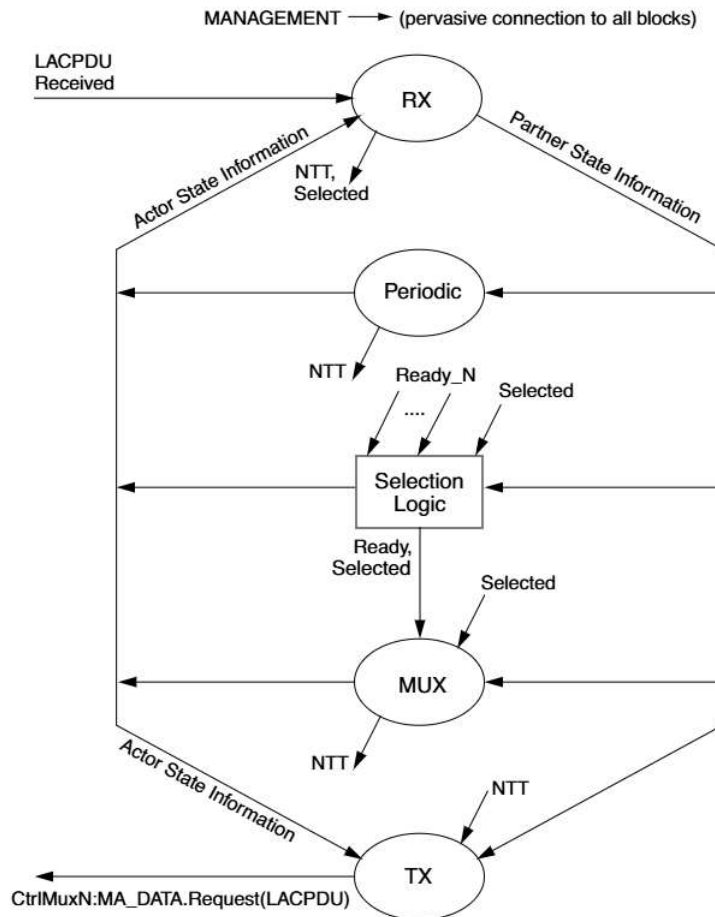


Figure 43–9—Interrelationships among state machines

Two further state machines are defined for diagnostic purposes. They are as follows:

- f) *Actor and Partner Churn Detection machines (43.4.17)*. These state machines make use of the IN\_SYNC and OUT\_OF\_SYNC states generated by the Actor and Partner Mux machines in order to detect the situation where the state machines are unable to resolve the state of a given link; e.g., because the Partner System repeatedly sends conflicting information in its LACPDU. As this situation can occur in a normally functioning link, particularly where either or both participating Systems have constrained aggregation capability (see 43.6), these state machines simply detect the presence of such a condition and signal its existence to management.

#### 43.4.4 Constants

All timers specified in this subclause have an implementation tolerance of  $\pm 250$  ms.

**Fast\_Periodic\_Time**

The number of seconds between periodic transmissions using Short Timeouts.

Value: Integer

1

**Slow\_Periodic\_Time**

The number of seconds between periodic transmissions using Long Timeouts.

Value: Integer

30

**Short\_Timeout\_Time**

The number of seconds before invalidating received LACPDU information when using Short Timeouts (3 x Fast\_Periodic\_Time).

Value: Integer

3

**Long\_Timeout\_Time**

The number of seconds before invalidating received LACPDU information when using Long Timeouts (3 x Slow\_Periodic\_Time).

Value: Integer

90

**Churn\_Detection\_Time**

The number of seconds that the Actor and Partner Churn state machines wait for the Actor or Partner Sync state to stabilize.

Value: Integer

60

**Aggregate\_Wait\_Time**

The number of seconds to delay aggregation, to allow multiple links to aggregate simultaneously.

Value: Integer

2

#### 43.4.5 Variables associated with the System

**Actor\_System**

The MAC address component of the System Identifier of the System.

Value: 48 bits

Assigned by administrator or System policy.

Actor\_System\_Priority  
The System Priority of the System.  
Value: Integer  
Assigned by administrator or System policy.

#### 43.4.6 Variables associated with each Aggregator

Aggregator\_MAC\_address  
The MAC address assigned to the Aggregator.  
Value: 48 bits  
Assigned by administrator or System policy.

Aggregator\_Identifier  
Used to uniquely identify an Aggregator within a System.  
Value: Integer  
Assigned by administrator or System policy.

Individual\_Aggregator  
The aggregation capability of the Aggregator.  
Value: Boolean  
TRUE if the port attached to this Aggregator is not capable of aggregation with any other port.  
FALSE if the port(s) attached to this Aggregator are capable of aggregation with other ports.

Actor\_Admin\_Aggregator\_Key  
The administrative Key value associated with the Aggregator.  
Value: Integer  
Assigned by administrator or System policy.

Actor\_Oper\_Aggregator\_Key  
The operational Key value associated with the Aggregator.  
Value: Integer  
Assigned by the Actor.

Partner\_System  
The MAC address component of the System Identifier of the remote System to which the Aggregator is connected. If the Aggregator has no attached ports, this variable is set to 0x00-00-00-00-00-00.  
Value: 48 bits

Partner\_System\_Priority  
The System Priority of the remote System to which the Aggregator is connected. If the Aggregator has no attached ports, this variable is set to zero.  
Value: Integer

Partner\_Oper\_Aggregator\_Key  
The operational Key assigned to an aggregation by the remote System to which this Aggregator is connected. If the Aggregator has no attached ports, this variable is set to zero.  
Value: Integer

**Receive\_State**

The Receive\_State of the Aggregator will be Enabled if one or more ports attached to the Aggregator are Collecting (i.e., Actor\_Oper\_Port\_State.Collecting is TRUE for any port). Otherwise, Receive\_State is Disabled.

Values: Enabled or Disabled

**Transmit\_State**

The Transmit\_State of the Aggregator will be Enabled if one or more ports attached to the Aggregator are Distributing (i.e., Actor\_Oper\_Port\_State.Distributing is TRUE for any port). Otherwise, Transmit\_State is Disabled.

Values: Enabled or Disabled

**LAG\_Ports**

The set of ports that belong to the Link Aggregation Group.

Value: Integer Array

**43.4.7 Variables associated with each port****Actor\_Port\_Number**

The port number assigned to the port.

Value: Integer

Assigned by administrator or System policy.

**Actor\_Port\_Priority**

The priority value assigned to the port, used to converge dynamic Key changes.

Value: Integer

Assigned by administrator or System policy.

**Actor\_Port\_Aggregator\_Identifier**

The identifier of the Aggregator that this port is attached to.

Value: Integer

**NTT**

Need To Transmit flag.

Value: Boolean

TRUE indicates that there is new protocol information that should be transmitted on the link, or that the Partner needs to be reminded of the old information.

FALSE otherwise.

**Actor\_Admin\_Port\_Key**

The administrative value of Key assigned to this port by administrator or System policy.

Value: Integer

**Actor\_Oper\_Port\_Key**

The operational value of Key assigned to this port by the Actor.

Value: Integer

**Actor\_Admin\_Port\_State**

The administrative values of the Actor's state parameters. This consists of the following set of variables, as described in 43.4.2.2:

LACP\_Activity

LACP\_Timeout

Aggregation

Synchronization

Collecting  
Distributing  
Defaulted  
Expired  
Value: 8 bits

**Actor\_Oper\_Port\_State**

The operational values of the Actor's state parameters. This consists of the following set of variables, as described in 43.4.2.2:

LACP\_Activity  
LACP\_Timeout  
Aggregation  
Synchronization  
Collecting  
Distributing  
Defaulted  
Expired  
Value: 8 bits

**Partner\_Admin\_System**

Default value for the MAC address component of the System Identifier of the Partner, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: 48 bits

**Partner\_Oper\_System**

The operational value of the MAC address component of the System Identifier of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_System.

Value: 48 bits

**Partner\_Admin\_System\_Priority**

Default value for the System Priority component of the System Identifier of the Partner, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

**Partner\_Oper\_System\_Priority**

The operational value of the System Priority of the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_System\_Priority.

Value: Integer

**Partner\_Admin\_Key**

Default value for the Partner's Key, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

**Partner\_Oper\_Key**

The operational value of the Key value assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_Key.

Value: Integer

**Partner\_Admin\_Port\_Number**

Default value for the Port Number component of the Partner's Port Identifier, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

**Partner\_Oper\_Port\_Number**

The operational value of the port number assigned to this link by the Partner. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_Port\_Number.

Value: Integer

**Partner\_Admin\_Port\_Priority**

Default value for the Port Priority component of the Partner's Port Identifier, assigned by administrator or System policy for use when the Partner's information is unknown or expired.

Value: Integer

**Partner\_Oper\_Port\_Priority**

The operational value of the priority value assigned to this link by the Partner, used to converge dynamic Key changes. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_Port\_Priority.

Value: Integer

**Partner\_Admin\_Port\_State**

Default value for the Partner's state parameters, assigned by administrator or System policy for use when the Partner's information is unknown or expired. The value consists of the following set of variables, as described in 43.4.2.2:

- LACP\_Activity
- LACP\_Timeout
- Aggregation
- Synchronization
- Collecting
- Distributing
- Defaulted
- Expired

The value of Collecting shall be set the same as the value of Synchronization.

Value: 8 bits

**Partner\_Oper\_Port\_State**

The operational value of the Actor's view of the current values of the Partner's state parameters. The Actor sets this variable either to the value received from the Partner in an LACPDU, or to the value of Partner\_Admin\_Port\_State. The value consists of the following set of variables, as described in 43.4.2.2:

- LACP\_Activity
- LACP\_Timeout
- Aggregation
- Synchronization
- Collecting
- Distributing
- Defaulted
- Expired

Value: 8 bits

**port\_enabled**

A variable indicating that the physical layer has indicated that the link has been established and the port is operable.

Value: Boolean

TRUE if the physical layer has indicated that the port is operable.

FALSE otherwise.

NOTE—The means by which the value of the port\_enabled variable is generated by the underlying MAC and/or PHY is implementation-dependent.

**43.4.8 Variables used for managing the operation of the state machines****BEGIN**

This variable indicates the initialization (or reinitialization) of the LACP protocol entity. It is set to TRUE when the System is initialized or reinitialized, and is set to FALSE when (re-)initialization has completed.

Value: Boolean

**LACP\_Enabled**

This variable indicates that the port is operating the LACP. If the port is operating in half duplex, the value of LACP\_Enabled shall be FALSE. Otherwise, the value of LACP\_Enabled shall be TRUE.

Value: Boolean

**actor\_churn**

This variable indicates that the Actor Churn Detection machine has detected that a local port configuration has failed to converge within a specified time, and that management intervention is required.

Value: Boolean

**partner\_churn**

This variable indicates that the Partner Churn Detection machine has detected that a remote port configuration has failed to converge within a specified time, and that management intervention is required.

Value: Boolean

**Ready\_N**

The port asserts Ready\_N TRUE to indicate to the Selection Logic that the wait\_while\_timer has expired and it is waiting (i.e., the port is in the WAITING state) to attach to an Aggregator. Otherwise, its value is FALSE. There is one Ready\_N value for each port.

Value: Boolean

**Ready**

The Selection Logic asserts Ready TRUE when the values of Ready\_N for all ports that are waiting to attach to a given Aggregator are TRUE. If any of the values of Ready\_N for the ports that are waiting to attach to that Aggregator are FALSE, or if there are no ports waiting to attach to that Aggregator, then the value of Ready is FALSE.

Value: Boolean

**Selected**

A value of SELECTED indicates that the Selection Logic has selected an appropriate Aggregator. A value of UNSELECTED indicates that no aggregator is currently selected. A value of STANDBY indicates that although the Selection Logic has selected an appropriate Aggregator, aggregation restrictions currently prevent the port from being enabled as part of the aggrega-

tion, and so the port is being held in a standby condition. This variable can only be set to SELECTED or STANDBY by the operation of the port's Selection Logic. It can be set to UNSELECTED by the operation of the port's Receive machine, or by the operation of the Selection Logic associated with another port.

NOTE—Setting Selected UNSELECTED in the Selection Logic associated with another port occurs if the Selection Logic determines that the other port has a stronger claim to attach to this port's current Aggregator.

Value: SELECTED, UNSELECTED, or STANDBY

#### port\_moved

This variable is set to TRUE if the Receive machine for a port is in the PORT\_DISABLED state, and the combination of Partner\_Oper\_System and Partner\_Oper\_Port\_Number in use by that port has been received in an incoming LACPDU on a different port. This variable is set to FALSE once the INITIALIZE state of the Receive machine has set the Partner information for the port to administrative default values.

Value: Boolean

### 43.4.9 Functions

#### recordPDU

This function records the parameter values for the Actor carried in a received LACPDU (Actor\_Port, Actor\_Port\_Priority, Actor\_System, Actor\_System\_Priority, Actor\_Key, and Actor\_State variables) as the current Partner operational parameter values (Partner\_Oper\_Port\_Number, Partner\_Oper\_Port\_Priority, Partner\_Oper\_System, Partner\_Oper\_System\_Priority, Partner\_Oper\_Key, and Partner\_Oper\_Port\_State variables with the exception of Synchronization) and sets Actor\_Oper\_Port\_State.Defaulted to FALSE.

This function also updates the value of the Partner\_Oper\_Port\_State.Synchronization using the parameter values carried in received LACPDU's. Parameter values for the Partner carried in the received PDU (Partner\_Port, Partner\_Port\_Priority, Partner\_System, Partner\_System\_Priority, Partner\_Key, and Partner\_State.Aggregation) are compared to the corresponding operational parameter values for the Actor (Actor\_Port\_Number, Actor\_Port\_Priority, Actor\_System, Actor\_System\_Priority, Actor\_Oper\_Port\_Key, and Actor\_Oper\_Port\_State.Aggregation). Partner\_Oper\_Port\_State.Synchronization is set to TRUE if all of these parameters match, Actor\_State.Synchronization in the received PDU is set to TRUE, and LACP will actively maintain the link in the aggregation.

Partner\_Oper\_Port\_State.Synchronization is also set to TRUE if the value of Actor\_State.Aggregation in the received PDU is set to FALSE (i.e., indicates an Individual link), Actor\_State.Synchronization in the received PDU is set to TRUE, and LACP will actively maintain the link.

Otherwise, Partner\_Oper\_Port\_State.Synchronization is set to FALSE.

LACP is considered to be actively maintaining the link if either the PDU's Actor\_State.LACP\_Activity variable is TRUE or both the Actor's Actor\_Oper\_Port\_State.LACP\_Activity and the PDU's Partner\_State.LACP\_Activity variables are TRUE.

#### recordDefault

This function records the default parameter values for the Partner carried in the Partner Admin parameters (Partner\_Admin\_Port\_Number, Partner\_Admin\_Port\_Priority, Partner\_Admin\_System, Partner\_Admin\_System\_Priority, Partner\_Admin\_Key, and

Partner\_Admin\_Port\_State) as the current Partner operational parameter values (Partner\_Oper\_Port\_Number, Partner\_Oper\_Port\_Priority, Partner\_Oper\_System, Partner\_Oper\_System\_Priority, Partner\_Oper\_Key, and Partner\_Oper\_Port\_State) and sets Actor\_Oper\_Port\_State.Defaulted to TRUE.

#### update\_Selected

This function updates the value of the Selected variable, using parameter values from a newly received LACPDU. The parameter values for the Actor carried in the received PDU (Actor\_Port, Actor\_Port\_Priority, Actor\_System, Actor\_System\_Priority, Actor\_Key, and Actor\_State.Aggregation) are compared with the corresponding operational parameter values for the port's Partner (Partner\_Oper\_Port\_Number, Partner\_Oper\_Port\_Priority, Partner\_Oper\_System, Partner\_Oper\_System\_Priority, Partner\_Oper\_Key, and Partner\_Oper\_Port\_State.Aggregation). If one or more of the comparisons show that the value(s) received in the PDU differ from the current operational values, then Selected is set to UNSELECTED. Otherwise, Selected remains unchanged.

#### update\_Default\_Selected

This function updates the value of the Selected variable, using the Partner administrative parameter values. The administrative values (Partner\_Admin\_Port\_Number, Partner\_Admin\_Port\_Priority, Partner\_Admin\_System, Partner\_Admin\_System\_Priority, Partner\_Admin\_Key, and Partner\_Admin\_Port\_State.Aggregation) are compared with the corresponding operational parameter values for the Partner (Partner\_Oper\_Port\_Number, Partner\_Oper\_Port\_Priority, Partner\_Oper\_System, Partner\_Oper\_System\_Priority, Partner\_Oper\_Key, and Partner\_Oper\_Port\_State.Aggregation). If one or more of the comparisons shows that the administrative value(s) differ from the current operational values, then Selected is set to UNSELECTED. Otherwise, Selected remains unchanged.

#### update\_NTT

This function updates the value of the NTT variable, using parameter values from a newly received LACPDU. The parameter values for the Partner carried in the received PDU (Partner\_Port, Partner\_Port\_Priority, Partner\_System, Partner\_System\_Priority, Partner\_Key, Partner\_State.LACP\_Activity, Partner\_State.LACP\_Timeout, Partner\_State.Synchronization, and Partner\_State.Aggregation) are compared with the corresponding operational parameter values for the Actor (Actor\_Port\_Number, Actor\_Port\_Priority, Actor\_System, Actor\_System\_Priority, Actor\_Oper\_Port\_Key, Actor\_Oper\_Port\_State.LACP\_Activity, Actor\_Oper\_Port\_State.LACP\_Timeout, Actor\_Oper\_Port\_State.Synchronization, and Actor\_Oper\_Port\_State.Aggregation). If one or more of the comparisons show that the value(s) received in the PDU differ from the current operational values, then NTT is set to TRUE. Otherwise, NTT remains unchanged.

#### Attach\_Mux\_To\_Aggregator

This function causes the port's Control Parser/Multiplexer to be attached to the Aggregator Parser/Multiplexer of the selected Aggregator, in preparation for collecting and distributing frames.

#### Detach\_Mux\_From\_Aggregator

This function causes the port's Control Parser/Multiplexer to be detached from the Aggregator Parser/Multiplexer of the Aggregator to which the port is currently attached.

#### Enable\_Collecting

This function causes the Aggregator Parser of the Aggregator to which the port is attached to start collecting frames from the port.

**Disable\_Collecting**

This function causes the Aggregator Parser of the Aggregator to which the port is attached to stop collecting frames from the port.

**Enable\_Distributing**

This function causes the Aggregator Multiplexer of the Aggregator to which the port is attached to start distributing frames to the port.

**Disable\_Distributing**

This function causes the Aggregator Multiplexer of the Aggregator to which the port is attached to stop distributing frames to the port.

**Enable\_Collecting\_Distributing**

This function causes the Aggregator Parser of the Aggregator to which the port is attached to start collecting frames from the port, and the Aggregator Multiplexer to start distributing frames to the port.

**Disable\_Collecting\_Distributing**

This function causes the Aggregator Parser of the Aggregator to which the port is attached to stop collecting frames from the port, and the Aggregator Multiplexer to stop distributing frames to the port.

**43.4.10 Timers****current\_while\_timer**

This timer is used to detect whether received protocol information has expired. If Actor\_Oper\_State.LACP\_Timeout is set to Short Timeout, the timer is started with the value Short\_Timeout\_Time. Otherwise, it is started with the value Long\_Timeout\_Time (see 43.4.4).

**actor\_churn\_timer**

This timer is used to detect Actor churn states. It is started using the value Churn\_Detection\_Time (see 43.4.4).

**periodic\_timer (time\_value)**

This timer is used to generate periodic transmissions. It is started using the value Slow\_Periodic\_Time or Fast\_Periodic\_Time (see 43.4.4), as specified in the Periodic Transmission state machine.

**partner\_churn\_timer**

This timer is used to detect Partner churn states. It is started using the value Churn\_Detection\_Time (see 43.4.4).

**wait\_while\_timer**

This timer provides hysteresis before performing an aggregation change, to allow all links that will join this Aggregation to do so. It is started using the value Aggregate\_Wait\_Time (see 43.4.4).

**43.4.11 Messages****CtrlMuxN:MA\_DATA.indicate(LACPDU)**

This message is generated by the Control Parser as a result of the reception of a LACPDU, formatted as defined in 43.4.2.

#### 43.4.12 Receive machine

The Receive machine shall implement the function specified in Figure 43–10 with its associated parameters (43.4.4 through 43.4.11).

