ST notes for WG meeting 1/13/98

Greg Chesson and Jim Pinkerton

1.0 Duplicate Processing and Alias Detection

Duplicated ST operations that could corrupt a data transfer or otherwise lead to a protocol error can arise when:

1. an operation is accepted with valid port and key fields, and
2. a sequence number wraps quickly, or
3. a sequence field does not increment, e.g. STU_num is zero for a sequence of blocks containing a single STU, or
4. a transaction id is reused before all DATA packets are flushed from the interconnect network..

Most of the fields in the protocol that can wrap or are subject to reuse are 32-bit fields: bufx, B_num, D_key. The STU_num field is 16-bits, but is qualified by other context in a message (transaction, B_num). The standard must specify that STU_num not be allowed to wrap. Fields that can be reused quickly, but are needed to qualify the other fields to avoid aliasing are D_id and S_id. These two fields are both 16-bit fields.

D_id and S_id must become 32-bit fields to solve the aliasing problems that arise in high-speed operation or in delayed operation over a WAN. This can be accomplished by reassigning and moving the existing header. fields. The “before” and “after” header diagrams are depicted below. Some

TABLE 1. Packet Field Changes

Before	
Op_Flags	Param
D_Port	S_Port
D_Key	
D_id	S_id
Bufx	
Offset	
Sync	
B_num	
Cksum	Op_len
Offset_2	

After	
Op_Flags	Param
D_Port	S_Port
D_Key	
Cksum	B_id
Bufx	
Offset	
Sync	
B_num	
D_id	
S_id	

consequences of the change are:

1. Op_len is deleted,
2. size of opaque data is reduced,
3. B_id has returned and contains the MX values,
4. D_id and S_id have moved and are both 32-bits.

The assignment of parameters in protocol operations is explained below for each operation. The tables replicate the format used in the protocol standard, with certain fields (Op, Flags, D_Port, S_Port, D_Key, Cksum) eliminated from each table. The suppressed fields function exactly the same as in ST Rev 1.4 (the position of Cksum has been moved).

TABLE 2. Connection Management sequences

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
Request Connection	***I-Slots***	***ethertype***	***I-Bufsize***	***I-Key***	***IMax_stu***	***IMax_blk***	*	*
Connection Answer	***R-Slots***	*	***R-Bufsize***	***R-Key***	***RMax_stu***	***RMax_Blk***	*	*
Request Disconnect	*	*	*	I-Key	*	*	*	*

The Connection Management Sequences differ from Table 3 of Rev 1.4 in that the ethertype parameter moves to B_id and the fields formerly in

Offset_2 are now in Sync. The disconnect operations follow the style of Request_Disconnect and are not shown here.

TABLE 3. Write sequences

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
RTS	***CTS_req***	***Max_blk***	*	*	***T_len***		*****	***I-id***
<RA>	*	*	*	*	*	*	I-id	*
CTS	***Blocksize***	***R-Mx***	***R-Bufx***	***R-Offset***	***F_Offset***	***B_num***	I-id	***R-id***
DATA	***STU_num***	R-Mx	R-Bufx	R-Offset	***Sync***	B_num	R-id	*opaque*
<RSR>	***R_Slots***	*	*	***B_seq***	Sync	B_num	I-id	R-id

The write sequences communicate the same information as described in Rev 1.4. The changes are:

1. R-MX values are communicated through the B_id field,
2. S_id is overloaded differently: it now contains opaque data.

The 64-bit T_len field occupies B_num and Sync.

The Read sequences are derived from the Write sequences, and do not need to be replicated here.

There are no suprises in the persistent memory management sequences. The allocation sequence is shown below. The fields are transcribed from

TABLE 4. Allocate Persistent Memory Region

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
RMR	*	***Max_blk***	*	*	***T_len***		*****	***I-id***
<RA>	*	*	*	*	*	*	I-id	*
MRA	*****	***R-Mx***	***R-Bufx***	***R-Offset***	*	*	I-id	***R-id***

Rev 1.4, the principal difference being that the Mx value is communicated through the B_id field. The Put, Get and FetchOp sequences are depicted below.

TABLE 5. Put Sequence

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
DATA	***STU_num***	R-Mx	***R-Bufx***	***R-Offset***	***Sync***	***B_num***	R-id	***Opaque***
<RSR>	***R-Slots***	*	*	***B_seq***	Sync	B_num	I-id	R_id

The Put sequence messages derive from the Write sequence. The Get Sequence uses the same parameters at Rev 1.4 with the exception that the R-Mx was squeezed out. The Get implementation will need to synthesize the R-Mx value from the saved state generated as a side-effect of the MRA operation.

TABLE 6. Get Sequence

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
GET	***T_len***	***I-Mx***	***R_Bufx***	***R_Offset***	***I-Bufx***	***I_Offset***	R-id	***G-id***
<RA>	*	*	*	*	*	*	G-id	*
DATA	***STU_num***	I-Mx	***I-Bufx***	***I-Offset***	***Sync***	*****	G-id	*****

Changes from the previous GET syntax are:

1. T_len moved from defunct Op_len field to param,
2. I-Mx parameter provided and echoed in B_id field,
3. no room for R-Mx parameter.

TABLE 7. FetchOP sequence

Operation	param	B_id	Bufx	Offset	Sync	B_Num	D_id	S_id
FetchOP	***x'0008'***	***I-Mx***	***R-Bufx***	***R-Offset***	***I-Bufx***	***I_Offset***	R-id	***F-id***
<RA>	*****	*****	*****	*****	*****	*****	F-id	*****
DATA	x'0000'	I-Mx	I-Bufx	I-Offset	***Sync***	*****	F-id	*****
FetchOP Complete	*	*	*	*	Sync	*****	R-id	F_id

FetchOP parameters are the same as for GET with the restriction on T_len of 8 byte transfers. The STU_num in DATA is fixed at zero.