Author Norman Finn

Port Aggregation Protocol

A protocol and hardware features are defined which allow multiple point-to-point links connecting two devices to be automatically discovered and aggregated into a single logical link, providing both load sharing and redundancy.

This document is presented to the IEEE 802.3 Link Aggregation Study Group. It documents a protocol developed by, and used by, Cisco Systems, as well as other companies licensed by Cisco Systems to do so. Distribution of this document does not constitute permission by Cisco Systems for other individuals or organizations to implement the procedures and protocols described herein. It is the intention of Cisco Systems to follow recommended IEEE 802 practices with regard to making reasonable and nodiscriminatory licensing arrangements for the use of any procedures or protocols described in this document which become incorporated into an IEEE 802.3 Link Aggregation standard, and which are covered by U.S. or foreign patents held or pending by Cisco Systems.

Port Aggregation Protocol

1.0 Introduction

This paper defines a Port Aggregation Protocol that can:

1. Allow individual point-to-point ports/interfaces to be aggregated into a group (an aggregation port, or agport) which is regarded as a single port/interface by all higher layers.
2. Interoperate, without aggregation, with known hardware and software.
3. Support all aggregation methods known to be implemented in hardware.
4. Allow the makeup of an agport to change as interfaces or wires fail and/or heal, with an absolute minimum of disruption to the higher layers.
5. Provide the ability for management to control and track the composition of an agport and obtain meaningful statistics on the agport and its member interfaces.
6. Allow the automatic determination of which physical ports can avoid the delay/disruption involved in the 802.1D blocked->listening->learning->forwarding progression.
7. Detect multidrop connections, and allow those connections to be used without port aggregation.
8. Provide a means to detect, report, and prevent the use of unidirectional interfaces, whether on point-to-point or multidrop connections, among all devices running the Port Aggregation Protocol.
9. Detect, report, and prevent the use of incorrectly-paired bidirectional point-to-point interfaces.
10. Provide the ability for a bridge to ensure that the ordered delivery of L2 frames is not compromised by port aggregation.

The Port Aggregation Protocol has four parts, the Frame Forwarding part (Section 3.0), the PAgP Port State part (Section 4.0), the PAgP Group Management part (Section 5.0), and the Flush protocol (Section 6.0). The PAgP Port State runs on each individual physical (or logical) port which may be grouped. The PAgP Group Management part forms agports from individual ports. Frame Forwarding covers the use of the agports to actually forward data. The Flush protocol may be required by some implementations and/or installations to ensure packet ordering is maintained.

Section 2.0 describes the Port Aggregation Protocol from a high level. Implementation notes are summarized in Section 7.0.

2.0 Overall Architecture

The principal elements of the Port Aggregation Protocol are physical ports and agports. Physical ports may be just that, in a typical bridge, or may be logical entities such as individual X.25 connections or IP encapsulated pipes in a router. This paper will assume from now on that physical ports are IEEE 802 physical ports. However, any logical or physical link can be a physical port if it:

1. provides a means (such as a peculiar MAC destination address) to ensure that PAgP packets are not passed transparently through a non-PAgP-capable device; and

2. provides a means (such as an Ethertype field) to distinguish PAgP packets from other traffic carried on the same physical port.

Agports are the bridge ports of a bridge, the router interfaces of a router, or the interfaces of an endstation. All are agports as viewed by PAgP.

The Port Aggregation Protocol is not recursive in the sense that it cannot run in multiple layers, in order to aggregate groups of PAgP-aggregated agports.

2.1 Aggregating Physical Ports with Agports

Interfacing to higher layers poses some difficult problems if the number of those interfaces can vary with time. Most of these problems are avoided by using the interface stack. We define as many agports as we have physical ports on which the Port Aggregation Protocol is enabled. The agports stand “above” the physical ports in the interface stack. Most higher-layer protocols, even bridges, bind to agports, and not to physical ports. (Physical topology discovery protocols would be an exception; they would bind to physical ports.) PAgP’s job is to define the connections between the agports and the physical ports.

The interface stack MIB is used to reflect the relationships between the agport interfaces and the physical interfaces. If PAgP is enabled, then the relationships between agports and physical interfaces are dynamic, and the state of the interface stack reflects changes in these relationships. If PAgP is disabled, and port aggregation is handled manually via the MIB or by operator commands, then the interface stack is static between manual changes. Thus, if PAgP is enabled, and if an aggregated physical port leaves the UpPAgP state, it is disconnected from its agport in the interface stack. If PAgP is disabled, then loss of a physical port may or may not change the interface stack and/or the agport’s ifOperStatus.

A given agport may be configured to be available for binding by PAgP to specific sets of physical ports. There are two obvious cases where this is required:

1. Devices such as routers require specific bindings to specific physical ports.
2. The aggregation of physical ports into agports may utilize hardware support that restricts which physical ports that may be grouped together.

Figure 1 illustrates this concept in a bridge which, over time, has had its physical ports and agports undergo a number of changes.

In Figure 1, the agports **a** through **h** are the bridge ports. Physical ports **1** through **8** are the bridge’s physical ports. Agport **a** comprises physical ports **1** and **2**, agport **d** has physical ports **3**, **5**, and **6**, agport **e** has only physical port **4**, and agport **g** only physical port **7**. Agports **b**, **c**, **f**, and **h** have no physical ports attached to them, and hence are in the operationally down state in the bridge’s port table. Physical port **8** is unattached to any agport, so is unavailable to the bridge.

Figure 2 illustrates the use of PAgP on a router with the same eight physical ports as the bridge in Figure 1. Because a router carries considerable configuration information on each of its router interfaces, its physical ports are not free to associate with just any agport, even though the hardware is capable of grouping them arbitrarily. In this example, the router has four router interfaces, and hence only four agports, **a** through **d**. The router’s physical ports are divided by configuration into separate sets, and each set is able to be grouped with only one particular agport. Agport (router interface) **a** has been connected by PAgP to both of its physical ports **1** and **2**, agport **b** to

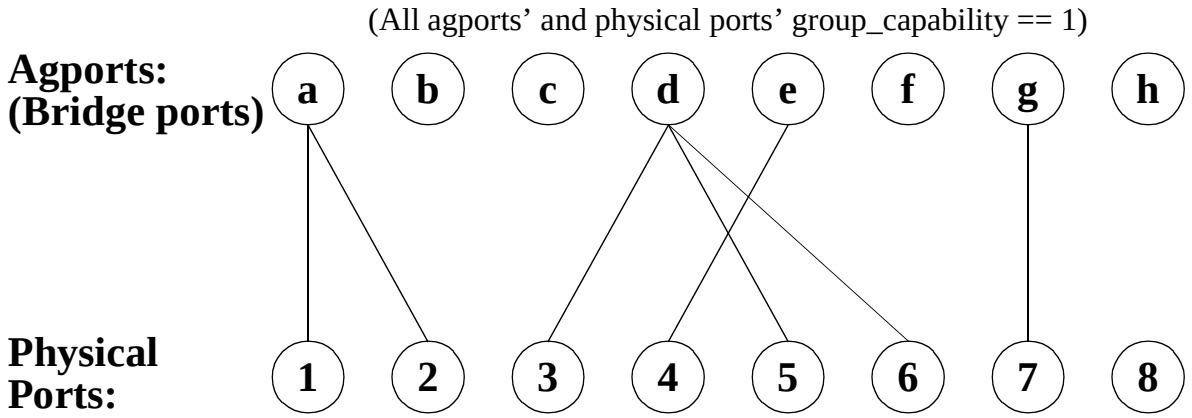

FIGURE 1. Agports and Physical Ports for a Bridge

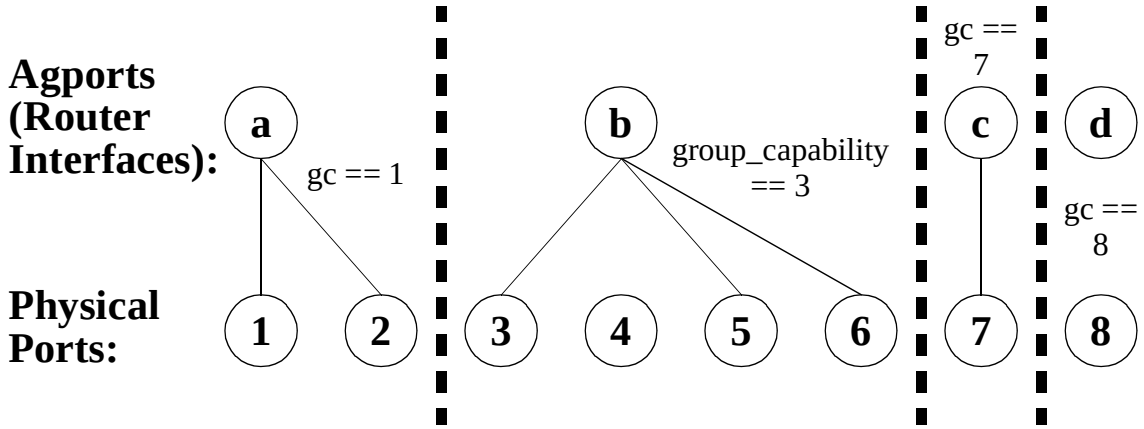

FIGURE 2. Agports and Physical Ports for a Router or File Server

physical ports **3**, **5**, and **6**, (**4** is unavailable, at present) and agport **c** is connected to physical port **7**. Agport **d** is unconnected to its physical port **8**, and hence its ifOperStatus is Down.

