



Serial Attached SCSI Link layer – part 1



by Rob Elliott

HP Industry Standard Servers

Server Storage Advanced Technology

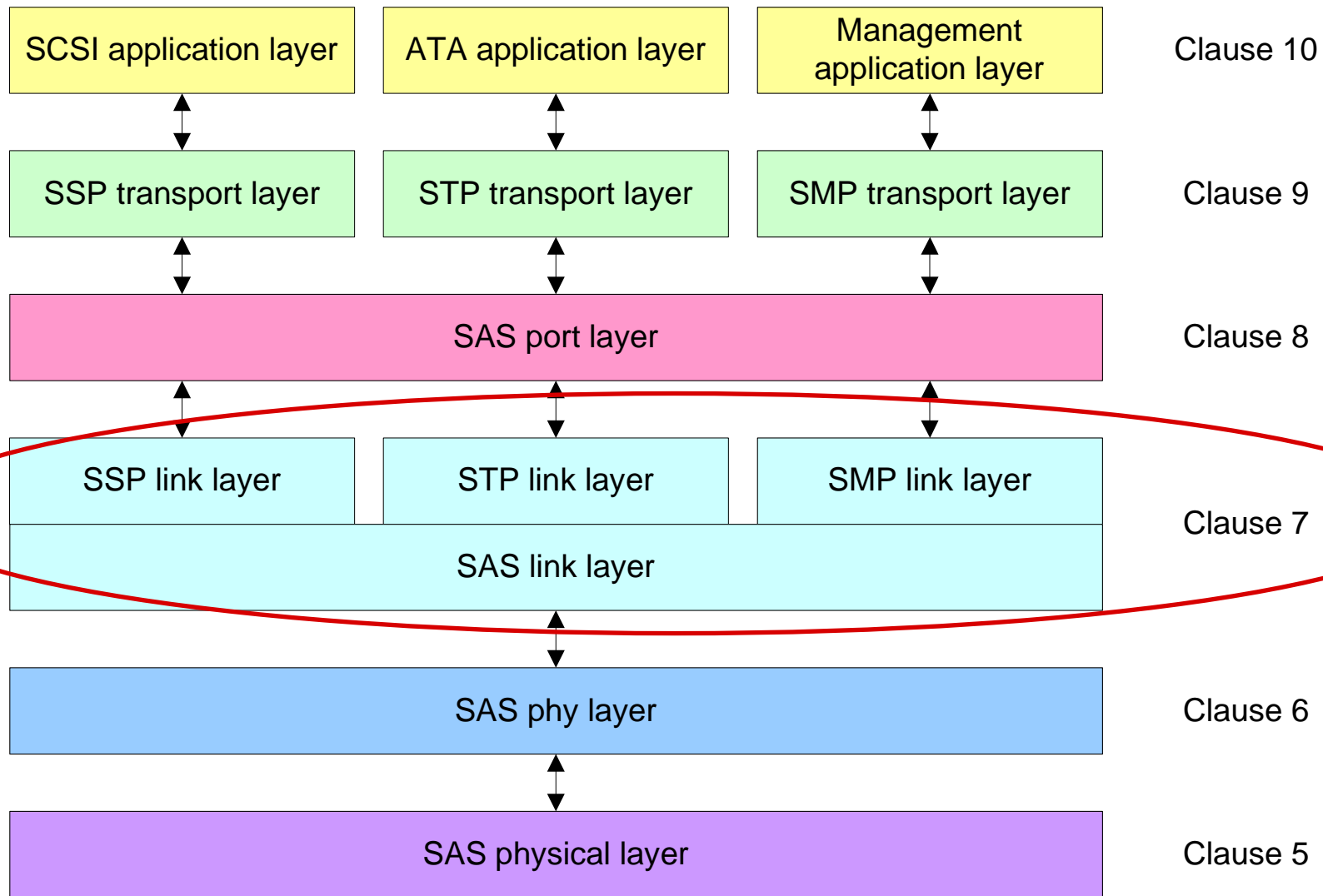
elliott@hp.com <http://www.hp.com>

30 September 2003

- These slides are freely distributed by HP through the SCSI Trade Association (<http://www.scsita.org>)
- STA members are welcome to borrow any number of the slides (in whole or in part) for other presentations, provided credit is given to the SCSI Trade Association and HP
- This compilation is © 2003 Hewlett-Packard Corporation



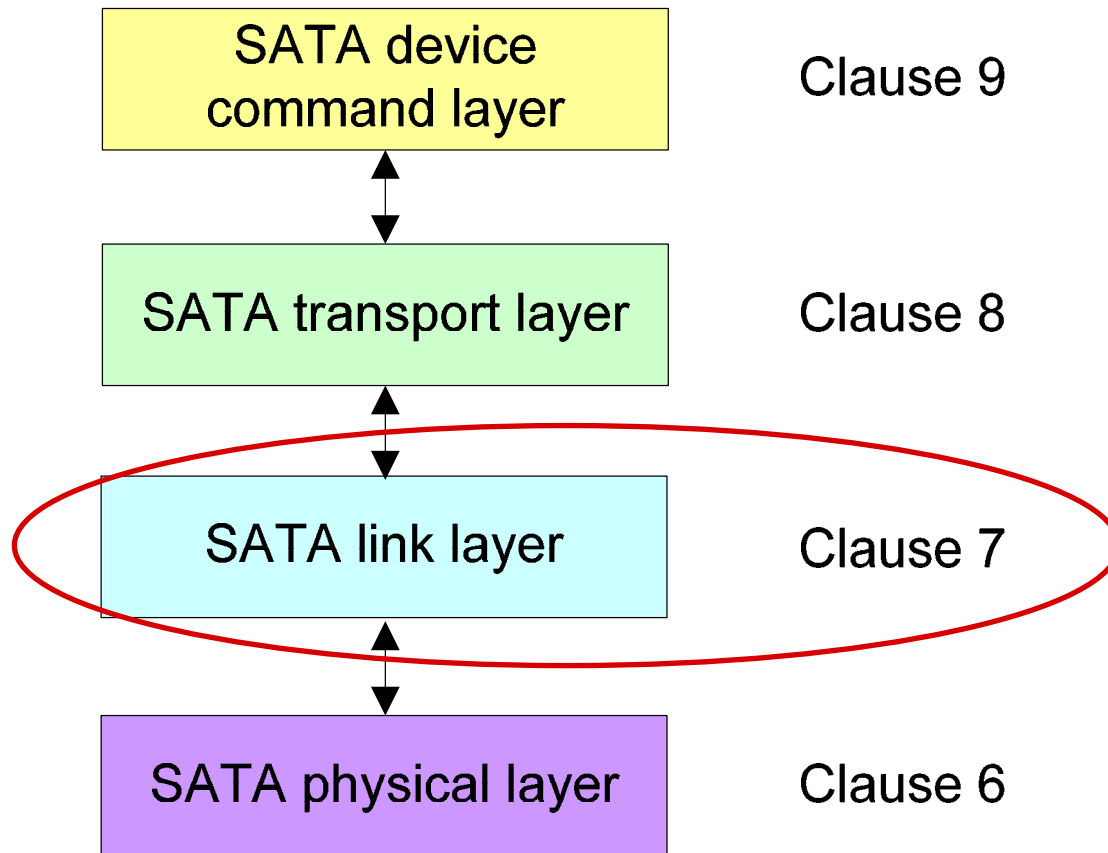
SAS standard layering



SATA 1.0a standard layering



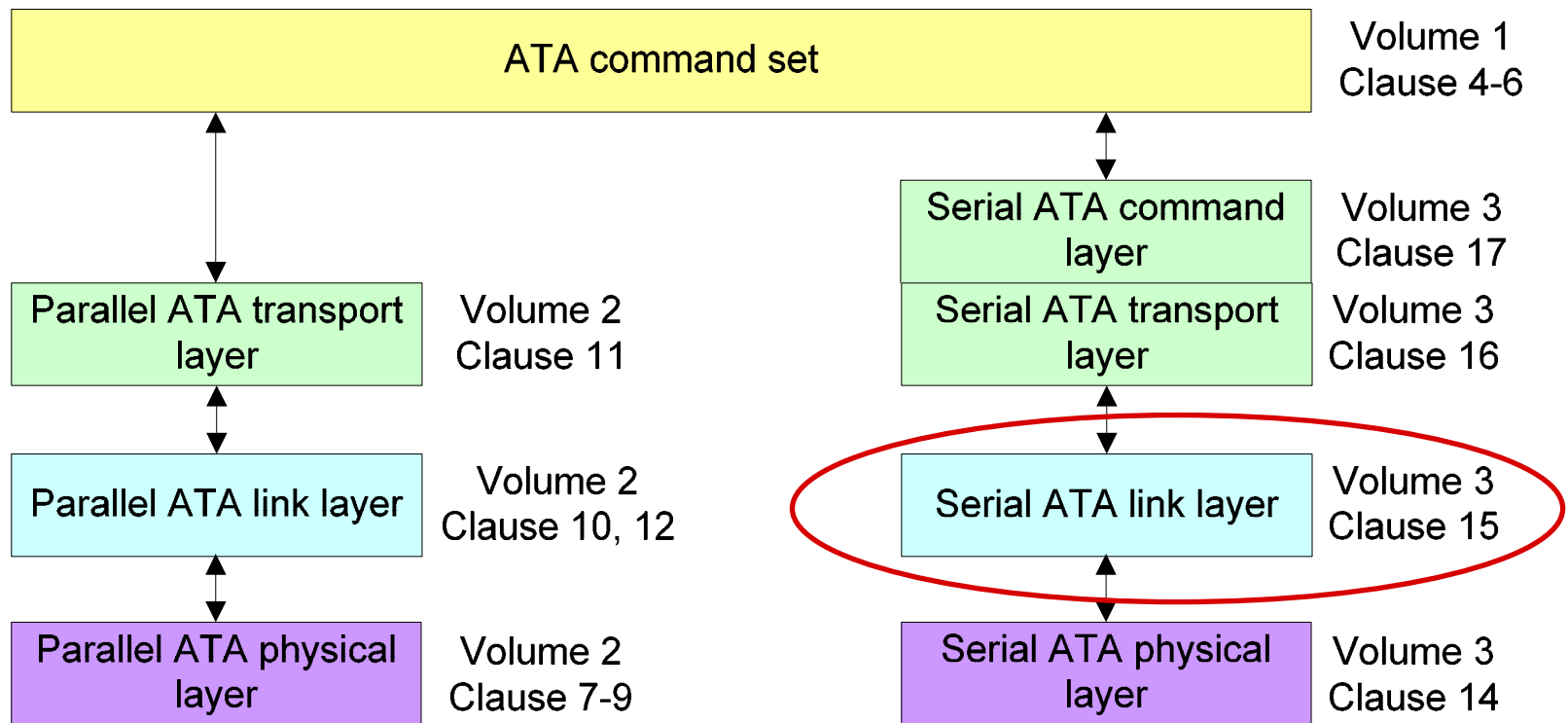
- For SATA 1.0a from the private Serial ATA working group



ATA/ATAPI-7 standard layering



- For the public standard ATA/ATAPI-7
- Subject to change by T13 standards committee



SAS clause 6 – Link layer (part 1)

- Primitives
- Clock skew management
- CRC
- Scrambling
- Bit order
- Address frames
- Identification and hard reset sequences
- Connections
- Connection management state machines
- Wrap up

Link layer – Primitives

Primitives (from phy layer)



- 1 control character and 3 data characters
 - First character is K28.5 (for SAS primitives), K28.3 (for SATA primitives), or K28.6 (special SATA error primitive)
 - K28.6 primitive serves as an invalid dword for SATA
 - Last three characters are data characters
 - Endianness does not matter
 - both SAS and SATA primitives always have the control character first on the wire

first	second	third	fourth
K28.5	Dxx.y	Dxx.y	Dxx.y
K28.3	Dxx.y	Dxx.y	Dxx.y
K28.6	Dxx.y	Dxx.y	Dxx.y

Primitive encodings

- SATA primitive table uses little-endian notation

Primitive name	Byte 3	Byte 2	Byte 1	Byte 0
ALIGN	D27.3	D10.2	D10.2	K28.5
CONT	D25.4	D25.4	D10.5	K28.3
...	K28.3

- SAS primitive tables use first byte/last byte notation

Primitive	1st	2nd	3rd	4 th (last)
ALIGN (0)	K28.5	D10.2	D10.2	D27.3
SATA_CONT	K28.3	D10.5	D25.4	D25.4
...	Kxx.y

- Functional categories
 - Primitives not tied to specific types of connections
 - Some are outside connections only, others inside or outside connections
 - Primitives used inside SSP and SMP connections
 - Primitives used inside STP connections and on SATA
 - Encodings and functionality defined by SATA
- There are multiple versions of many primitives
 - Communicate functional differences (e.g. different reasons for an OPEN_REJECT)
 - EMI (electromagnetic interference) reduction (e.g. rotate through 4 different ALIGNs)

Primitives not tied to specific types of connections



- Connection management
 - **AIP** – arbitration in progress
 - **BREAK** – break a connection (emergency close)
 - **CLOSE** – close a connection
 - **OPEN_ACCEPT** – accept a connection request
 - **OPEN_REJECT** – reject a connection request
 - **SOAF, EOAF** – start/end of address frame
 - Used for connection requests and identification sequence
- General
 - **ALIGN** – clock skew management, rate matching
 - **NOTIFY** – version of ALIGN that communicates information
 - **BROADCAST** – expander broadcasts to all initiators
 - **ERROR** – for forwarding invalid dwords
 - **HARD_RESET** – force a low level reset

Primitives for SSP and SMP connections



- Only used inside SSP and SMP connections
 - **RRDY** – extend permission to send a frame
 - **CREDIT_BLOCKED** – warning that an RRDY is not coming soon
 - **SOF, EOF** – start and end of frame
 - **ACK, NAK** – positive and negative frame acknowledgement
 - **DONE** – prepare to close connection
- SMP only uses **SOF** and **EOF**

Primitives for STP connections and SATA



- SAS uses SATA_ prefix when referring to all SATA primitives except ALIGN
 - SATA_X_RDY – request to send frame
 - SATA_R_RDY – grant permission to send frame
 - SATA_SOF, SATA_EOF – start and end of frame
 - SATA_HOLD, SATA_HOLDA – temporarily pause frame transmission
 - SATA_WTRM – frame transmission complete
 - SATA_R_IP, SATA_R_OK, SATA_R_ERR – frame receipt in progress, received OK (ACK), received with error (NAK)
 - SATA_PMREQ_P, SATA_PMREQ_S, SATA_PMACK, SATA_PMNAK – power management requests and acknowledgements
 - SATA_ERROR – used to forward invalid dword to SATA
 - SATA_SYNC – idle
 - SATA_DMAT – terminate a DMA operation in progress
 - SATA_CONT – enter idle scrambled data mode