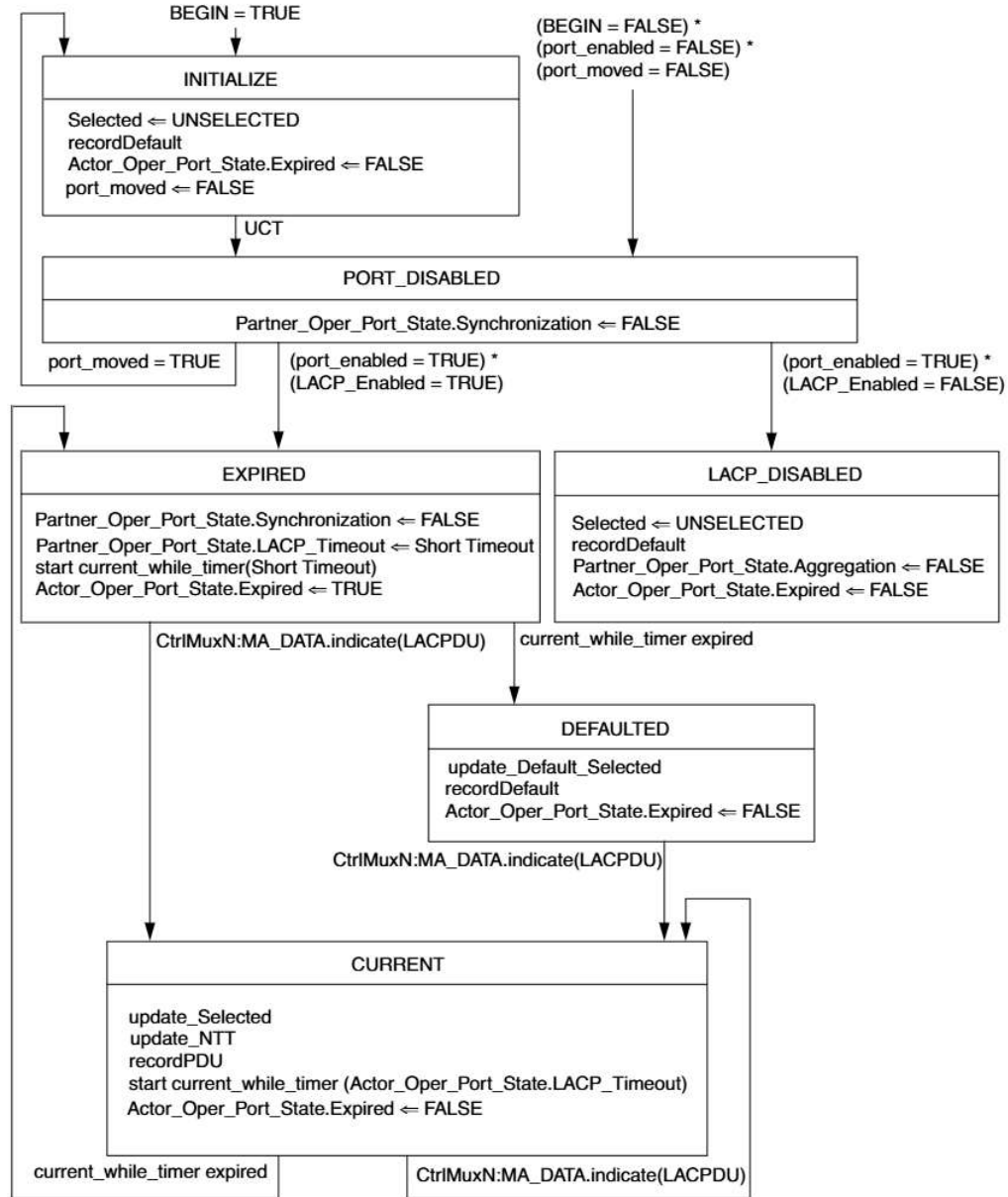


Figure 43–10—Receive machine state diagram

On receipt of a LACPDU, the state machine enters the CURRENT state. The update\_Selected function sets the Selected variable UNSELECTED if the Link Aggregation Group identified by the combination of the protocol Partner's own information and the Actor's own information has changed. The Selected variable is used by the Mux machine (43.4.15).

NOTE—The Receive machine may set the Selected variable to UNSELECTED; however, setting this variable to SELECTED or STANDBY is the responsibility of the Selection Logic.

The update\_NTT function is used to determine whether further protocol transmissions are required; NTT is set to TRUE if the Partner's view of the Actor's operational parameters is not up to date. The recordPDU function records the information contained in the LACPDU in the Partner operational variables, and the current\_while timer is started. The value used to start the timer is either Short\_Timeout\_Time or Long\_Timeout\_Time, depending upon the Actor's operational value of LACP\_Timeout.

In the process of executing the recordPDU function, a Receive machine compliant to this standard shall not validate the Version Number, TLV\_type, or Reserved fields in received LACPDUs. The same actions are taken regardless of the values received in these fields. A Receive machine may validate the Actor\_Information\_Length, Partner\_Information\_Length, Collector\_Information\_Length, or Terminator\_Length fields. These behaviors, together with the constraint on future protocol enhancements, are discussed in 43.4.2.2.

NOTE—The rules expressed above allow Version 1 devices to be compatible with future revisions of the protocol.

If no LACPDU is received before the current\_while timer expires, the state machine transits to the EXPIRED state. The Partner\_Oper\_Port\_State.Synchronization variable is set to FALSE, the current operational value of the Partner's LACP\_Timeout variable is set to Short Timeout, and the current\_while timer is started with a value of Short\_Timeout\_Time. This is a transient state; the LACP\_Timeout settings allow the Actor to transmit LACPDUs rapidly in an attempt to re-establish communication with the Partner.

If no LACPDU is received before the current\_while timer expires again, the state machine transits to the DEFAULTED state. The recordDefault function overwrites the current operational parameters for the Partner with administratively configured values. This allows configuration of aggregations and individual links when no protocol Partner is present, while still permitting an active Partner to override default settings. The update\_Default\_Selected function sets the Selected variable UNSELECTED if the Link Aggregation Group has changed. Since all operational parameters are now set to locally administered values, there can be no disagreement as to the Link Aggregation Group, so the Partner\_Oper\_Port\_State.Synchronization variable is set to TRUE.

If the port becomes inoperable and a BEGIN event has not occurred, the state machine enters the PORT\_DISABLED state. Partner\_Oper\_Port\_State.Synchronization is set to FALSE. This state allows the current Selection state to remain undisturbed, so that, in the event that the port is still connected to the same Partner and Partner port when it becomes operable again, there will be no disturbance caused to higher layers by unnecessary re-configuration. If the same Actor System ID and Port are seen in a LACPDU received on a different Port (port\_moved is set to TRUE), this indicates that the physical connectivity has changed, and causes the state machine to enter the INITIALIZE state. This state is also entered if a BEGIN event occurs.

The INITIALIZE state causes the administrative values of the Partner parameters to be used as the current operational values, and sets Selected to UNSELECTED. These actions force the Mux machine to detach the port from its current Aggregator. The variable port\_moved is set to FALSE; if the entry to INITIALIZE occurred as a result of port\_moved being set to TRUE, then the state machine will immediately transition back to the PORT\_DISABLED state.

If the port is operating in half duplex, the operation of LACP is disabled on the port (LACP\_Enabled is FALSE) and the LACP\_DISABLED state is entered. This state is entered following a BEGIN or Port Enabled event. This state is similar to the DEFAULTED state, except that the port is forced to operate as an Individual port, as the value of Partner\_Oper\_Port\_State.Aggregation is forced to Individual. Exit from this state occurs on a BEGIN or Port Disabled event.

#### 43.4.13 Periodic Transmission machine

The Periodic Transmission machine shall implement the function specified in Figure 43–11 with its associated parameters (43.4.4 through 43.4.11).

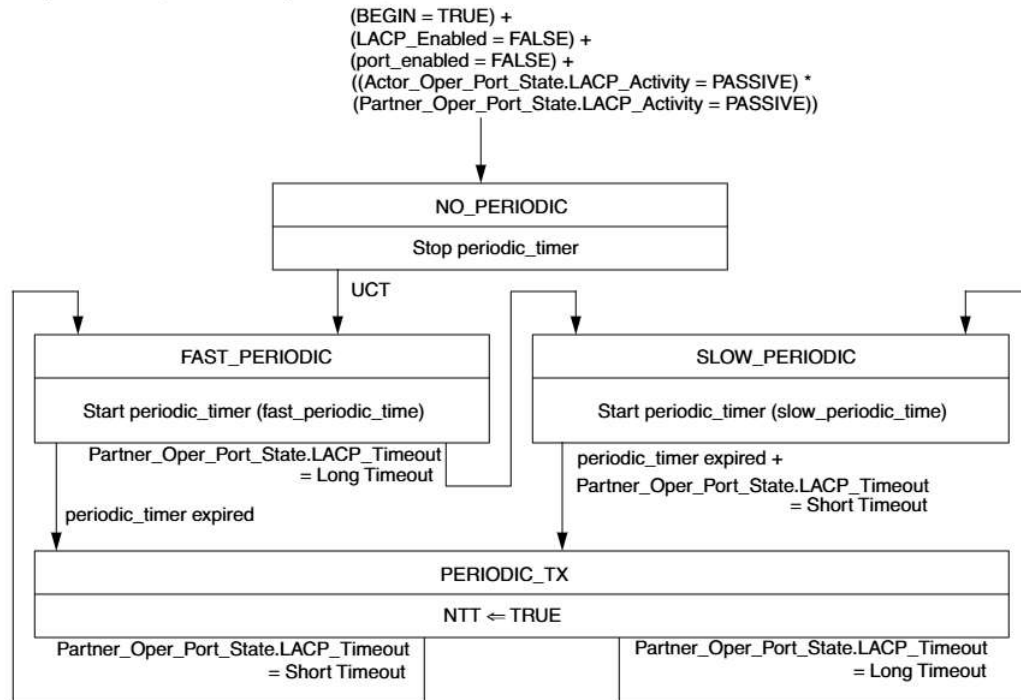


**Figure 43–11 – Periodic Transmission machine state diagram**

The Periodic Transmission machine establishes the desire of the Actor and Partner to exchange periodic LACPDUs on the link in order to maintain an aggregation, and establishes 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 Partner; this rate is linked to the speed at which the Partner will time out received information.

The state machine has four states. They are as follows:

- NO\_PERIODIC*. While in this state, periodic transmissions are disabled.
- FAST\_PERIODIC*. While in this state, periodic transmissions are enabled at a fast transmission rate.
- SLOW\_PERIODIC*. While in this state, periodic transmissions are enabled at a slow transmission rate.
- PERIODIC\_TX*. This is a transitory state entered on periodic timer expiry, that asserts NTT and then exits to *FAST\_PERIODIC* or *SLOW\_PERIODIC* depending upon the Partner's LACP\_Timeout setting.

The values of `Partner_Oper_Port_State.LACP_Activity` and `Actor_Oper_Port_State.LACP_Activity` determine whether periodic transmissions take place. If either or both parameters are set to Active LACP, then periodic transmissions occur; if both are set to Passive LACP, then periodic transmissions do not occur. Similarly, if either of the `LACP_Enabled` or `port_enabled` variables is set to FALSE, indicating that LACP has been disabled on the port or that the port is non-operational, then no periodic transmissions take place.

If periodic transmissions are enabled, the rate at which they take place is determined by the value of the `Partner_Oper_Port_State.LACP_Timeout` variable. If this variable is set to Short Timeout, then the value `fast_periodic_time` is used to determine the time interval between periodic transmissions. Otherwise, `slow_periodic_time` is used to determine the time interval.

#### 43.4.14 Selection Logic

The Selection Logic selects a compatible Aggregator for a port, using the port's LAG ID. The Selection Logic may determine that the link should be operated as a standby link if there are constraints on the simultaneous attachment of ports that have selected the same Aggregator.

NOTE—There will never be more than one Aggregator with the same LAG ID, but there may be none. Normally, the latter will be a temporary state, caused by the fact that it takes a finite time for ports to be moved to the correct aggregators during reconfiguration.

The Mux machine controls the process of attaching the port to a selected Aggregator, after first detaching the port from any prior Aggregator if the port's LAG ID has changed.

NOTE—A port is always detached from its prior Aggregator when the LAG ID changes, even if the same Aggregator is selected later; to do otherwise would be to risk misdelivery of frames. Selection of a new Aggregator cannot take place until the port is detached from any prior Aggregator; other Aggregators may become free while the port is detaching, and other ports may attach to some of the available Aggregators during this time interval.

The operation of the Selection Logic is separated into the following two subclauses:

- a) The requirements for the correct operation of the Selection Logic are defined in 43.4.14.1.
- b) The recommended default operation of the Selection Logic is described in 43.4.14.2.

This separation reflects the fact that a wide choice of selection rules is possible within the proper operation of the protocol. An implementation that claims conformance to this standard may support selection rules other than the recommended default; however, any such rules shall meet the requirements stated in 43.4.14.1.

##### 43.4.14.1 Selection Logic—Requirements

Aggregation is represented by a port selecting an appropriate Aggregator, and then attaching to that Aggregator. The following are required for correct operation of the selection and attachment logic:

- a) The implementation shall support at least one Aggregator per System.
- b) Each port shall be assigned an operational Key (43.3.5). Ports that can aggregate together are assigned the same operational Key as the other ports with which they can aggregate; ports that cannot aggregate with any other port are allocated unique operational Keys.
- c) Each Aggregator shall be assigned an operational Key.
- d) Each Aggregator shall be assigned an identifier that distinguishes it among the set of Aggregators in the System.
- e) A port shall only select an Aggregator that has the same operational Key assignment as its own operational Key.

- f) Subject to the exception stated in item g), ports that are members of the same Link Aggregation Group (i.e., two or more ports that have the same Actor System ID, Actor Key, Partner System ID, and Partner Key, and that are not required to be Individual) shall select the same Aggregator.
- g) Any pair of ports that are members of the same Link Aggregation Group, but are connected together by the same link, shall not select the same Aggregator (i.e., if a loopback condition exists between two ports, they shall not be aggregated together. For both ports, the Actor System ID is the same as the Partner System ID; also, for port A, the Partner's port identifier is port B, and for port B, the Partner's port identifier is port A).

NOTE—This exception condition prevents the formation of an aggregated link, comprising two ends of the same physical link aggregated together, in which all frames transmitted through an Aggregator are immediately received through the same Aggregator. However, it permits the aggregation of multiple links that are in loopback; for example, if port A is looped back to port C and port B is looped back to port D, then it is permissible for A and B (or A and D) to aggregate together, and for C and D (or B and C) to aggregate together.

- h) Any port that is required to be Individual (i.e., the operational state for the Actor or the Partner indicates that the port is Individual) shall not select the same Aggregator as any other port.
- i) Any port that is Aggregatable shall not select an Aggregator to which an Individual port is already attached.
- j) If the above conditions result in a given port being unable to select an Aggregator, then that port shall not be attached to any Aggregator.
- k) If there are further constraints on the attachment of ports that have selected an Aggregator, those ports may be selected as standby in accordance with the rules specified in 43.6.1. Selection or deselection of that Aggregator can cause the Selection Logic to re-evaluate the ports to be selected as standby.
- l) The Selection Logic operates upon the operational information recorded by the Receive state machine, along with knowledge of the Actor's own operational configuration and state. The Selection Logic uses the LAG ID for the port, determined from these operational parameters, to locate the correct Aggregator to attach the port to.
- m) The Selection Logic is invoked whenever a port is not attached to and has not selected an Aggregator, and executes continuously until it has determined the correct Aggregator for the port.

NOTE—The Selection Logic may take a significant time to complete its determination of the correct Aggregator, as a suitable Aggregator may not be immediately available, due to configuration restrictions or the time taken to re-allocate ports to other Aggregators.

- n) Once the correct Aggregator has been determined, the variable Selected shall be set to SELECTED or to STANDBY (43.4.8, 43.6.1).

NOTE—If Selected is SELECTED, the Mux machine will start the process of attaching the port to the selected Aggregator. If Selected is STANDBY, the Mux machine holds the port in the WAITING state, ready to be attached to its Aggregator once its Selected state changes to SELECTED.

- o) The Selection Logic is responsible for computing the value of the Ready variable from the values of the Ready\_N variable(s) associated with the set of ports that are waiting to attach to the same Aggregator (see 43.4.8).
- p) Where the selection of a new Aggregator by a port, as a result of changes to the selection parameters, results in other ports in the System being required to re-select their Aggregators in turn, this is achieved by setting Selected to UNSELECTED for those other ports that are required to re-select their Aggregators.

NOTE—The value of Selected is set to UNSELECTED by the Receive machine for the port when a change of LAG ID is detected.

- q) A port shall not be enabled for use by the MAC Client until it has both selected and attached to an Aggregator.

#### 43.4.14.2 Selection Logic—Recommended default operation

The recommended default behavior provides an element of determinism (i.e., history independence) in the assignment of ports to Aggregators. It also has the characteristic that no additional MAC addresses are needed, over and above those already assigned to the set of underlying MACs.

NOTE—This standard does not specify any alternative selection rules beyond the recommended set. A wide variety of selection rules are possible within the scope of the requirements stated in 43.4.14.1. In particular, it is possible within these requirements to support implementations that provide fewer Aggregators than ports, as well as implementations designed to minimize configuration changes at the expense of less deterministic behavior.

Each port has an Aggregator associated with it, (i.e., the number of Aggregators in the System equals the number of ports supported). Each port/Aggregator pair is assigned the same operational Key and port number. When there are multiple ports in an aggregation, the Aggregator that the set of ports selects is the Aggregator with the same port number as the lowest-numbered port in the aggregation. Note that this lowest numbered port may not be in a state that allows data transfer across the link; however, it has selected the Aggregator in question. This is illustrated in Figure 43–12.

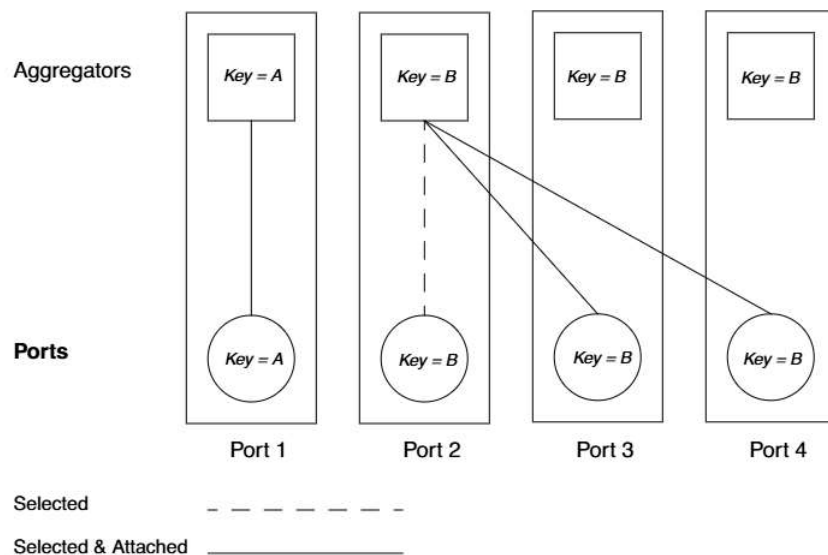


**Figure 43–12—Selection of Aggregators**

If the port is Individual, then the Aggregator selected is always the port's own Aggregator. Otherwise, an Aggregator is selected from the set of Aggregators corresponding to the set of ports that will form the aggregation. The Aggregator selected is the lowest numbered Aggregator with the same selection parameters as those of the port. These selection parameters are

- The Actor's System ID
- The Actor's operational Key
- The Partner's System ID
- The Partner's operational Key
- The Individual\_Aggregator state (which must be FALSE)

#### 43.4.15 Mux machine

The Mux machine shall implement the function specified in either of the Mux machine state diagrams, Figure 43–13 and Figure 43–14, with their associated parameters (43.4.4 through 43.4.11).

The state machine conventions in 21.5 assert that all in-state actions are instantaneous, are performed in sequence, and are performed prior to evaluating any exit conditions. While the Mux machine will operate correctly if all actions can be performed instantaneously, this will not be realistic in many implementations. Correct operation is maintained even if actions are not completed instantaneously, as long as each action completes prior to initiating the next sequential action, and all actions complete prior to evaluating any exit conditions.

The independent control state diagram (Figure 43–13) is suitable for use implementations in which it is possible to control enabling and disabling of frame collection from a port, and frame distribution to a port, independently. The coupled control state diagram (Figure 43–14) is suitable for use implementations where collecting and distributing cannot be controlled independently with respect to a port. It is recommended that the independent control state diagram be implemented in preference to the coupled control state diagram.