The purpose of this two-layer approach provides some degree of stability to the upper layers' agports. Rather than having these interfaces appear, disappear, or change identities as PAgP associations change, the agports merely change operational state.

This mapping technique has the disadvantage that there is no straightforward or predictable correlation between agports and physical ports. However, it more than makes up for this by minimizing the disruption of agports as physical ports come and go. Figure 3 illustrates this feature. Bridge **A**, with four groupable ports, is connected, via two ports each, to bridges **B** and **C**. The wires **2-6** and **3-7** could be swapped, and then the wires **1-5** and **4-8** swapped, thus completely reconfiguring the connections between **A**, **B**, and **C**, without ever causing the agports **a**, **c**, **e**, and **g** to change ifOperStatus. No spanning tree topology change would take place during the swap.

Note finally that VLANs require a certain amount of configuration of specific bridge ports similar to, though smaller in scope than, the configuration of router interfaces. It is left to individual implementations whether to model a bridge as in Figure 1 or in Figure 2, or in some intermediate manner.

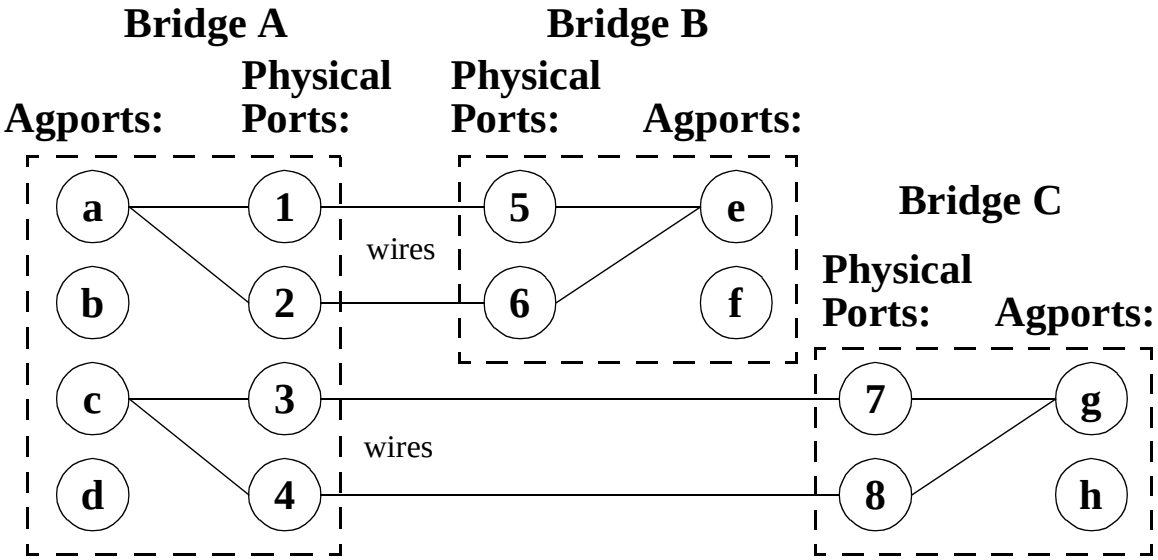

FIGURE 3. Example Bridge Connection

2.2 Statistics

Each agport and physical port has its own set of standard SNMPv2 interface statistics (ifInOctets, ifOutFrames, etc.). However, it is inefficient to count each packet twice, once for the physical port, and once for the agport. There is a trick that allows us to keep the right statistics without double counting.

For each statistic kept, the agport has a corresponding counter. This counter is equal to the negative of the sum of all of the agport's component physical ports' counters at the time the agport was formed. The number reported for any agport statistic is the sum of all its component physical ports, plus the agport's counter. Whenever a physical port is removed from an agport, each of its stats is added to the corresponding agport counter. Whenever a physical port is added to an agport, its current stats are subtracted from the agport's counter.

To be fully accurate, we should count PAgP packets differently from data packets passed through both interfaces; PAgP packets traverse the physical port but not the agport. Also, packets received on the port interface but thrown away without passing up through the agport should be properly counted. Each packet sent or received through the physical port, but not through the agport, is counted *negatively* in the agport's counter. By counting the exceptional packets (packets thrown away or generated by PAgP) twice instead of once, once positively and once negatively, normal data packets can be counted only once.

2.3 Configuration Parameters

Several configuration parameters are required to implement the Port Aggregation Protocol.

2.3.1 Interface Stack

The SNMPv2 interface stack provides the means to display to the system administrator the association of physical ports and agports. Controlling these associations, however, requires variables particular to PAgP.

It must be possible to turn PAgP on and off. When PAgP is turned off on physical ports, there may be no need for agports to exist. There are several obvious means for controlling PAgP and the existence of agports:

1. On a router, the agports could be created and named explicitly, as it is the router interfaces which carry much of the router's configuration. Physical ports would be connected to agports via the CLI with a command on the physical port which refers to the agport to which it may be attached.

   It may not be trivial to separate those parameters that should be attached to the (logical) agport and which to the physical port. But the requirements of VLANs and LANE ELANs are similar to the separation required for PAgP.
2. On a bridge, the agports might be created only if and when PAgP is enabled globally in the device, with one agport created for each physical port. Whether the bridge's ports are associated with physical ports (which may be VLAN virtual trunk ports) or agports depends on whether PAgP is turned on globally. Turning PAgP on or off on a physical port forces that physical port to be connected to its corresponding agport.

2.3.2 Group_capability

Each physical port and agport possesses a configuration parameter called the group_capability. In both Figure 1 and Figure 2, the agports and physical ports are labelled with their group_capability values. A physical port can be aggregated with any other physical port that has the same group_capability, and only with such a physical port. It may be attached to only an agport that shares the same group_capability.

Group_capability values are arbitrary identifiers with no required association with physical port or agport identifiers. Note, however, that the maximum useful number of agports is equal to the number of physical ports. That maximum could be used, for example, in a bridge where no physical ports are aggregated together, so that each is attached to a different agport. An implementation may find it convenient, therefore, to use the physical port number of one member of a set of interfaces of like grouping ability as the value for the group_capability. This technique is used to assign the group_capability values in Figure 1 and Figure 2. It is not a requirement that this technique be used for assigning group_capability values.

A physical port's group_capability may be a function both of hardware limitations and operator configuration. For example:

1. A given implementation may require that only the physical ports residing on the same circuit board are capable of being aggregated into an agport. In that case, physical ports on different circuit boards would never be allowed to have the same values for group_capability.
2. An implementation's software may not be able to properly assign flows among physical ports of differing speeds, even though the hardware is capable of aggregation. The implementation would adjust the group_capability values to reflect this. Another implementation might be able to properly apportion traffic among physical ports of different speeds, and set the group_capability values to allow such aggregation.
3. The system administrator may wish to overwrite the group_capability values to further restrict the grouping possibilities, say, to limit aggregation to pairs of physical ports.

4. The group_capability values may not even be static. A device might, for example, discover by some means outside PAgP that a hub is present on a given port, and alter the group_capabilities to prevent this port from being aggregated.

Developers who implement dynamic group_capability values must be very careful not to introduce instability in the protocol. See Section 5.2.

In any case, the purpose of the group_capability is to express the willingness of one end to aggregate ports. If the other end of the connection finds port differences to be insurmountable, then its group_capability values will reflect this, and the ports will not be aggregated.