Primitive sequences



- To reduce problems from errors, some primitives must be sent multiple times
 - Missing an SOF just messes up that frame and that connection
 - Missing a CLOSE or BREAK confuses lots of devices

Type	Transmit	Receive
Single	1	1
Repeated	2	1
Triple	3	3
Redundant	6	3

Single primitive sequence

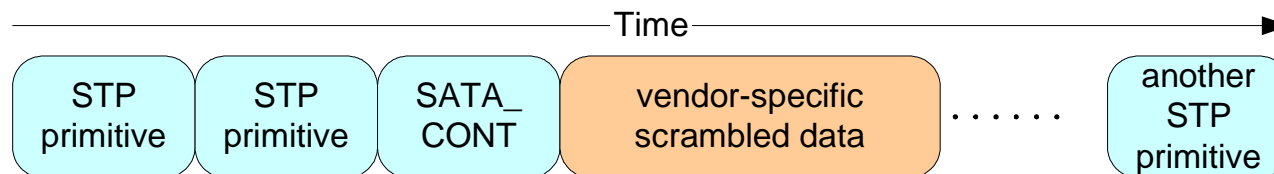


- Just one instance of the primitive is transmitted
- Subject to single-bit errors
- Most primitives independent of connections are single primitive sequences
 - AIP, ALIGN, NOTIFY, OPEN_ACCEPT, OPEN_REJECT, SOAF, EOAF, ERROR
- All primitives used only inside SSP and SMP connections are single primitive sequences
- SATA_SOF, SATA_EOF, and SATA_ERROR inside STP connections are single primitive sequences

Repeated primitive sequence



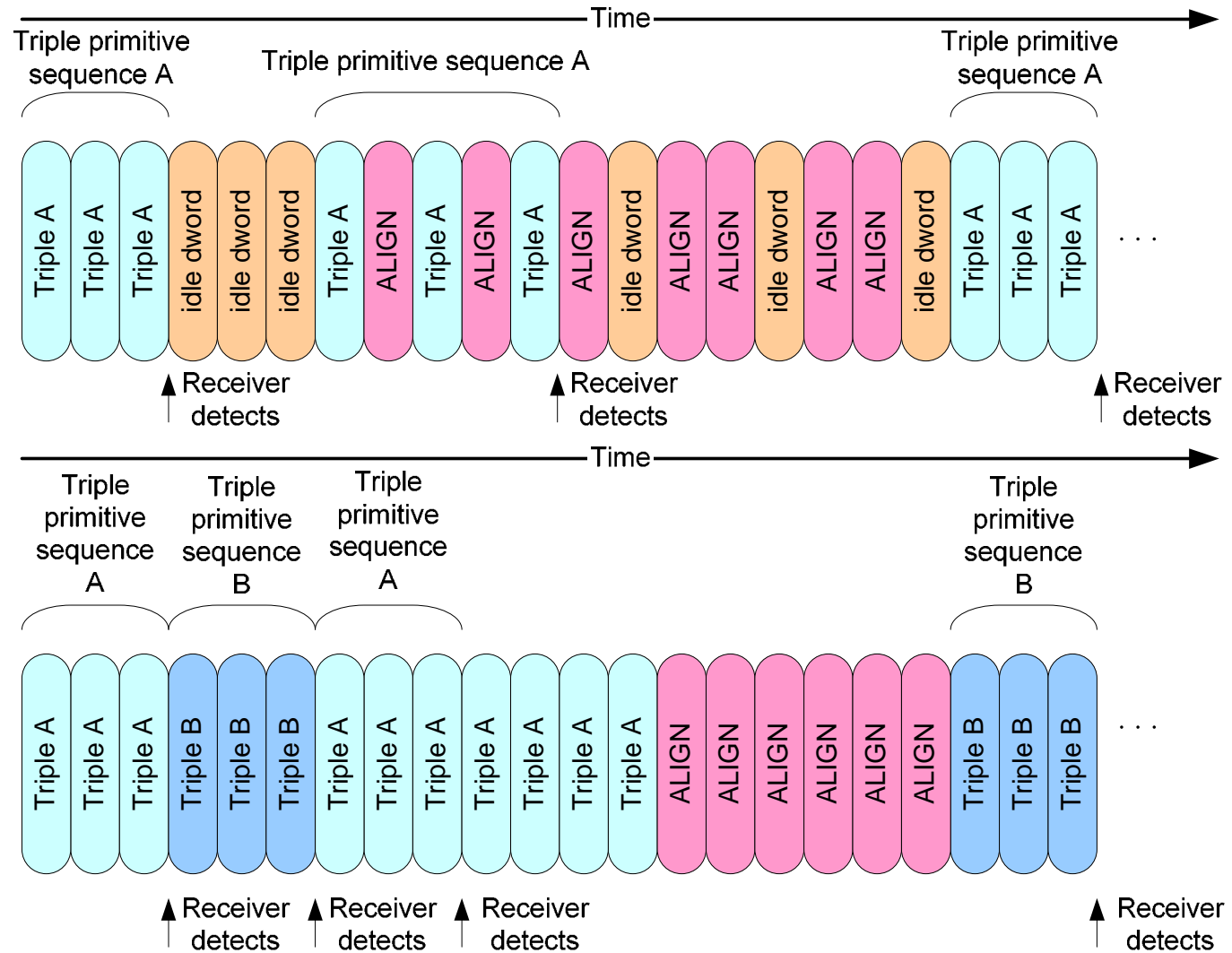
- For STP connections and SATA
 - Send the primitive at least twice
 - Send SATA_CONT once
 - Send vendor-specific scrambled data dwords
 - Ignored by receiver
 - Send a different primitive to exit this mode
- Avoids EMI problems from sending the same primitive over and over
- ALIGNs still appear inside the scrambled data
- Most SATA primitives can be repeated



Triple primitive sequence



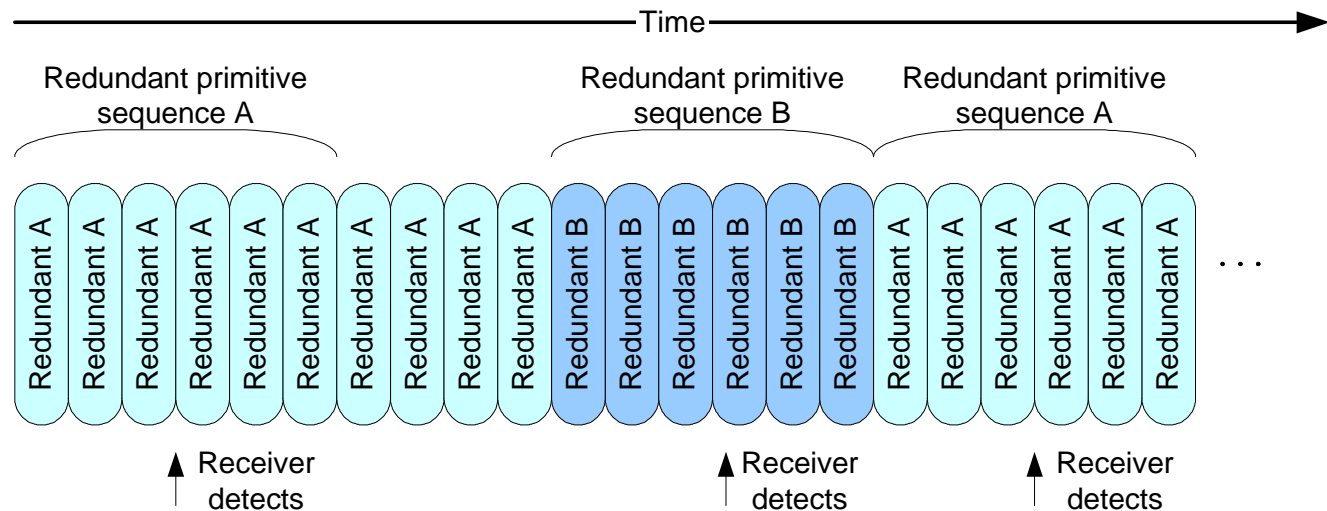
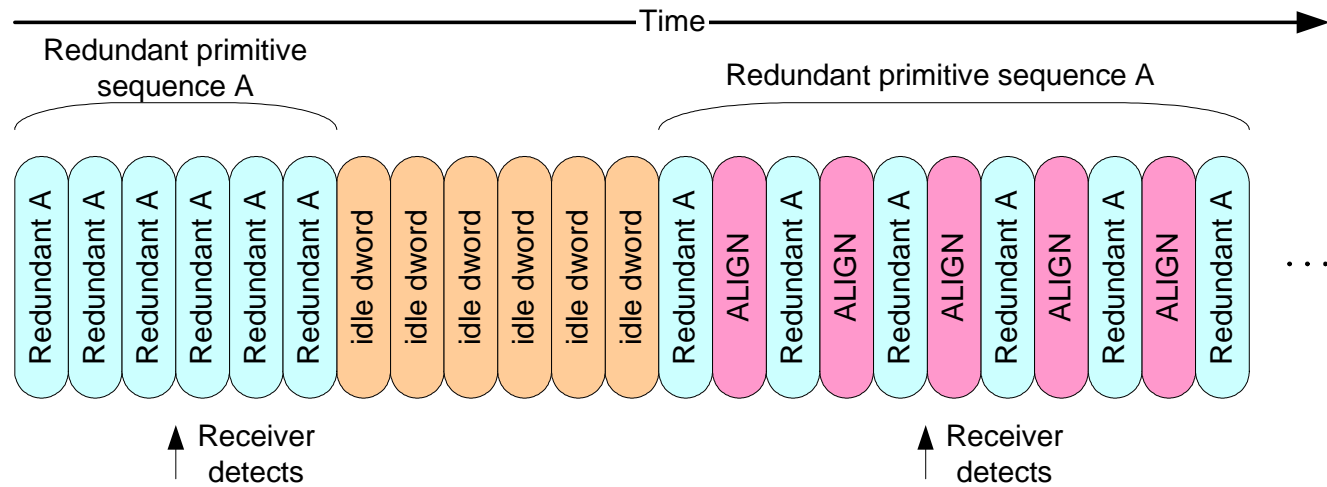
- Send-3, receive-3
- Error causes primitive to be missed
- Don't detect again until 3 other dwords arrive
- CLOSE is the only triple primitive sequence



Redundant primitive sequence



- Send-6, receive-3
- Tolerates some errors
- Don't detect again until 6 other dwords arrive
- BREAK, HARD_RESET, and BROADCAST are the only redundant primitive sequences



Link layer – Clock skew management

Clock frequency tolerances

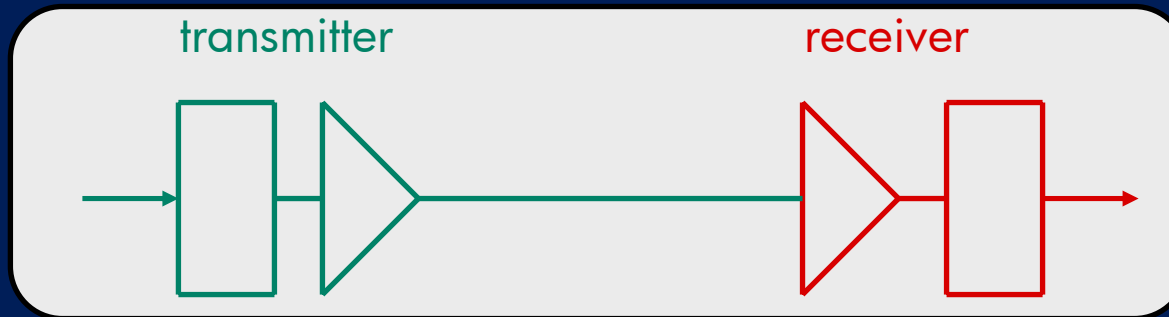


- Clock frequencies are imperfect
- PPM = parts per million, shorthand for a percentage
- $100 \text{ ppm} = 100/1,000,000 = 1/10,000 = 0.0001 = 0.01\%$
- Unit interval = the time to transmit one bit

Type	Parts per million	Unit interval (nominal)	Unit interval (minimum to maximum)
SAS 3 Gbps	100 ppm	$333.333 \pm 0.01\%$	333.300 to 333.367 ps
SAS 1.5 Gbps	100 ppm	$666.666 \pm 0.01\%$	666.600 to 666.733 ps
SATA 1.5 Gbps	+350/ -5,350 ppm	666.666 +.035%/-0.535%	666.43 to 670.23 ps (spec says 670.12)

Clock skew management

- Clocks in the transmitter and receiver are not exactly the same
- A slow transmitter will eventually underrun a fast receiver
 - Not a problem (assuming receiver designers realized the issue)
- A fast transmitter will eventually overrun a slow receiver
 - No low-level solution possible in the receiver alone



Type	Fastest	Slowest	Difference
SAS 3 Gbps	300.03 MBps	299.97 MBps	60 KBps
SAS 1.5 Gbps	150.015 MBps	149.99 MBps	25 KBps
SATA 1.5 Gbps	150.053 MBps	149.20 MBps	853 KBps

Clock skew management solution



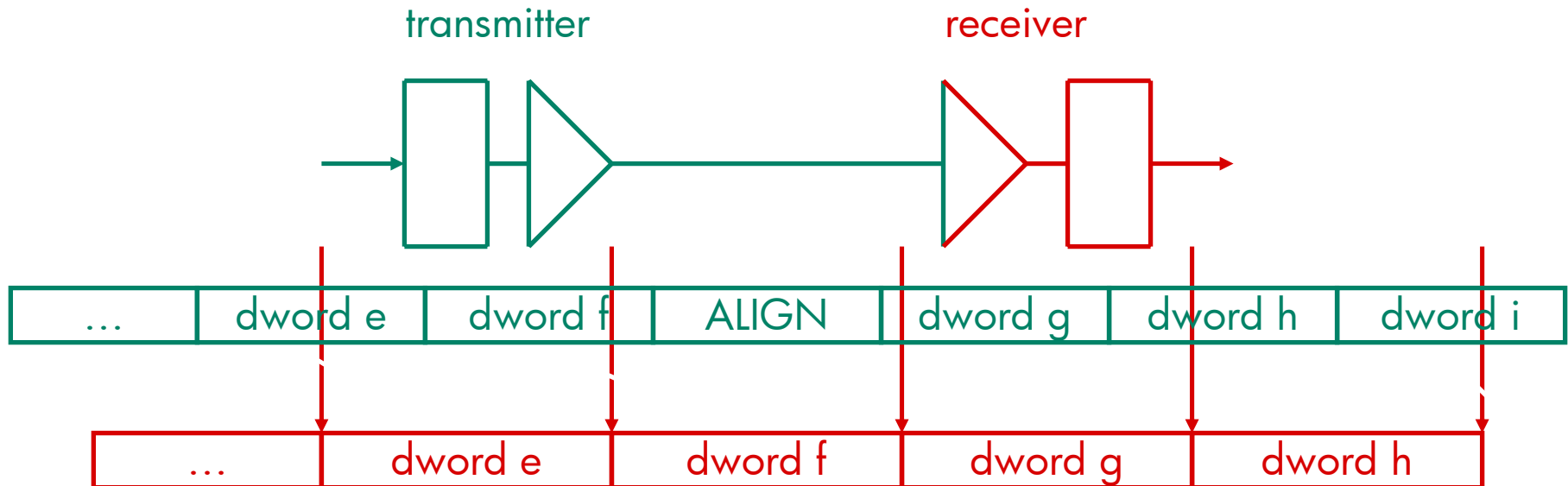
- Slow down the transmitter by including dwords that can be deleted
 - ALIGN (and NOTIFY for SAS) primitives inserted in the data stream
 - Throttles the transmitter to be slower than the slowest receiver
 - SATA (and SAS for STP initiators) requires 2 per 256 dwords (0.7812%)
 - Must always appear in pairs
 - SAS requires 1 per 2048 dwords (0.0488%)
 - Data stream never faster than the slowest receiver can receive
 - Receiver throws away incoming ALIGNs (and NOTIFYs for SAS) as needed
 - Receiver ignore ALIGNs that are not deleted
 - Receiver may even insert ALIGNs internally as a placeholder when it underruns

