F Expander Operation (informative)

F.1 Expander Connection Management examples

The following examples are intended to provide additional insight into how SAS Expanders process connection requests.

F.1.1 Basic Connection example

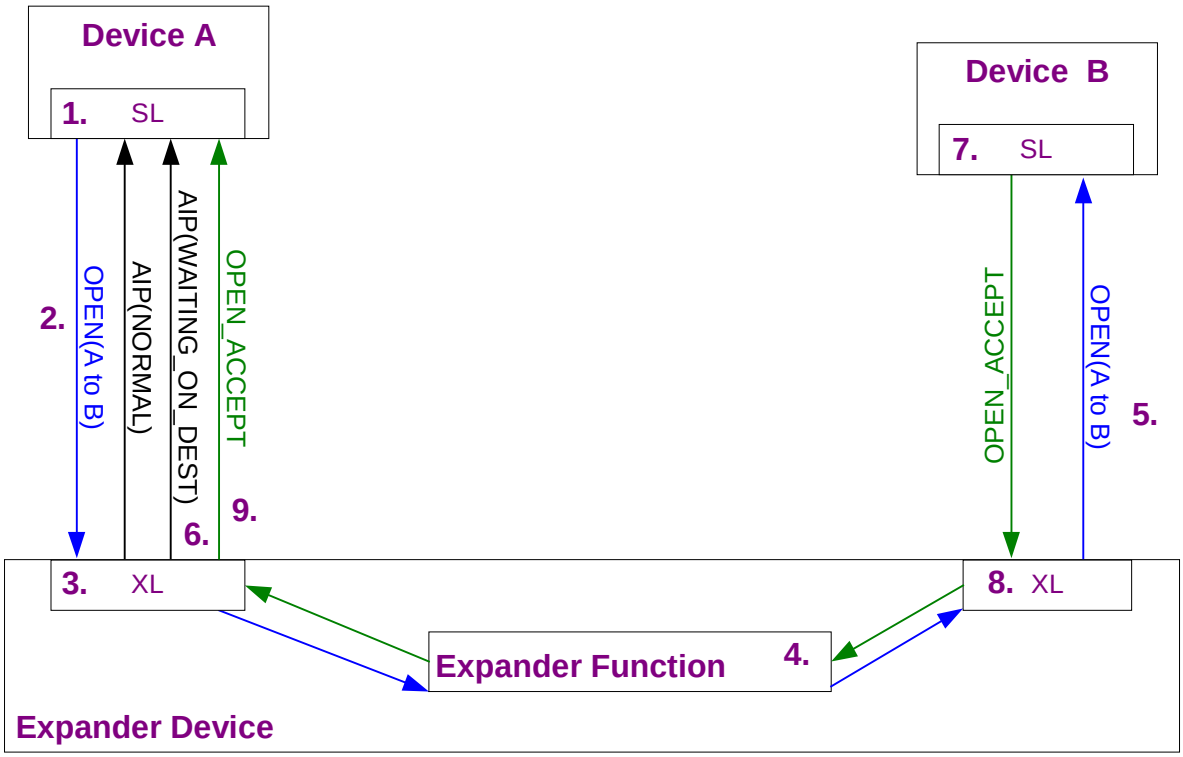

Figure 1. Basic Connection Through Expander

Figure 1 illustrates a simple connection between two SAS devices. The sequence of events involved in servicing a connection request are summarized in Table 1.

Table 1: Connection Establishment Sequence

Step	Device A SL state	Device A TX	Device A RX	Expander Link A XL state	Expander Link B XL state	Device B RX	Device B TX	Device B SL state
1	Idle			Idle	Idle			Idle
2	Arbsel	SOAF						
		OPEN (A to B) address frame						
		EOAF						
3			AIP (NORMAL)	Request_ Path				
4				Request_ Open				
5					Forward_ Open	SOAF		
				Open_ Confirm_ Wait		OPEN (A to B) address frame		
						EOAF		
6, 7			AIP (WAITING_ ON_DEST)		Open_ Response_ Wait			Selected
8							OPEN_ ACCEPT	
								Connected
					Connected			
9			OPEN_ ACCEPT	Connected				
	Connected							