The value of the Selected variable may be changed by the following:

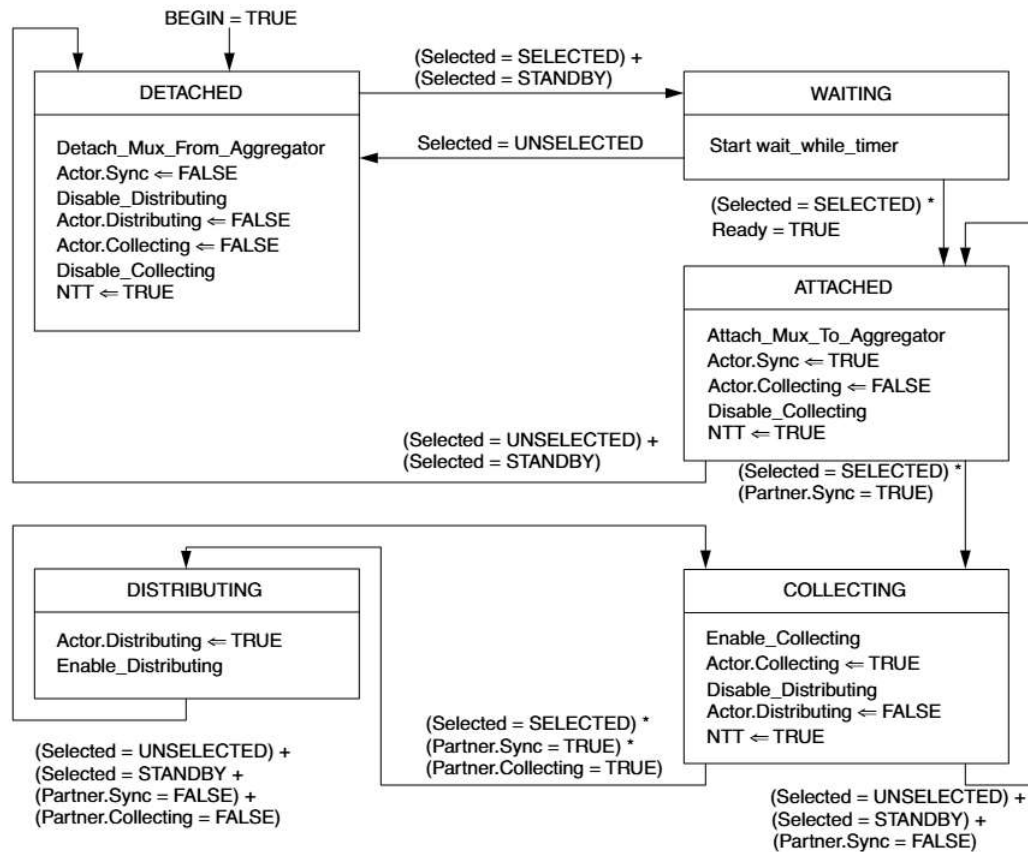
- a) *The Receive machine.* The Receive machine can set Selected to UNSELECTED at any time if any of the following change: The Partner System ID, the Partner Priority, the Partner Key, the Partner\_State.Aggregation, the Actor System ID, the Actor Priority, the Actor Key, or the Actor\_State.Aggregation.
- b) *The Selection Logic, in the process of selecting an Aggregator.* The Selection Logic will select an Aggregator when the Mux machine is in the DETACHED state and the value of the Selected variable is UNSELECTED.
- c) *The Selection Logic, in the process of selecting or de-selecting standby links.* If the value of the Selected variable is SELECTED or STANDBY, the Selection Logic can change the value to STANDBY or SELECTED.

The Mux machine enters the WAITING state from the DETACHED state if the Selection Logic determines that Selected is either SELECTED or STANDBY. The WAITING state provides a holding state for the following two purposes:

- d) If Selected is SELECTED, the wait\_while\_timer forces a delay to allow for the possibility that other ports may be reconfiguring at the same time. Once the wait\_while\_timer expires, and once the wait\_while\_timers of all other ports that are ready to attach to the same Aggregator have expired, the process of attaching the port to the Aggregator can proceed, and the state machine enters the ATTACHED state. During the waiting time, changes in selection parameters can occur that will result in a re-evaluation of Selected. If Selected becomes UNSELECTED, then the state machine re-enters the DETACHED state. If Selected becomes STANDBY, the operation is as described in item c).

NOTE—This waiting period reduces the disturbance that will be visible to higher layers; for example, on start-up events. However, the selection need not wait for the entire waiting period in cases where it is known that no other ports will attach; for example, where all other ports with the same operational Key are already attached to the Aggregator.

- e) If Selected is STANDBY, the port is held in the WAITING state until such a time as the selection parameters change, resulting in a re-evaluation of the Selected variable. If Selected becomes UNSELECTED, the state machine re-enters the DETACHED state. If SELECTED becomes SELECTED, then the operation is as described in item d). The latter case allows a port to be brought into operation from STANDBY with minimum delay once Selected becomes SELECTED.



The following abbreviations are used in this diagram:

- Actor.Sync:** Actor\_Oper\_Port\_State.Synchronization
- Actor.Collecting:** Actor\_Oper\_Port\_State.Collecting
- Actor.Distributing:** Actor\_Oper\_Port\_State.Distributing
- Partner.Sync:** Partner\_Oper\_Port\_State.Synchronization
- Partner.Collecting:** Partner\_Oper\_Port\_State.Collecting

**Figure 43–13—Mux machine state diagram (independent control)**

On entry to the ATTACHED state, the Mux machine initiates the process of attaching the port to the selected Aggregator. Once the attachment process has completed, the value of Actor\_Oper\_Port\_State.Synchronization is set to TRUE indicating that the Actor considers the port to be IN\_SYNC, and Actor\_Oper\_Port\_State.Collecting is set to FALSE. Collection of frames from the port is disabled. In the coupled control state machine, Distribution of frames to the port is also disabled, and Actor\_Oper\_Port\_State.Distributing is set to FALSE.

A change in the Selected variable to UNSELECTED or to STANDBY causes the state machine to enter the DETACHED state. The process of detaching the port from the Aggregator is started. Once the detachment process is completed, Actor\_Oper\_Port\_State.Synchronization is set to FALSE indicating that the Actor considers the port to be OUT\_OF\_SYNC, distribution of frames to the port is disabled, Actor\_Oper\_Port\_State.Distributing and Actor\_Oper\_Port\_State.Collecting are set to FALSE, and collection of frames from the port is disabled. The state machine remains in the DISABLED state until such time as the Selection logic is able to select an appropriate Aggregator.

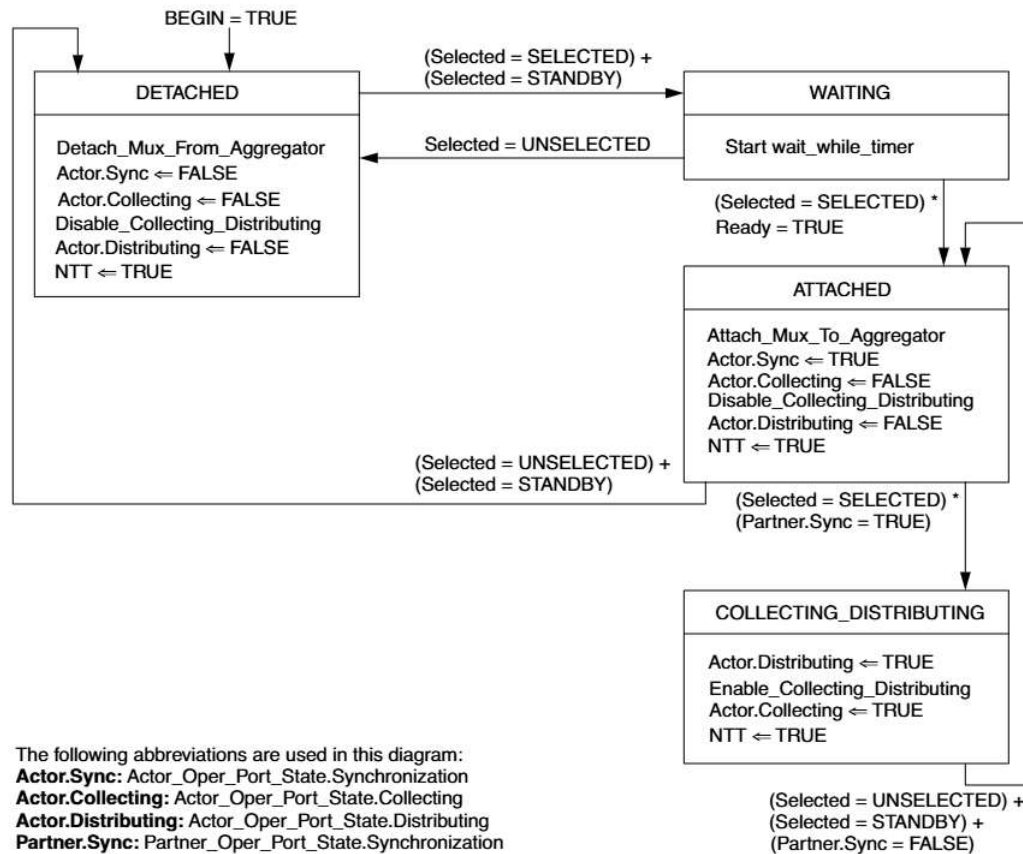


Figure 43-14—Mux machine state diagram (coupled control)

While in the ATTACHED state, a TRUE value for Partner\_Oper\_Port\_State.Synchronization causes the state machine to transition to the COLLECTING state (independent control) or the COLLECTING\_DISTRIBUTING state (coupled control).

In the COLLECTING state, collection of frames from the port is enabled, then Actor\_Oper\_Port\_State.Collecting is set to TRUE, then distribution of frames to the port is disabled and Actor\_Oper\_Port\_State.Distributing is set to FALSE. The state machine will return to the ATTACHED state if Selected changes to UNSELECTED or STANDBY, or if the Partner's synchronization state becomes FALSE. Once the Partner has signalled that collecting has been enabled (Partner\_Oper\_Port\_State.Collecting is TRUE), the state machine transitions to the DISTRIBUTING state. The value of Actor\_Oper\_Port\_State.Distributing is set to TRUE, and then distribution of frames to the port is enabled. From DISTRIBUTING, a return to the COLLECTING state occurs if the value of Selected becomes UNSELECTED or STANDBY, if the Partner's synchronization state becomes FALSE, or if the Partner signals that collection has been disabled (Partner\_Oper\_Port\_State.Collecting is FALSE).

In the COLLECTING\_DISTRIBUTING state, the value of Actor\_Oper\_Port\_State.Distributing is set to TRUE, distribution of frames to the port and collection of frames from the port are both enabled, and then Actor\_Oper\_Port\_State.Collecting is set to TRUE. The state machine will return to the ATTACHED state if Selected changes to UNSELECTED or STANDBY, or if the Partner's synchronization state becomes FALSE.

The sequence of operations and transitions defined for the COLLECTING and DISTRIBUTING states in the independent control version of this state machine ensures that frames are not distributed to a port until the Partner has enabled collection, and that distribution is stopped as soon as the Partner's state indicates that collection has been disabled. This sequence minimizes the possibility that frames will be misdelivered during the process of bringing the port into operation or taking the port out of operation. In the coupled control version of the state machine, the COLLECTING and DISTRIBUTING states merge together to form the combined state, COLLECTING\_DISTRIBUTING. As independent control is not possible, the coupled control state machine does not wait for the Partner to signal that collection has started before enabling both collection and distribution.

The NTT variable is set to TRUE in the DETACHED, ATTACHED, COLLECTING, and COLLECTING\_DISTRIBUTING states in order to ensure that the Partner is made aware of the changes in the Actor's state variables that are caused by the operations performed in those states.

#### 43.4.16 Transmit machine

When the Transmit machine creates a LACPDU for transmission, it shall fill in the following fields with the corresponding operational values for this port:

- a) Actor\_Port and Actor\_Port\_Priority
- b) Actor\_System and Actor\_System\_Priority
- c) Actor\_Key
- d) Actor\_State
- e) Partner\_Port and Partner\_Port\_Priority
- f) Partner\_System and Partner\_System\_Priority
- g) Partner\_Key
- h) Partner\_State
- i) CollectorMaxDelay

When the Periodic machine is in the NO\_PERIODIC state, the Transmit machine shall

- Not transmit any LACPDU, and
- Set the value of NTT to FALSE.

When the LACP\_Enabled variable is TRUE and the NTT (43.4.7) variable is TRUE, the Transmit machine shall ensure that a properly formatted LACPDU (43.4.2) is transmitted (i.e., issue a Ctrl-MuxN:MA\_DATA.Request(LACPDU) service primitive), subject to the restriction that no more than three LACPDU, may be transmitted in any Fast\_Periodic\_Time interval. If NTT is set to TRUE when this limit is in force, the transmission shall be delayed until such a time as the restriction is no longer in force. The NTT variable shall be set to FALSE when the Transmit machine has transmitted a LACPDU.

If the transmission of a LACPDU is delayed due to the above restriction, the information sent in the LACPDU corresponds to the operational values for the port at the time of transmission, not at the time when NTT was first set to TRUE. 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.

When the LACP\_Enabled variable is FALSE, the Transmit machine shall not transmit any LACPDU, and shall set the value of NTT to FALSE.

#### 43.4.17 Churn Detection machines

If implemented, the Churn Detection machines shall implement the functions specified in Figure 43–15 and Figure 43–16 with their associated parameters (43.4.4 through 43.4.11). Implementation of the Churn Detection machines is mandatory if the associated management functionality (the Aggregation Port Debug Information package) is implemented; otherwise, implementation of the Churn Detection machines is optional.

The Churn Detection machines detect the situation where a port is operable, but the Actor and Partner have not attached the link to an Aggregator and brought the link into operation within a bounded time period. Under normal operation of the LACP, agreement between Actor and Partner should be reached very rapidly. Continued failure to reach agreement can be symptomatic of device failure, of the presence of non-standard devices, or of misconfiguration; it can also be the result of normal operation in cases where either or both Systems are restricted in their ability to aggregate. Detection of this condition is signalled by the Churn Detection machines to management in order to prompt administrative action to further diagnose and correct the fault.

NOTE—One of the classes of problems that will be detected by this machine is the one where the implementation has been designed to support a limited number of Aggregators (fewer than the number of ports—see 43.6.4.2) and the physical topology is such that one or more ports end up with no Aggregator to attach to. This may be the result of a wiring error or an error in the allocation of operational Key values to the ports and Aggregators. Alternatively, failure of a link to aggregate may be the result of a link being placed in standby mode by a System that has hardware limitations placed on its aggregation ability, leading it to make use of the techniques described in 43.6 to find the ideal configuration. Given that the time taken by an aggregation-constrained System to stabilize its configuration may be relatively large, the churn detection timers allow 60 seconds to elapse before a Churn condition is signalled.

The symptoms that the Actor Churn Detection state machine detects is that the Actor's Mux has determined that it is OUT\_OF\_SYNC, and that condition has not resolved itself within a period of time equal to Short\_Timeout\_Time (43.4.4). Under normal conditions, this is ample time for convergence to take place. Similarly, the Partner Churn Detection state machine detects a failure of the Partner's Mux to synchronize.

The Actor Churn Detection state machine is depicted in Figure 43–15. The Partner Churn Detection state machine is depicted in Figure 43–16.

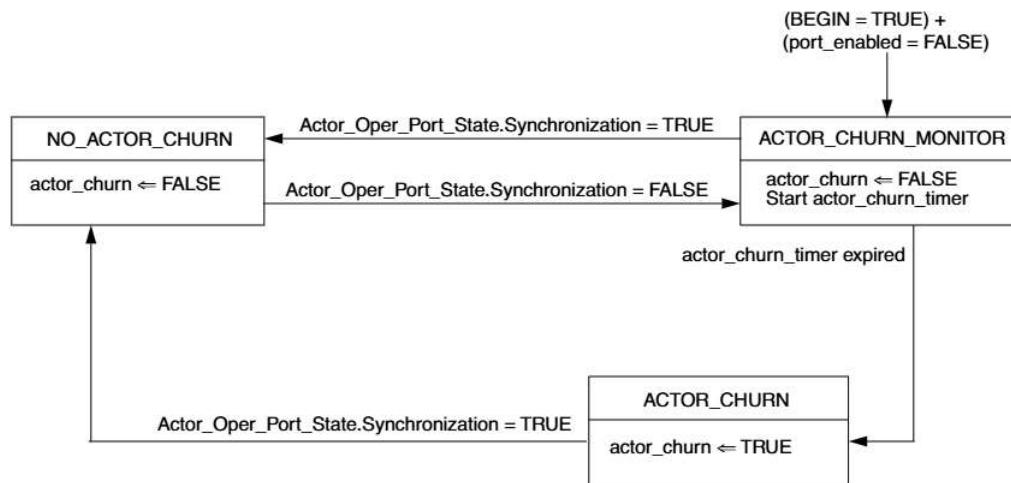


Figure 43–15—Actor Churn Detection machine state diagram

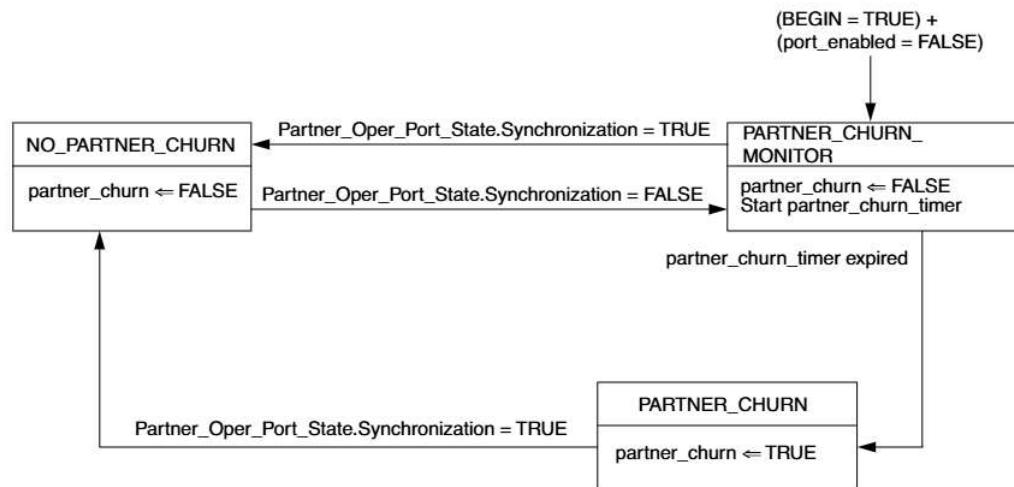


**Figure 43-16—Partner Churn Detection machine state diagram**

## 43.5 Marker protocol

### 43.5.1 Introduction

The Marker protocol allows the distribution function of an Actor's Link Aggregation sublayer to request the transmission of a Marker PDU on a given link. The Marker PDU is received by the Partner's collection function and a Marker Response PDU is returned on the same link to the initiating Actor's distribution function. Marker and Marker Response PDUs are treated by the underlying MACs at each end of the link as normal MAC Client PDUs; i.e., there is no prioritization or special treatment of Marker or Marker Response PDUs relative to other frames. Marker/Marker Response 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 Response PDU is received from that link, then it can be certain that there are no MAC Client PDUs related to those conversations still to be received by the Partner's collection function. The use of the Marker 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 mis-ordered at the collector.

NOTE—The use of the Marker protocol is further discussed in Annex 43A.

The operation of the Marker protocol is unaffected by any changes in the Collecting and Distributing states associated with the port. Therefore, Marker and Marker Response PDUs can be sent on a port whose distribution function is disabled; similarly, such PDUs can be received and passed to the relevant Aggregator's collection or distribution function on a port whose collection function is disabled.

The use of the Marker protocol is optional; however, the ability to respond to Marker PDUs, as defined for the operation of the Marker Responder (see 43.5.4.1 and 43.5.4.2), is mandatory. Some distribution algorithms may not require the use of a marker; other mechanisms (such as 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 the Marker protocol is used. (See Annex 43A for an informative discussion of distribution algorithms.).

The Marker protocol does not provide a guarantee of a response from the Partner; 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 cases.

### 43.5.2 Sequence of operations

Figure 43–17 illustrates the sequence of marker operations between an initiating and responding System. Time is assumed to flow from the top of the diagram to the bottom.

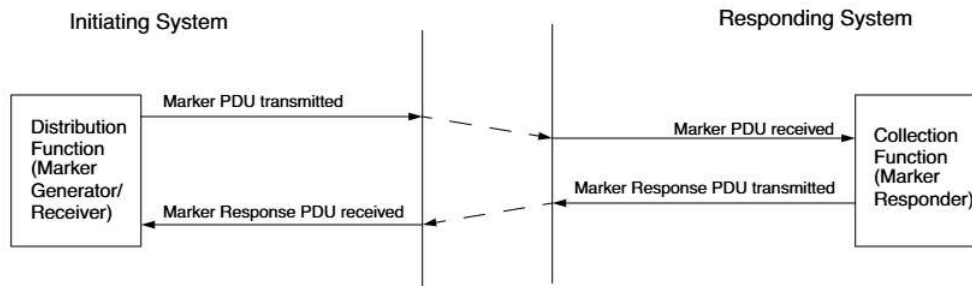


Figure 43–17—Marker protocol time sequence diagram

### 43.5.3 Marker and Marker Response PDU structure and encoding

#### 43.5.3.1 Transmission and representation of octets

All Marker and Marker Response PDUs comprise an integral number of octets. The bits in each octet are numbered from 0 to 7, where 0 is the low-order bit. When consecutive octets are used to represent a numerical value, the most significant octet is transmitted first, followed by successively less significant octets.

When the encoding of (an element of) a Marker or Marker Response PDU is depicted in a diagram, then

- Octets are transmitted from top to bottom.
- Within an octet, bits are shown with bit 0 to the left and bit 7 to the right, and are transmitted from left to right.
- Numerical values are encoded as binary numbers.