Type	Fastest transmitter	Fastest with required ALIGNs	Slowest receiver
SAS 3 Gbps	300.03 MBps	299.88 MBps	299.97 MBps
SAS 1.5 Gbps	150.015 MBps	149.94 MBps	149.99 MBps
SATA 1.5 Gbps	150.053 MBps	148.88 MBps	149.20 MBps

Clock skew management example



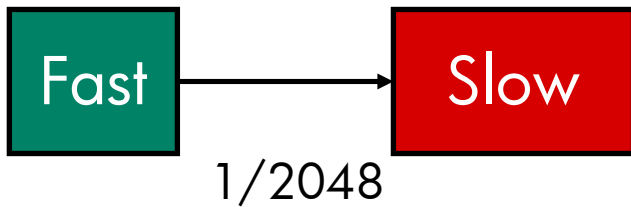
- Assume the receiver is latching dwords successfully but drifting behind the transmitter
- Dword f is just barely latched OK
- If dword g followed, it would be lost (overrun)
- Instead, an ALIGN shows up, thrown away by the receiver
- Dword g can then be latched OK



Clock skew source rate is fixed

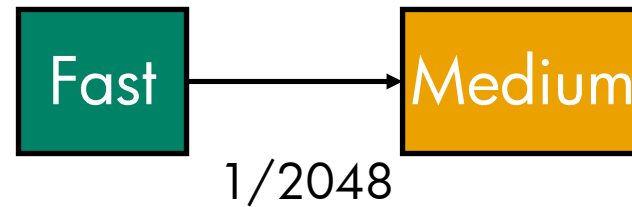


- Source phy always inserts the same number of ALIGNs
- Phys don't know if they are running slower or faster than the other
 - it doesn't matter
- Underruns are okay



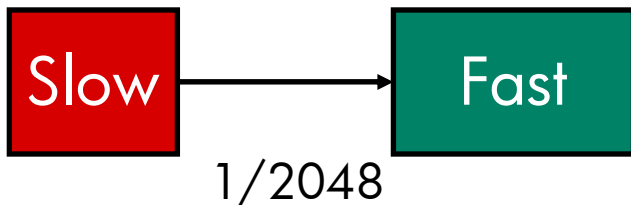
Source always inserts 1/2048

Strips off most ALIGNs



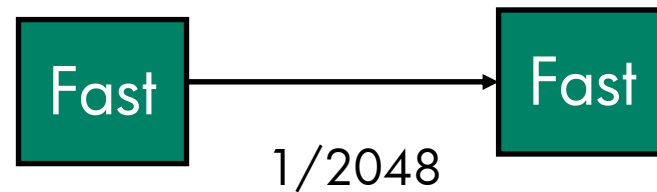
Source always inserts 1/2048

Strips off some ALIGNs



Source always inserts 1/2048

Underruns on each ALIGN and other times



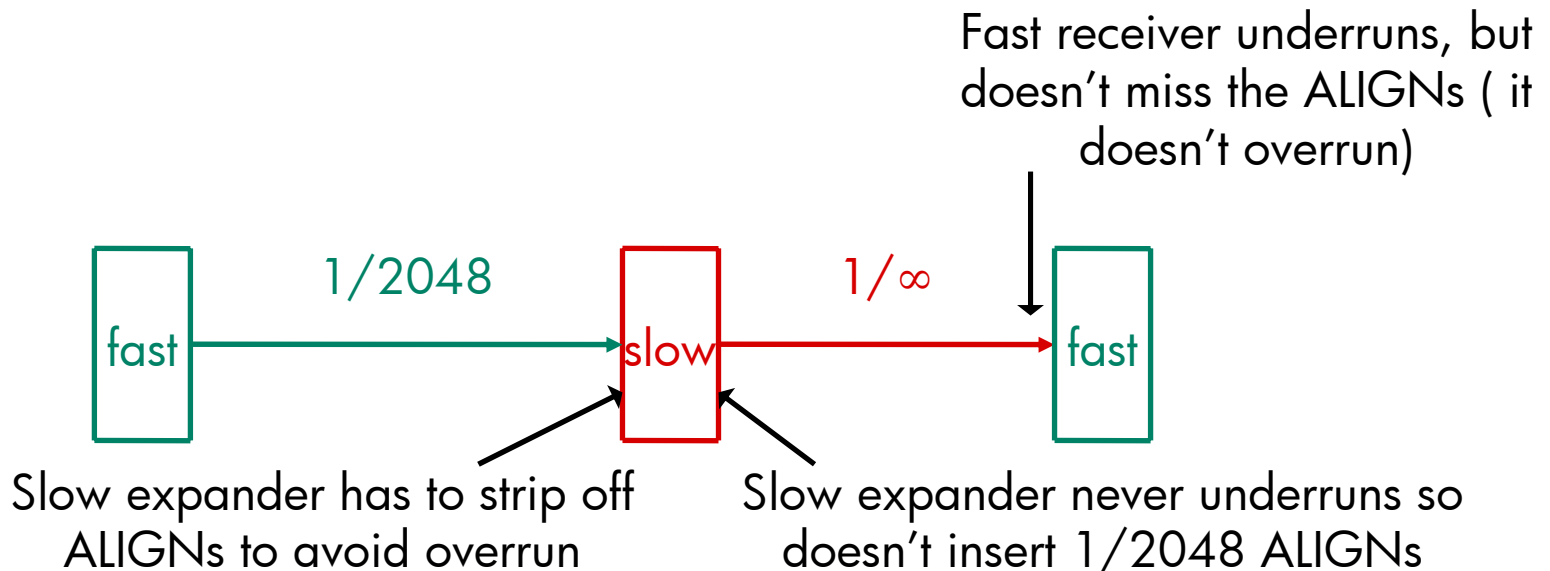
Source always inserts 1/2048

Underruns on each ALIGN

Clock skew management implementations



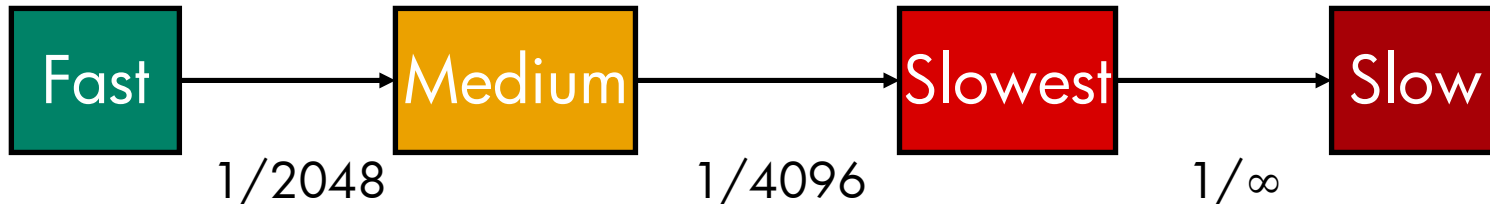
- Any mix of fast, medium, and slow expanders and end devices possible
- Expanders strip off ALL incoming ALIGNs and NOTIFYs and generate them only when the output phy underruns
 - Not required to guarantee 1/2048 on **every** physical link - just the source physical link
 - A slow expander transmitting to a fast expander cannot cause an overrun, so a lack of ALIGNs doesn't hurt



Clock skew and multiple expanders



- This works with any combination of different frequencies

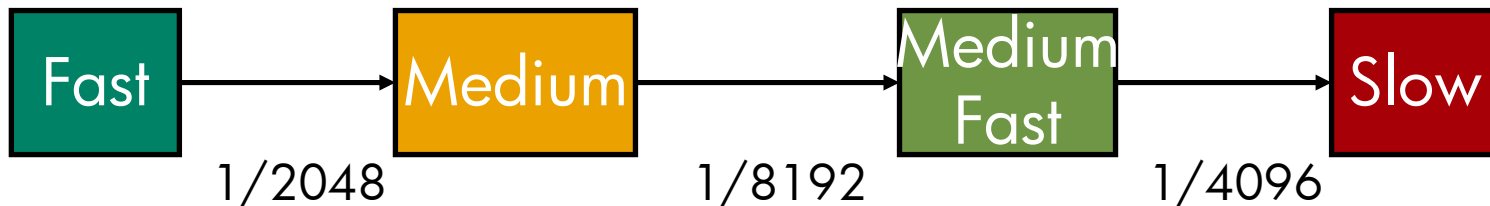


Source always starts with 1/2048

Assume Medium strips off half the ALIGNs

Slowest strips off all the ALIGNs

Slow will underflow occasionally



Source always starts with 1/2048

Assume Medium strips off 1/4 the ALIGNs

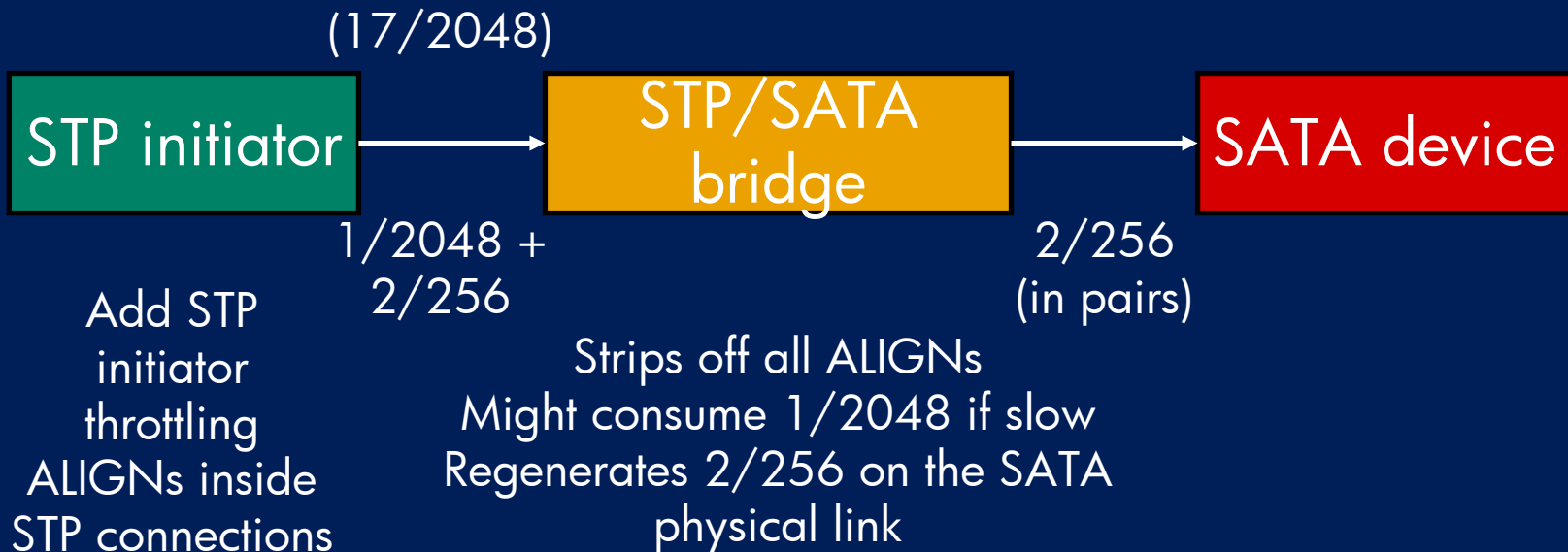
Medium Fast forwards ALIGNs and adds some of its own

Slow will underflow occasionally

STP initiator throttling



- SATA 2/256 rule must be met on SATA physical links
 - SAS STP initiators must insert 2/256 plus 1/2048 during STP connections
 - Expander with STP/SATA bridge must honor the pair rule when inserting
 - STP target ports need not insert 2/256 extra, because they don't talk to a SATA host on a SATA physical link where 2/256 is expected

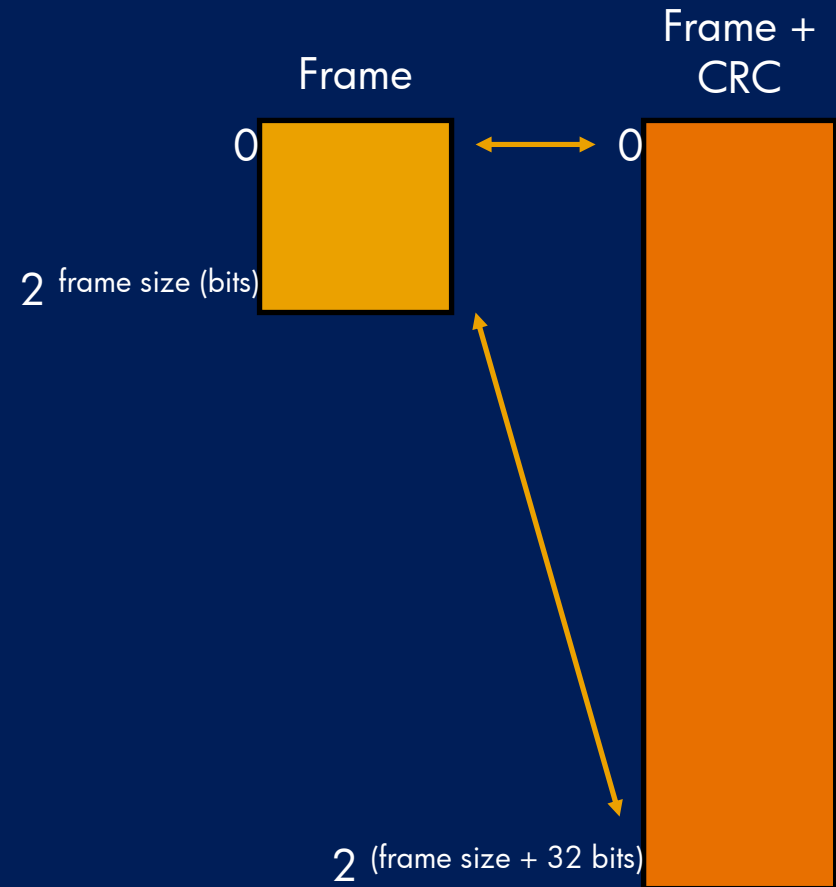


Link layer – CRC

CRC



- Each frame of data is protected by a cyclic redundancy check (CRC)
 - Address frames
 - SSP frames
 - SMP frames
 - STP/SATA frames
- CRC-32 generator polynomial
- In the wider space, many patterns are invalid
- CRC background
 - RFC 3385 – iSCSI CRC considerations
 - Discusses coverage and implementations
 - <http://www.ietf.org>