F.1.2 Backoff and Retry example

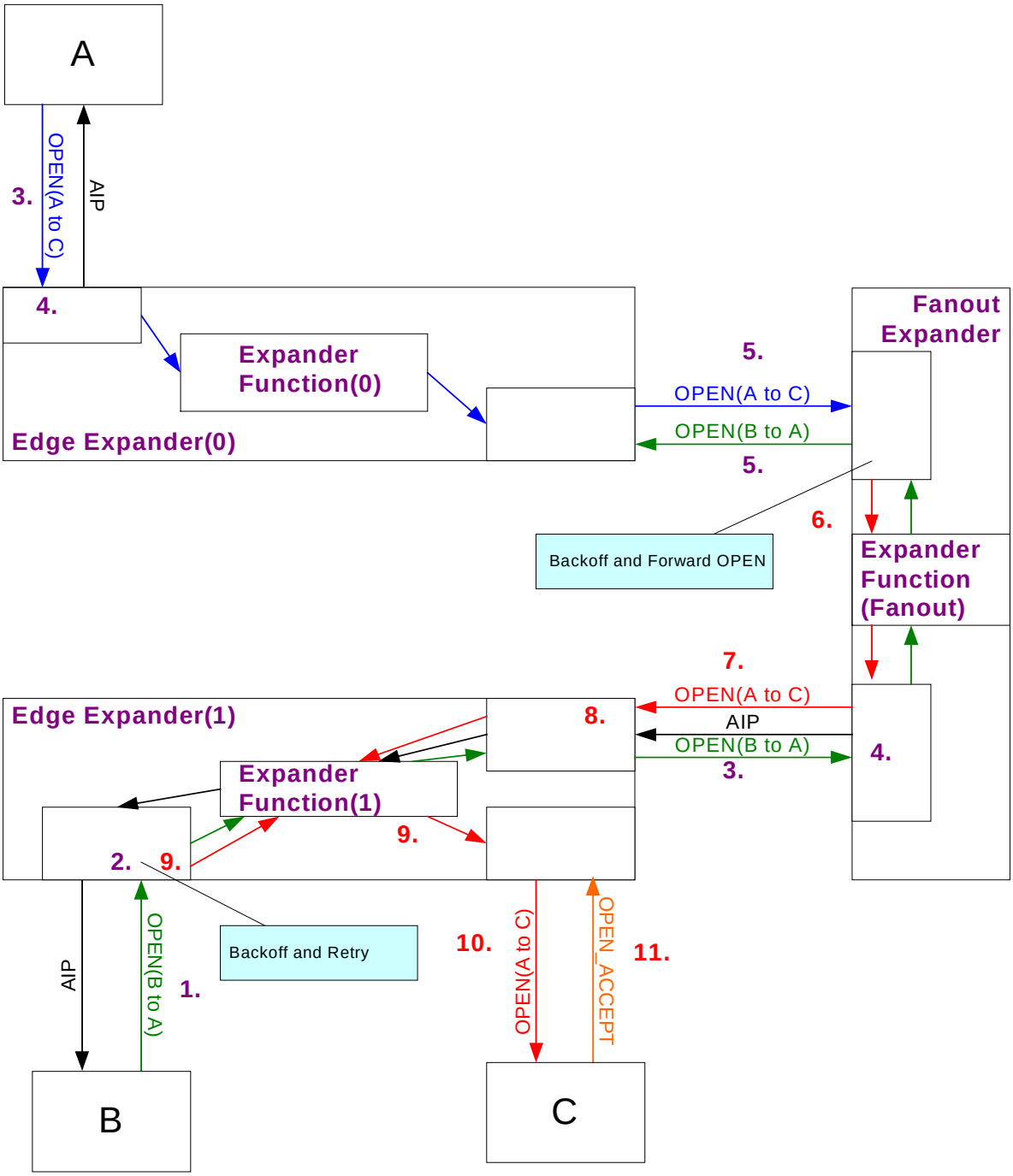

Figure 2. Backoff and Retry Expander Example

The example represented by Figure 2 and subsequent text is intended to provide an overview of the sequence of events surrounding the backoff and retry semantics which are encapsulated within the Expander Link (XL) state machine.