2.3.3 Timer Values

Altering the timer values of the Port State protocol is dangerous; it can adversely affect the interoperability of the protocol. The timers are all constant and defined by the protocol definition.

2.3.4 Partner Limitations

It may be useful for any device, but especially for a router, to be able to specify in either the physical port or agport configuration, some limitation on the external devices to which the physical port or agport may be attached. This might be in the form of a list of allowed or disallowed MAC addresses or L3 addresses. Presumably, MAC addresses would be learned through the Port State protocol.

This subject is not addressed by the Port Aggregation Protocol. The problem can be defined in terms of binding agports to configurations.

3.0 Data Frame Forwarding

There are three elements to the forwarding of data frames on PAgP-aggregated agports. Frames must be sent, frames must be received, and it must be possible for the upper layers using PAgP to meet requirements to deliver frames in the same order in which they were transmitted.

3.1 Frame Reception

All frames received from any physical port in an agport are considered to have been received on the agport itself. No distinction, e.g. IEEE 802.1D bridge source address learning, should be based on physical port. Only the agport matters. In particular, device must be prepared to receive broadcast, multicast, unicast floods, and BPDUs on any member physical port of an agport.

3.2 Frame Transmission

A device may use any means it cares to use to select among the physical ports of an agport to transmit a given frame. It should, however, use a different source MAC address on PAgP packets on each different physical port.

3.2.1 Physical Port Selection by Source MAC Address Only

Selecting the physical port for transmission based solely on the source MAC address of a packet makes it slightly easier for some receiving bridge implementations, because source addresses do not skip around from physical port to physical port. However, traffic from one source, e.g. traffic from a router using the same MAC address for multiple VLANs, will not be distributed or load shared. However, that router could, of course, implement another distribution plan, so that the problem would only affect trunks between two switches. Furthermore, this plan may not satisfy the needs of all traffic with regard to packet ordering.

Consider the configuration illustrated in Figure 4. Station A is connected to a switch using the Port Aggregation Protocol on physical ports 1 and 2. Stations B and C are connected by simple wires. Suppose the following sequence of packets is sent:

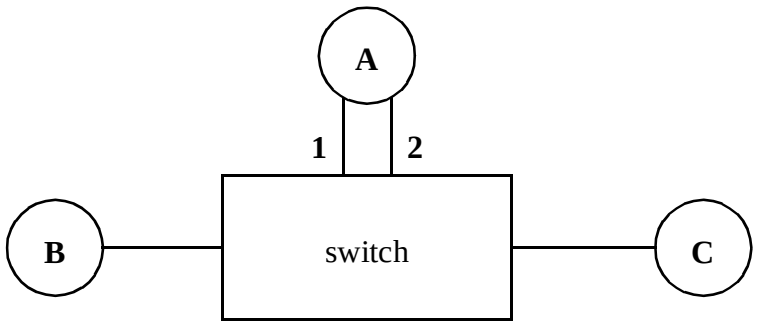

FIGURE 4. Packet Ordering Problems

1. B sends a unicast to A (via port 1).
2. B sends a unicast to C.
3. C sends a unicast containing information obtained from the B->C packet to A (via port 2).

Because physical port 1 could be congested while the others are not, it is entirely possible that station A receives packet 3 from C before receiving packet 2 from B. Similarly, if packets 1 and 2 are combined into a single multicast packet, A could receive the packet from C before receiving the packet from A.

3.2.2 Physical Port Selection by Destination MAC Address Only

The authors know of no misordering problems that can be caused if the physical port is selected based solely on the destination MAC address of a packet. Furthermore, this makes it slightly easier for some transmitting bridge implementations, because a bridge table entry can be tied to a particular physical port. However, traffic addressed to a single destination, e.g. traffic to a router using the same MAC address for multiple VLANs, will not be distributed or load shared. Furthermore, all broadcast traffic is tied to one particular physical port.

3.2.3 Physical Port Selection by both Source and Destination MAC Addresses

Selecting the physical port for transmission based on some function of both the source and destination MAC addresses distributes the traffic across multiple physical ports even if one or the other

interface leads to or from a router. It suffers from the same packet ordering problem, however, as the source MAC address only solution.

3.2.4 Physical Port Selection by Other Means

If two routers are connected by a series of switches, and both use the same MAC address for a number of VLANs, then an aggregated trunk between switches cannot load share based on MAC addresses alone; every packet has the same source and destination. In this case, information from higher layers, such as IP addresses, could be used to distribute the packets. A switch must be careful when doing this, however. If it merely obtains bits from the place where the IP addresses should go, without verifying that a packet is, indeed, an IP packet, it could disrupt a NetBIOS or LAT stream by fetching garbage data on which to base its distribution.

If maintaining the order of packet delivery is not a consideration, one could distribute the packets in many ways, even pseudo-randomly. A round-robin distribution plan, sending each packet in turn to the next physical port attached to an agport, is simple and distributes the load reasonably well. For ideal bandwidth utilization, a device could distribute the frames according to frame size (plus a byte-size equivalent of the inter-frame gaps), and send each frame down the physical port which has sent the least total number of bytes. However, even though routed protocols such as IP usually can tolerate out-of-order delivery of packets, the performance of the protocol stacks in the receiving host may suffer significantly when receiving out-of-order packets.

3.2.5 Hot Standby Transmission Method

Hot Standby transmission mode is the minimal supported use of an aggregated group. In this mode, a device sends all data frames on one of the physical ports within an aggregated group. The other physical ports in the group are used for frame transmission only if the originally chosen physical port becomes unusable. Nonetheless, the device operating frame transmission in Hot Standby mode is required to be able to receive traffic on all of the ports in the aggregated group.

3.2.6 Is The Multicast Packet Ordering Problem Significant?

The multicast ordering question raised in Section 3.2.1 cannot be answered without further investigation. No protocol is known to this author that fails in this manner. However, one can easily imagine a token passing protocol using multicasts that might be disrupted by this ordering problem: B has the token. B sends a data packet to A. B sends the token to C. C sends the token to A. A receives the token from C, then receives the data packet from B. A initiates an expensive and time-consuming token reset/re-acquisition sequence.

It may be perfectly reasonable to use any of the above schemes, (3.2.3 Physical Port Selection by both Source and Destination MAC Addresses is best) and merely caution the user not to use PAgP (or use it only in hot standby mode) if it causes ordering problems. This answer depends, of course, on the unanswered question of whether there are any *common* protocols that would be disrupted by this problem.

3.2.7 Perfect Distribution

A “perfect” distribution scheme can be imagined. The transmitting device attaches ordering information to each frame sent, perhaps using an inserted tag, similar to an 802.1Q tag. It distributes

the packets according to frame size (see Section 3.2.4), thus achieving perfect bandwidth utilization. At the receiving end, the ordering information is used to extract the packets in the proper sequence. This method is not chosen for implementation or further description at this time, because it requires considerably more complex hardware and/or software than the other distribution methods described.

4.0 PAgP Port State

The Port Aggregation Protocol is controlled by a state machine, described in Section 4.2, and a set of per-Physical Port and per-Agport variables, described in Section 4.3.

4.1 PAgP Packet Transmission

The Port Aggregation Protocol would work with an extremely simple algorithm for transmitting PAgP packets: transmit one PAgP on each physical port every time T_H. This method works, but suffers from two drawbacks. Firstly, it is Yet Another Hello protocol, which wastes processing time and bandwidth when it is not required. Secondly, if and when PAgP is changed and/or standardized, both the old and the new versions of the protocol must be supported.

4.1.1 Using a Larger T_H to Minimize PAgP Hellos

A system administrator may wish to make a trade-off between the processing and transmission bandwidth consumed by PAgP packets and the time it takes to react to a loss of connectivity that is not signalled by hardware. That is, the system administrator may wish to increase the time between PAgP packets after the ports have been aggregated and things have settled down.

To enable this feature, the FLAGS field in the PAgP packet defines a slow_hello bit. If a physical port on device A is connected to exactly one other PAgP device B, and if device B's last PAgP packet had the slow_hello bit set in the FLAGS field, then device A may use the alternative, long value, for transmitting its PAgP packets. The slow_hello bit indicates that the device transmitting the bit is using the alternative value T_A for T_H to time out receipt of PAgP packets. If the device receiving the slow_hello bit ignores the bit, nothing bad happens. A device must never use the alternative value T_A for T_H for sending PAgP packets unless it has received the slow_hello bit, and it must revert to the normal, short time, whenever it receives a PAgP packet with the slow_hello bit clear.