Modulo-2 arithmetic



- Modulo-2 arithmetic
 - Modulo arithmetic introduced by Carl Friedrich Gauss in 1801
 - Addition = subtraction = XOR
 - $0 + 0 = 0 = 0 - 0$, $0 + 1 = 1 + 0 = 1 = 0 - 1 = 1 - 0$, $1 + 1 = 0 = 1 - 1$
 - No carries for addition, no borrows for subtraction
 - Multiplication
 - $0 * 0 = 0$, $0 * 1 = 0 = 1 * 0$, $1 * 1 = 1$
 - by power of 2, shift bits left
 - Division
 - in long division, a divisor “goes into” a dividend if the dividend has the same number of bits as the divisor
- Polynomials
 - Treat bit string as a polynomial with coefficients of 0s and 1s
 - 6 bits 110011b (33h) represented as $A(x) = x^5 + x^4 + x + 1$
 - ($0 * x^3 + 0 * x^2$ not shown since their coefficients are 0)

CRC generation frame and generator



- $F(x)$ = the frame (expressed as a polynomial)
 - SAS SSP frames are up to 1048 bytes (8384 bits) long
 - $x^{32} * F(x)$ is the frame with 32 trailing 0s
 - Polynomial of order 8416

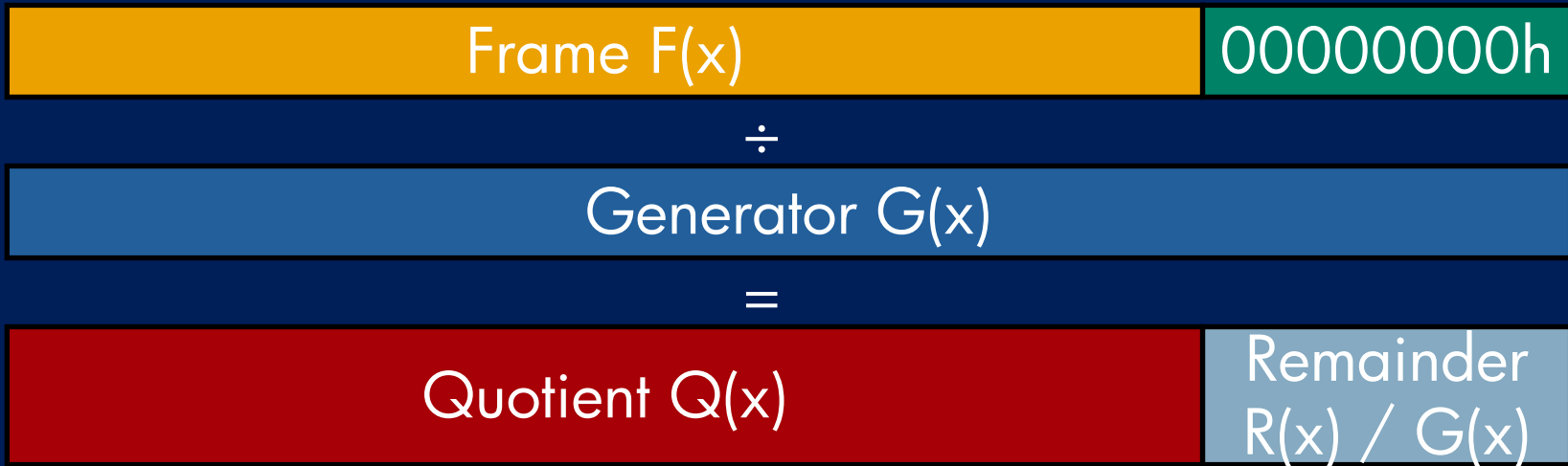


- $G(x)$ = generator polynomial
 - $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
 - $G(x) = 1_04C11DB7h$ (bits 32:1 contain 82608EDBh)
 - Called CRC-32
 - Same as that used by Serial ATA, Parallel SCSI, Fibre Channel, and Ethernet
 - Different than iSCSI, which found a better polynomial

CRC core equation



- CRC core equation
 - Divide the frame polynomial by the generator polynomial
 - $x^{32} * F(x)/G(x) = Q(x) + R(x)/G(x)$



- Result is an integer and a fraction
 - $Q(x)$ = integer = the greatest multiple of $G(x)$ in $(x^{32} * F(x))$
 - $R(x)/G(x)$ = fraction
 - $R(x)$ = the remainder (32 bits)

CRC remainder

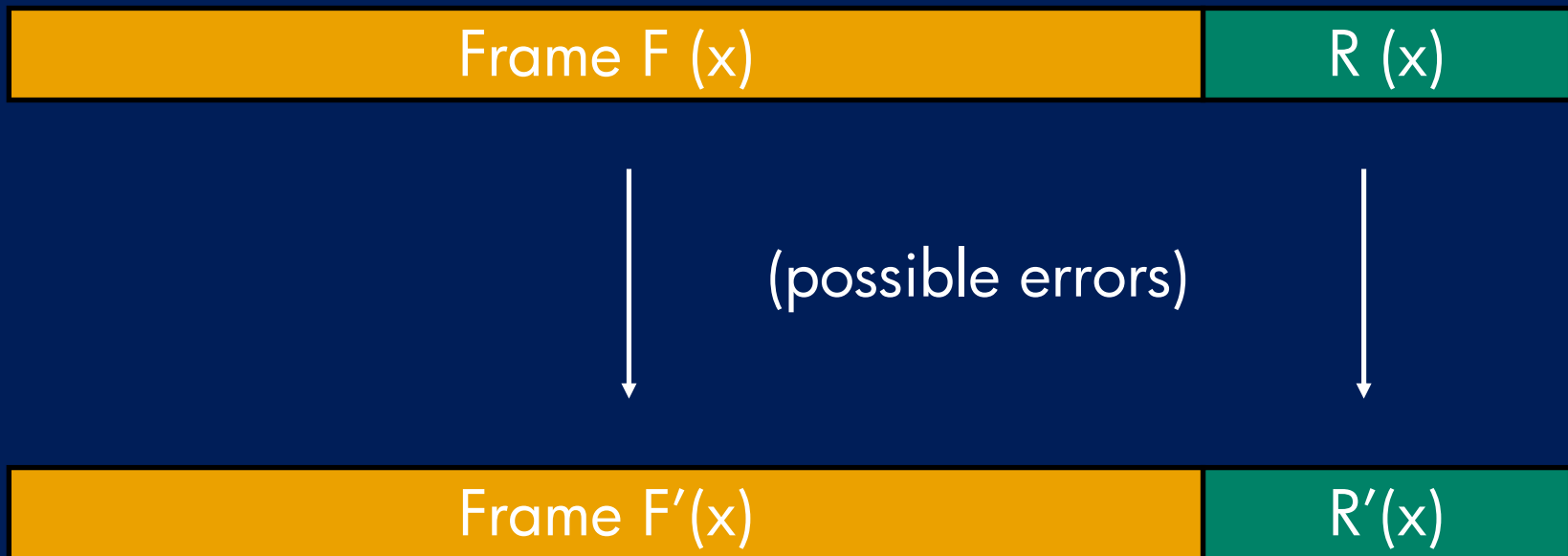


- After calculating the quotient and remainder, instead of transmitting $x^{32} * F(x)$, replace the trailing 0s with the remainder $R(x)$
 - $M(x) = x^{32} * F(x) + R(x)$
- Since $+$ is same as $-$, the core CRC equation can be viewed as
 - $Q(x) = x^{32} * F(x)/G(x) + R(x)/G(x)$
- Receiver divides $M(x)$ by the same generator polynomial
 - $(x^{32} * F(x) + R(x))/G(x) = Q(x) + R(x)/G(x)$
 - $x^{32} * F(x) + R(x) = Q(x) * G(x) + R(x)$
 - $x^{32} * F(x) = Q(x) * G(x)$
 - Notice no remainder remains

CRC checking



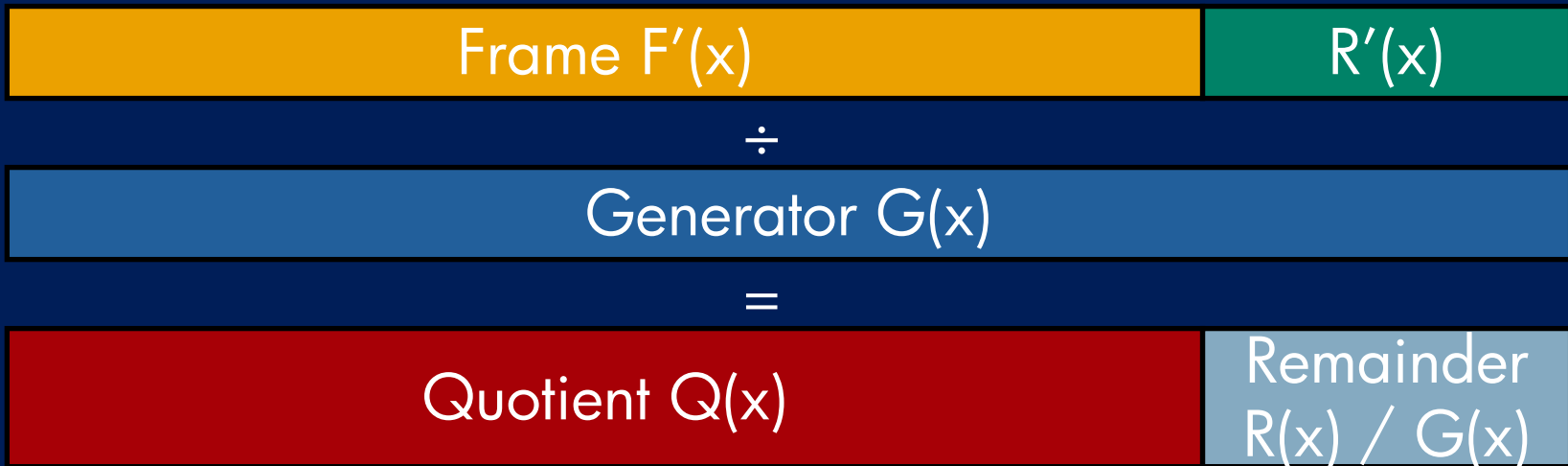
- Transmit $M(x) = F(x) + R(x)$
- Receive $M'(x) = F'(x) + R'(x)$
 - With no errors, $M(x) = M'(x)$
 - With errors, they are different



CRC checking



- Receiver divides $M'(x)$ by $G(x)$
 - if the remainder is zero, $M'(x)$ is probably correct
 - If the remainder is not zero, must be an error
 - $M(x)$ is directly divisible (no remainder) by $G(x)$ since it has $R(x)$ in it
 - $M(x) / G(x) =$
 - $(x^{32} * F(x) + R(x)) / G(x) =$
 - $x^{32} * F(x)/G(x) + R(x)/G(x) = Q(x)$



CRC checking enhancements

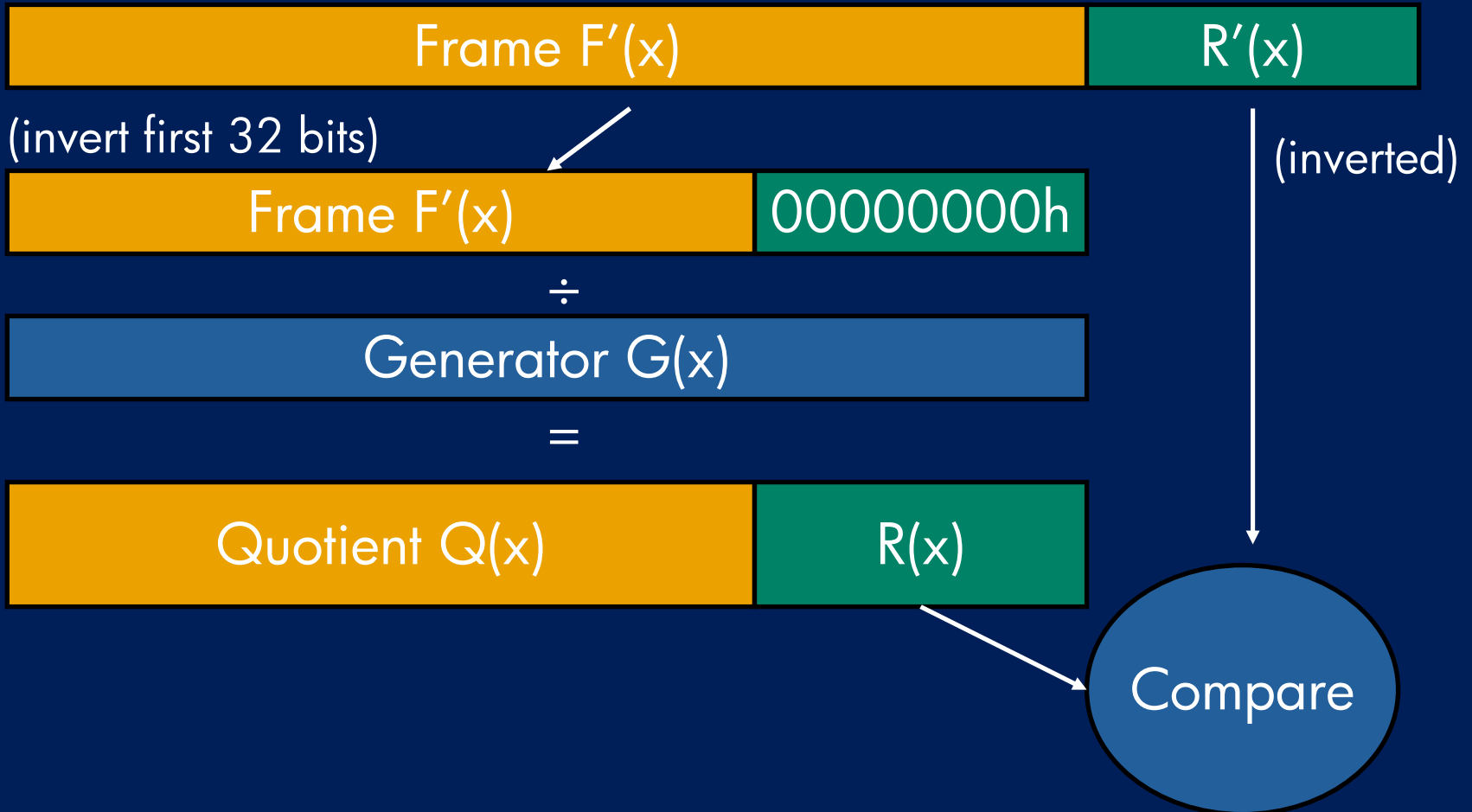


- SAS CRC is slightly modified from the ideal mathematics
 - Bits within each byte of $F(x)$ are transposed to counteract the 8b10b mapping problem (A to h, ... H to a)
 - First 32 bits in $F(x)$ are inverted when calculating $R(x)$
 - $L(x) = x^{32} + x^{31} + \dots + x + 1$ [all ones]
 - $(x^{32} * F(x) + x^K * L(x)) / G(x) = Q(x) + R(x) / G(x)$
 - Detects leading 0s in $F(x)$
 - $R(x)$ is transmitted as inverted
 - $M(x) = x^{32} * F(x) + L(x) / R(x)$
 - Expected remainder is not zero due to the inversions
 - $C(x) = x^{32} * L(x) / G(x)$
 - $C(x) = x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$
 - $C(x) = C703DD7Bh$