1. Device B sends OPEN address frame (B to A).
2. Edge Expander(1) receives OPEN address frame, validates CRC, allocates path resources, and forwards to destination link.
3. Edge Expander(1) sends OPEN address frame (B to A) to Fanout Expander. Device A sends OPEN address frame (A to C).
4. Fanout Expander receives OPEN address frame (B to A), validates CRC, allocates path resources, and forwards to destination link. Edge Expander(0) receives OPEN address frame, validates CRC, allocates path resources, and forwards to destination link.
5. Fanout Expander sends OPEN address frame (B to A) to Edge Expander(0). Edge Expander(0) sends OPEN address frame (A to C) to Fanout Expander. The two OPEN address frames pass on the wire with no intervening AIP. Both Edge Expander(0) and Fanout Expander must determine what to do next.
6. Assume that the arbitration priority for OPEN(B to A) is lower than for OPEN(A to C). Edge Expander(0) determines that the OPEN address frame it sourced is of higher priority, so it just waits. Fanout Expander receives a higher priority OPEN with a destination address along the same pathway already allocated for this connection attempt. Fanout Expander retains path resources and causes the two links to effectively switch roles between connection initiator and responder.
7. Fanout Expander sends OPEN address frame (A to C) to Edge Expander(1).
8. Edge Expander(1) receives OPEN address frame (A to C). Because it previously received AIP from the Fanout Expander, it must unconditionally back off it's outstanding connection request. Further, because the destination address requires different pathway resources, Edge Expander(1) arbitrates for link(c) and instructs link(a) to release it's path resources and re-issue its request for a new pathway.
9. Edge Expander(1) allocates pathway and indicates OPEN address frame (A to C). Edge Expander(1) re-requests pathway resources for OPEN (B to A).
10. Edge Expander(1) sends OPEN address frame (A to C) to Device C.
11. Device C sends OPEN_ACCEPT and the OPEN_ACCEPT propagates back to Device A.