Please note that transmitting the slow_hello bit gives permission to one's partners to transmit slowly; it does not indicate that the device transmitting the slow_hello bit is transmitting slowly.

4.1.2 Auto_mode versus Desirable_mode

PAgP can be configured to operate in either of two modes, auto_mode or desirable_mode. In auto_mode, a device sends PAgP packets if and only if the device receives PAgP packets from a device in desirable_mode. In desirable_mode, PAgP packet transmission follows the normal rules in this section. The purpose of this mode difference is to enable the system administrator to absolutely minimize the transmission of PAgP packets. One might, for example, set all edge switches' modes to auto_mode, and all backbone switches to desirable_mode. This would restrict trunking to the backbone and edge-to-backbone links.

Note that turning PAgP to auto_mode (or turning it off) greatly limits its ability to automatically form aggregations and to detect uni-directional links.

4.1.3 Silent partners

It is possible that a physical link (or multiple physical links) are tied to a non-PAgP capable device which seldom, if ever, transmits packets. For example, the link may be connected to a packet analyzer, or to a file server which never speaks unless spoken to. In this case, running PAgP on a physical port will prevent that port from ever becoming operational. This is because the link is indistinguishable from a uni-directional link, which PAgP is designed to protect against.

The system administrator may disable PAgP on ports which may be connected to these “silent partners”. Alternatively, PAgP can be configured to give a reduced capability of detecting uni-directional links, but gain the ability to work with silent partners. This is accomplished by enabling the operation of a link on which no packets have been received after a timeout period.

If a PAgP device is connected to another PAgP device, this configuration option does not disable the ability for PAgP to detect a uni-directional link. Suppose A and B have a uni-directional link, with A able to send to B, but unable to receive from B. A enables the link after the timeout period, and believes it to be operational. B, however, sees A’s PAgP packets, which never reflect B’s presence, so B’s end of the link stays in the S5 state (see Section 4.2) and never becomes operational.

If a PAgP device is connected to a non-PAgP device, and the PAgP device is configured to allow “silent partners”, then no protection against a uni-directional link possible.

4.1.4 PAgP Transmission Configurations

The various combinations of auto_mode and silent partners lead to four possible configurations of the PAgP protocol, of which only three are distinct. These are summarized in Table 1. For each configuration, a state machine is defined in the following sections.

Mode	Silent Partner	Configuration Name	See
auto	uni-directional links detected	TAU	Figure 5
	silent partner OK	TAS	
desirable	uni-directional links detected	TDU	Figure 6
	silent partner OK	TDS	Figure 7

TABLE 1. Allowed Transmission Configurations

4.1.4.1 TAS, TAU: Auto

In these configurations (Figure 5, Table 2), no PAgP packets are sent unless or until a PAgP packet is received. Once PAgP has started exchanging packets, it slows down to one packet every time

T_A, about 30 seconds. Whether or not detection of uni-directional links is performed does not affect the transmission state machine.

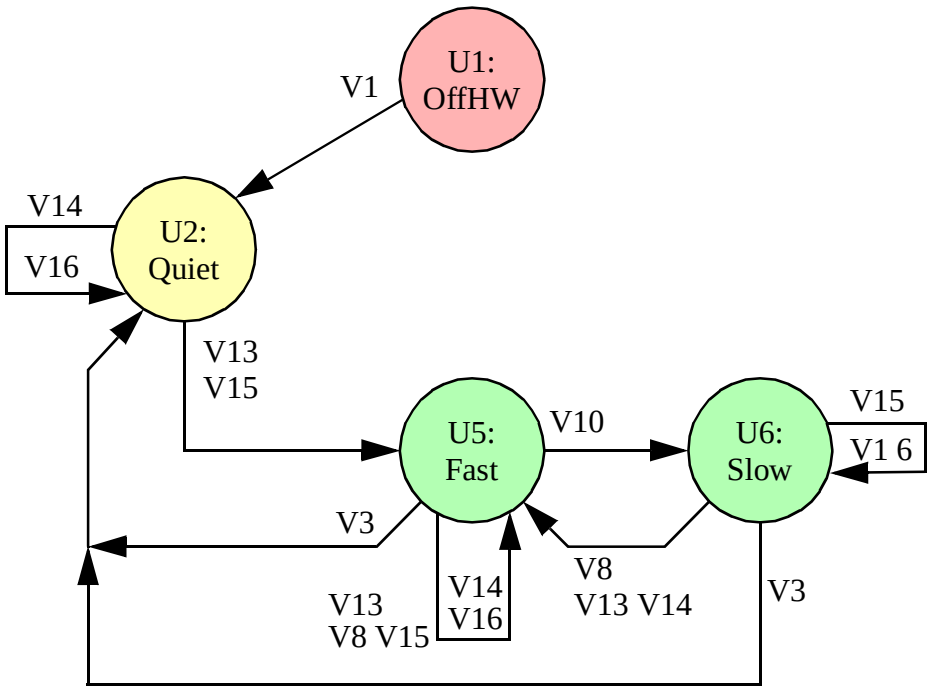

FIGURE 5. PAgP Transmission State Machine: Configuration = TAU or TAS

Old State: Event:	**U1**	**U2**	**U5**	**U6**
V1	B0 U2	Error	Error	Error
V3	Error	Error	B2 U2	B2 U2
V8	Error	Error	B0 U5	B3 U5
V10	Error	Error	B4 U6	Error
V13	Error	B3 U5	B1 U5	B3 U5
V14	Error	B0 U2	B0 U5	B8 U6
V15	Error	B3 U5	B1 U5	B1 U6
V16	Error	B0 U2	B0 U5	B0 U6

TABLE 2. PAgP Transmission State Table: TAS, TAU

4.1.4.2 TDU: Desirable, Uni-detect

In this configuration (Figure 6, Table 3), PAgP packets are transmitted every time T_H (1 second) after the physical link comes up, or every time T_A, about 30 seconds, if no response is received. Uni-directional links are detected and never made operational, so silent partners are not allowed.

Once PAgP has started exchanging packets, it slows down to one packet every time T_A, about 30 seconds.

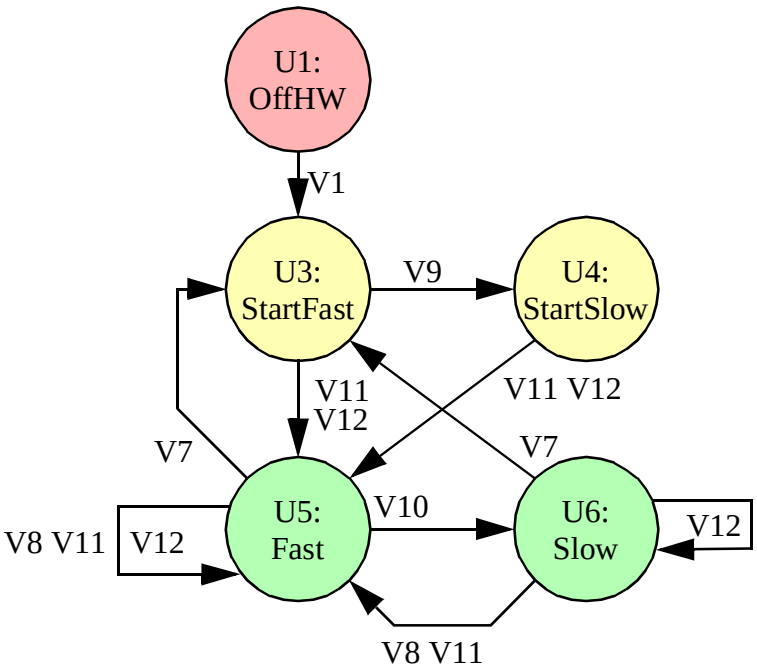

FIGURE 6. PAgP Transmission State Machine: Configuration = TDU

Old State: Event:	U1	U3	U4	U5	U6
V1	B6 U3	Error	Error	Error	Error
V7	Error	Error	Error	B5 U3	B6 U3
V8	Error	Error	Error	B0 U5	B8 U5
V9	Error	B7 U4	Error	Error	Error
V10	Error	Error	Error	B9 U6	Error
V11	Error	B10 U5	B8 U5	B0 U5	B8 U5
V12	Error	B10 U5	B8 U5	B0 U5	B0 U6

TABLE 3. PAgP Transmission State Table: TDU

4.1.4.3 TDS: Desirable, Silent OK

In this configuration (Figure 7, Table 4), PAgP packets are transmitted every time T_H (1 second) after the physical link comes up and continue to be transmitted at this rate as long as the link remains up. Silent partners are allowed.