CRC checking – compare approach



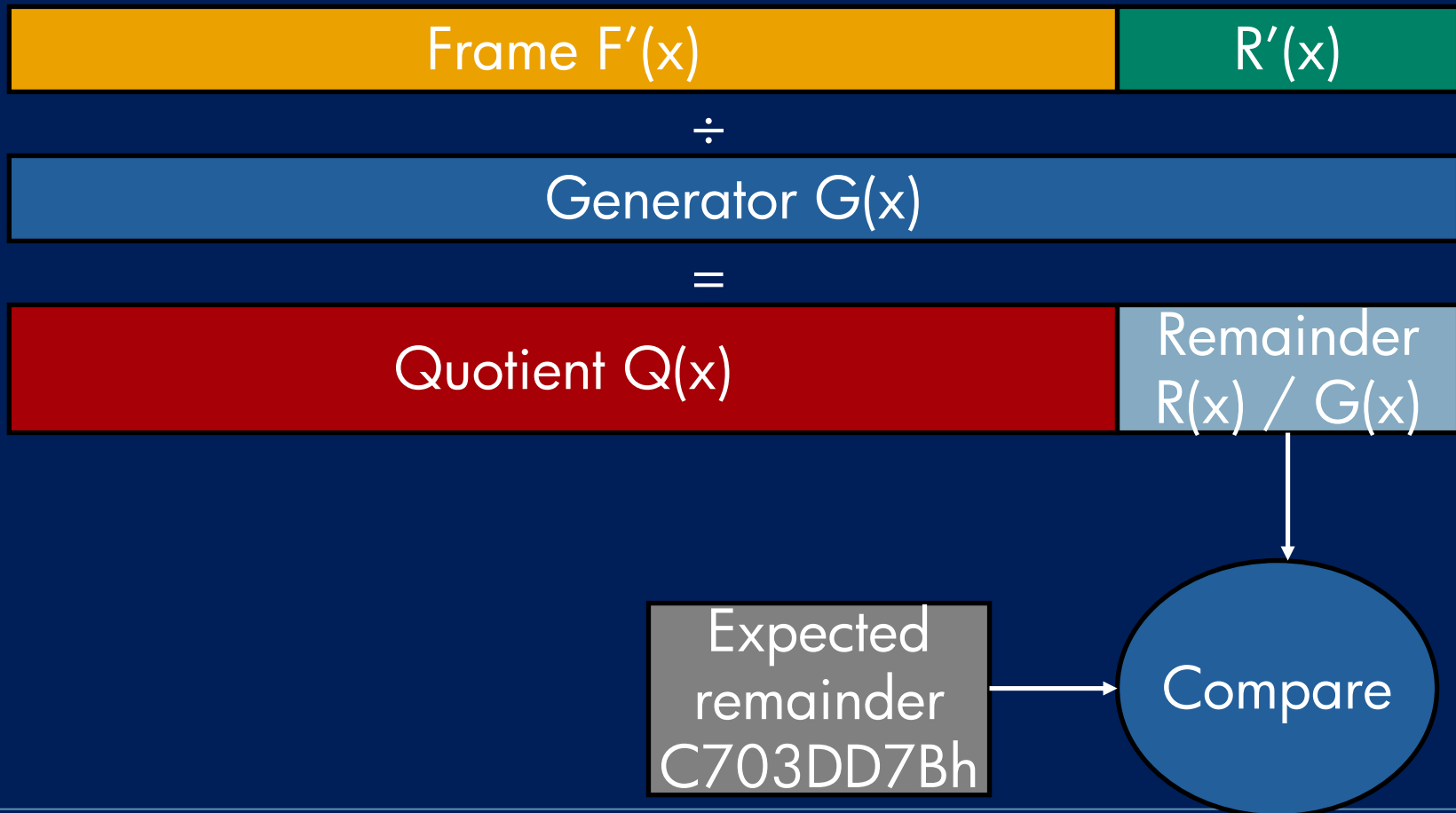
- Calculate the CRC over the frame only
- Compare the calculated remainder to the remainder that was received



CRC checking – constant approach



- Calculate the CRC over the frame plus the remainder
- Compare with the expected (constant) remainder



CRC-32 coverage



- All bursts of up to 32 bit inversions are detected
- All odd numbers of bit inversions detected
 - Some even numbers of bit inversions go undetected
- Hamming distance = # bits that have to change to convert one valid value into another valid value
 - Hamming distance of 4 for up to 91,607 bits (11,450 bytes)
 - Detects all 2-bit and 3-bit errors
 - 4 bit errors can convert one valid frame into another
 - Slightly more than 1 out of 2^{32} possible 4-bit errors are undetectable

CRC-32 coverage



- According to an iSCSI-related paper, this CRC has...

Hamming distance	Number of bits in the frame
15	8 to 10
14	-
13	-
12	11 to 12
11	13 to 21
10	22 to 34
9	35 to 57
8	58 to 91
7	92 to 171
6	172 to 268
5	269 to 2974
4	2975 to 91,607
3	91,608 to more than 131,072

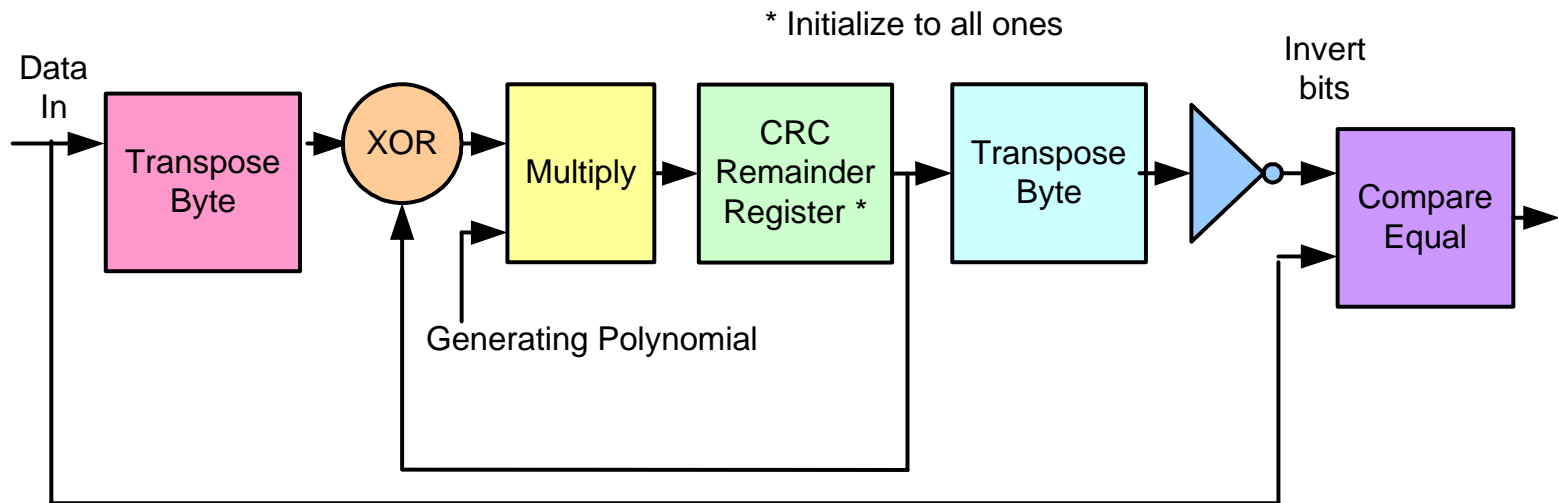
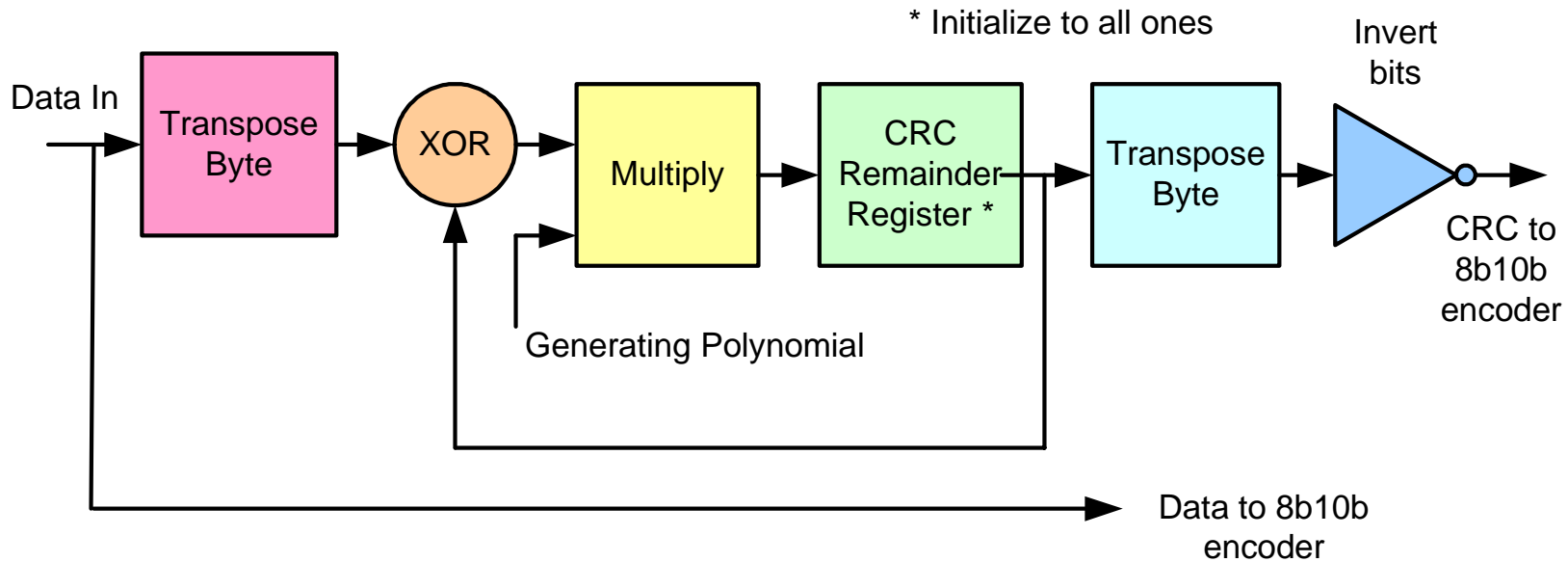
- "32-Bit Cyclic Redundancy Codes for Internet Applications" by Philip Koopman
- http://www.ece.cmu.edu/~koopman/networks/dsn02/dsn02_koopman.pdf
- SAS SSP frames are up to 8384 bits long

CRC implementations



- Serial XORs
 - Shift register with XOR feedback
 - One bit at a time
 - Mainly a conceptual reference
 - SAS Annex C contains C code
- Parallel XORs
 - Best for hardware implementations
 - One dword (32 bits) at a time
 - Sample implementations
 - SATA Annex A contains C code
 - SAS Annex C contains Verilog code
 - Free CRC32 synthesizable code generator
 - <http://www.easics.com>
 - Enter polynomial coefficients
 - Generate Verilog or VHDL
- Table lookup
 - Used for software implementations
 - One byte, word, or dword at a time

CRC serial XOR implementations



Link layer – Scrambling

Scrambling



- All data dwords in SAS and SATA are transmitted after being XORed with a pattern
- Reduces EMI problems from repeated data patterns
 - With 8b10b coding, long strings of 0s and 1s are already eluded
 - There is a worst-case pattern that undoes scrambling
- Receiver must unscramble with same pattern before using the data
- Primitives are not scrambled
 - SATA uses SATA_CONT primitive to handle repeated primitives
 - SAS avoids defining primitives that repeat
 - In cases where problems could occur, like ALIGNs, multiple primitives are defined (transmitter rotates through ALIGN(0), ALIGN(1), ALIGN(2), ALIGN(3))
- Scrambling is mandatory and always used in production
 - For debug, most devices will be able to turn it off in the lab

Scrambling data dword types



- SAS SSP works with just one scrambling value
- SATA (and SAS STP) require two scrambling values
 - one for the STP frame
 - another for repeated SATA primitives (which can be sent inside an STP frame)

Connection state	Data dword type	Description
Outside connections	SAS idle dword	Physical link is idle
	Address frame	After SOAF until EOAF
Inside SSP connection	SSP frame	After SOF until EOF
	SSP idle dword	Physical link is idle
Inside SMP connection	SMP frame	After SOF until EOF
	SMP idle dword	Physical link is idle
Inside STP connection, and on SATA physical links	STP frame	After SATA_SOF until SATA_EOF
	Repeated SATA primitive	After SATA_CONT until primitive other than ALIGN or NOTIFY