F.1.3 Pathway Blocked / Pathway Recovery example

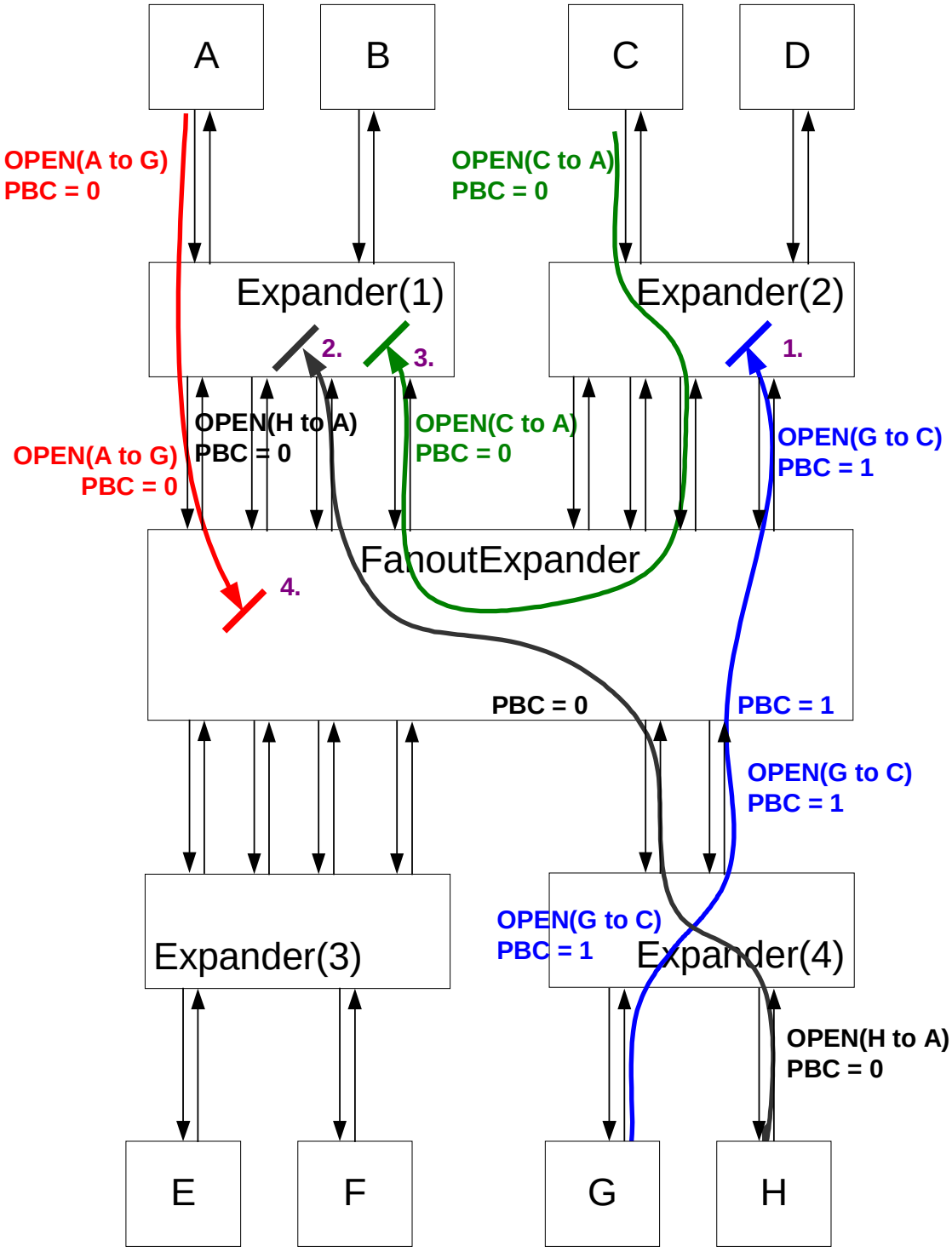

Figure 3. Pathway Recovery Example

The example represented by Figure 3 and subsequent text has several partial pathways in the topology with a circular dependency. The following possible sequence of events illustrate how path conflicts are resolved.

1. PPT timer expires for the OPEN(G to C) connection request in Expander(2). Because the Pathway Blocked Count (PBC) value of OPEN(G to C) > PBC of OPEN(C to A) Expander(2) will not change state, i.e. it will wait for resolution elsewhere in the topology.
2. PPT timer expires for the OPEN(H to A) connection request in Expander(1). Because the PBC of OPEN(H to A) = PBC of OPEN(A to G), Expander(1) must compare SAS addresses to determine whether OPEN(H to A) should wait or backoff (cause OPEN_REJECT(PATHWAY BLOCKED) to be sent to source of connection request).

   Assume SAS address H > SAS address A so H waits, resolution must come elsewhere in the topology.

   If SAS address H < SAS address A then H would have sent OPEN_REJECT(PATHWAY BLOCKED) which would have then freed resources to enable OPEN(A to G) to complete a connection, removing the deadlock condition…

3. PPT timer expires for the OPEN(C to A) connection request in Expander(1). Because the PBC of OPEN(C to A) = PBC of OPEN(A to G), Expander(1) must compare SAS addresses to determine whether OPEN(C to A) should wait or backoff (cause OPEN_REJECT(PATHWAY BLOCKED) to be sent to source of connection request).

   Assume SAS address C > SAS address A so C waits, resolution must come elsewhere in the topology.

   If SAS address C < SAS address A then C would have sent OPEN_REJECT(PATHWAY BLOCKED) which would have then freed resources to enable OPEN(G to C) to complete a connection, removing the deadlock condition…

4. PPT timer expires for the OPEN(A to G) connection request in Fanout Expander. The destination port is a wide link with each link allocated to a partial pathway. The Fanout Expander observes that the PBC of OPEN(G to C) > PBC of OPEN(A to G) so the OPEN(G to C) is unaffected. The PBC of OPEN(H to A) = PBC of OPEN(A to G) so the Fanout Expander must compare SAS addresses to determine whether OPEN(A to G) should wait or backoff.

   It has been previously established that SAS address H > SAS address A, therefore OPEN(A to G) must backoff by sending OPEN_REJECT(PATHWAY_BLOCKED). This frees resources in Expander(1) which then enables either OPEN(C to A) or OPEN(H to A) to win the connection (as determined by arbitration fairness rules).