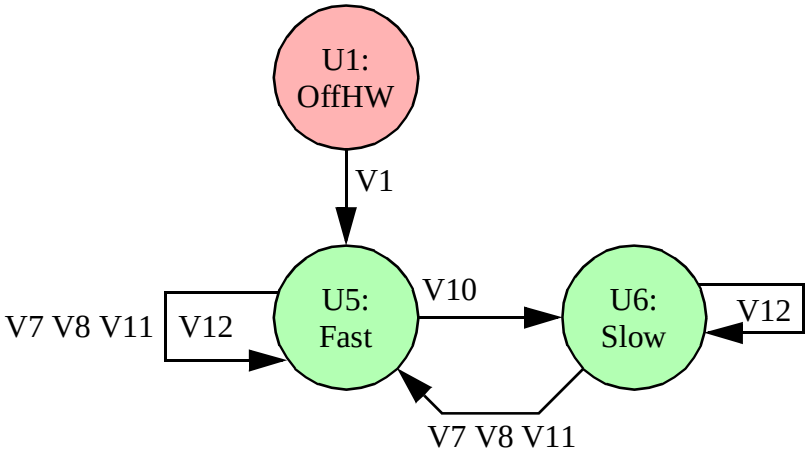

FIGURE 7. PAgP Transmission State Machine: Configuration = TDS

Old State: Event:	U1	U5	U6
V1	B8 U3	Error	Error
V7	Error	B0 U3	B8 U3
V8	Error	B0 U5	B8 U5
V10	Error	B9 U6	Error
V11	Error	B0 U5	B8 U5
V12	Error	B0 U5	B0 U6

TABLE 4. PAgP Transmission State Table: TDS

4.1.5 PAgP Packet Transmission State Machine

Table 5, Table 6, Table 7, and Table 12 contain the keys for interpreting the state machine descriptions in Section 4.1.

State	Name	Meaning
U1	OffHW	Physical port's ifOperStatus is not Up. Nothing is sent through the physical port.
U2	Quiet	Physical port's ifOperStatus is Up. No PAgP packets are sent through the physical port.

TABLE 5. Port Aggregation Protocol Port Transmission States

State	Name	Meaning
U3	StartFast	Use timer T_H to control transmission of PAgP packets. No PAgP partner has been discovered.
U4	StartSlow	Use timer T_A to control transmission of PAgP packets. No PAgP partner has been discovered.
U5	Fast	Use timer T_H to control transmission of PAgP packets.
U6	Slow	Use timer T_A to control transmission of PAgP packets.

TABLE 5. Port Aggregation Protocol Port Transmission States

Name	Action
Error	Report failure of state machine. This state machine reverts momentarily to state U1; it then advances to the next state if the physical port is Up. The main PAgP state machine (see Section 4.2) reverts momentarily to state S1 HWOff; it then advances to state S2 HWOn if the physical port is Up.
B0	Do nothing.
B1	Restart timer T_Q.
B2	Clear timer T_Q.
B3	Restart timer T_Q. Start using timer T_H for transmitting PAgP packets.
B4	Restart timer T_Q. Start using timer T_A for transmitting PAgP packets.
B5	Restart timer T_S.
B6	Restart timer T_S. Start using timer T_H for transmitting PAgP packets.
B7	Clear timer T_S. Start using timer T_A for transmitting PAgP packets.
B8	Start using timer T_H for transmitting PAgP packets.
B9	Start using timer T_A for transmitting PAgP packets.
B10	Clear timer T_S. Start using timer T_H for transmitting PAgP packets.

TABLE 6. PAgP Transmission State Machine Actions

Event	Meaning
V1	Physical port’s ifOperStatus enters the On state.
V2	A valid PAgP packet is received with the auto_mode bit = 0.
V3	Timer T_Q expires.
V5	A valid PAgP packet is received with the auto_mode bit = 1.
V7	The last PAgP partner on this port is lost (same as event E6).
V8	A previously-unknown partner is added to the PAgP partners list
V9	T_S expires.
V10	All partners are tracked, and all have set the slow_hello bit
V11	A PAgP packet is received with the slow_hello bit = 0.
V12	A PAgP packet is received with the slow_hello bit = 1.

TABLE 7. PAgP Transmission State Machine Events

Event	Meaning
V13	A PAgP packet is received with the slow_hello bit = 0 and auto_mode bit = 0.
V14	A PAgP packet is received with the slow_hello bit = 0 and auto_mode bit = 1.
V15	A PAgP packet is received with the slow_hello bit = 1 and auto_mode bit = 0.
V16	A PAgP packet is received with the slow_hello bit = 1 and auto_mode bit = 1.

TABLE 7. PAgP Transmission State Machine Events

4.2 PAgP Port State Machine

The Port Aggregation Protocol Port state machine controls each individual physical (or logical) port to be grouped by the protocol. The basic protocol is pretty simple: As long as the physical port is up (its ifOperStatus is On), PAgP packets are periodically transmitted. If data packets, but no PAgP packets, are received for a while, then assume that the port is connected to non-PAgP-capable devices. Otherwise, listen for PAgP packets that prove that the physical port has a bi-directional connection to another PAgP-capable device. As soon as two such packets have been received on some physical port, try to form an agport. If the PAgP packets stop for a long time, destroy the agport.

Figure 8, Table 8, Table 9, Table 10, Table 11, and Table 12, taken together, largely define the Port Aggregation Protocol state machine.

Old State: Event:	S1	S2	S3	S4	S5	S6	S7	S8
E0	A0 S1	A1 S1	A1 S1	A1 S1	A1 S1	A1 S1	A1 S1	A1 S1
E1	A0 S2	Error	Error	Error	Error	Error	Error	Error
E2	Error	A2 S3	A3 S3	A4 S4	A3 S5	A3 S6	A4 S7	A4 S8
E3	Error	Error	A11 S4	Error	Error	Error	Error	Error
E4	Error	A5 S5	A5 S5	A10 S5	A5 S5	A5 S5	A10 S5	A5 S8
E5	Error	A6 S6	A6 S6	A10 S6	A6 S6	A7 S7	A6 S7	Error
E6	Error	Error	Error	Error	A1 S2	A1 S2	A1 S2	Error
E7	Error	Error	Error	Error	Error	A8 S7	Error	Error
E8	Error	Error	Error	Error	A9 S8	A9 S8	A1 S2	A5 S8
E9	Error	Error	Error	Error	Error	Error	Error	A1 S2
E10	Error	Error	Error	Error	A1 S2	A1 S2	A1 S2	A1 S2