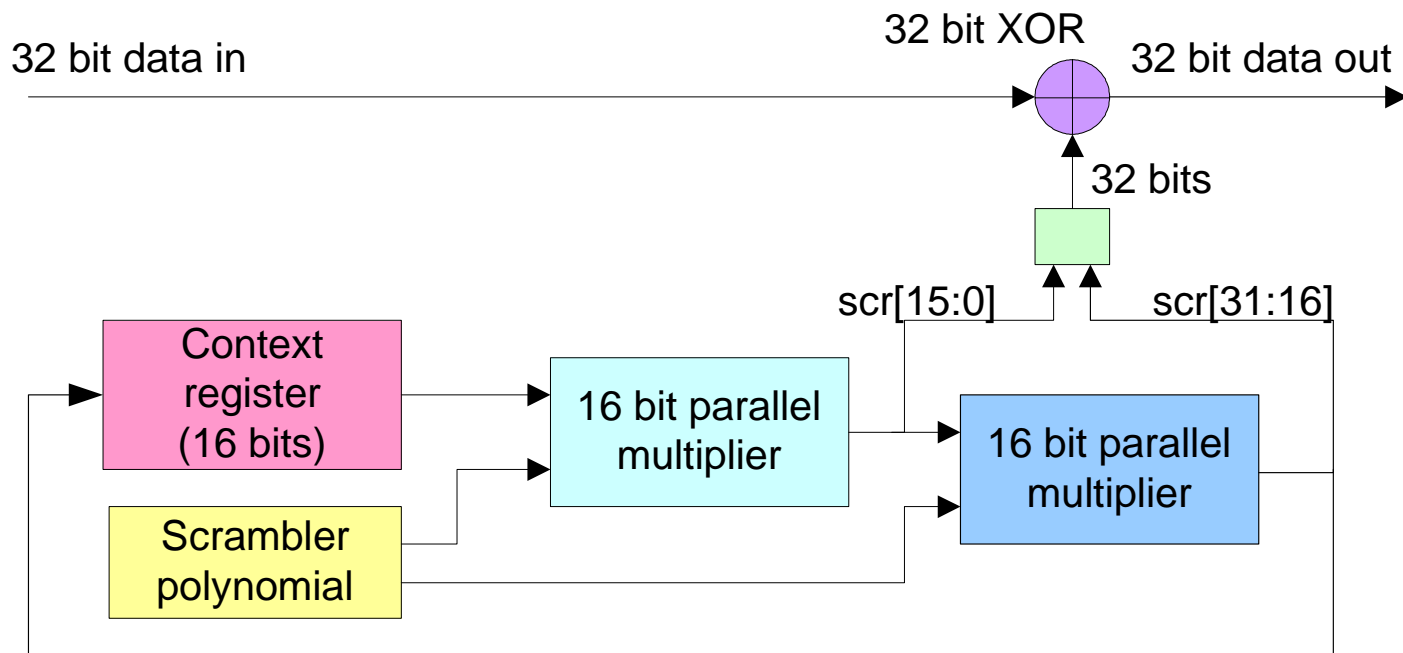
Scrambling generation

- Scrambling pattern reset at start of each frame
 - Initialized to FFFFh
- Scrambling acts on 16 bits at a time
 - Bits 15:0 first
 - Bits 31:16 next
- Scrambling generator polynomial
 - $G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$
- Scrambling pattern is fixed, not based on the data being transmitted

Scrambling implementation



- SATA Annex A contains C source code
- SAS Annex D contains Verilog source code

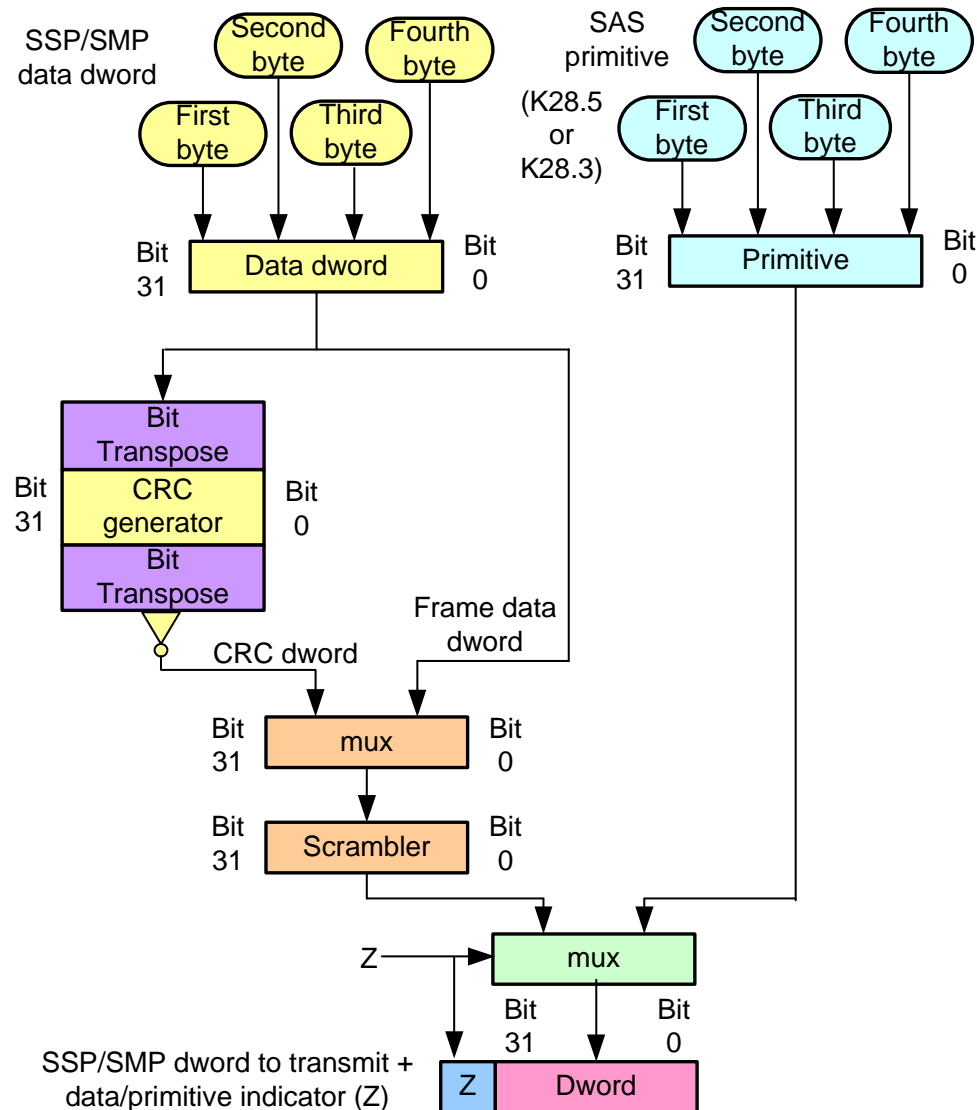


Link layer – Bit order

SAS transmit bit order



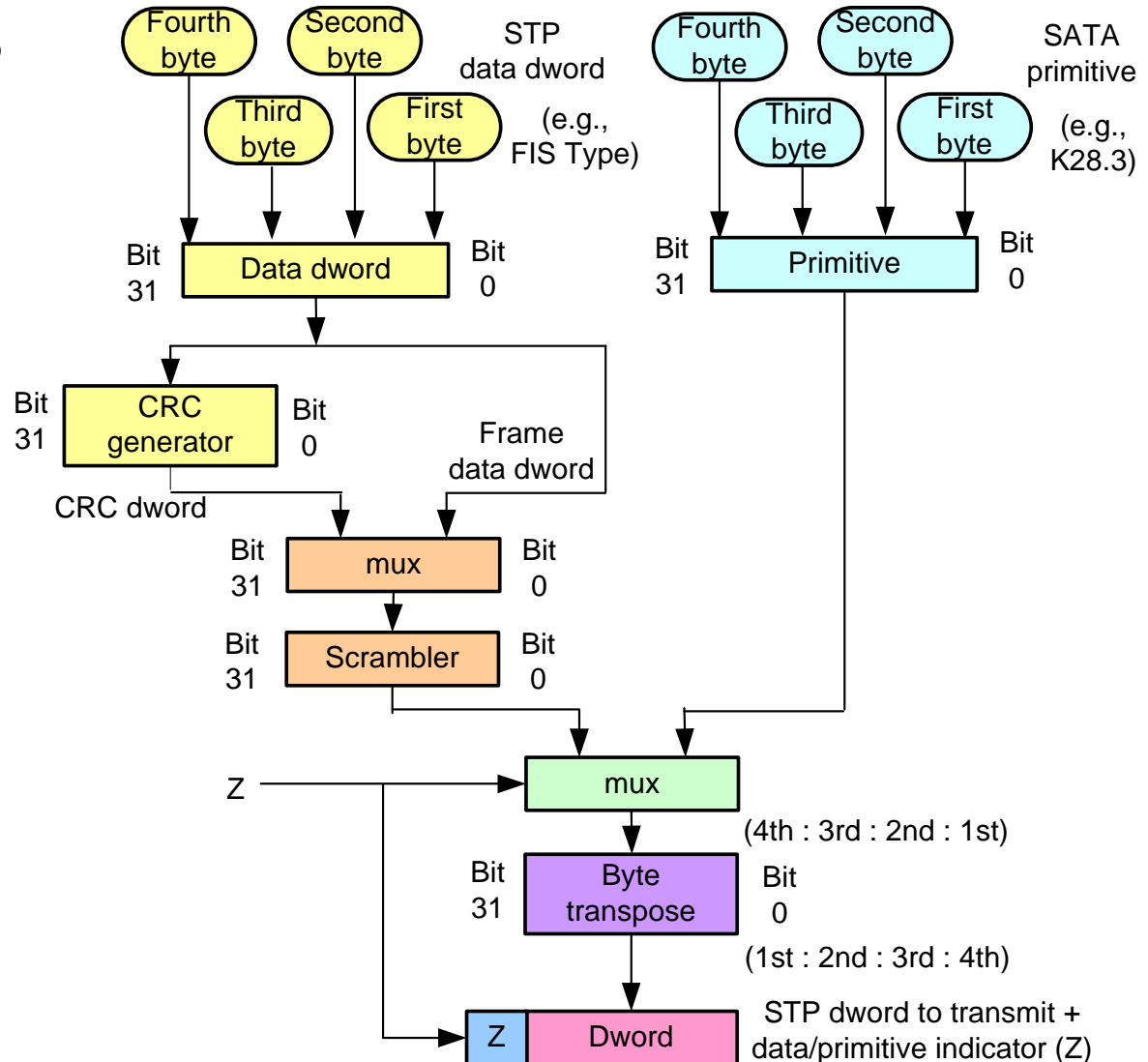
- Address frame, SSP frame, and SMP frame data dwords are big-endian
- Primitives outside connections and inside SSP and SMP connections can be interpreted as big-endian to match
- The CRC covers the unscrambled data
- Note that primitives are not involved in CRC calculations and are not scrambled



SAS STP transmit bit order



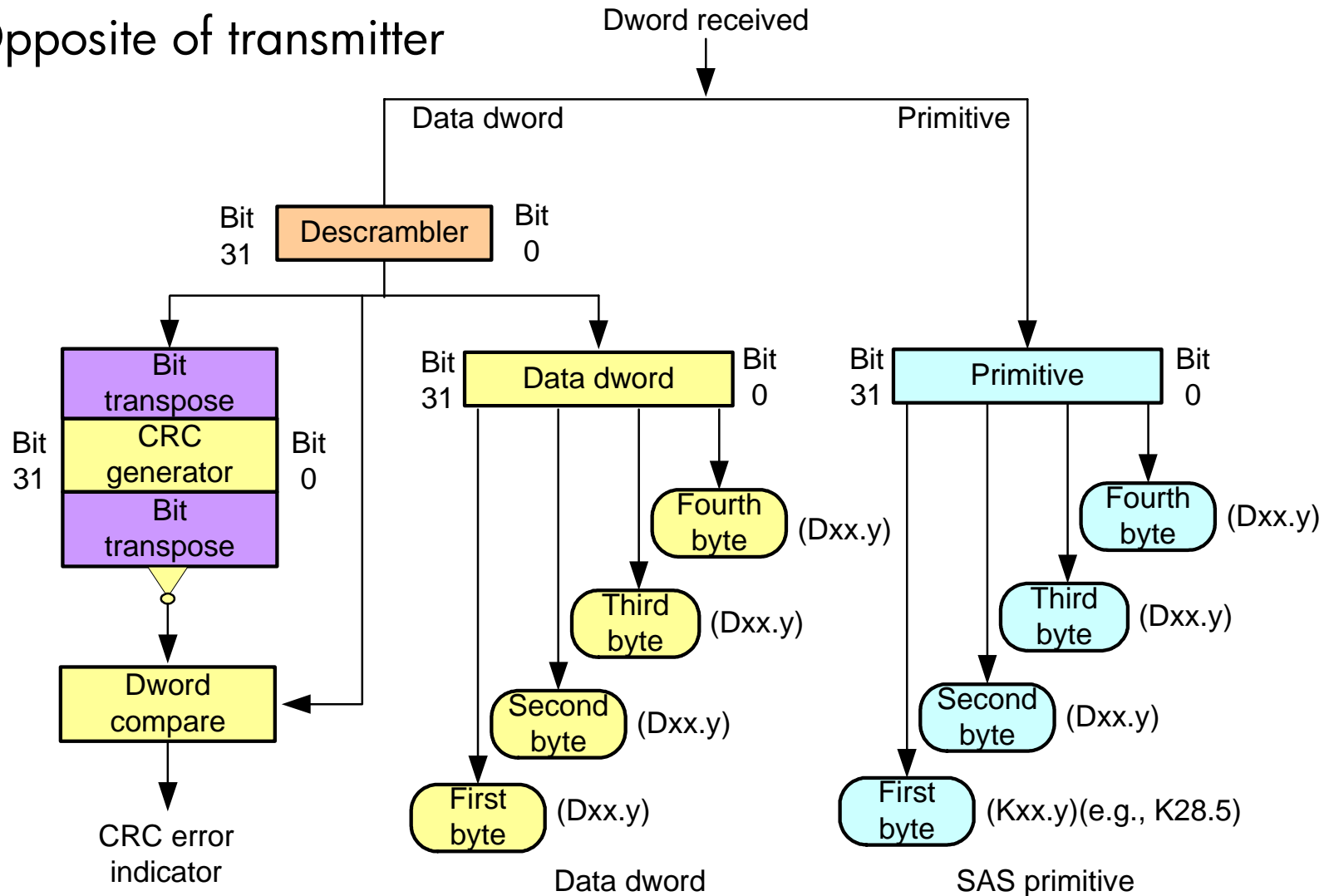
- Little-endian inside STP connections
- Bytes are transposed because the SAS dword transmitter (see phy layer) always transmits bits 31:24 first, 23:16 second, etc. (big-endian)
 - A pure SATA transmitter would not transpose
 - its phy layer would transmit the dword in little-endian byte order