#### 43.5.3.2 Marker and Marker Response PDU structure

Marker PDUs and Marker Response PDUs are basic IEEE 802.3® frames; they shall not be tagged (see Clause 3). The Marker PDU and Marker Response PDU structure shall be as shown in Figure 43–18 and as further described in the following field definitions:

- Destination Address.* The DA in all Marker and Marker Response PDUs is the Slow\_Protocols\_Multicast address. Its use and encoding are specified in Annex 43B.
- Source Address.* The SA in Marker and Marker Response PDUs is the individual MAC address associated with the port from which the PDU is transmitted.
- Length/Type.* Marker and Marker Response PDUs are always Type encoded, and carry the Slow Protocol type field. The use and encoding of this type field is specified in Annex 43B.
- Subtype.* The Subtype field identifies the specific Slow Protocol being encapsulated. Both Marker and Marker Response PDUs carry the Marker\_subtype value 0x02.

- e) *Version number*. This identifies the Marker protocol version; implementations conformant to this version of the standard carry the value 0x01.
- f) *TLV\_type = Marker Information/Marker Response Information*. This indicates the nature of the information carried in this TLV-tuple. Marker Information is encoded as the integer value 0x01; Marker Response Information is encoded as the integer value 0x02.
- g) *Marker\_Information\_Length/Marker\_Response\_Information\_Length*. This field indicates the length (in octets) of this TLV-tuple. Both Marker and Marker Response information use a length value of 16 (0x10).
- h) *Requester\_Port*. The port number assigned to the port by the Requester (the System sending the initial Marker PDU), encoded as an unsigned integer.

| Marker PDU                    | Octets | Marker Response PDU                     |
|-------------------------------|--------|-----------------------------------------|
| Destination Address           | 6      | Destination Address                     |
| Source Address                | 6      | Source Address                          |
| Length/Type                   | 2      | Length/Type                             |
| Subtype = Marker              | 1      | Subtype = Marker                        |
| Version Number                | 1      | Version Number                          |
| TLV_type = Marker Information | 1      | TLV_type = Marker Response Information  |
| Marker_Information_Length= 16 | 1      | Marker_Response_Information_Length = 16 |
| Requester_Port                | 2      | Requester_Port                          |
| Requester_System              | 6      | Requester_System                        |
| Requester_Transaction_ID      | 4      | Requester_Transaction_ID                |
| Pad = 0                       | 2      | Pad = 0                                 |
| TLV_type = Terminator         | 1      | TLV_type = Terminator                   |
| Terminator_Length = 0         | 1      | Terminator_Length = 0                   |
| Reserved                      | 90     | Reserved                                |
| FCS                           | 4      | FCS                                     |

**Figure 43–18—Marker PDU and Marker Response PDU structure**

- i) *Requester\_System*. The Requester's System ID, encoded as a MAC address.
- j) *Requester\_Transaction\_ID*. The transaction ID allocated to this request by the requester, encoded as an integer.
- k) *Pad*. This field is used to align TLV-tuples on 16-byte memory boundaries. It is transmitted as zeroes in Marker PDUs; in Marker Response PDUs, this field may be transmitted as zeroes, or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.  
 NOTE—The difference in handling of the Pad field in Marker Response PDUs allows an implementation to reflect the contents of the received Marker PDU in its response, without enforcing the requirement to transmit the field as zeroes.
- l) *TLV\_type = Terminator*. This field indicates the nature of the information carried in this TLV-tuple. Terminator (end of message) information carries the integer value 0x00.
- m) *Terminator\_Length*. This field indicates the length (in octets) of this TLV-tuple. Terminator information uses a length value of 0 (0x00).

- n) *Reserved*. These 90 octets are reserved for use in future extensions to the protocol. It is transmitted as zeroes in Marker PDUs; in Marker Response PDUs, this field may be either transmitted as zeroes, or with the contents of this field from the Marker PDU triggering the response. The field is ignored on receipt in all cases.

NOTE—These trailing reserved octets are included in all Marker and Marker Response PDUs in order to force a fixed PDU size, regardless of the version of the protocol. Hence, a Version 1 implementation is guaranteed to be able to receive version N PDUs successfully, although version N PDUs may contain additional information that cannot be interpreted (and will be ignored) by the Version 1 implementation. A crucial factor in ensuring backwards compatibility is that any future version of the protocol is required not to re-define the structure or semantics of information defined for the previous version; it may only add new information elements to the previous set. Hence, in a version N PDU, a Version 1 implementation can expect to find the Version 1 information in exactly the same places as in a Version 1 PDU, and can expect to interpret that information as defined for Version 1.

- o) *FCS*. This field is the Frame Check Sequence, typically generated by the underlying MAC.

#### 43.5.4 Protocol definition

##### 43.5.4.1 Operation of the marker protocol

Marker PDUs may be generated by the Frame Distribution function to provide a sequence marker in the stream of frames constituting a conversation or set of conversations. Received Marker PDUs are delivered to the Marker Responder within the Frame Collection function of the Partner System.

On receipt of a valid Marker PDU, the Frame Collection function issues a Marker Response PDU, in the format specified in Figure 43–18, to the same port from which the Marker PDU was received. The **Requester\_Port**, **Requester\_System**, and **Requester\_Transaction\_ID** parameter in the Marker Response PDU carry the same values as those received in the corresponding Marker PDU.

Received Marker Response PDUs are passed to the Marker Receiver within the Frame Distribution function. Implementation of the Marker Generator and Receiver is optional.

The Marker Generator, if implemented, shall comply with the frame rate limitation constraint for Slow Protocols, as specified in Annex 43B.3. A Marker Responder may (but is not required to) control its Marker Response transmissions to conform to this Slow Protocols timing constraint when faced with Marker messages not in compliance with this constraint (i.e., to send fewer Marker Response PDUs than Marker PDUs received). If the Marker Responder is controlling its responses in this manner, Marker Response PDUs corresponding to Marker PDUs received in excess of the Slow Protocols timing constraint shall not be sent.

NOTE—It is important that Marker Response PDUs not be queued indefinitely, and sent long after the corresponding Marker PDU that triggered the response.

Frames generated by the Marker Responder do not count towards the rate limitation constraint for Slow Protocols, as specified in Annex 43B.3.

##### 43.5.4.2 Marker Responder state diagram

The Marker Responder shall implement the function specified in Figure 43–19, with its associated parameters (43.5.4.2.1 through 43.5.4.2.3).

###### 43.5.4.2.1 Constants

**Slow\_Protocols\_Multicast**

The value of the Slow Protocols reserved multicast address. (See Table 43B–1.)

#### 43.5.4.2.2 Variables

DA  
SA  
m\_sdu  
service\_class  
status

The parameters of the MA\_DATA.request and MA\_DATA.indication primitives, as defined in Clause 2.

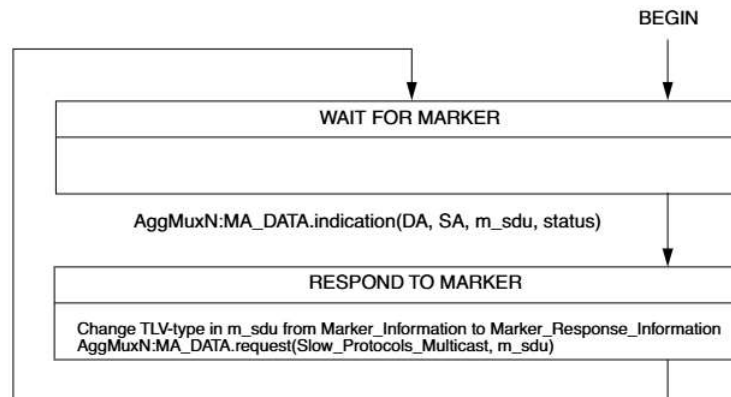
#### 43.5.4.2.3 Messages

AggMuxN:MA\_DATA.request

The service primitive used to transmit a frame with the specified parameters.

AggMuxN:MA\_DATA.indication

The service primitive used to pass a received frame to a client with the specified parameters.



The value of N (the port number) in the AggMuxN:MA\_DATA.request primitive shall be the same as that of the received AggMuxN:MA\_DATA.indication

**Figure 43–19—Marker Responder state diagram**

Upon receipt of an AggMuxN:MA\_DATA.indication primitive, the Marker Responder shall not validate the Version Number, Pad, or Reserved fields in the contained Marker Request PDU. The same actions are taken regardless of the values received in these fields. A Marker Responder may validate the Marker\_Information\_Length field. These behaviors, together with the constraint on future protocol enhancements discussed in the Note in 43.5.3.2, allow Version 1 devices to be compatible with future revisions of the protocol.

### 43.6 Configuration capabilities and restrictions

#### 43.6.1 Use of system and port priorities

The formulation chosen for the Link Aggregation Group identifier (43.3.6) has the consequence that it is not possible to represent two or more Link Aggregation Groups, comprising aggregatable links, 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 6 members, restricting the size of any aggregation to 4 members or fewer) 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 subdividing Key Groups and allocating different operational 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 (e.g., it might be possible to aggregate only those links that are on the same interface card).

In Systems that have limited aggregation capability of this form, the following algorithm shall be used to determine the subset of ports that will be aggregated together:

- a) The System Aggregation Priority of each System is an eight octet binary number, formed by using the Actor\_System\_Priority as the two most significant octets and the Actor's System ID as the least significant six octets. For a given Actor and Partner, the System with the numerically lower value of System Aggregation Priority has the higher priority.
- b) The Port Aggregation Priority of each port is a four octet binary number, formed by using the Actor\_Port\_Priority as the two most significant octets and the port number as the two least significant octets. For any given set of ports, the port with the numerically lower value of Port Aggregation Priority has the higher priority.
- c) Ports shall be selected for aggregation by each System based upon the Port Aggregation Priority assigned by the System with the higher System Aggregation Priority, starting with the highest priority port of the System with the higher priority, and working downward through the ordered list of Port Aggregation Priority values for the N ports, applying the particular constraints imposed on the System concerned.
- d) For each link that a given System cannot include in the aggregation, the Selection Logic identifies the Selection state of the corresponding port as STANDBY, preventing the link from becoming active. The synchronization state signalled in transmitted LACPDUs for such links will be OUT\_OF\_SYNC.
- e) The selection algorithm is reapplied upon changes in the membership of the Link Aggregation Group (for example, if a link fails, or if a new link joins the group) and any consequent changes to the set of active links are made accordingly.

A port that is selected as standby as a result of limitations on aggregation capability can be viewed as providing a "hot standby" facility, as it will be able to take part in the aggregation upon failure of one of the active links in the aggregation. The ability to hold links in a standby mode in this way provides the possibility of using LACP even where the System is incapable of supporting distribution and collection with more than one port. Parallel links could be automatically configured as standby links, and deployed to mask link failures without any disruption to higher layer protocols.

#### 43.6.2 Dynamic allocation of operational Keys

In some circumstances, the use of System and port priorities may prove to be insufficient to generate the optimum aggregation among the set of links connecting a pair of Systems. A System may have a limited aggregation capability that cannot be simply expressed as a limit on the total number of links in the aggregation. The full description of its restrictions may be that it can only aggregate together particular subsets of links, and the sizes of the subsets need not all be the same.

NOTE—An example would be an implementation organized such that, for a set of four links A through D, it would be possible to operate with {A+B+C+D} as a single aggregation, or operate with {A+B} and {C+D} as two separate aggregations, or operate as four individual links; however, all other aggregation possibilities (such as {A+C} and {B+D}) would not be achievable by the implementation.

In such circumstances, it is permissible for the System with the higher System Aggregation Priority (i.e., the numerically lower value) to dynamically modify the operational Key value associated with one or more of

the ports; the System with the lower priority shall not attempt to modify operational Key values for this purpose. Operational Key changes made by the higher priority System should be consistent with maintaining its highest priority port in the aggregate as an active link (i.e., in the IN\_SYNC state). Successive operational Key changes, if they occur, should progressively reduce the number of ports in the aggregation. The original operational Key value should be maintained for the highest priority port thought to be aggregatable.

NOTE—Restricting operational Key changes in the manner described prevents the case where both Partner Systems involved have limited capability and both attempt to make operational Key changes; this could be a non-converging process, as a change by one participant can cause the other participant to make a change, which in turn causes the first participant to make a change—and so on, ad infinitum.

This approach effectively gives the higher priority System permission to search the set of possible configurations, in order to find the best combination of links given its own and its Partner's configuration constraints. The reaction of the Partner System to these changes can be determined by observing the changes in the synchronization state of each link. A System performing operational Key changes should allow at least 4 seconds for the Partner System to change an OUT\_OF\_SYNC state to an IN\_SYNC state.

In the course of normal operation a port can dynamically change its operating characteristics (e.g., data rate, full or half duplex operation). It is permissible (and appropriate) for the operational Key value associated with such a port to change with the corresponding changes in the operating characteristics of the link, so that the operational Key value always correctly reflects the aggregation capability of the link. Operational Key changes that reflect such dynamic changes in the operating characteristics of a link may be made by either System without restriction.

#### **43.6.3 Link Aggregation on shared-medium links**

The Link Aggregation Control Protocol cannot detect the presence of multiple Aggregation-aware devices on the same link. Hence, shared-medium links shall be treated as Individual, with transmission/reception of LACPDUs disabled on such ports.

#### **43.6.4 Selection Logic variants**

Two variants of the Selection Logic rules are described as follows:

- a) The first accommodates implementations that may wish to operate in a manner that minimizes disturbance of existing aggregates, at the expense of the deterministic characteristics of the logic described in 43.4.14.2.
- 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 ports supported.

##### **43.6.4.1 Reduced reconfiguration**

By removing the constraint that the Aggregator chosen is always the lowest numbered Aggregator associated with the set of ports in an aggregation, an implementation can minimize the degree to which changes in the membership of a given aggregation result in changes of connectivity at higher layers. As there would still be the same number of Aggregators and ports with a given operational Key value, any port will still always be able to find an appropriate Aggregator to attach to, however the configuration achieved over time (i.e., after a series of link disconnections, reconnections, or reconfigurations) with this relaxed set of rules would not necessarily be the same as the configuration achieved if all Systems involved were reset, given the rules stated in 43.4.15.

#### 43.6.4.2 Limited Aggregator availability

By removing the constraint that there are always as many Aggregators as ports, an implementation can limit the number of MAC Client interfaces available to higher layers while maintaining the ability for each Aggregator to serve multiple ports. This has the same effect as removing the assumption that Aggregators and their associated ports have the same operational Key value; Aggregators can be effectively disabled (and therefore ignored) by configuring their Keys to be different from any operational Key value allocated to any of the ports.

In this scenario, any port(s) that cannot find a suitable Aggregator to attach to will simply wait in the DETACHED state until an Aggregator becomes available, with a synchronization state of OUT\_OF\_SYNC.

### 43.7 Protocol Implementation Conformance Statement (PICS) proforma for Clause 43, Aggregation of Multiple Link Segments<sup>13</sup>

#### 43.7.1 Introduction

The supplier of an implementation that is claimed to conform to Clause 43, Link Aggregation, shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

#### 43.7.2 Identification

##### 43.7.2.1 Implementation identification

|                                                                                                                                                                                                                                                                                           |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Supplier (Note 1)                                                                                                                                                                                                                                                                         |  |
| Contact point for queries about the PICS (Note 1)                                                                                                                                                                                                                                         |  |
| Implementation Name(s) and Version(s) (Notes 1 and 3)                                                                                                                                                                                                                                     |  |
| Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names (Note 2)                                                                                                                                                       |  |
| <b>NOTES</b><br>1—Required for all implementations.<br>2—May be completed as appropriate in meeting the requirements for the identification.<br>3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model). |  |

<sup>13</sup>Copyright release for PICS proformas: Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

**43.7.2.2 Protocol summary**

|                                                                                                                                                                                                                                          |                                                    |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------|
| Identification of protocol specification                                                                                                                                                                                                 | IEEE Std 802.3-2002®, Clause 43, Link Aggregation. |
| Identification of amendments and corrigenda to the PICS proforma which have been completed as part of the PICS                                                                                                                           |                                                    |
| Have any Exception items been required? No <input type="checkbox"/> Yes <input type="checkbox"/><br>(See Clause 21: the answer Yes means that the implementation does not conform to IEEE Std 802.3-2002®, Clause 43, Link Aggregation.) |                                                    |

|                   |  |
|-------------------|--|
| Date of Statement |  |
|-------------------|--|

**43.7.3 Major capabilities/options**

| Item | Feature                                            | Subclause | Value/Comment                                                    | Status          | Support                                                                                                                        |
|------|----------------------------------------------------|-----------|------------------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------|
| *MG  | Marker Generator/Receiver                          | 43.2.5    |                                                                  | O               | Yes <input type="checkbox"/><br>No <input type="checkbox"/>                                                                    |
| *AM  | Aggregation Port Debug Information package support | 30.7      |                                                                  | O               | Yes <input type="checkbox"/><br>No <input type="checkbox"/>                                                                    |
| *CM  | Churn Detection machine                            | 43.4.17   | Required if Aggregation Port Debug Information package supported | AM: M<br>!AM: O | N/A <input type="checkbox"/><br>M: Yes <input type="checkbox"/><br>Yes <input type="checkbox"/><br>No <input type="checkbox"/> |

**43.7.4 Frame Collector**

| Item | Feature                                    | Subclause  | Value/Comment                                                                                 | Status | Support                      |
|------|--------------------------------------------|------------|-----------------------------------------------------------------------------------------------|--------|------------------------------|
| FC1  | Frame Collector function                   | 43.2.3     | As specified in the state diagram shown in Figure 43-3 and associated definitions in 43.2.3.1 | M      | Yes <input type="checkbox"/> |
| FC2  | Frame Collector function—CollectorMaxDelay | 43.2.3.1.4 | Deliver or discard frames within CollectorMaxDelay                                            | M      | Yes <input type="checkbox"/> |

**43.7.5 Frame Distributor**

| Item | Feature                                                                                | Subclause | Value/Comment                                                                                          | Status | Support |
|------|----------------------------------------------------------------------------------------|-----------|--------------------------------------------------------------------------------------------------------|--------|---------|
| FD1  | Distribution algorithm ensures the following, when frames received by Frame Collector: | 43.2.4    |                                                                                                        |        |         |
|      | Frame mis-ordering                                                                     |           | None                                                                                                   | M      | Yes [ ] |
| FD2  | Frame duplication                                                                      |           | None                                                                                                   | M      | Yes [ ] |
| FD3  | Frame Distributor function                                                             | 43.2.4    | Function as specified in the state diagram shown in Figure 43-4 and associated definitions in 43.2.4.1 | M      | Yes [ ] |

**43.7.6 Marker protocol**

| Item | Feature                   | Subclause | Value/Comment                  | Status | Support              |
|------|---------------------------|-----------|--------------------------------|--------|----------------------|
| MGR1 | Marker Generator/Receiver | 43.2.5    |                                | MG:M   | N/A [ ]<br>M:Yes [ ] |
| MGR2 | Marker Responder          | 43.2.6    | Function specified in 43.5.4.2 | M      | Yes [ ]              |

**43.7.7 Aggregator Parser/Multiplexer**

| Item | Feature                | Subclause | Value/Comment                                                                                    | Status | Support |
|------|------------------------|-----------|--------------------------------------------------------------------------------------------------|--------|---------|
| APM1 | Aggregator Multiplexer | 43.2.7    | Transparent pass-through of frames                                                               | M      | Yes [ ] |
| APM2 | Aggregator Multiplexer | 43.2.7    | Discard of TX frames when port not Distributing                                                  | M      | Yes [ ] |
| APM3 | Aggregator Parser      | 43.2.7    | Function specified by state diagram shown in Figure 43-5 and associated definitions in 43.2.7.1. | M      | Yes [ ] |
| APM4 | Aggregator Parser      | 43.2.7    | Discard of RX frames when port not Collecting                                                    | M      | Yes [ ] |

**43.7.8 Control Parser/Multiplexer**

| Item | Feature             | Subclause | Value/Comment                                                                                    | Status | Support |
|------|---------------------|-----------|--------------------------------------------------------------------------------------------------|--------|---------|
| CPM1 | Control Multiplexer | 43.2.9    | Transparent pass-through of frames                                                               | M      | Yes [ ] |
| CPM2 | Control Parser      | 43.2.9    | Function specified by state diagram shown in Figure 43-6 and associated definitions in 43.2.9.1. | M      | Yes [ ] |

**43.7.9 System identification**

| Item | Feature                    | Subclause | Value/Comment                                                     | Status | Support           |
|------|----------------------------|-----------|-------------------------------------------------------------------|--------|-------------------|
| SID1 | Globally unique identifier | 43.3.2    | Globally administered individual MAC address plus System Priority | M      | Yes [ ]           |
| SID2 | MAC address chosen         | 43.3.2    | MAC address associated with one of the ports                      | O      | Yes [ ]<br>No [ ] |

**43.7.10 Aggregator identification**

| Item  | Feature                     | Subclause | Value/Comment                                          | Status      | Support              |
|-------|-----------------------------|-----------|--------------------------------------------------------|-------------|----------------------|
| AID1  | Globally unique identifier  | 43.3.3    | Globally administered individual MAC address           | M           | Yes [ ]              |
| AID2  | Integer identifier          | 43.3.3    | Uniquely identifies the Aggregator within the System   | M           | Yes [ ]              |
| *AID3 | Unique identifier allocated | 43.3.3    | Unique identifier assigned to one of its bound ports   | O           | Yes [ ]<br>No [ ]    |
| AID4  |                             |           | Unique identifier not assigned to any other Aggregator | !AID3<br>:M | N/A [ ]<br>M:Yes [ ] |