TABLE 8. Port Aggregation Protocol State Table

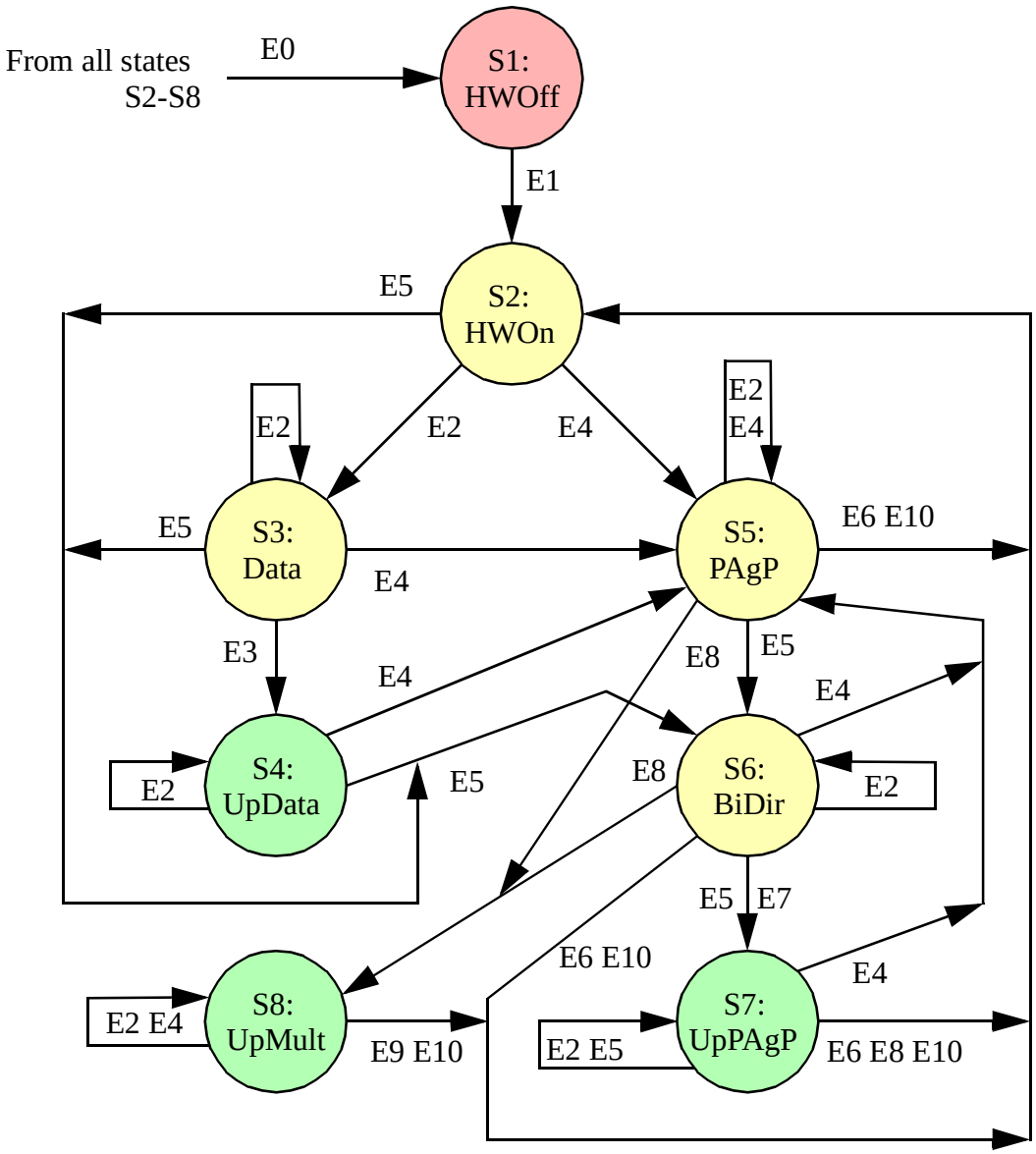

FIGURE 8. PAgP Port State Engine

Name	Action
Error	Report failure of state machine. This state machine reverts momentarily to state S1 HWOff; it then advances to state S2 HWOn if the physical port is Up. The transmission state machine (see Section 4.1) reverts momentarily to state U1; it then advances to the next state if the physical port is Up.
A0	Do nothing.
A1	Disconnect physical port from any agport. Clear PAgP variables. Reset timers.
A2	Start timer T_I. Drop received packet.
A3	Drop received packet.

TABLE 9. Port Aggregation Protocol Actions

Name	Action
A4	Pass received packet up to agport
A5	Store received my_data in other_data[0]. Start T_p. Clear T_i.
A6	Re-start T_p. Clear T_i.
A7	Re-start T_p. Attach this physical port, along with any other physical ports which have reached the BiDir or UpPAgP states and are groupable with this port, to an agport.
A8	Attach this physical port, along with any other physical ports which have reached the BiDir state and the one which reached the UpPAgP state, and which are groupable with this port, to an agport. Re-start T_p.
A9	Store received my_data in other_data[0]. Start T_p. Attach this physical port, and this physical port only, to an agport.
A10	Disconnect physical port from any agport. Store received my_data in other_data[0]. Start T_p. Clear T_i.
A11	Attach this physical port, and this physical port only, to an agport. Clear T_i.

TABLE 9. Port Aggregation Protocol Actions

State	Name	Meaning
S1	HWOff	Physical port's ifOperStatus is Down. Nothing is sent or received through the physical port.
S2	HWOn	No packets of any kind have been received. Physical port is not connected to any agport. PAgP packets are sent and may be received.
S3	Data	One or more non-PAgP packets have been received, but no PAgP packet has been received. Timer T_I has not expired. Physical port is not connected to any agport. PAgP packets are sent and may be received.
S4	UpData	No PAgP packets have been received. PAgP packets are sent. Physical port is the only physical port connected to its agport. Non-PAgP packets are passed in and out between physical port and agport.
S5	PAgP	One or more PAgP packets have been received which do not (yet) prove a bi-directional connection exists, or a PAgP packet has been received which contradicts information received in previous PAgP packets. Physical port is not connected to any agport. PAgP packets are sent and may be received.
S6	BiDir	Exactly one PAgP packet has been received that proves a bi-directional connection to exactly one neighbor exists since state S5 was entered. Physical port is not connected to any agport. PAgP packets are sent and may be received.

TABLE 10. Port Aggregation Protocol Port States

State	Name	Meaning
S7	UpPAgP	This physical port, perhaps in association with other physical ports, is connected to an agport. PAgP packets are sent and received on the physical port. Non-PAgP packets are passed in and out between physical port and agport.
S8	UpMult	PAgP packets have been received from multiple neighbors. Physical port is connected to one agport, and that agport is not connected to any other physical port. Non-PAgP packets are passed in and out between physical port and agport.

TABLE 10. Port Aggregation Protocol Port States

Event	Meaning
E0	Physical port's ifOperStatus leaves the On state.
E1	Physical port's ifOperStatus enters the On state.
E2	Any non-PAgP packet is received. (Packets for protocols which run "below" PAgP, on the physical port, do not trigger event E2.)
E3	Timer T_I expires.
E4	A PAgP packet is received that 1) does not prove that a bi-directional connection exits; or 2) is inconsistent with the last PAgP packet received. See also E8.
E5	A PAgP packet is received that 1) proves that a bi-directional connection exists with exactly one neighbor; and 2) [is compatible with the last-received PAgP packet or is the first PAgP packet received].
E6	Either the last neighbor's timer T_P expires on this physical port, or the last timer T_P expires on another physical port, and less than T_H (T_A if the physical port's pp_partner_data[0].slow_hello flag is set) remains before this physical port's last neighbor's timer T_P is to expire. See also Section 4.7.
E7	Another physical port receives a PAgP packet and enters the UpPAgP state, and this physical port is in the BiDir state.
E8	A PAgP packet is received from a second (or higher) neighbor. That is, at least two different {device_id, sent_port_ifindex} pairs have been received on this physical port and timer T_P has not expired for either neighbor.
E9	A timer T_P expires on this physical port, leaving only one neighbor in the port's list of neighbors.
E10	Some unspecified condition is detected, presumably one related to data received in a PAgP packet, which causes the physical port's group_capability value to change. This event takes precedence over any other events that would otherwise have been caused by the reception of the PAgP packet.

TABLE 11. PAgP Port State Events

Timer	Usage
T_H	Interval between sending PAgP packets. **Value 1 second.**
T_A	Alternate interval between sending PAgP packets. **Value 30 seconds.**

TABLE 12. Port Aggregation Protocol Timers

Timer	Usage
T_I	Interval to wait after first data packet is received and during which, no PAgP packets are received, before giving up on PAgP and turning on the interface. There is one T_I timer per physical port. **Value 3 seconds.**
T_P	Interval after receiving the last PAgP packet before giving up on PAgP for a specific neighbor. There is one T_P timer for each neighbor being tracked. **Value $3.5T_H$ or $3.5T_A$, as appropriate.**
T_F	Interval after sending a flush message before giving up and retrying. **Value 1 second.**
T_Q	Interval to wait before halting the transmission of PAgP packets. Receiving a PAgP packet resets T_Q. **Value $7T_A$ == 270 seconds or $7T_H$ == 7 seconds, as appropriate.**
T_S	Interval to wait before switching from using T_H to T_A. **Value $8T_H$ == 8 seconds.**

TABLE 12. Port Aggregation Protocol Timers

4.3 PAgP Physical Port Variables

Table 13 shows examples of the data structures associated with each physical port, along with the Port Aggregation Protocol packet format, itself.

```
#define NPARTS 10         /* max partners tracked (min 2) */
typedef uchar  unsigned char;
typedef uint   unsigned int;

typedef struct PAgP_end_data_ {
    uchar   device_id[6];          /* Perhaps the MAC address */
                                   /* of first physical port */
    uint    sent_port_ifindex;     /* Port packet was sent on */
    uint    group_capability;      /* See text */
    uint    consistent_state;      /* See Text */
    uint    group_ifindex;         /* See text */
    uint    slow_hello;            /* Use TA for TH */
} PAgP_end_data;

typedef struct PAgP_phys_port_data_ {
    PAgP_end_data  pp_my_data;            /* myself */
    timer          pp_timer_i;            /* no-PAgP timer */
    PAgP_end_data  pp_partner_data[NPARTS];/* my partners */
    timer          pp_timer_p[NPARTS];    /* T[P] timers */
}
```

TABLE 13. Port Aggregation Protocol Packet and Variable Formats

The meanings of the variables:

device_id: Same as pagpDeviceId. A six-character value which globally and uniquely identifies a physical device. A six-character format is chosen to fit an IEEE 48-bit address. This does not imply that PAgP can only be used on IEEE 802 LANs; it merely means that every

device requires a globally-unique identifier.