SAS receive bit order



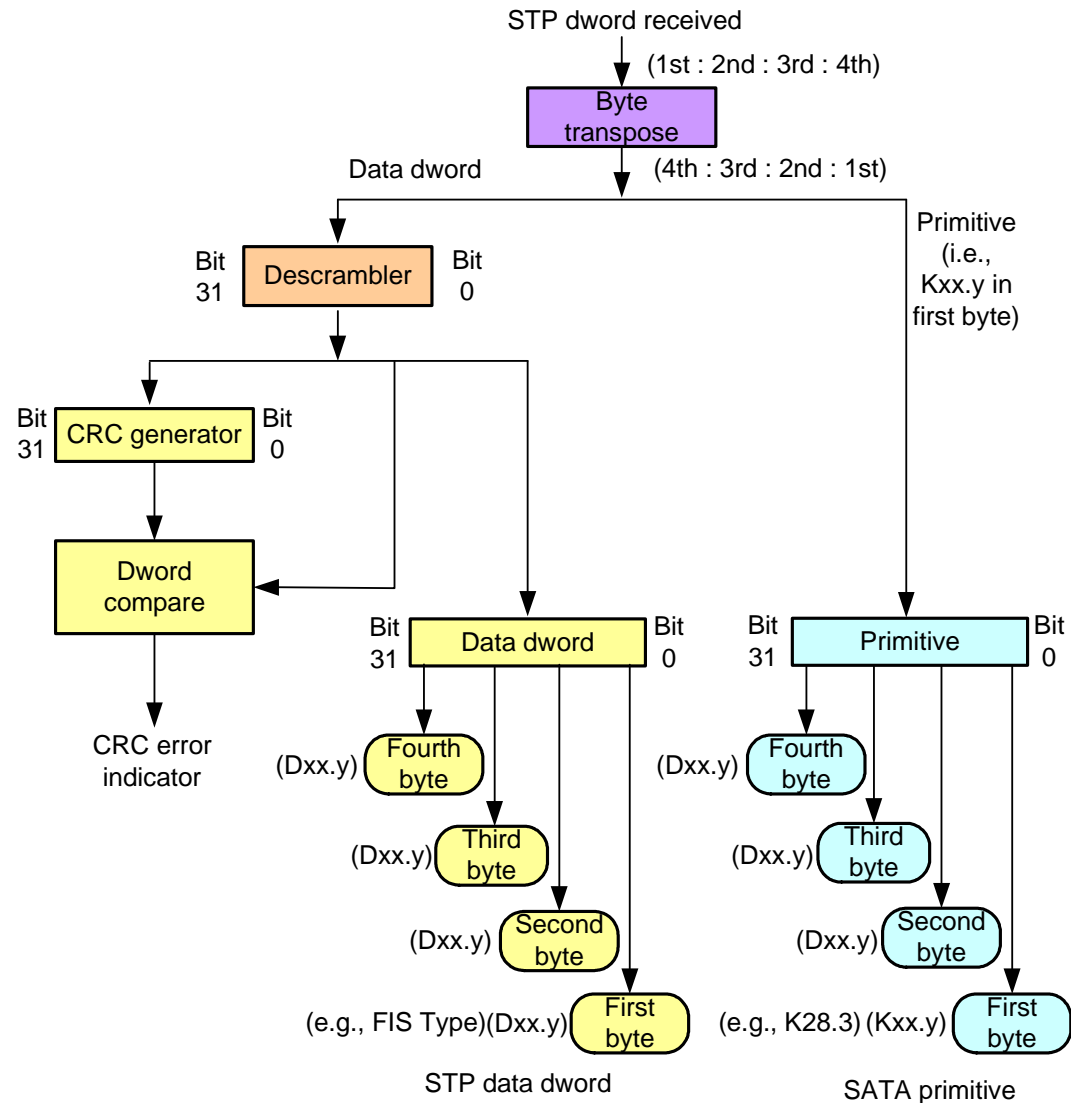
- Opposite of transmitter



SAS STP receive bit order



- Little-endian inside STP connections
- Byte transpose only present in SAS designs, where the phy layer receiver assumes big-endian byte ordering
 - Actual STP implementations can use big-endian ordering throughout more of the logic
 - Primitives could always be big-endian
 - Data dwords eventually must be treated as little-endian



Link layer – Address frames

Address frames



- SAS only
- Outside connections
- 32 bytes long (including CRC)
- **IDENTIFY address frame**
 - Exchanged by SAS phys after the phy reset sequence completes
 - Used to exchange SAS addresses
- **OPEN address frame**
 - Transmitted when the physical link is idle
 - Idle dwords being transmitted
 - No connection currently open
 - Used to establish a new connection
 - Specifies source and destination SAS addresses

IDENTIFY address frame



- Contains information about the phy transmitting the IDENTIFY
- **Device type**
 - End device, edge expander, or fanout expander
- **Protocol bits**
 - SSP, STP, and SMP initiator
 - SSP, STP, and SMP target
- **SAS address**
 - For SAS phys, indicates the SAS address of the SAS port containing the phy
 - For expander phys, indicates the SAS address of the expander device containing the phy
- **Phy identifier**
 - the phy identifier relative to the device (0..n)
 - Used for SMP functions

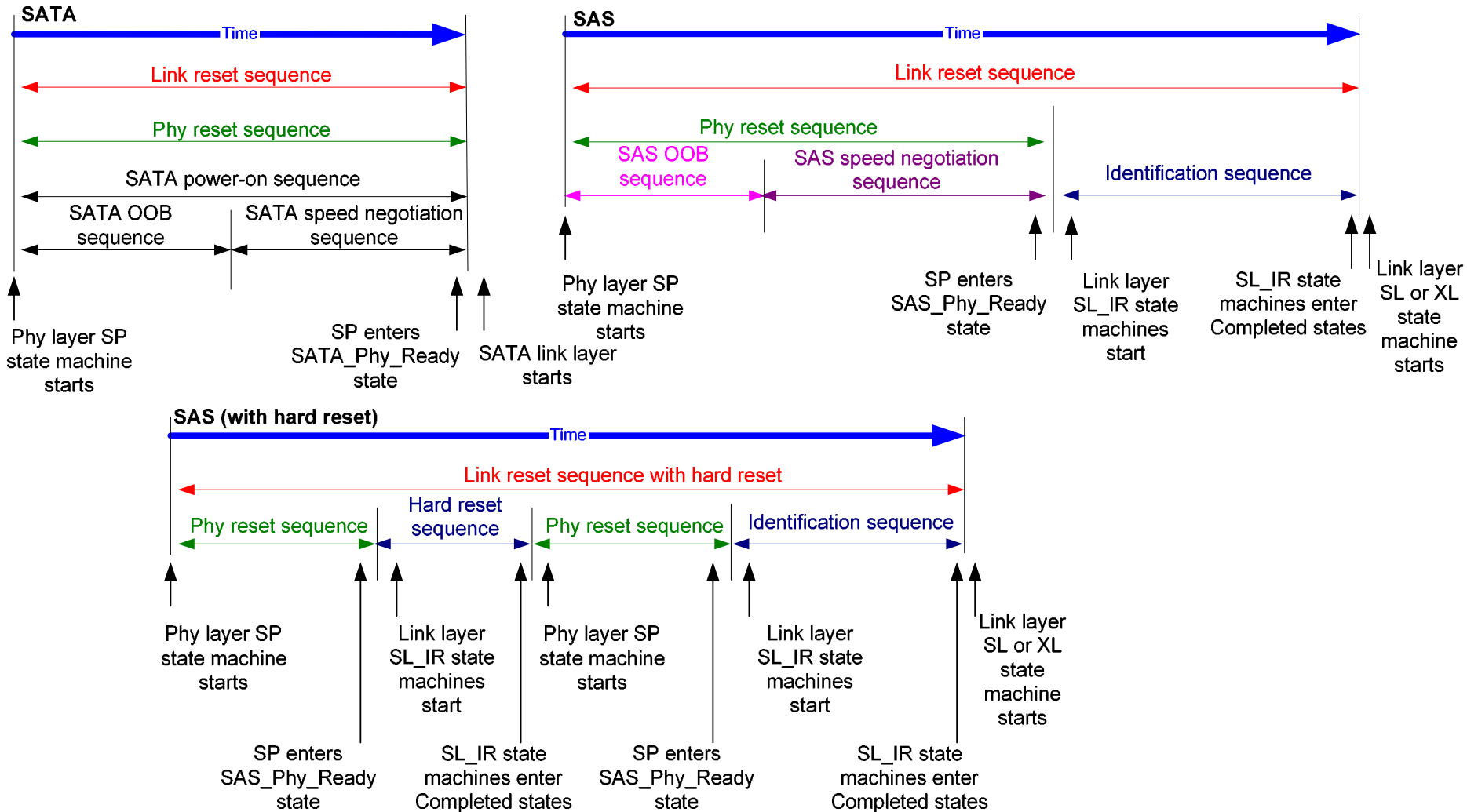
OPEN address frame



- Carries information needed to establish a connection
- **Protocol**
 - SSP, STP, or SMP
- **Connection rate**
 - Connection bandwidth 1.5 or 3.0 Gbps
- **Initiator connection tag**
 - Initiators may set to a unique cookie per target
 - Target uses the cookie when opening the initiator
 - Eliminates 64-bit SAS address comparison in initiator on incoming OPENs
- **Destination SAS address**
 - Tells the expanders where to send this request
- **Source SAS address**
 - Tells the recipient who is opening it
- **Pathway blocked count**
 - Used for arbitration fairness
- **Arbitration wait time**
 - Used for arbitration fairness

Link layer – Identification and hard reset sequences

Reset sequences (from General clause)



Identification and hard reset sequences



- After phy reset sequence completes, a phy can run either
 - **Hard reset sequence**
 - Transmit a HARD_RESET primitive
 - **Identification sequence**
 - Transmit an IDENTIFY address frame
 - If one phy transmits an IDENTIFY but the other transmits HARD_RESET, the HARD_RESET wins
- SATA phys do not implement either
 - SATA devices don't have SAS addresses
 - Running a phy reset sequence itself is considered a hard reset

Identification sequence



- Exchange IDENTIFY address frames
 - Expanders can direct-route to the attached SAS address
 - Initiators know what is attached and can perform discovery starting with that SAS address
 - Targets don't do much with this information
 - They can report their "attached" information in mode pages and log pages
 - If the target had commands queued and reran the link reset sequence, it can resume opening connections if the attached SAS address is the same

Hard reset sequence

- HARD_RESET in at least one direction
- Forces a low-level reset
 - Similar to parallel SCSI bus reset line
 - Not recommended for general use
 - Purposely difficult to transmit – only honored after phy reset sequence
 - Afterwards, another phy reset sequence must be run

SAS SL_IR state machines

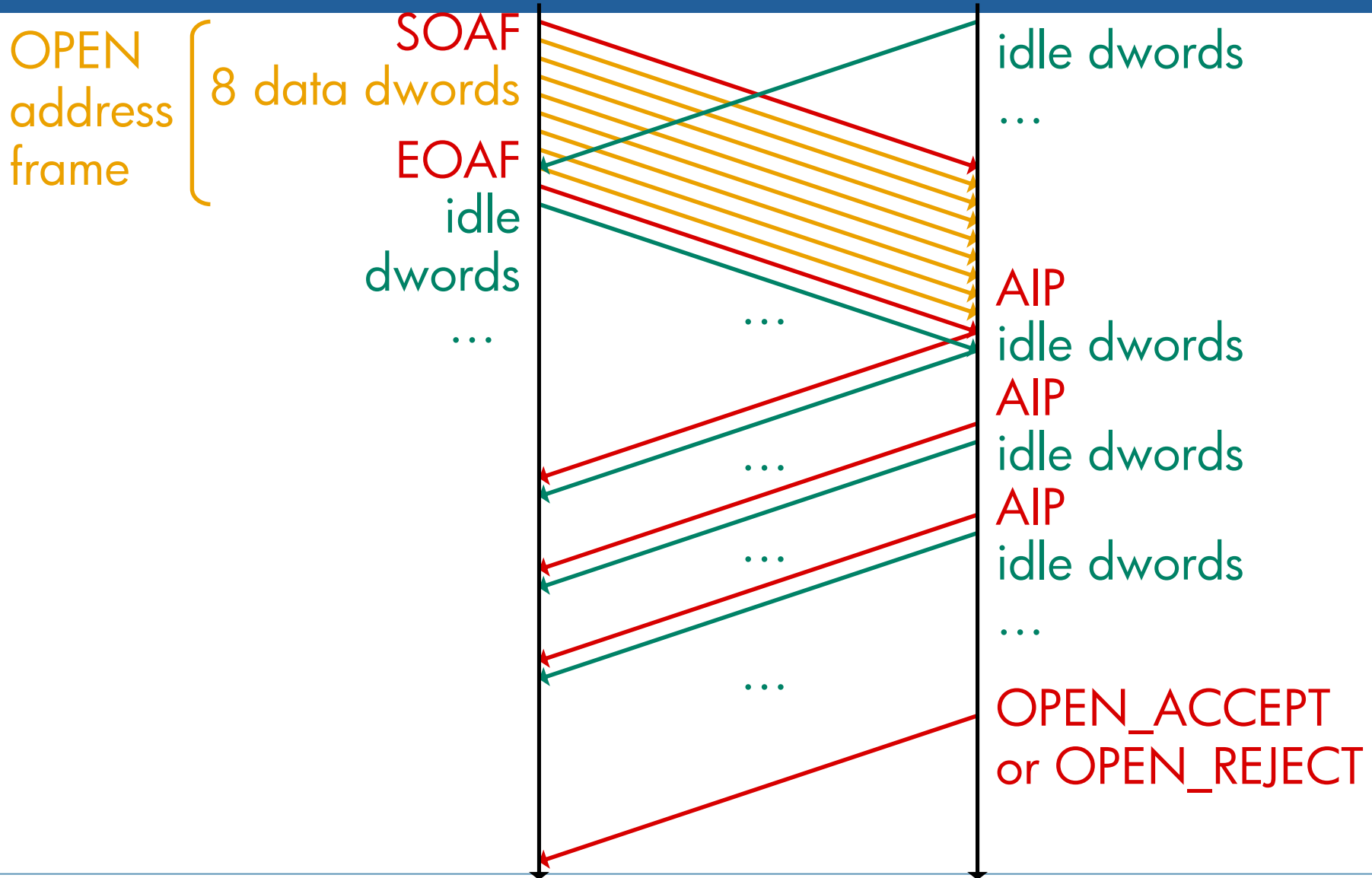


- Runs identification and hard reset sequences
- Three state machines
 - **SL_IR_TIR** – transmits IDENTIFY address frame or HARD_RESET primitive
 - **SL_IR_RIF** – receives an IDENTIFY address frame
 - Supports both SATA and SAS speed negotiation
 - **SL_IR_IRC** - coordinator
- Receive Identify Timeout timer
 - Must receive an IDENTIFY address frame within 1 ms, or rerun the phy reset sequence
- When complete, allows the SL state machines to transmit and receive dwords
- IDENTIFY address frames and HARD_RESET primitives ignored after that