**43.7.11 Port identification**

| Item | Feature          | Subclause | Value/Comment                                               | Status | Support |
|------|------------------|-----------|-------------------------------------------------------------|--------|---------|
| PID1 | Port identifiers | 43.3.4    | Unique within a System; port number 0 not used for any port | M      | Yes [ ] |

**43.7.12 Capability identification**

| Item | Feature                                                                   | Subclause | Value/Comment | Status | Support |
|------|---------------------------------------------------------------------------|-----------|---------------|--------|---------|
| CID1 | Administrative and operational Key values associated with each port       | 43.3.5    |               | M      | Yes [ ] |
| CID2 | Administrative and operational Key values associated with each Aggregator | 43.3.5    |               | M      | Yes [ ] |

**43.7.13 Link Aggregation Group identification**

| Item | Feature                 | Subclause | Value/Comment                                                                  | Status | Support |
|------|-------------------------|-----------|--------------------------------------------------------------------------------|--------|---------|
| LAG1 | LAG ID component values | 43.3.6.1  | Actor's values non-zero. Partner's admin values only zero for Individual ports | M      | Yes [ ] |

**43.7.14 Detaching a link from an Aggregator**

| Item | Feature                                                | Subclause | Value/Comment            | Status | Support |
|------|--------------------------------------------------------|-----------|--------------------------|--------|---------|
| DLA1 | Effect on conversation reallocated to a different link | 43.3.13   | Frame ordering preserved | M      | Yes [ ] |

**43.7.15 LACPDU structure**

| Item | Feature             | Subclause | Value/Comment                                                  | Status | Support |
|------|---------------------|-----------|----------------------------------------------------------------|--------|---------|
| LPS1 | LACPDU Frame format | 43.4.2.2  | Not Tagged                                                     | M      | Yes [ ] |
| LPS2 | LACPDU structure    | 43.4.2.2  | As shown in Figure 43-7 and as described                       | M      | Yes [ ] |
| LPS3 | LACPDU structure    | 43.4.2    | All Reserved octets ignored on receipt and transmitted as zero | M      | Yes [ ] |

**43.7.16 State machine variables**

| Item | Feature                  | Subclause | Value/Comment                                       | Status | Support |
|------|--------------------------|-----------|-----------------------------------------------------|--------|---------|
| SMV1 | Partner_Admin_Port_State | 43.4.7    | Collecting set to the same value as Synchronization | M      | Yes [ ] |
| SMV2 | LACP_Enabled             | 43.4.8    | FALSE for half duplex ports, otherwise TRUE         | M      | Yes [ ] |

**43.7.17 Receive machine**

| Item | Feature               | Subclause | Value/Comment                                                 | Status | Support |
|------|-----------------------|-----------|---------------------------------------------------------------|--------|---------|
| RM1  | Receive machine       | 43.4.12   | As defined in Figure 43–10 and associated parameters          | M      | Yes [ ] |
| RM2  | Validation of LACPDUs |           | No validation of Version Number, TLV_type, or Reserved fields | M      | Yes [ ] |

**43.7.18 Periodic Transmission machine**

| Item | Feature                       | Subclause | Value/Comment                                        | Status | Support |
|------|-------------------------------|-----------|------------------------------------------------------|--------|---------|
| PM1  | Periodic Transmission machine | 43.4.13   | As defined in Figure 43–11 and associated parameters | M      | Yes [ ] |

**43.7.19 Selection Logic**

| Item  | Feature                                                   | Subclause | Value/Comment                                            | Status | Support           |
|-------|-----------------------------------------------------------|-----------|----------------------------------------------------------|--------|-------------------|
| SLM1  | Selection logic requirements                              | 43.4.14.1 | At least one Aggregator per System                       | M      | Yes [ ]           |
| SLM2  | Aggregator support                                        |           | Each port assigned an operational Key                    | M      | Yes [ ]           |
| SLM3  | Port Keys                                                 |           | Each Aggregator assigned an operational Key              | M      | Yes [ ]           |
| SLM4  | Aggregator Keys                                           |           | Each Aggregator assigned an identifier                   | M      | Yes [ ]           |
| SLM5  | Aggregator Identifiers                                    |           | If same Key assignment as port                           | M      | Yes [ ]           |
| SLM6  | Aggregator selection                                      |           | Ports select same Aggregator                             | M      | Yes [ ]           |
| SLM7  | Ports that are members of the same Link Aggregation Group |           | Not select same Aggregator as each other                 | M      | Yes [ ]           |
| SLM8  | Pair of ports connected in loop-back                      |           | Not select same Aggregator as any other port             | M      | Yes [ ]           |
| SLM9  | Port required to be Individual                            |           | Not select same Aggregator as any Individual port        | M      | Yes [ ]           |
| SLM10 | Port is Aggregatable                                      |           | Port not attached to any Aggregator                      | M      | Yes [ ]           |
| SLM11 | Port unable to select an Aggregator                       |           | Ports may be selected as standby                         | O      | Yes [ ]<br>No [ ] |
| SLM12 | Further aggregation constraints                           |           | Set to SELECTED or STANDBY once Aggregator is determined | M      | Yes [ ]           |
| SLM13 | Selected variable                                         |           | Only when selected and attached to an Aggregator         | M      | Yes [ ]           |
| SLM14 | Port enabled                                              |           |                                                          |        |                   |
| SLM14 | Recommended default operation of Selection Logic          | 43.4.14.2 | Meets requirements of 43.4.14.2                          | O      | Yes [ ]<br>No [ ] |

**43.7.20 Mux machine**

| Item | Feature     | Subclause | Value/Comment                                                         | Status | Support |
|------|-------------|-----------|-----------------------------------------------------------------------|--------|---------|
| XM1  | Mux machine | 43.4.15   | As defined in Figure 43–13 or Figure 43–14, and associated parameters | M      | Yes [ ] |

**43.7.21 Transmit machine**

| Item | Feature                                                                       | Subclause | Value/Comment                                          | Status | Support |
|------|-------------------------------------------------------------------------------|-----------|--------------------------------------------------------|--------|---------|
| TM1  | Transmitted in outgoing LACP-DUs<br>Actor_Port and Actor_Port_Priority        | 43.4.16   |                                                        | M      | Yes [ ] |
| TM2  | Actor_System and Actor_System_Priority                                        |           |                                                        | M      | Yes [ ] |
| TM3  | Actor_Key                                                                     |           |                                                        | M      | Yes [ ] |
| TM4  | Actor_State                                                                   |           |                                                        | M      | Yes [ ] |
| TM5  | Partner_Port and Partner_Port_Priority                                        |           |                                                        | M      | Yes [ ] |
| TM6  | Partner_System and Partner_System_Priority                                    |           |                                                        | M      | Yes [ ] |
| TM7  | Partner_Key                                                                   |           |                                                        | M      | Yes [ ] |
| TM8  | Partner_State                                                                 |           |                                                        | M      | Yes [ ] |
| TM9  | CollectorMaxDelay                                                             |           |                                                        | M      | Yes [ ] |
| TM10 | Action when Periodic machine is in the NO_PERIODIC state                      | 43.4.16   | Set NTT to FALSE, do not transmit                      | M      | Yes [ ] |
| TM11 | Action when LACP_Enabled is TRUE, NTT is TRUE, and not rate limited           | 43.4.16   | Properly formatted LACPDU transmitted                  | M      | Yes [ ] |
| TM12 | Action when LACP_Enabled is TRUE and NTT is TRUE, when rate limit is in force | 43.4.16   | Transmission delayed until limit is no longer in force | M      | Yes [ ] |
| TM13 | Action when LACPDU has been transmitted                                       | 43.4.16   | Set NTT to FALSE                                       | M      | Yes [ ] |
| TM14 | Action when LACP_Enabled is FALSE                                             | 43.4.16   | Set NTT to FALSE, do not transmit                      | M      | Yes [ ] |

**43.7.22 Churn Detection machines**

| Item | Feature                  | Subclause | Value/Comment                               | Status | Support            |
|------|--------------------------|-----------|---------------------------------------------|--------|--------------------|
| CM1  | Churn Detection machines | 43.4.17   | As defined in Figure 43–15 and Figure 43–16 | CM:M   | N/A [ ]<br>Yes [ ] |

**43.7.23 Marker protocol**

| Item | Feature                                        | Subclause  | Value/Comment                                                                   | Status | Support            |
|------|------------------------------------------------|------------|---------------------------------------------------------------------------------|--------|--------------------|
| FP1  | Respond to all received Marker PDUs            | 43.5.1     | As specified by 43.5.4                                                          | M      | Yes [ ]            |
| FP2  | Use of the Marker protocol                     | 43.5.1     | As specified by 43.5.4                                                          | O      | Yes [ ]<br>No [ ]  |
| FP3  | MARKER.request service primitives request rate | 43.5.4.1   | Maximum of five during any one-second period                                    | MG:M   | N/A [ ]<br>Yes [ ] |
| FP4  | Marker PDU Frame format                        | 43.5.3.2   | Not Tagged                                                                      | MG:M   | N/A [ ]<br>Yes [ ] |
| FP5  | Marker Response PDU Frame format               | 43.5.3.2   | Not Tagged                                                                      | M      | Yes [ ]            |
| FP6  | Marker PDU structure                           | 43.5.3.2   | As shown in Figure 43–18 and as described                                       | MG:M   | N/A [ ]<br>Yes [ ] |
| FP7  | Marker Response PDU structure                  | 43.5.3.2   | As shown in Figure 43–18 and as described                                       | M      | Yes [ ]            |
| FP8  | Marker Responder State Diagram                 | 43.5.4.2   | As specified in Figure 43–19 and 43.5.4.2.1 through 43.5.4.2.3                  | M      | Yes [ ]            |
| FP9  | Validation of Marker Request PDUs              | 43.5.4.2.3 | Marker Responder shall not validate the Version Number, Pad, or Reserved fields | M      | Yes [ ]            |

**43.7.24 Configuration capabilities and restrictions**

| Item | Feature                                                                                                                 | Subclause | Value/Comment                            | Status | Support              |
|------|-------------------------------------------------------------------------------------------------------------------------|-----------|------------------------------------------|--------|----------------------|
| CCR1 | Algorithm used to determine subset of ports that will be aggregated in Systems that have limited aggregation capability | 43.6.1    | As specified in items a) to e) of 43.6.1 | M      | Yes [ ]              |
| CCR2 | Key value modification to generate optimum aggregation                                                                  | 43.6.2    |                                          | O      | Yes [ ]<br>No [ ]    |
| CCR3 | Key value modification when System has higher System Aggregation Priority                                               |           |                                          | CCR2:M | N/A [ ]<br>M:Yes [ ] |
| CCR4 | Key value modification when System has lower System Aggregation Priority                                                |           |                                          | CCR2:X | N/A [ ]<br>X:Yes [ ] |

**43.7.25 Link Aggregation on shared-medium links**

| Item | Feature                                   | Subclause | Value/Comment                     | Status | Support |
|------|-------------------------------------------|-----------|-----------------------------------|--------|---------|
| LSM1 | Shared-medium links—<br>Configuration     | 43.6.3    | Configured as<br>Individual links | M      | Yes [ ] |
| LSM2 | Shared-medium links—<br>Operation of LACP | 43.6.3    | LACP is disabled                  | M      | Yes [ ] |



NOTE—The derivation of this pattern may be found in *Fibre Channel Jitter Working Group Technical Report* [B36]. This technical report modified the original RPAT as defined by Fibre Channel so that it would maintain its intended qualities but fit into a Fibre Channel frame. This annex uses similar modifications to fit the same RPAT into an 802.3® frame.

The long continuous random test pattern consists of a continuous stream of identical packets, separated by a minimum IPG. Each packet is encapsulated within SPD and EPD delimiters as specified in Clause 36 in the ordinary way. The contents of each packet is composed of the following octet sequences, as observed at the GMII, before 8B/10B coding.

Each packet in the long continuous random test pattern consists of 8 octets of PREAMBLE/SFD, followed by 1512 data octets (126 repetitions of the 12-octet modified RPAT sequence), plus 4 CRC octets, followed by a minimum IPG of 12 octets of IDLE.

PREAMBLE/SFD:

55 55 55 55 55 55 55 D5

MODIFIED RPAT SEQUENCE (LOOP 126 TIMES)

BE D7 23 47 6B 8F B3 14 5E FB 35 59

CRC

94 D2 54 AC

IPG (TX\_EN and TX\_ER low)

00 00 00 00 00 00 00 00 00 00 00 00

END

### 36A.5 Short continuous random test pattern

The short continuous random test pattern is a random test pattern intended to provide broad spectral content and minimal peaking that can be used for the measurement of jitter at either a component or system level.

NOTE—The derivation of this pattern may be found in *Fibre Channel Jitter Working Group Technical Report* [B36]. This technical report modified the original RPAT as defined by Fibre Channel so that it would maintain its intended qualities but fit into a Fibre Channel frame. This annex uses similar modifications to fit the same RPAT into an 802.3® frame.

The short continuous random test pattern consists of a continuous stream of identical packets, separated by a minimum IPG. Each packet is encapsulated within SPD and EPD delimiters as specified in Clause 36 in the ordinary way. The contents of each packet is composed of the following octet sequences, as observed at the GMII, before 8B/10B coding.

Each packet in the short continuous random test pattern consists of 8 octets of PREAMBLE/SFD, followed by 348 data octets (29 repetitions of the 12-octet modified RPAT sequence), plus 4 CRC octets, followed by a minimum IPG of 12 octets of IDLE.

The format of this packet is such that the PCS will generate the following ordered sets for the IPG: / T/ /R/ /11/ /12/ /12/ /12/ /12/

PREAMBLE/SFD:

55 55 55 55 55 55 55 D5

MODIFIED RPAT SEQUENCE (LOOP 29 TIMES)

BE D7 23 47 6B 8F B3 14 5E FB 35 59

CRC

2F E0 AA EF

IPG (TX\_EN and TX\_ER low)

00 00 00 00 00 00 00 00 00 00 00 00

END

## Annex 36B

(informative)

### 8B/10B transmission code running disparity calculation examples

Detection of a invalid code-group in the 8B/10B transmission code does not necessarily indicate that the code-group in which the error was detected was the one in which the error occurred. Invalid code-groups may result from a prior error that altered the running disparity of the bit stream but that did not result in a detectable error at the code-group in which the error occurred. The examples shown in Tables 36B-1 and 36B-2 exhibit this behavior. The example shown in Table 36B-3 exhibits the case where a bit error in a received code-group is detected in that code-group, affects the next code group, and error propagation is halted upon detection of the running disparity error.

**Table 36B-1—RD error detected two code-groups following error**

| Stream                 | Code-group |        | Code-group |                   | Code-group |        |
|------------------------|------------|--------|------------|-------------------|------------|--------|
|                        | RD         | RD     | RD         | RD                | RD         | RD     |
| Transmitted code-group | –          | D21.1  | –          | D10.2             | –          | D23.5  |
| Transmitted bit stream | –          | 101010 | –          | 1001              | –          | 010101 |
| Received bit stream    | –          | 101010 | –          | 1011 <sup>a</sup> | –          | 010101 |
| Received code-group    | –          | D21.0  | –          | D10.2             | –          | D23.5  |

<sup>a</sup>Bit error introduced (1001  $\Rightarrow$  1011)

<sup>b</sup>Nonzero disparity blocks must alternate in polarity (+  $\Rightarrow$  –). In this case, RD does not alternate (+  $\Rightarrow$  +), the received code group is not found in the Current RD+ column in either Table 36-1a or Table 36-2, and an invalid code-group is recognized.

<sup>c</sup>Running disparity is calculated on the received code-group regardless of the validity of the received code-group. Nonzero disparity blocks prevent the propagation of errors and normalize running disparity to the transmitted bit stream (i.e., equivalent to the received bit stream had an error not occurred).

**Table 36B-2—RD error detected in next code-group following error**

| Stream                 | Code-group |        | Code-group |                   | Code-group |        |
|------------------------|------------|--------|------------|-------------------|------------|--------|
|                        | RD         | RD     | RD         | RD                | RD         | RD     |
| Transmitted code-group | –          | D21.1  | –          | D23.4             | –          | D23.5  |
| Transmitted bit stream | –          | 101010 | –          | 1001              | –          | 111010 |
| Received bit stream    | –          | 101010 | –          | 1011 <sup>a</sup> | –          | 111010 |
| Received code-group    | –          | D21.0  | –          | D23.4             | –          | D23.5  |

<sup>a</sup>Bit error introduced (1001  $\Rightarrow$  1011)

<sup>b</sup>Nonzero disparity blocks must alternate in polarity (+  $\Rightarrow$  –).

**Table 36B–3—A single bit error affects two received code-groups**

| Stream                 | Code-group |                                 | RD | Code-group                      |    | RD          | Code-group |      | RD |
|------------------------|------------|---------------------------------|----|---------------------------------|----|-------------|------------|------|----|
|                        | RD         | RD                              |    | RD                              | RD |             | RD         | RD   |    |
| Transmitted code-group | –          | D3.6 (FCS3)                     | –  | K29.7 (/T/)                     | –  | K23.7 (/R/) | –          |      |    |
| Transmitted bit stream | –          | 110001                          | –  | 0110                            | –  | 101110      | +          | 1000 | –  |
| Received bit stream    | –          | 110001                          | –  | 0111 <sup>a</sup>               | +  | 101110      | +          | 1000 | –  |
| Received code-group    | –          | invalid code-group <sup>d</sup> | +  | invalid code-group <sup>c</sup> | –  | K23.7 (/R/) | –          |      |    |

<sup>a</sup>Bit error introduced (0110 ⇒ 0111).<sup>b</sup>Nonzero disparity blocks must alternate in polarity (– ⇒ +). Received RD differs from transmitted RD.<sup>c</sup>Nonzero disparity blocks must alternate in polarity (+ ⇒ –). Invalid code-group due to RD error since RD remains at +.<sup>d</sup>Received code-group is not found in either Table 36–1a or Table 36–2.<sup>e</sup>Nonzero disparity blocks prevent the propagation of errors and normalize running disparity to the transmitted bit stream (i.e. equivalent to the received bit stream had an error not occurred). All code-groups contained in PCS End\_of\_Packet delimiters (/T/R/R or /T/R/K28.5/) include nonzero disparity blocks.

## **Annex 38A**

(informative)

### **Fiber launch conditions**

#### **38A.1 Overfilled Launch**

Overfilled Launch (OFL), as described in IEC 60793-1-4 [B24], is the standard launch used to define optical fiber bandwidth. This launch uniformly overfills the fiber both angularly and spatially. It excites both radial and azimuthal modes of the fiber equally, thus providing a reproducible bandwidth which is insensitive to small misalignments of the input fiber. It is also relatively insensitive to microbending and macrobending when they are not sufficient to affect power distribution carried by the fiber. A restricted launch gives a less reproducible bandwidth number and is dependent on an exact definition of the launch. Overfilled launch is commonly used to measure the bandwidth of LED-based links.

#### **38A.2 Radial Overfilled Launch (ROFL)**

A Radial Overfilled Launch is created when a multimode optical fiber is illuminated by the coherent optical output of a source operating in its lowest order transverse mode in a manner that excites predominantly the radial modes of the multimode fiber. This contrasts with the OFL, which is intended to excite both radial and azimuthal modes of the fiber equally. In practice an ROFL is obtained when

- a) A spot of laser light is projected onto the core of the multimode fiber,
- b) The laser spot is approximately symmetrical about the optical center of the multimode fiber,
- c) The optical axis of both the fiber and the laser beam are approximately aligned,
- d) The angle of divergence of the laser beam is less than the numerical aperture of the multimode fiber,
- e) The laser spot is larger than the core of the multimode fiber.

An ROFL cannot be created using a multi-transverse mode laser or by simply projecting a speckle pattern through an aperture.

## Annex 40A

(informative)

### Additional cabling design guidelines

This annex provides additional cabling guidelines when installing a new Category 5 balanced cabling system or using an existing Category 5 balanced cabling system. These guidelines are intended to supplement those in Clause 40. 1000BASE-T is designed to operate over 4-pair unshielded twisted-pair cabling systems that meet both the Category 5 requirements described in ANSI/TIA/EIA-568A (1995) and ISO/IEC 11801:1995, and the additional transmission parameters of return loss, ELFEXT loss, and MDLFXEXT loss specified in 40.7. There are additional steps that may be taken by network designers that provide additional operating margins and ensure the objective BER of  $10^{-10}$  is achieved. Cabling systems that meet or exceed the specifications in 40.7 for a worst case 4-connector topology are recommended for new installations. Whether installing a new Category 5 balanced cabling system or reusing one that is already installed, it is *highly recommended* that the cabling system be measured/certified before connecting 1000BASE-T equipment following the guidelines in (proposed) ANSI/TIA/EIA TSB95.