The identifier may refer to the “box”, or to an individual circuit card. However, if two circuit cards in the same box have different device_id values, they cannot be aggregated together.

sent_port_ifindex: The interface MIB ifIndex of the physical port on which the PAgP packet was sent. Each physical port must have a sent_port_ifindex that is unique over all physical ports sharing the same device_id.

group_capability: See Section 2.3.2. Same as pagpOperGroupCapability.

consistent_state: The agport’s ifOperStatus must not reach the On state until the last PAgP received on each of its component physical ports had the consistent_state flag set.

group_ifindex: Same as pagpGroupIfIndex. The ifIndex of the agport to which this physical port is attached. Contains 0 if the physical port is not currently attached to any agport. May contain the interface’s own ifIndex if the interface is “aggregated” only to itself.

slow_hello: Same semantics as pagpHelloFrequency. If set in pp_partner_data[0], the device may use T_A, instead of T_H, for transmitting PAgP packets. If set in pp_my_data, the device uses 3.5 T_A, instead of 3.5 T_H, for the value of the T_P, used to time out the other device’s PAgP packets.

partner_count: Same as pagpPartnerCount. The number of non-empty pp_partner_data entries on this physical port, or all-1s, if all pp_partner_data entries are non-empty.

4.4 Receiving PAgP Packets

Each time a PAgP packet is received on a PAgP-enabled physical port, the following steps must be taken:

1. If the received my_data.device_id is zero, the PAgP packet is invalid.
2. The received my_data.device_id and my_data.sent_port_index are compared against the physical port’s pp_partner_data[i] values for all i. If they match an entry i, that entry is selected. If they match no entry, and some entry is empty, then that empty entry is selected. If they match no entry, and no entry is empty, then the entry i whose timer is nearest expiring is cleared to zero, and that entry is selected.
3. If the selected i is not zero, then the contents of pp_partner_data[0] and pp_timer_p[0] are swapped with pp_partner_data[i] and pp_timer_p[i].
4. The detailed tests described in Section 4.5 are performed to determine the exact event (E4, E5, or E8) that this received PAgP packet is to trigger.
5. The contents of the received PAgP packet’s my_data is copied to the physical port’s pp_partner_data[0].
6. pp_partner_data[0].timer is reset to T_P.
7. The physical port’s partner_count variable is updated.

Note that, with these rules, the physical port’s pp_partner_data[0] always contains information relevant to the last-received PAgP packet.

It is possible that the receiver is unable to distinguish among the physical ports of an agport when receiving PAgP packets. In this case, the receiving device must scan the pp_partner_data[i] structures of the physical ports belonging to the agport on which the PAgP packet was received. The

PAgP packet should match only one physical port's state. The algorithm can then be applied as if the PAgP packet were received on that physical port.

All devices participating in PAgP must be able to transmit PAgP packets on individual physical ports. A device may utilize a range of source MAC addresses in order to accomplish this.

The number of pp_partner_data structures kept by a device is an implementation matter. A minimum of two of these structures must be kept in order to reliably distinguish between point-to-point connections and multidrop connections. See also Section 4.8.

4.5 Finding Bi-Directional PAgP Ports

After all the steps in Section 4.4 have been completed, the received PAgP packet must be examined to see whether it does or does not indicate a bi-directional physical port. The tests which must match to identify the physical port as bi-directional are:

1. Either the received my_data.device_id, my_data.sent_port_ifindex, and my_data.group_capability fields must match the corresponding values in the physical port's pp_partner_data[0], or this physical port's pp_partner_data[0] must contain all 0s.
2. All of the physical port's other pp_partner_data[i].device_id variables must contain all 0s.
3. The received your_data.device_id, your_data.sent_port_ifindex, and your_data.group_capability values must match the corresponding values in the physical port's pp_my_data structure. The value of slow_hello need not match.
4. Either the received my_data.group_ifindex matches the port's pp_partner_data[0].group_ifindex, or the latter is 0 and the former is non-zero.
5. The received my_data.sent_port_ifindex must be different from the my_data.sent_port_ifindex saved from this same device on any other physical port.
6. The received packet's partner_count variable must contain the value "1".

The check in point 4. above is not strictly necessary. The purpose of trading group_ifindex variables is really to provide a debug tool. It allows us to be sure that the protocol did, in fact, work. It should not be possible to override a device's group_ifindex variables independently of the other parameters. This could introduce a potential instability in the protocol.

Note that these checks can result in the formation of a loop-back connection from one or more ports of a device to one or more other ports of the same device. This is perfectly allowable in many situations; it is up to the protocol layers above PAgP to determine whether this is an allowable condition. The information in the PAgP variables can be used to detect loop-back connections and report them to higher layers.

4.6 Consistent_state Flag

It is possible that reprogramming the hardware in a device to change its port aggregation state may take some time, even seconds. If one device believes that the other device is aggregated, but that other device's hardware is not yet aggregated, then loops and/or duplicate frames can result. Therefore, a device should transmit PAgP with the consistent_state flag cleared if it is transmitting PAgP packets that indicate aggregation is possible, but its hardware is not yet aggregated. This is most easily accomplished by setting the consistent_state flag only after reaching state S7 UpPAgP

or S8 UpMult. When all of the partners' PAgP packets on that agport indicate that the partner is in the consistent state, then the agport can be safely enabled.

4.7 Timer T_P

Whenever any pp_partner_data[i].timer expires, that pp_partner_data entry is reset to all 0s, and the physical port's partner_count is updated. Whenever the partner_count transitions from non-zero to zero in this manner, event "E6" in the PAgP state machine is triggered. Event "E6" is not triggered if multiple partners are known to the physical port and one of them times out. Whenever the partner_count transitions from two to one in this manner, event "E9" in the PAgP state machine is triggered.

4.8 Detecting Uni-directional Physical Ports on a Multidrop Connection

If every device on a multidrop connection (e.g. devices connected to an Ethernet hub with several ports) is running PAgP, and if every device is maintaining as many pp_partner_data structures as there are total PAgP devices on the hub, then it is possible to get an indication if any one of those devices' physical port interfaces are (erroneously) operating in a unidirectional manner, either transmit only or receive only. This is accomplished by comparing the received and transmitted partner_count values.

If all the partner_count values are equal, then there is a high likelihood that all devices' interfaces are bi-directional. If one transmitted partner_count is zero and all others are the same value, the odd device likely has a transmit-only interface, but does not know it. In the case of IEEE 802.1D bridges in this situation, the other devices must be very careful; the odd device is likely to transmit data on the connection.

If one device's own partner_count is one higher than all its neighbors' partner_count values, then that one device is likely correct in assuming that it has a receive-only interface, and the other devices are unaware of this. This situation is easier to deal with; the device suspecting the defect can keep the physical port disconnected from any agport.

If one assumes that only single points of failure are likely, this technique is helpful. However, if multiple failures are possible (e.g. if the hub breaks), then the counts are much less reliable, and transmitting a full list of neighbors in every PAgP packet would be required. This is a straightforward extension of PAgP, but it is not recommended for implementation in Cisco at this time.

Unless we add the list of neighbors' device_ids to the PAgP packets, it is recommended that the number of pp_partner_data structures maintained by any PAgP device be two. More than that are of limited value, and fewer make the detection of multiple partners unreliable. Should we add a list of device_ids to the PAgP packets, a higher number would be appropriate. Two hundred neighbors could easily be accommodated in one 1500-byte PAgP packet.

5.0 Port Aggregation Protocol Group Management

When a physical port reaches the "S4: UpData", "S7: UpPAgP", or "S8: UpMult" states, it is attached to an appropriate agport. When it leaves either of those states for another state, it is detached from its agport.