#### 40A.1 Alien crosstalk

##### 40A.1.1 Multipair cabling (i.e., greater than 4-pair)

Multiple Gigabit Ethernet links [(n\*4-Pair) with n greater than 1] should not share a common sheath as in a 25-pair binder group in a multipair cable. When the multipair cable is terminated into compliant connecting hardware (TIA does not specify 25 position connecting hardware), the NEXT loss contributions between the adjacent 4-pair gigabit ethernet link, from connecting hardware and the cable combined, cannot be completely cancelled.

##### 40A.1.2 Bundled or hybrid cable configurations

Another source of alien crosstalk can occur in a bundled or hybrid cable configuration where two or more 4-pair cables are assembled together.

In order to limit the noise coupled between adjacent 1000BASE-T link segments in a bundled or hybrid cable configuration, the PSNEXT loss between a 1000BASE-T duplex channel in a link segment and all duplex channels in adjacent 1000BASE-T link segments should be greater than  $35 - 15 \cdot \log(f/100)$  (dB) at all frequencies from 1 MHz to 100 MHz.

#### 40A.2 Cabling configurations

The primary application for the Clause 40 specification is expected to be between a workstation and the local telecommunications closet. In commercial buildings this application is generally referred to as the horizontal cabling subsystem. As specified in ANSI/TIA/EIA-568-A and ISO/IEC 11801: 1995 the maximum length of a horizontal subsystem building wiring channel is 100 m. The channel consists of cords, cables, and connecting hardware. The maximum configuration for this channel is shown in Figure 40A-1.

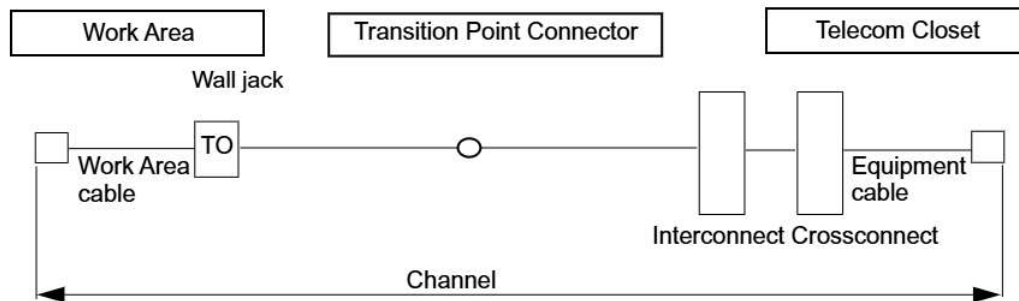


**Figure 40A-1 — Maximum horizontal subsystem configuration**

On the other hand, a minimum configuration can be achieved by removing the patch cord and transition point, which is shown in Figure 40A-2.

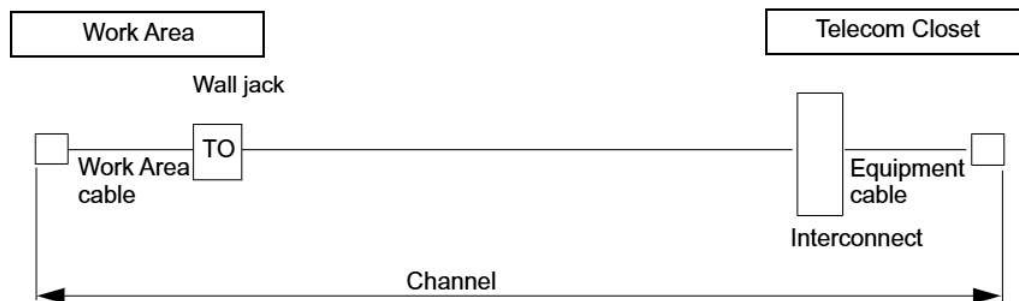


**Figure 40A-2 — Minimum horizontal subsystem configuration**

1000BASE-T is designed to operate over a channel that meets the specifications of 40.7 and the channel configuration shown in Figure 40A-1. However, if the channel specifications of 40.8 cannot be met when using the channel configuration shown in Figure 40A-1, the configuration shown in Figure 40A-2 is recommended. This optimized channel for a 1000BASE-T link segment deletes the transition point and runs an equipment patch cord directly between the LAN equipment and the connector termination of the permanent link. This reduces the number of connectors and their associated crosstalk in the link. The minimum link configuration:

- a) Minimizes crosstalk, both near-end and far-end, which maximizes the BER margin; and
- b) Minimizes link insertion loss.

## Annex 40B

(informative)

### Description of cable clamp

This annex describes the cable clamp used in the common-mode noise rejection test of 40.6.1.3.3, which is used to determine the sensitivity of the 1000BASE-T receiver to common-mode noise from the link segment. As shown in Figure 40B-1, the clamp is 300 mm long, 58 mm wide, 54 mm high with a center opening of 6.35 mm (0.25 in). The clamp consists of two halves that permit the insertion of a cable into the clamp.

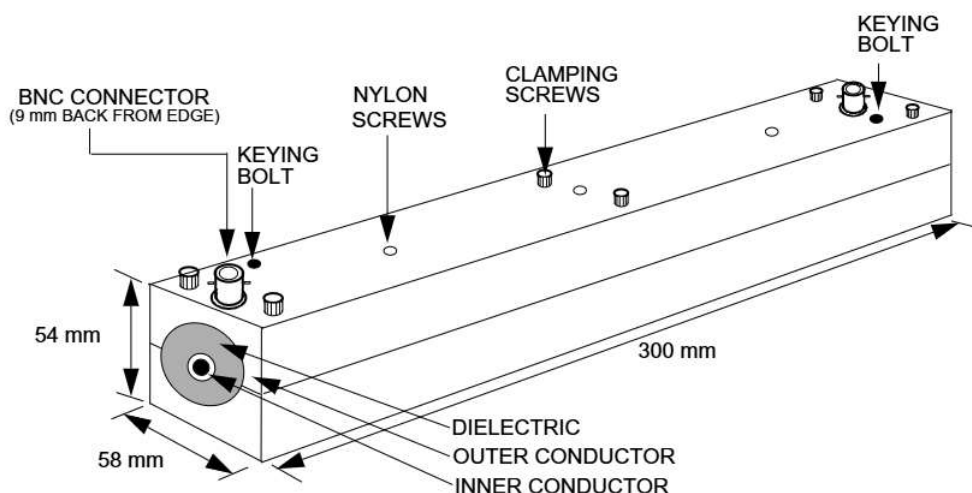


Figure 40B-1—Cable clamp

The clamp has a copper center conductor and an aluminum outer conductor with a high density polyethylene dielectric. The following is a review of the construction and materials of the clamp:

- a) *Inner conductor*—Copper tubing with an inner diameter of 6.35 mm (0.25 in) and an outer diameter of 9.4 mm (0.37 in).
- b) *Outer conductor*—Aluminum bar that is 300 mm long and approximately 54 mm by 58 mm. The bar is milled to accept the outer diameter of the dielectric material.
- c) *Dielectric*—High Density Polyethylene (Residual, TypeF) with dielectric constant of 2.32. An outside diameter of 33.5 mm and an inner diameter that accepts the outside diameter of the copper inner conductor.
- d) *Connectors*—BNC connectors are located 9 mm (0.39 in) from each end of the clamp and are recessed into the outer conductor. The center conductor of the connector is connected to the inner conductor as shown in Figure 40B-2.
- e) *Clamping screws*—Six screws are used to connect the two halves of the clamp together after the cable has been inserted. Although clamping screws are shown in Figure 40B-1, any clamping method may be used that ensures the two halves are connected electrically and permits quick assembly and disassembly.
- f) *Nylon screws*—Used to align and secure the inner conductor and dielectric to the outer conductor. The use and location of the screws is left to the manufacturer.
- g) *Keying bolts*—Two studs used to align the two halves of the clamp.

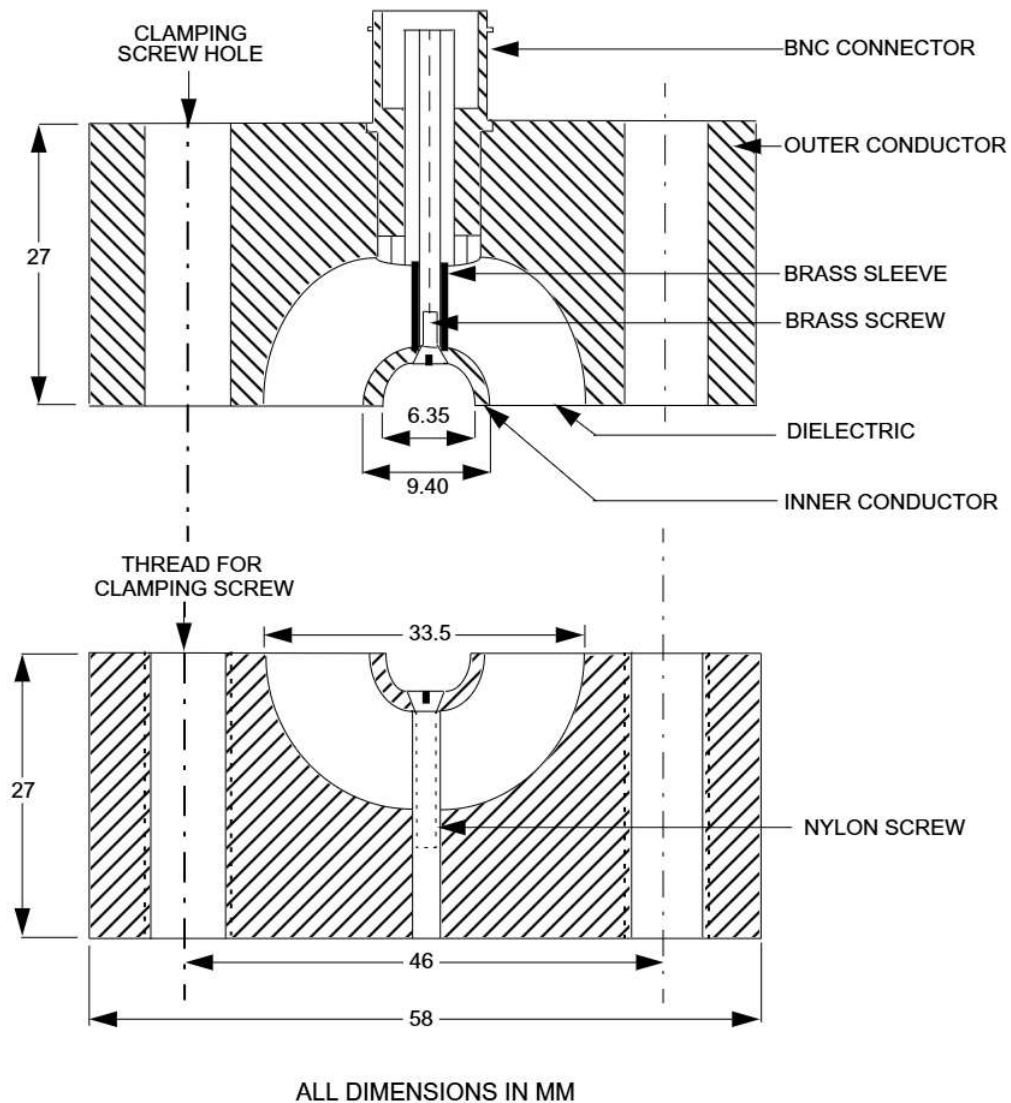


**Figure 40B-2—Cross-section of cable clamp**

As shown in Figure 40B-2 the inner conductor on the bottom half of the clamp extends slightly ( $\sim 0.1\text{mm}$ ) above the dielectric to ensure there is good electrical connection with the inner conductor of the top half of the clamp along the full length of the conductor when the two halves are clamped together.

The electrical parameters of the clamp between 1MHz and 250 MHz are as follows:

- a) Insertion loss:  $< 0.2\text{ dB}$
- b) Return loss:  $> 20.0\text{ dB}$

### 40B.1 Cable clamp validation

In order to ensure the cable clamp described above is operating correctly, the following test procedure is provided. Prior to conducting the following test shown in Figure 40B-3, the clamp should be tested to ensure the insertion loss and return loss are as specified above. The cable clamp validation test procedure uses a well-balanced 4-pair Category 5 unshielded test cable or better that meets the specifications of 40.7. The test hardware consists of the following:

- a) *Resistor network*—Network consists of three  $50 \pm 0.1\%$   $\Omega$  resistors; two resistors are connected in series as a differential termination for cable pairs and the other resistor is connected between the two and the ground plane as a common-mode termination.
- b) *Balun*—3 ports, laboratory quality with a  $100\ \Omega$  differential input,  $50\ \Omega$  differential output, and a  $50\ \Omega$  common-mode output:
  - Insertion Loss ( $100\ \Omega$  balanced  $\leftrightarrow$   $50\ \Omega$  unbalanced):  $<1.2\ \text{dB}$  (1-350 MHz)
  - Return Loss:  $>20\ \text{dB}$  (1-350 MHz)
  - Longitudinal Balance:  $>50\ \text{dB}$  (1-350 MHz)
- c) *Test cable*—4-pair  $100\ \Omega$  UTP category 5 balanced cable at least 30 m long.
- d) *Chokes* (2)—Wideband Ferrite Material:
  - Inter-diameter: 6.35 to 6.86 mm
  - Impedance:  $250\ \Omega$  @ 100 MHz
- e) *Ground plane*—Copper sheet or equivalent.
- f) *Signal generator*
- g) *Oscilloscope*
- h) *Receiver*

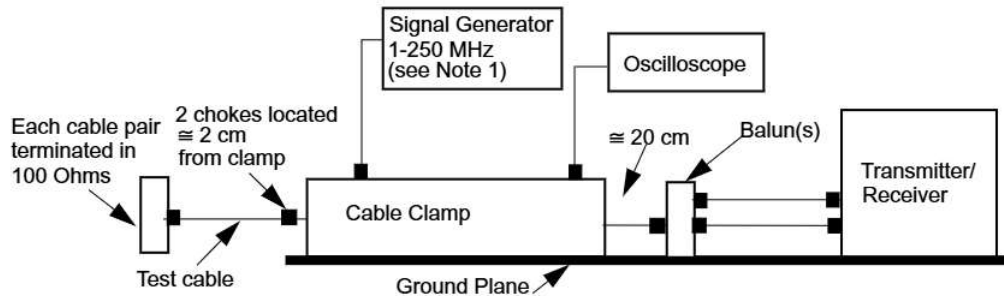


**Figure 40B-3—Cable clamp validation test configuration**

With the test cable inserted in the cable clamp, a signal generator with a  $50\ \Omega$  output impedance is connected to one end of the cable clamp and an oscilloscope with a  $50\ \Omega$  input impedance is connected to the other end. The signal generator shall be capable of providing a sine wave signal of 1 MHz to 250 MHz. The output of the signal generator is adjusted for a voltage of  $1.0\ \text{V}_{\text{rms}}$  ( $2.83\ \text{V}_{\text{pp}}$ ) at 20 MHz on the oscilloscope. The remainder of the test is conducted without changing the signal generator voltage. The cable pairs not connected to the balun are terminated in a resistor network, although when possible it is recommended that each cable pair be terminated in a balun. It is very important that the cable clamp, balun, receiver, and resistor networks have good contact with the ground plane. The two chokes, which are located next to each other, are located approximately 2.0 cm from the clamp. The cable between the clamp and the balun should be straight and not in contact with the ground plane.

The differential-mode and common-mode voltage outputs of the balun should meet the limits shown in Table 40B–1 over the frequency range 1 MHz to 250 MHz for each cable pair. The differential mode voltage at the output of the balun must be increased by 3 dB to take into account the 100-to-50 impedance matching loss of the balun.

**Table 40B–1 — Common- and differential-mode output voltages**

| Frequency ( <i>f</i> ) | Common-mode voltage                     | Differential-mode voltage             |
|------------------------|-----------------------------------------|---------------------------------------|
| 1-30 MHz               | $<0.1 + 0.97(f/30) \text{ V}_{pp}$      | $<2.4 + 19.68 (f/30) \text{ mV}_{pp}$ |
| 30-80 MHz              | $<1.07 \text{ V}_{pp}$                  | $<22 \text{ mV}_{pp}$                 |
| 80-250 MHz             | $<1.07 - 0.6 (f-80)/170 \text{ V}_{pp}$ | $<22 \text{ mV}_{pp}$                 |

NOTE—Prior to conducting the validation test the cable clamp should be tested without the cable inserted to determine the variation of the signal generator voltage with frequency at the output of the clamp. The signal generator voltage should be adjusted to 1 Vrms (2.83 Vpp) at 20 MHz on the oscilloscope. When the frequency is varied from 20 MHz to 250 MHz, the voltage on the oscilloscope should not vary more than  $\pm 7.5\%$ . If the voltage varies more than  $\pm 7.5\%$ , then a correction factor must be applied at each measurement frequency.

## **Annex 40C**

(informative)

### **Add-on interface for additional Next Pages**

This annex describes a technique for implementing Auto-Negotiation for 1000BASE-T when the implementor wishes to send additional Next Pages (other than those required to configure for 1000BASE-T operation). To accomplish this mode of Auto-Negotiation, the implementor must ensure that the three Next Pages required for 1000BASE-T configuration are sent first.

The add-on interface described in this annex shows one technique for supporting the transmission of additional Next Pages. This mechanism utilizes the existing Clause 28 Auto-Negotiation mechanism and variables defined in Clause 28. Its purpose is merely to provide optional NEXT PAGE WAIT responses to the Auto-Negotiation Arbitration state diagram (see Figure 28–16).

The add-on interface for Auto-Negotiation is intended to interface directly between the defined registers and the Auto-Negotiation mechanism defined in Clause 28. The mechanism described includes five main blocks (see Figure 40C–1).

The first three blocks are used by the MASTER-SLAVE resolution function. They are used to generate and store random seeds and to resolve the status of the MASTER-SLAVE relationship. Their operation is described later in this annex. The final two blocks, the transmit state machine for the 1000BASE-T Next Page exchange and the receive state machine for the 1000BASE-T Next Page exchange, are described in this annex.

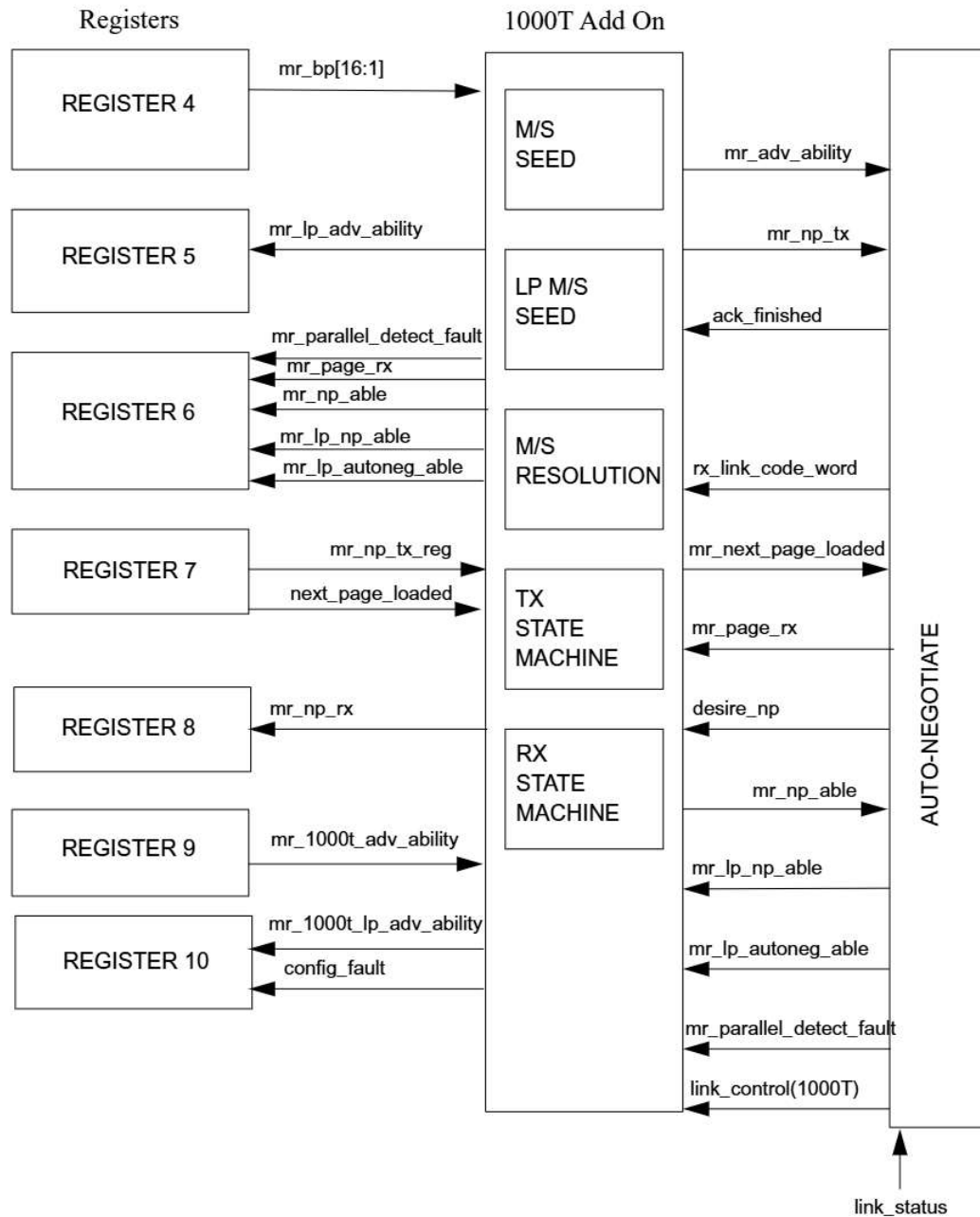


**Figure 40C–1 — Auto-negotiate add-on diagram for 1000BASE-T**

NOTE—When the exchange of Next Pages is complete, the MASTER-SLAVE relationship can be determined using Table 40–5 with the 1000BASE-T Technology Ability Next Page bit values specified in Table 40–4 and information received from the link partner. This process is conducted at the entrance to the FLP LINK GOOD CHECK state shown in the Auto-Negotiation Arbitration State Diagram (Figure 28–16).