5.1 Normal Aggregation Procedures

The outline of the procedure is:

1. Both ends of both connections must agree on what the grouping is going to be. The grouping agreed to is simply the largest group that can be formed that is permitted by both ends of the connection. The two ends do not have to turn on grouping at the same time. Rather, each receiver must be programmed to know the grouping before the corresponding transmitter can be enabled for data and BPDUs.
2. When a physical port with a given pp_my_data.group_capability value reaches the "S4: UpData" state, it may be assigned to any agport which has no physical ports assigned to it.
3. When a physical port reaches the "S7: UpPAgP" state, it is assigned to the agport all of whose member physical ports match the new physical port's pp_my_data.group_capability, pp_partner_data[0].device_id, and pp_partner_data[0]. group_capability fields, and which are in the "S6: BiDir" or "S7: UpPAgP" states. (Any such BiDir ports are moved to the UpPAgP state at the same time.)

   If there are no agports whose constituent physical ports' parameters are compatible with the newly-ready physical port, that physical port is assigned to an agport with suitable parameters which has no associated physical ports. (If none can be found, the protocol fails. The device's configuration may be in error.)

   The agport selected may already be connected to other physical ports. If not, the means for selecting an agport is not specified by this document.

   By triggering the aggregation on state S7, but including S6 ports in the aggregation, flapping of the agport is minimized. In normal circumstances, all of the ports will be aggregated at once, instead of having the agport come up and down, growing by one physical port at a time.
4. There is no need for numeric correspondence between physical ports and agports. You start things off in this way, but if the cables are swapped around, it is possible to maintain continuous connections (and cause no spanning tree interruptions) only by allowing the physical and agports to get mixed up.
5. Whenever event E6 (PAgP timeout) occurs on the last neighbor known on the physical port, the port timing out is removed from the agport. At the same time, all physical ports on the same agport whose timers have less than T_H remaining are also removed. This enables an agport whose other end has died to come down all at once, instead of coming down one physical port at a time.
6. Some means of notifying the Spanning Tree Protocol when the speed of an agport changes is needed. STP may want to recalculate the spanning tree when the addition or subtraction of a physical port to/from and agport changes the speed of the agport. This is an implementation question, however, not a protocol question.
7. When multiple neighbors are known via PAgP, the physical port is attached to exactly one agport, and never aggregated.
8. If no suitable agport can be found for a physical port (or ports) that reach the appropriate state, then an error has been made either in the configuration of the devices' agports and/or physical ports, or in the wiring connections between devices. In this case, the device discovering the error should alter its group_capability values to correct the error, and presumably,

report the error. Correcting the error might entail preventing the new aggregation from coming up, or might entail destroying an existing aggregation to allow the new one to usurp an agport.

5.2 Dynamic Group_capability Values

Changing group_capability values dynamically, as suggested at several points in this document, introduces the possibility of unstable oscillations that would prevent PAgP from converging. In order to prevent oscillations, the following rules are required of any device which can change group_capability for any reason other than operator intervention:

1. All changes to group_capability as a result of the exchange of PAgP packets must be made in the direction of preventing aggregations. That is, physical ports with the same group_capability values may be changed to have different values, but a physical port may never be changed (as a result of a PAgP exchange) to have the same group_capability value as another physical port.
2. Changes to group_capability that allow greater aggregation can be made only to ports which have been physically disconnected, as a result of operator intervention via operator command or SNMP, or by a change in the state of a lower level protocol.

The presumption here is that dynamic group_capability changes are required only when the devices or the wiring is improperly configured, and that operator intervention is an acceptable means of correcting the situation.

5.3 Line Speed and Bridge Port Costs

The line speed of an agport is the total of the line speeds of each of its physical ports As physical ports join and leave an agport in a router, the routing algorithms will probably wish to follow the changes in the agport's line speed, perhaps with some hysteresis in order to avoid excessively frequent changes.

In a bridge, each bridge port has an associated port cost, which is related to the port's line speed. A change to a bridge port's port cost may well trigger a spanning tree topology change, either in that bridge, or in a bridge further from the root bridge. If the port cost of a bridge port, which is attached to an agport, varies each time that a physical port joins or leaves that agport, then each join or leave has a potential of causing a spanning tree topology change. But, one of the goals of PAgP is to avoid spanning tree topology changes.

In order to avoid these spanning tree topology changes, it is recommended that implementors of PAgP in bridges take the following steps:

1. Port costs should be calculated form the bandwidth of the agport. 1000/speed (speed in Mbits/s) is the standard means of calculating port cost from bandwidth. If a table is employed that violates this calculation, the implementation can interpolate in the table.
2. Manually-configured aggregations should have a port cost equivalent to the maximum line speed possible in the configuration.
3. Aggregations formed by PAgP should use the line speed calculated when the agport is first formed.
4. The line speed should change as little as possible after the first BPDU is sent, and preferably, only when manual configuration changes are made.

It is largely a local matter to the bridge whether this advice is followed, although a diversity of methods for handling speed changes across multiple bridges will increase the frequency of spanning tree topology changes. There are possibilities for improvement to the basic behavior, above:

1. If an agport's ifOperStatus goes to Down, then its port cost should be re-calculated when it comes back Up.
2. If an agport's bandwidth degrades substantially (e.g. 400 Mb/s down to 100 Mb/s), and remains there for some time, a bridge may well wish to make the change.

Other methods can be imagined, such as a bridge updating all its port costs when it first becomes aware of a spanning tree topology change. Such techniques are for further experimentation and study.

6.0 Flush Protocol

Given that physical ports can be added and removed from agports, it would be very limiting to say that a device could never change the assignment of a given source/destination/priority triplet from one physical port to another within an agport. If such a change is made, and the transmitting device is to ensure that frames with the switched flow triplet are delivered in order, then either a transmission delay or a flush mechanism similar to that used in ATM LANE is required.

To use the flush protocol, a device sends a flush packet down the old physical port, the one that the flow is moving from. The flush packet must contain the device_id of both devices and a transaction_id which is changed (presumably incremented) each time a flush message is sent on any member of that agport. The receiving device returns that flush message to the sender. Once the original sender receives the flush message, it can start sending flow data down the new physical port. Between sending and receiving the flush message, the sender must either 1) hold in a queue; or 2) discard any frames on that flow. If the frames are held in a queue, the sender must be careful to drain the queue before allowing new packets to use the new path.

Note that a device must examine both the my_data.

If time T_F expires after sending the flush message without receiving a reply, then the sending device can retransmit the flush message. The sending device must not switch from the old path to the new without waiting a full IEEE 802.1D packet delay time.

The value of 1 second is chosen for T_F to match the "Maximum Bridge Transit Delay" of IEE 802.1D. A bridge is guaranteed not to hold any frame for longer than this period without either forwarding or discarding the frame. If a device is transmitting frames to another device which is not an 802.1D bridge, there is no well-defined time sufficient to guarantee against out-of-order packet reception. One second should, however, suffice for any reasonable implementation.

It is not required that a device send flush messages. It may instead discard frames for a moved flow (or for all flows on the agport) for one 802.1D forwarding delay time after switching paths. But is should not simply take its chances and (sometimes) deliver out-of-order packets.

It is required that a device always return flush messages to the sender. If flush messages are not echoed, then serious delays in traffic could be caused by a device that does use flush messages.

Note that there is no problem caused by non-PAgP devices which do not echo flush messages. Any device that operates PAgP is required to echo flush messages, and no device is allowed to send flush messages unless PAgP is in use.

7.0 Implementation Notes

This section is reserved for outstanding design and/or implementation issues. It may be used for specific implementation notes, as well.

1. It may be possible to make some additional transitions in the state machine summarized in Table 8. In particular, if an agport has only a single physical port associated with it, and if another device neighbor appears on that port, then a transition directly from state S7: UpPAgP to S8: UpMult could be made. This might happen, for example, if several PAgP-capable devices each have a single physical port attached to the same hub.
2. It is not clear that it is always an error condition when physical ports are operational, but have no suitable agport to which they can be attached. This can be an implementation decision. However, the use of PAgP to make connections to specific neighbors is not intended.
3. For some implementations and/or installations, the Flush protocol may be dispensed with. For example, if IP is the only protocol in use, Flush is not necessary. However, the ability to recognize Flush messages and return them to the sender must not be omitted from an implementation. This could severely impact the recovery time of a device that does use the Flush protocol.