### 40C.1 State variables

**mr\_bp**

This variable is used as an intermediate signal from register 4. Normally register 4 would directly source the mr\_adv\_ability information. This information is now sourced from the transmit state machine.

**mr\_1000t\_adv\_ability**

A 16-bit array used to store and indicate the contents of register 9.

**mr\_1000t\_lp\_adv\_ability**

A 16-bit array used to write information to register 10.

**mr\_np\_tx\_reg**

This variable is an intermediate signal from register 7. Normally register 7 would directly source the information to the Auto-Negotiation function via mr\_np\_tx. This information is now sourced from the transmit state machine.

**mr\_np\_rx**

A 16-bit array used to write information to register 8.  
Values: Zeros; data bit is logical zero.  
One; data bit is logical one.

**config\_fault**

This variable indicates the result of the MASTER-SLAVE resolution function.

**next\_page\_loaded**

This variable is an intermediate signal from register 7. Normally register 7 would directly source the information to the Auto-Negotiation function via mr\_next\_page\_loaded. This information is now sourced from the transmit state machine.

**reg\_random**

An 11-bit array used to store the received random seed from the link partner. It is used by the MASTER-SLAVE resolution function.

**1000T\_capable**

This variable is used merely to show the local device is 1000Base-T capable. It is shown to illustrate the path that a non-1000Base-T device would take within the auto negotiation mechanism.

**ATMP\_CNT**

This variable is used to count the number of failed MASTER-SLAVE resolutions. It has a maximum value of 7.

All other signals are defined in Clause 28.

## **40C.2 State diagrams**

### **40C.2.1 Auto-Negotiation Transmit state machine add-on for 1000BASE-T**

The Auto-Negotiation transmit state machine add-on (see Figure 40C–2) is responsible for sending the Base Page, 1000BASE-T Next Pages, as well as additional Next Pages as specified by the management interface. 1000BASE-T Next Pages will automatically be sent by the PHY whenever there are no additional Next Pages to be sent. If the user desires to send additional Next Pages, then the user must first send three pages of any format. Management will automatically replace these three pages with the appropriate 1000BASE-T Message Page and the two following unformatted pages and then will send the additional Next Pages as specified by the user. All other steps are performed by the management interface. The management interface is now required to complete the Next Page exchange by sourcing its own NULL page. This is shown in Figure 40C–2 for illustration only.

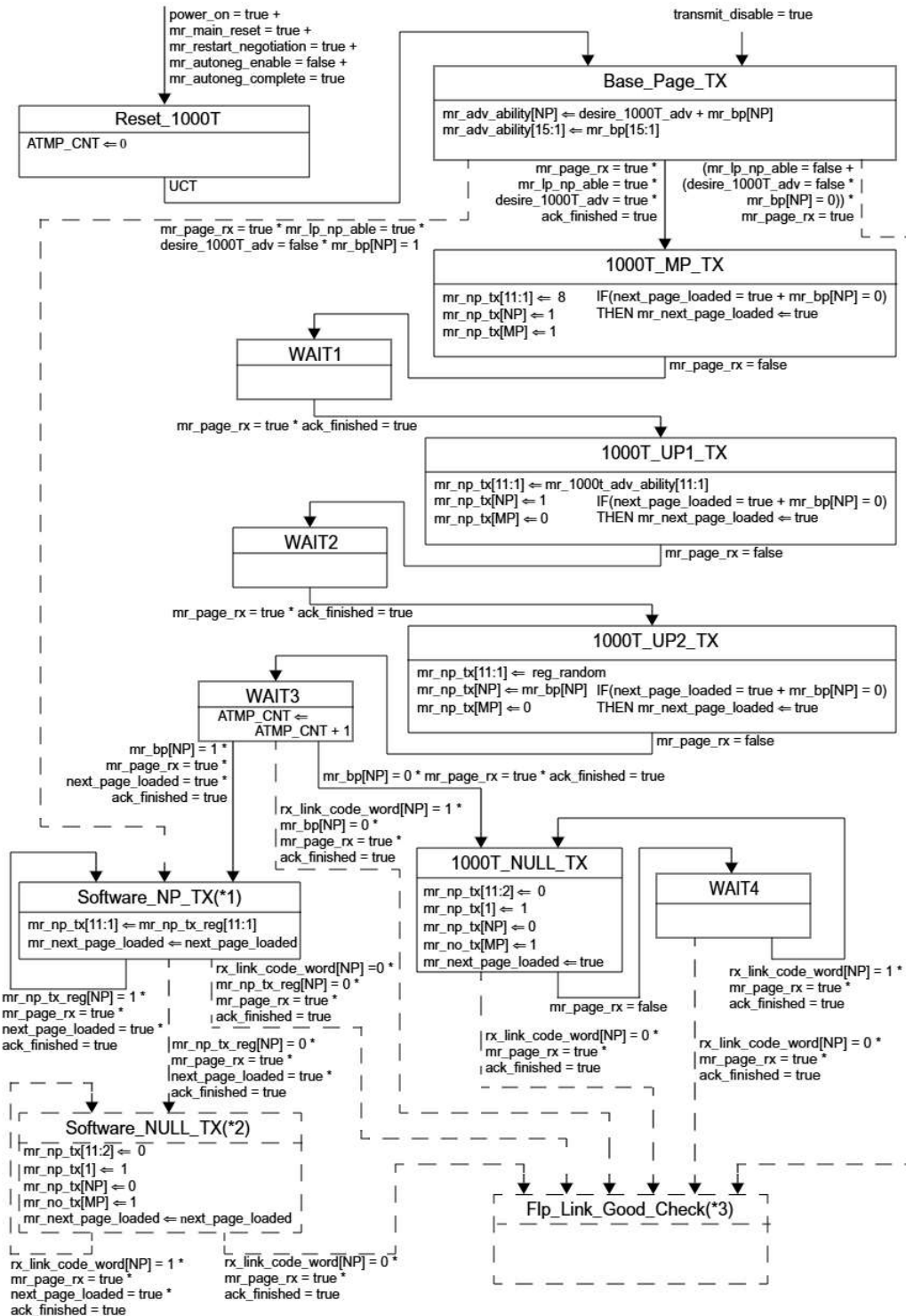


Figure 40C-2—Auto-Negotiation Transmit state diagram add-on for 1000BASE-T

## NOTES for Figure 40C-2

1—(Software\_NP\_TX) If the user desires to send additional Next Pages, then the contents of the first three Next Pages will be overwritten by the three 1000BASE-T Next Pages. In this case, the user is responsible for stepping through the Next Page sequence (by creating the initial three Next Pages to be overwritten by the three 1000BASE-T Next Pages); otherwise the process is automatic. (next\_page\_loaded signals clear operation as per Clause 28.)

2—(Software\_NULL\_TX) This is shown for illustration only. This is done in software and is required.

3—(Flp\_Link\_Good\_Check) This is shown for illustration only. This state is from the Auto-Negotiation arbitration state diagram and indicates the conclusion of pages being sent. (The transition 1000T\_capable = false is to show sequence for non 1000BASE-T implementations.)

**40C.2.2 Auto-Negotiation receive state diagram add-on for 1000BASE-T**

The Auto-Negotiation receive state machine add-on for 1000BASE-T Next Pages (see Figure 40C-3) is responsible for receiving the Base Page, 1000BASE-T Next Pages, and any additional Next Pages received. 1000BASE-T Next Pages will automatically be received whenever the user does not wish to participate in Next Page exchanges. In this case, the appropriate 1000BASE-T message page and its two unformatted pages will automatically be received and stored in their appropriate registers. Any additional Next Pages received will still be placed in register 8, but will be overwritten automatically when a new page is received. The net result is that the management interface does not need to poll registers 6 and 8. The information in register 8 will be meaningless in this case. If the user desires to participate in additional Next Page exchanges via setting the appropriate bit in register 4, the user now becomes responsible (as was previously the case) for defining how this will be accomplished. In this situation, the first three Next Pages received may be 1000BASE-T and should be discarded. This information will automatically be stored internally in the appropriate register 10 and reg\_random. The management interface/user can ignore the information received for the 1000BASE-T Next Pages.

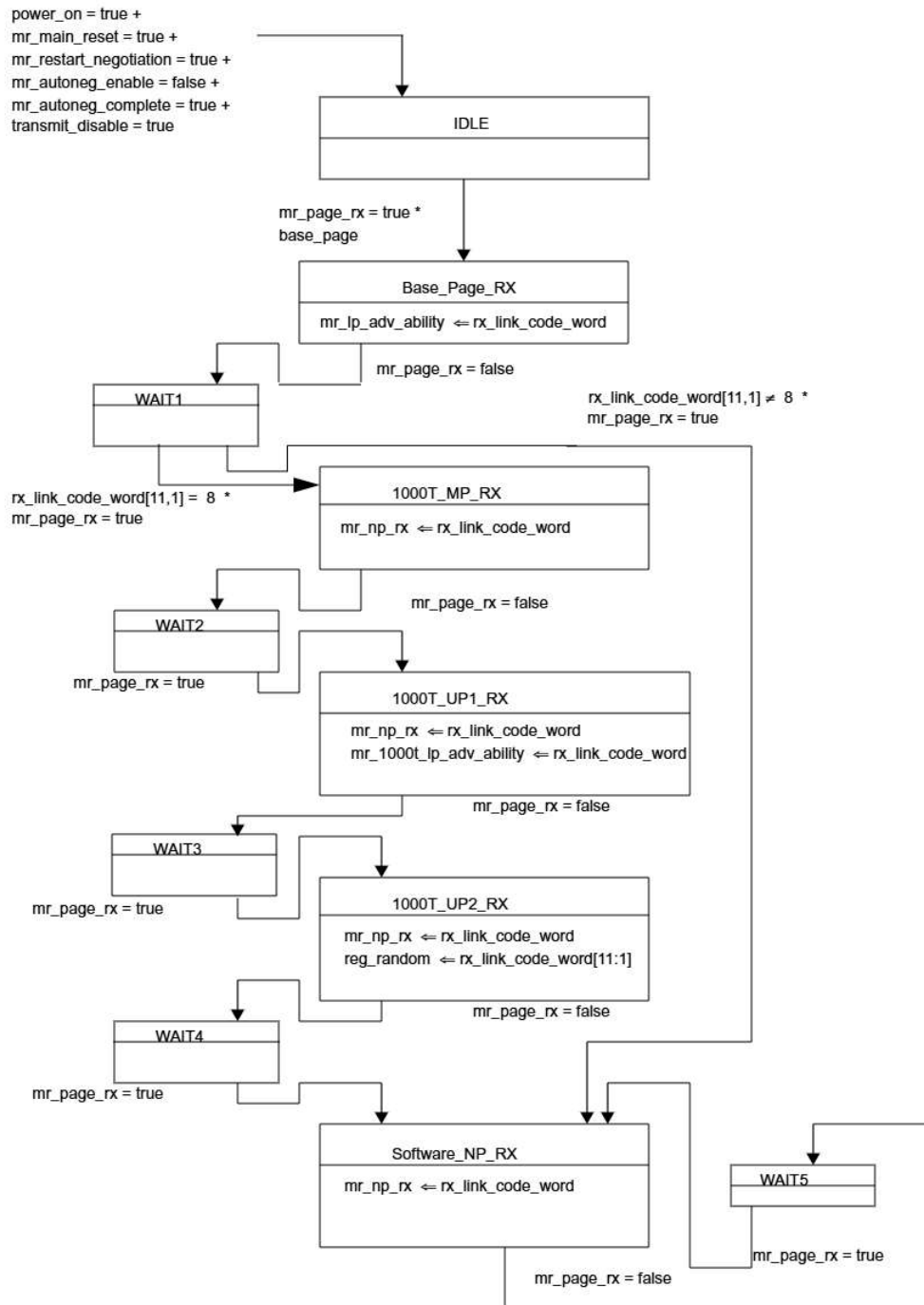


Figure 40C-3—Auto-Negotiation Receive state diagram add-on for 1000BASE-T

## Annex 43A

(informative)

### Collection and Distribution functions

#### 43A.1 Introduction

The specification of the Collection and Distribution functions was defined with the following considerations in mind:

- a) Frame duplication is not permitted.
- b) Frame ordering must be preserved in aggregated links. Strictly, the MAC service specification (ISO/IEC 15802-1) states that order must be preserved for frames with a given SA, DA, and priority; however, this is a tighter constraint than is absolutely necessary. There may be multiple, logically independent conversations in progress between a given SA-DA pair at a given priority; the real requirement is to maintain ordering within a conversation, though not necessarily between conversations.
- c) A single algorithm can be defined for the collection function that is independent of the distribution function(s) employed by the Partner System.
- d) In the interests of simplicity and scalability, the collection function should not perform re-assembly functions, re-order received frames, or modify received frames. Distribution functions, therefore, do not make use of segmentation techniques, do not label or otherwise modify transmitted frames in any way, and must operate in a manner that will inherently ensure proper ordering of received frames with the specified collector.
- e) The distribution and collection functions need to be capable of handling dynamic changes in aggregation membership.
- f) There are expected to be many different topologies and many different types of devices in which Link Aggregation will be employed. It is therefore unlikely that a single distribution function will be applicable in all cases.

A simple collection function has been specified. The Collector preserves the order of frames received on a given link, but does not preserve frame ordering amongst links. The distribution function maintains frame ordering by

- Transmitting frames of a given conversation on a single link at any time.
- Before changing the link on which frames of a given conversation are transmitted, ensuring that all previously transmitted frames of that conversation have been received to a point such that any subsequently transmitted frames received on a different links will be delivered to the MAC Client at a later time.

Given the wide variety of potential distribution algorithms, the normative text in Clause 43 specifies only the requirements that such algorithms must meet, and not the details of the algorithms themselves. To clarify the intent, this informative annex gives examples of distribution algorithms, when they might be used, and the role of the Marker protocol (43.5) in their operation. The examples are not intended to be either exhaustive or prescriptive; implementors may make use of any distribution algorithms as long as the requirements of Clause 43 are met.

### 43A.2 Port selection

A distribution algorithm selects the port used to transmit a given frame, such that the same port will be chosen for subsequent frames that form part of the same conversation. The algorithm may make use of information carried in the frame in order to make its decision, in combination with other information associated with the frame, such as its reception port in the case of a MAC Bridge.

The algorithm may assign one or more conversations to the same port, however, it must not allocate some of the frames of a given conversation to one port and the remainder to different ports. The information used to assign conversations to ports could include the following:

- a) Source MAC address
- b) Destination MAC address
- c) The reception port
- d) The type of destination address (individual or group MAC address)
- e) Ethernet Length/Type value (i.e., protocol identification)
- f) Higher layer protocol information (e.g., addressing and protocol identification information from the LLC sublayer or above)
- g) Combinations of the above

One simple approach applies a hash function to the selected information to generate a port number. This produces a deterministic (i.e., history independent) port selection across a given number of ports in an aggregation. However, as it is difficult to select a hash function that will generate a uniform distribution of load across the set of ports for all traffic models, it might be appropriate to weight the port selection in favor of ports that are carrying lower traffic levels. In more sophisticated approaches, load balancing is dynamic; i.e., the port selected for a given set of conversations changes over time, independent of any changes that take place in the membership of the aggregation.

### 43A.3 Dynamic reallocation of conversations to different ports

It may be necessary for a given conversation or set of conversations to be moved from one port to one or more others, as a result of

- a) An existing port being removed from the aggregation,
- b) A new port being added to the aggregation, or
- c) A decision on the part of the Distributor to re-distribute the traffic across the set of ports.

Before moving conversation(s) to a new port, it is necessary to ensure that all frames already transmitted that are part of those conversations have been successfully received. The following procedure shows how the Marker protocol (43.5) can be used to ensure that no mis-ordering of frames occurs:

- 1) Stop transmitting frames for the set of conversations affected. If the MAC Client requests transmission of further frames that are part of this set of conversations, these frames are discarded.
- 2) Start a timer, choosing the timeout period such that, if the timer expires, the destination System can be assumed either to have received or discarded all frames transmitted prior to starting the timer.
- 3) Use the Marker protocol to send a Marker PDU on the port previously used for this set of conversations.
- 4) Wait until either the corresponding Marker Response PDU is received or the timer expires.
- 5) Restart frame transmission for the set of conversations on the newly selected port.

The appropriate timeout value depends on the connected devices. For example, the recommended maximum Bridge Transit Delay is 1 second; if the receiving device is a MAC Bridge, it may be expected to have

forwarded or discarded all frames received more than 1 second ago. The appropriate timeout value for other circumstances could be smaller or larger than this by several orders of magnitude. For example, if the two Systems concerned are high-performance end stations connected via Gigabit Ethernet links, then timeout periods measured in milliseconds might be more appropriate. In order to allow an appropriate timeout value to be determined, the Frame Collector parameter `CollectorMaxDelay` (see 43.2.3) defines the maximum delay that the collector can introduce between receiving a frame from a port and either delivering it to the MAC Client or discarding it. This value will be dependent upon the particular implementation choices that have been made in a System. As far as the operation of the Collector state machine is concerned, `CollectorMaxDelay` is a constant; however, a management attribute, `aAggCollectorMaxDelay` (30.7.1.1.32), is provided that allows interrogation and administrative control of its value. Hence, if a System knows the value of `CollectorMaxDelay` that is in use by a Partner System, it can set the value of timeout used when flushing a link to be equal to that value of `CollectorMaxDelay`, plus sufficient additional time to allow for the propagation delay experienced by frames between the two Systems. A value of zero for the `CollectorMaxDelay` parameter indicates that the delay imposed by the Collector is less than the resolution of the parameter (10 microseconds). In this case, the delay that must be considered is the physical propagation delay of the channel. Allowing management manipulation of `CollectorMaxDelay` permits fine-tuning of the value used in those cases where it may be difficult for the equipment to pre-configure a piece of equipment with a realistic value for the physical propagation delay of the channel.

The Marker protocol provides an optimization that can result in faster reallocation of conversations than would otherwise be possible—without the use of markers, the full timeout period would always have to be used in order to be sure that no frames remained in transit between the local Distributor and the remote Collector. The timeout described recovers from loss of Marker or Marker Response PDUs that can occur.

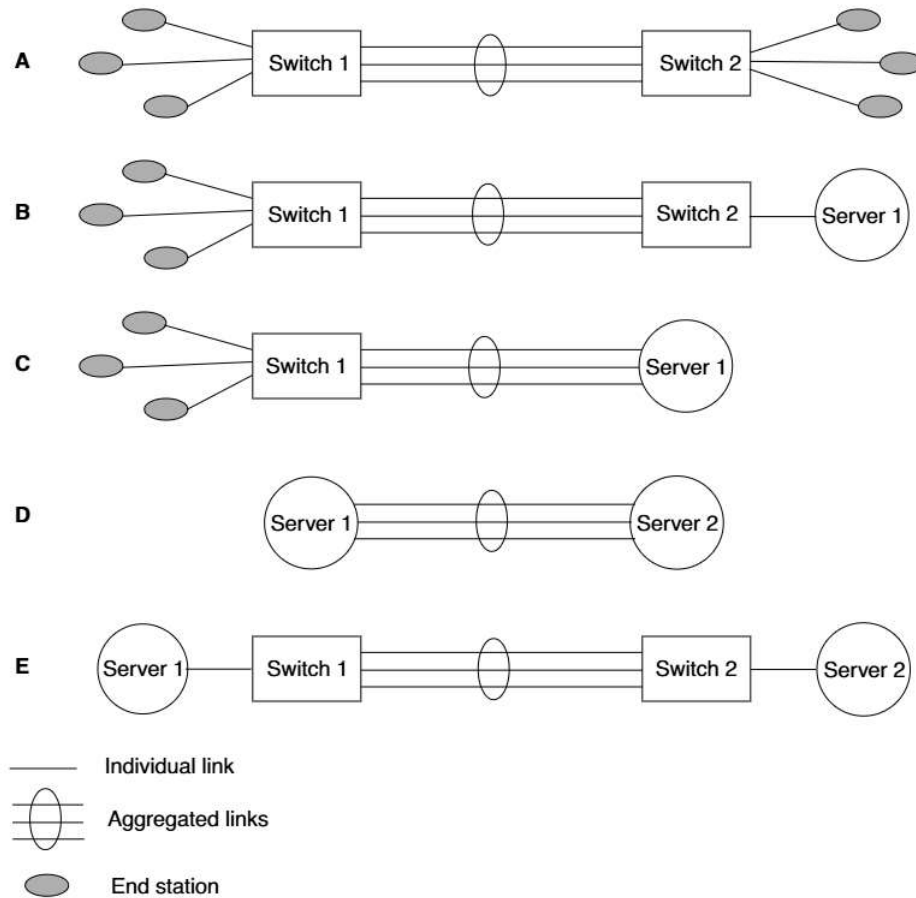
#### 43A.4 Topology considerations in the choice of distribution algorithm

Figure 43A-1 gives some examples of different aggregated link scenarios. In some cases, it is possible to use distribution algorithms that use MAC frame information to allocate conversations to links; in others, it is necessary to make use of higher-layer information.

In example A, there is a many-to-many relationship between end stations communicating over the aggregated link. It would be possible for each switch to allocate conversations to links simply on the basis of source or destination MAC addresses.

In examples B and C, a number of end stations communicate with a single server via the aggregated link. In these cases, the distribution algorithm employed in the server or in Switch 2 can allocate traffic from the server on the basis of destination MAC address; however, as one end of all conversations constitutes a single server with a single MAC address, traffic from the end stations to the server would have to be allocated on the basis of source MAC address. These examples illustrate the fact that different distribution algorithms can be used in different devices, as appropriate to the circumstances. The collection function is independent of the distribution function(s) that are employed.

In examples D and E, assuming that the servers are using a single MAC address for all of their traffic, the only appropriate option is for the distribution algorithm used in the servers and switches to make use of higher-layer information (e.g., Transport Layer socket identifiers) in order to allocate conversations to links. Alternatively, in example E, if the servers were able to make use of multiple MAC addresses and allocate conversations to them, then the switches could revert to MAC Address-based allocation.

**Figure 43A-1—Link aggregation topology examples**

## Annex 43B

(normative)

### Requirements for support of Slow Protocols

#### 43B.1 Introduction and rationale

There are two distinct classes of protocols used to control various aspects of the operation of IEEE 802.3® devices. They are as follows:

- a) Protocols such as the MAC Control PAUSE operation (Annex 31B) that need to process and respond to PDUs rapidly in order to avoid performance degradation. These are likely to be implemented as embedded hardware functions, making it relatively unlikely that existing equipment could be easily upgraded to support additional such protocols.

NOTE—This consideration was one of the contributing factors in the decision to use a separate group MAC address to support LACP and the Marker protocol, rather than re-using the group MAC address currently used for PAUSE frames.

- b) Protocols such as LACP, with less stringent frequency and latency requirements. These may be implemented in software, with a reasonable expectation that existing equipment be upgradeable to support additional such protocols, depending upon the approach taken in the original implementation.

In order to place some realistic bounds upon the demands that might be placed upon such a protocol implementation, this annex defines the characteristics of this class of protocols and identifies some of the behaviors that an extensible implementation needs to exhibit.

#### 43B.2 Slow Protocol transmission characteristics

Protocols that make use of the addressing and protocol identification mechanisms identified in this annex are subject to the following constraints:

- a) No more than 5 frames shall be transmitted in any one-second period.
- b) The maximum number of Slow Protocols is 10.  
NOTE—This is the maximum number of Slow Protocols that use the specified protocol type defined here. That is, there may be more than 10 slow protocols in the universe, but no more than 10 may map to the same Ethernet Length/Type field.
- c) The MAC Client data generated by any of these protocols shall be in the normal length range for an IEEE 802.3® MAC frame, as specified in 4.4.2. It is recommended that the maximum length for a Slow Protocol frame be limited to 128 octets.  
NOTE—The Slow Protocols specified in Clause 43 (i.e., LACP and Marker) conform to this recommended maximum.
- d) PDUs generated by these protocols shall use the Basic and not the Tagged frame format (see Clause 3).

The effect of these restrictions is to restrict the bandwidth consumed and performance demanded by this set of protocols; the absolute maximum traffic loading that would result is 50 maximum length frames per second per link.

### 43B.3 Addressing

The Slow\_Protocols\_Multicast address has been allocated exclusively for use by Slow Protocols PDUs; its value is identified in Table 43B-1.

**Table 43B-1—Slow\_Protocols\_Multicast address**

| Name                             | Value             |
|----------------------------------|-------------------|
| Slow_Protocols_Multicast address | 01-80-C2-00-00-02 |

#### NOTES

1—This address is within the range reserved by ISO/IEC 15802-3 (MAC Bridges) for link-constrained protocols. As such, frames sent to this address will not be forwarded by conformant MAC Bridges; its use is restricted to a single link.

2—Although the two currently existing Slow Protocols (i.e., LACP and the Marker protocol) always use this MAC address as the destination address in transmitted PDUs, this may not be true for all Slow Protocols. In some yet-to-be-defined protocol, unicast addressing may be appropriate and necessary. Rather, the requirement is that this address not be used by any protocols that are not Slow Protocols.

### 43B.4 Protocol identification

All Slow Protocols use Type-field encoding of the Length/Type field, and use the Slow\_Protocols\_Type value as the primary means of protocol identification; its value is shown in Table 43B-2.

**Table 43B-2—Slow\_Protocols\_Type field**

| Name                      | Value |
|---------------------------|-------|
| Slow_Protocols_Type field | 88-09 |

The first octet of the MAC Client data following the Length/Type field is a protocol subtype identifier that distinguishes between different Slow Protocols. Table 43B-3 identifies the semantics of this subtype.

NOTE—Although this mechanism is defined as part of an IEEE 802.3® standard, it is the intent that the reserved code points in this table will be made available to protocols defined by other working groups within IEEE 802®, should this mechanism be appropriate for their use.

**Table 43B–3—Slow Protocols subtypes**

| Protocol Subtype value | Protocol name                            |
|------------------------|------------------------------------------|
| 0                      | Unused—Illegal value                     |
| 1                      | Link Aggregation Control Protocol (LACP) |
| 2                      | Link Aggregation—Marker Protocol         |
| 3                      | Reserved for future use                  |
| 4                      | Reserved for future use                  |
| 5                      | Reserved for future use                  |
| 6                      | Reserved for future use                  |
| 7                      | Reserved for future use                  |
| 8                      | Reserved for future use                  |
| 9                      | Reserved for future use                  |
| 10                     | Reserved for future use                  |
| 11–255                 | Unused—Illegal values                    |

### 43B.5 Handling of Slow Protocol frames

Any received MAC frame that carries the `Slow_Protocols_Type` field value is assumed to be a Slow Protocol frame. An implementation that claims conformance to this standard shall handle all Slow Protocol frames as follows:

- Discard any Slow Protocol frame that carries an illegal value of Protocol Subtype (see Table 43B–3). Such frames shall not be passed to the MAC Client.
- Pass any Slow Protocol frames that carry Protocol Subtype values that identify supported Slow Protocols to the protocol entity for the identified Slow Protocol.
- Pass any Slow Protocol frames that carry Protocol Subtype values that identify unsupported Slow Protocols to the MAC Client.

**NOTE**—The intent of these rules is twofold. First, they rigidly enforce the maximum number of Slow Protocols, ensuring that early implementations of this mechanism do not become invalidated as a result of “scope creep.” Second, they make it clear that the appropriate thing to do in any embedded frame parsing mechanism is to pass frames destined for unsupported protocols up to the MAC Client rather than discarding them, thus allowing for the possibility that, in soft configurable systems, the MAC Client might be enhanced in the future in order to support protocols that were not implemented in the hardware.

## 43B.6 Protocol Implementation Conformance Statement (PICS) proforma for Annex 43B, Requirements for support of Slow Protocols<sup>14</sup>

### 43B.6.1 Introduction

The supplier of an implementation that is claimed to conform to Annex 43B shall complete the following Protocol Implementation Conformance Statement (PICS) proforma.

A detailed description of the symbols used in the PICS proforma, along with instructions for completing the PICS proforma, can be found in Clause 21.

### 43B.6.2 Identification

#### 43B.6.2.1 Implementation identification

|                                                                                                                                                                                                                                                                                           |  |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| Supplier (Note 1)                                                                                                                                                                                                                                                                         |  |
| Contact point for queries about the PICS (Note 1)                                                                                                                                                                                                                                         |  |
| Implementation Name(s) and Version(s) (Notes 1 and 3)                                                                                                                                                                                                                                     |  |
| Other information necessary for full identification—e.g., name(s) and version(s) of machines and/or operating system names (Note 2)                                                                                                                                                       |  |
| <b>NOTES</b><br>1—Required for all implementations.<br>2—May be completed as appropriate in meeting the requirements for the identification.<br>3—The terms Name and Version should be interpreted appropriately to correspond with a supplier's terminology (e.g., Type, Series, Model). |  |

#### 43B.6.2.2 Protocol summary

|                                                                                                                                                                                                                                                                    |                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Identification of protocol specification                                                                                                                                                                                                                           | IEEE Std 802.3-2002®, Annex 43B, Requirements for support of Slow Protocols. |
| Identification of amendments and corrigenda to the PICS proforma which have been completed as part of the PICS                                                                                                                                                     |                                                                              |
| Have any Exception items been required? No <input type="checkbox"/> Yes <input type="checkbox"/><br>(See Clause 21: the answer Yes means that the implementation does not conform to IEEE Std 802.3-2002®, Annex 43B, Requirements for support of Slow Protocols.) |                                                                              |

|                   |  |
|-------------------|--|
| Date of Statement |  |
|-------------------|--|

<sup>14</sup>Copyright release for PICS proformas: Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

**43B.6.2.3 Transmission characteristics**

| Item | Feature           | Subclause | Value/Comment                                   | Status | Support |
|------|-------------------|-----------|-------------------------------------------------|--------|---------|
| SP1  | Transmission rate | 43B.2     | Max 5 frames in any one-second period           | M      | Yes [ ] |
| SP2  | Frame size        | 43B.2     | Normal IEEE 802.3® frame size range (see 4.4.2) | M      | Yes [ ] |
| SP3  | Frame format      | 43B.2     | Basic (not Tagged) frame format                 | M      | Yes [ ] |

**43B.6.2.4 Frame handling**

| Item | Feature                          | Subclause | Value/Comment         | Status | Support |
|------|----------------------------------|-----------|-----------------------|--------|---------|
| FH1  | Handling of Slow Protocol frames | 43B.5     | As specified in 43B.5 | M      | Yes [ ] |

## **Annex 43C**

(informative)

### **LACP standby link selection and dynamic Key management**

#### **43C.1 Introduction**

While any two ports on a given system that have been assigned the same administrative Key may be capable of aggregation, it is not necessarily the case that an arbitrary selection of such ports can be aggregated. (Keys may have been deliberately assigned to allow one link to be operated specifically as a hot standby for another). A system may reasonably limit the number of ports attached to a single Aggregator, or the particular way more than two ports can be combined.

In cases where both communicating systems have constraints on aggregation, it is necessary for them both to agree to some extent on the links to be selected for aggregation and on which not to use. Otherwise it might be possible for the two systems to make different selections, possibly resulting in no communication at all.

When one or more links have to be selected as standby, it is possible that they could be used as part of a different Link Aggregation Group. For this to happen, one or another of the communicating systems has to change the operational Key values used for the ports attached to those links.

If the operational Key values were to be changed independently by each system, the resulting set of aggregations could be unpredictable. It is possible that numerous aggregations, each containing a single link, may result. Worse, with no constraint on changes, the process of both systems independently searching for the best combination of operational Key values may never end.

This annex describes protocol rules for standby link selection and dynamic Key management. It provides examples of a dynamic Key management algorithm applied to connections between systems with various aggregation constraints.

#### **43C.2 Goals**

The protocol rules presented

- a) Enable coordinated, predictable, and reproducible standby link selections.
- b) Permit predictable and reproducible partitioning of links into aggregations by dynamic Key management.

They do not require

- c) A LACP system to understand all the constraints on aggregations of multiple ports that might be imposed by other systems.
- d) Correct configuration of parameters, i.e., they retain the plug and play attributes of LACP.

### 43C.3 Standby link selection

Every link between systems operating LACP is assigned a unique priority. This priority comprises (in priority order) the System Priority, System ID, Port Priority, and Port Number of the higher-priority system. In priority comparisons, numerically lower values have higher priority.

Ports are considered for active use in an aggregation in link priority order, starting with the port attached to the highest priority link. Each port is selected for active use if preceding higher priority selections can also be maintained, otherwise the port is selected as standby.

### 43C.4 Dynamic Key management

Dynamic Key management changes the Key values used for links that either system has selected as a standby to allow use of more links. Whether this is desirable depends on their use. For example, if a single spanning tree is being used throughout the network, separating standby links into a separate aggregation serves little purpose. In contrast, if equal cost load sharing is being provided by routing, making additional bandwidth available in a separate Link Aggregation Group may be preferable to holding links in standby to provide link resilience.

The communicating system with the higher priority (as determined by System Priority and unique System ID) controls dynamic Key changes. Dynamic Key changes may only be made by this controlling system.

NOTE—The controlling system can observe the port priorities assigned by the Partner system, if it wishes to take these into account.

This rule prevents the confusion that could arise if both systems change Keys simultaneously. In principle the controlling system might search all possible Key combinations for the best way to partition the links into groups. In practice the number of times that Keys may have to be changed to yield acceptable results is small.

After each Key change, the controlling system assesses which links are being held in standby by its Partner. Although there is no direct indication of this decision, standby links will be held OUT\_OF\_SYNC. After matched information is received from the protocol Partner, and before acting on this information, a “settling time” allows for the Partner’s aggregate wait delay, and for the selected links to be aggregated. Twice the Aggregate Wait Time (the expiry period for the wait\_while\_timer), i.e., 4 seconds, should be ample. If matched Partner information indicates that all the links that the Actor can make active have been brought IN\_SYNC, it can proceed to change Keys on other links without further delay.

### 43C.5 A dynamic Key management algorithm

The following algorithm is simple but effective.

After the “settling time” (see 43C.4) has elapsed, the controlling system scans its ports in the Link Aggregation Group (i.e., all those ports with a specific operational Key value that have the same Partner System Priority, System ID, and Key) in descending priority order.

For each port, it may wish to know

- a) Is the port (i.e., the Actor) *capable* of being aggregated with the ports already selected for aggregation with the current Key? Alternatively is the Actor *not capable* of this aggregation?
- b) Is the port’s Partner IN\_SYNC or is the Partner OUT\_OF\_SYNC?

And as it inspects each port it may

- c) *Select* the port to be part of the aggregation with the current Key.
- d) *Retain* the current Key for a further iteration of the algorithm, without selecting the port to be part of the current aggregation.
- e) *Change* the operational Key to a new value. Once a new value is chosen, all the ports in the current Link Aggregation Group that have their Keys changed will be changed to this new value.

As the ports are scanned for the first time

- 1) The highest priority port is always selected.

If it is capable and IN\_SYNC, move to step 2).

Otherwise, **change** the operational Key of all other ports (if any) in this Link Aggregation Group, and apply this dynamic Key algorithm to those ports, beginning with step 1), after the settling time.

- 2) Move to the next port.

IF there is a next port, continue at step 3).

Otherwise, dynamic Key changes for ports with this operational Key are complete.

Note that ports that were once in the same aggregation may have had their operational Keys changed to (further) new values. If so, apply the dynamic Key management algorithms to those ports, beginning with step 1), after the settling time.

- 3) If this port is capable and IN\_SYNC:

**select** it, and repeat from step 2).

If this port is OUT\_OF\_SYNC:

**change** the operational Key, and repeat from step 2).

If this port is not capable but IN\_SYNC:

**change** the operational Key, move to step 4).

- 4) Move to the next port.

If there is a next port, continue at step 5).

Otherwise If there are still ports in the current Link Aggregation Group (which will have the current operational Key), wait for the settling time and apply the dynamic Key management algorithm, beginning with the first such port, at step 3).

Otherwise, dynamic Key changes for ports with this operational Key are complete.

- 5) **If** this port is capable:

retain the current Key and repeat from step 2).

Otherwise, **change** the operational Key and repeat from step 2).

This procedure is repeated until no OUT\_OF\_SYNC links remain, or a limit on the number of steps has been reached.

If the Partner's System ID changes on any link at any time, the Actor's operational Key for that link should revert to the administrative Key value, and the dynamic Key procedure should be rerun. This may involve changing the operational Key values for all the links that were assigned Key values subsequent to the change in Key for the link with the new Partner.

### 43C.6 Example 1

Two systems, A and B, are connected by four parallel links. Each system can support a maximum of two links in an aggregation. They are connected as shown in Figure 43C-1. System A is the higher priority system.

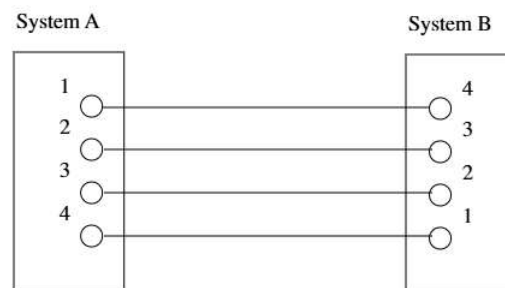


Figure 43C-1—Example 1

The administrative Key for all of System A and System B's ports is 1. Neither system knows before the configuration is chosen that all its ports would attach to links of the same Partner system. Equally, if the links were attached to two different systems, it is not known which pair of links (e.g., 1 and 2, or 1 and 4) would be attached to the same Partner. So choosing the administrative Keys values to be identical for four ports, even though only two could be actively aggregated, is very reasonable.

If there was no rule for selecting standby links, System A and System B might have both selected their own ports 1 and 2 as the active links, and there would be no communication. With the rule, the links A1-B4 and A2-B3 will become active, while A3-B2 and A4-B1 will be standby.

Since System A is the higher-priority system, System B's operational Key values will remain 1 while System A may dynamically change Keys, though it may choose to retain the standby links. Following the Key management algorithm suggested, System A would be able to change the Keys for A3 and A4 in a little over 2 seconds (depending on how fast System B completes the process of attaching its ports to the selected Aggregator) after the connections were first made, and both aggregations could be operating within 5 seconds.

If System A's aggregations were to be constrained to a maximum of three links, rather than two, while System B's are still constrained to two, the suggested algorithm would delay for 4 seconds before changing Keys. Both aggregations could be operating within 7 seconds.

### 43C.7 Example 2

A system has the odd design constraint that each of its four ports may be aggregated with one other as follows:

- a) Port 1 with port 2, or port 4.
- b) Port 2 with port 3, or port 1.
- c) Port 3 with port 4, or port 2.
- d) Port 4 with port 1, or port 3.

This is equivalent to each port being able to aggregate with either neighbor, understanding the ports to be arranged in a circle.

Two such systems are connected with four parallel links as shown in Figure 43C–2.

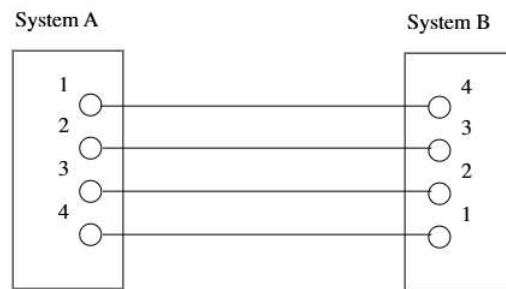


**Figure 43C–2—Example 2a**

Just as for Example 1, links A1-B4 and A2-B3 become active without changing the operational Key from its original administrative value. The Key for A3 and A4 is changed as soon as they become active, and a few seconds later A3-B2 and A4-B1 become active in a separate aggregation.

If the two systems had been connected as shown in Figure 43C–3:

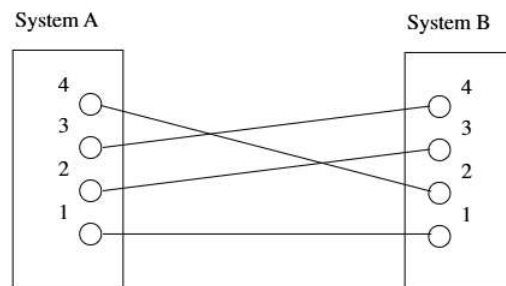


**Figure 43C–3—Example 2b**

the following would occur, assuming that System A operates the simple algorithm already described.

Initially System A advertises an operational Key equal to the administrative Key value of 1 on all ports. System B first selects B1 as active; since the link connects to A1 it has the highest priority. The next highest priority link is B3-A2, but System B cannot aggregate B3 with B1, so System B makes this port standby. System B can aggregate B4-A3, so the port is made active. Finally if B4 is aggregated with B1, B2 cannot be aggregated, so B2 is made standby.

System A, observing the resulting synchronization status from System B, assigns a Key value of 2 to ports 2 and 3, retaining the initial Key of 1 for ports 1 and 4. System B will remove B4 from the aggregation with B1, and substitute B2. B3 and B4 will be aggregated. In the final configuration A1-B1 and A4-B2 are aggregated, as are A2-B3 and A3-B4.