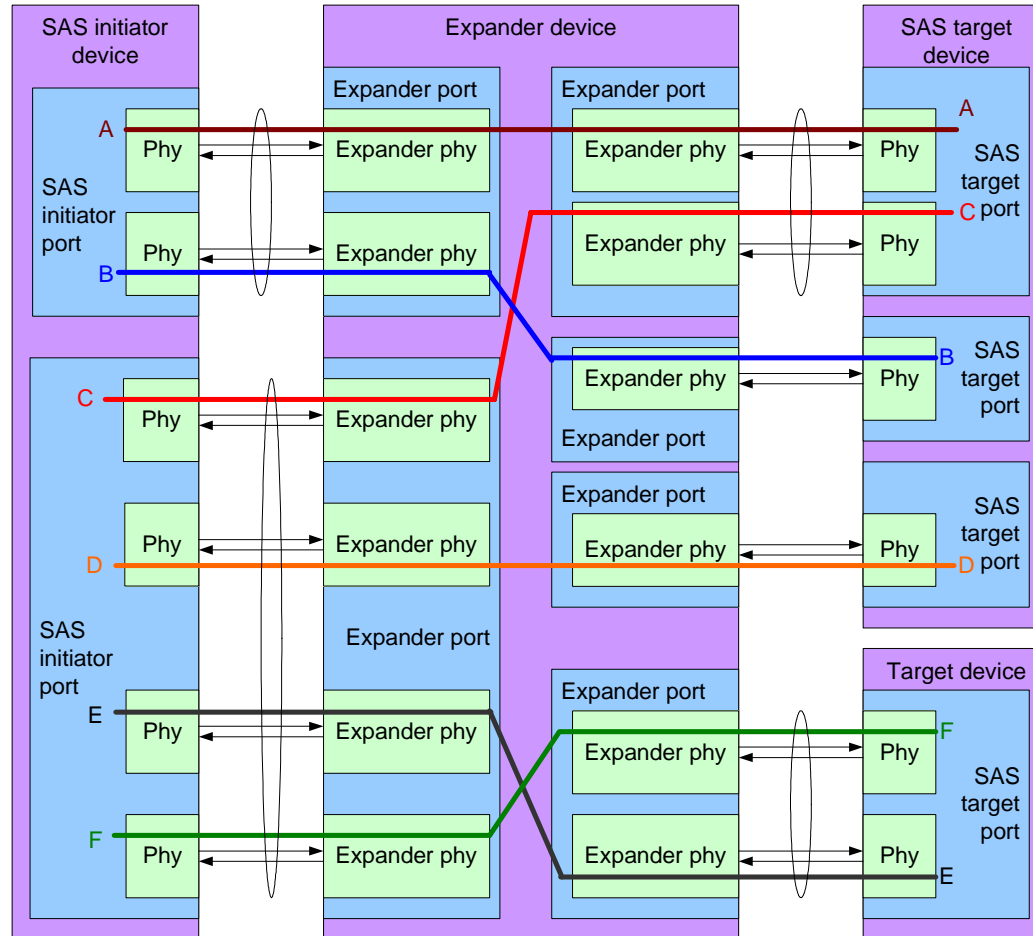
Link layer – Connections

- Established between a SAS initiator phy and a SAS target phy
- Three types of connections
 - SSP (Serial SCSI protocol)
 - SMP (Serial management protocol)
 - STP (Serial ATA Tunneling protocol)
- Basic sequence
 - 1. Transmit OPEN address frame
 - 2. Receive AIP (arbitration in progress) primitives if expanders are involved
 - 3. Receive final result
 - OPEN_ACCEPT primitive – connection established
 - OPEN_REJECT primitive – rejected, go back to idle
- SATA has no concept of connections

Connection – basic sequence



Connection examples



Key: \longleftrightarrow Single physical link \bigcirc Wide link X — X Connection

Notes: The expander device has a unique SAS address. Each SAS initiator port and SAS target port has a unique SAS address. Connections E and F represent a wide SAS initiator port with two simultaneous connections to a wide SAS target port.

Opening a connection

- Connection requests are directed to a SAS port, not a specific phy
 - Multiple phys may share the same SAS address
 - Expanders are free to route the connection request to any matching phy
- Open Timeout timer
 - 1 ms timer started after transmitting OPEN address frame
 - Reinitialized and restarted when an AIP primitive arrives
- Expanders store and forward OPEN address frames

Connection response – AIP (arbitration in progress)



- Expanders are working on the request
- Resets the Open Timeout timer
- AIP (NORMAL) transmitted as soon as the expander decodes the OPEN address frame
- AIP transmitted every 128 dwords while expander is picking an output phy
 - AIP (WAITING ON CONNECTION) if the output phy is occupied with another connection
 - AIP (WAITING ON PARTIAL) if the output phy is already being used for a different incomplete connection request
 - AIP (WAITING ON DEVICE) as the OPEN address frame is transmitted out the output phy
- Once OPEN address frame is transmitted out the output phy, no more AIPs from this expander
 - All dwords received by that output phy are forwarded back
 - May include AIPs from the next expander

Connection response – OPEN_ACCEPT



- The connection request was accepted
- A bidirectional dword stream is now open from source phy to destination phy
- Start following SSP, SMP, or STP protocol

Connection response – OPEN_REJECT



- The connection request failed
- Classes
 - Abandon – give up the connection request
 - Unsupported protocol, routing problem, etc.
 - Retry – resend the connection request
 - Destination said “Retry”, temporary blockage in the domain

Connection response – OPEN_REJECT abandon responses



- OPEN_REJECT (BAD DESTINATION)
 - Expander routing tables indicate a request would have to go back out the same port it came in on
- OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)
 - Connection rate is faster than physical link rate
 - (Special rules for connection rates faster than 1.5 Gbps)
- OPEN_REJECT (PROTOCOL NOT SUPPORTED)
 - End device doesn't support the requested SSP, STP, or SMP protocol, initiator/target role, or initiator connection tag
- OPEN_REJECT (STP RESOURCES BUSY)
 - STP target port has an active "affiliation" with a different initiator
- OPEN_REJECT (WRONG DESTINATION)
 - Destination SAS address doesn't match the address of the phy in an end device that receives the request

Connection response – OPEN_REJECT retry responses



- OPEN_REJECT (NO DESTINATION)
 - No such destination SAS address in the domain
 - Connection request arrives through a subtractive phy, and there is no hit
 - Bound to an STP/SATA bridge, but the SATA device has not yet delivered its initial Register FIS
- OPEN_REJECT (PATHWAY BLOCKED)
 - Partial Pathway Timer expired in an expander
- OPEN_REJECT (RETRY)
 - Destination is temporarily busy; try again
 - Destination must eventually accept the request

OPEN_REJECT priorities



- If there are multiple reasons a request could be rejected
 - By a destination phy
 - WRONG DESTINATION (highest priority)
 - PROTOCOL NOT SUPPORTED
 - CONNECTION RATE NOT SUPPORTED
 - STP RESOURCES BUSY
 - RETRY (lowest priority)
 - By an expander device between the source phy and destination phy
 - BAD DESTINATION or NO DESTINATION (highest priority)
 - CONNECTION RATE NOT SUPPORTED
 - STP RESOURCES BUSY or PATHWAY BLOCKED (lowest priority)

Connection response – OPEN address frame



- Since SAS is full duplex, OPENs can cross on the wire
- If OPEN arrives before an AIP arrives
 - Compare Arbitration wait time, then Source SAS address (from the OPEN address frames)
 - Highest value wins

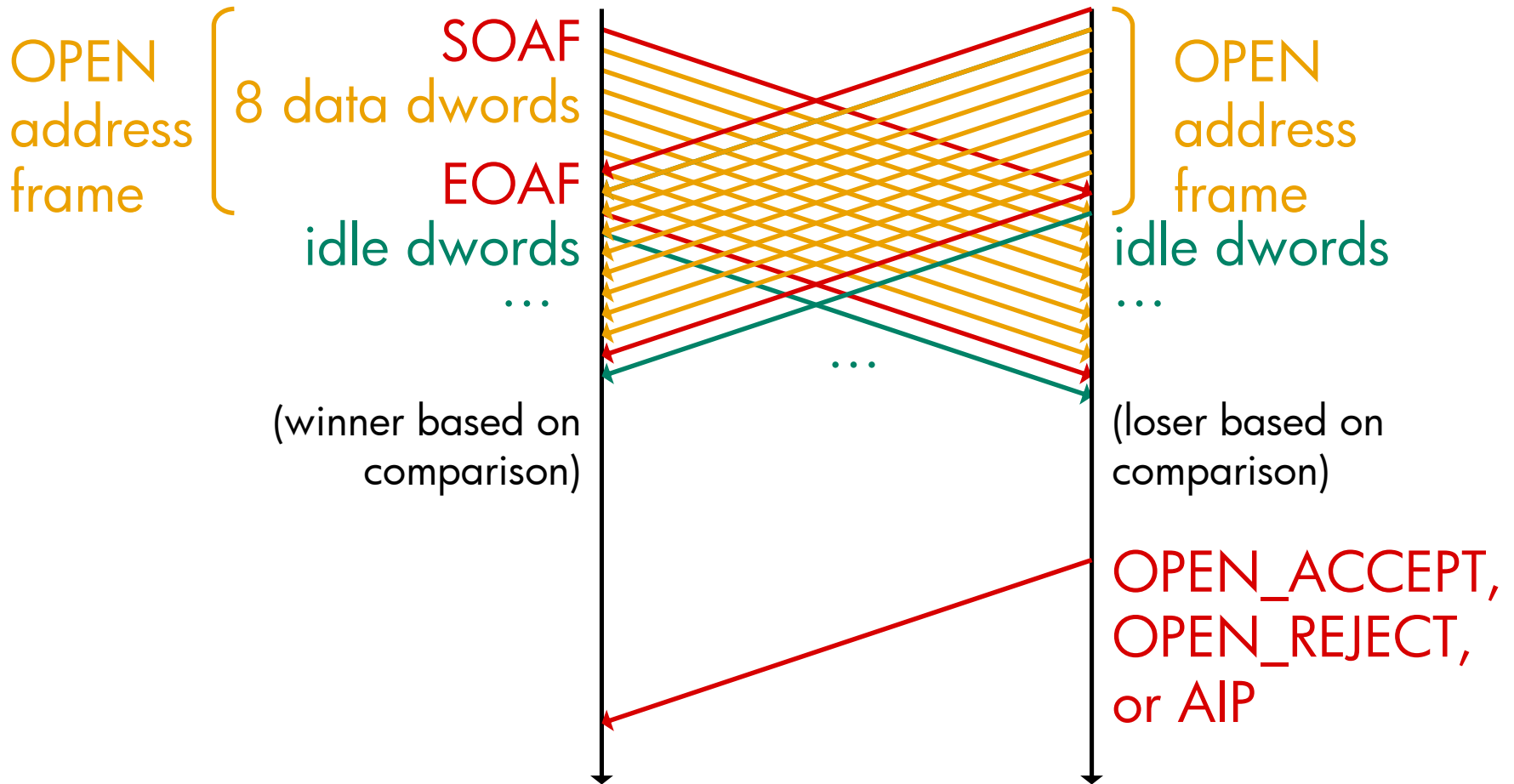


- If OPEN arrives after AIP arrives
 - Assume the incoming OPEN wins
- Loser behavior
 - If loser is an end device, generate OPEN_ACCEPT or OPEN_REJECT
 - If loser is an expander device, generate AIP (NORMAL)

Connection response – OPEN address frame before AIP



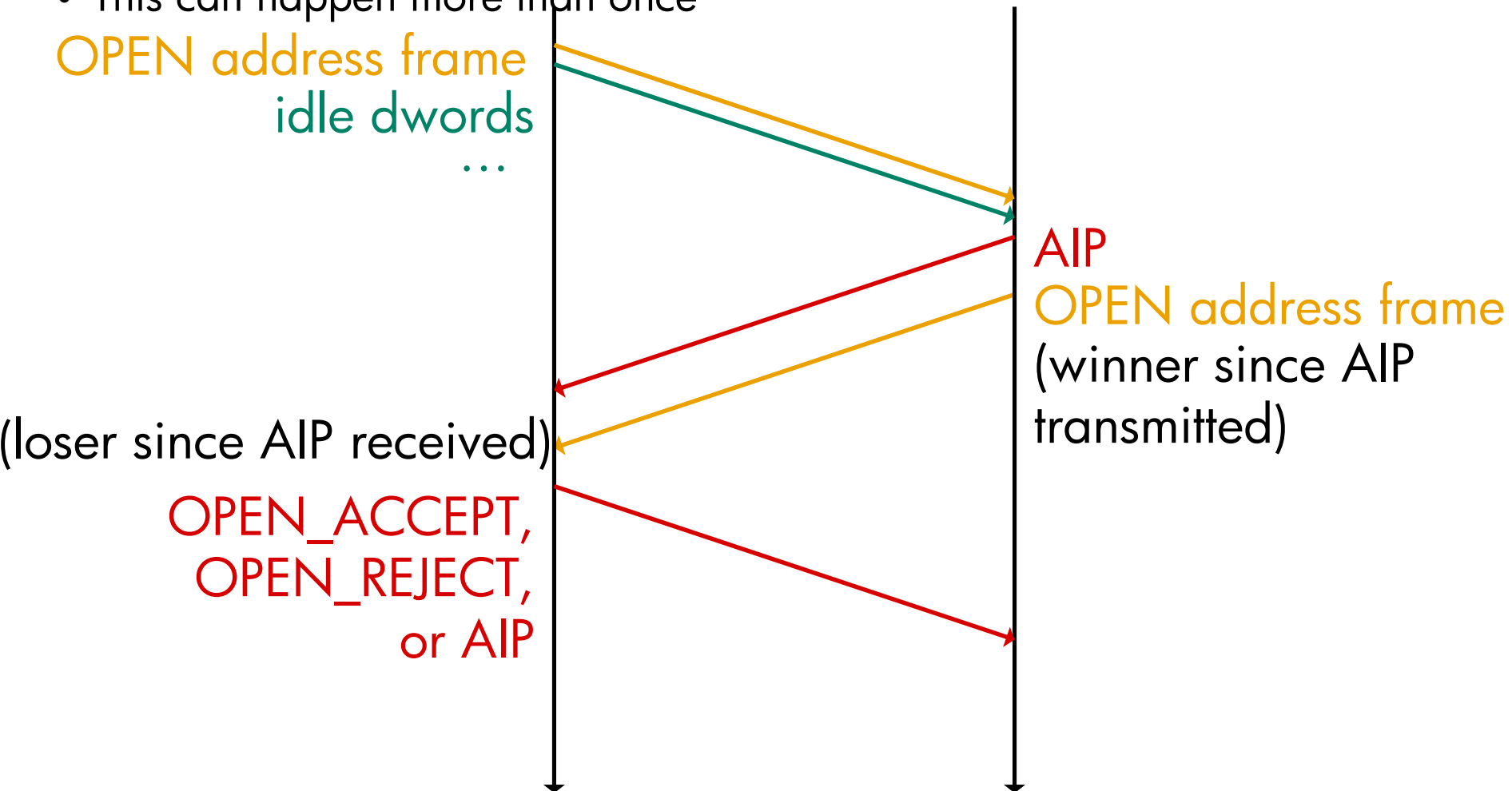
- Arbitration fairness comparison dictates the winner



Connection response – OPEN address frame after AIP



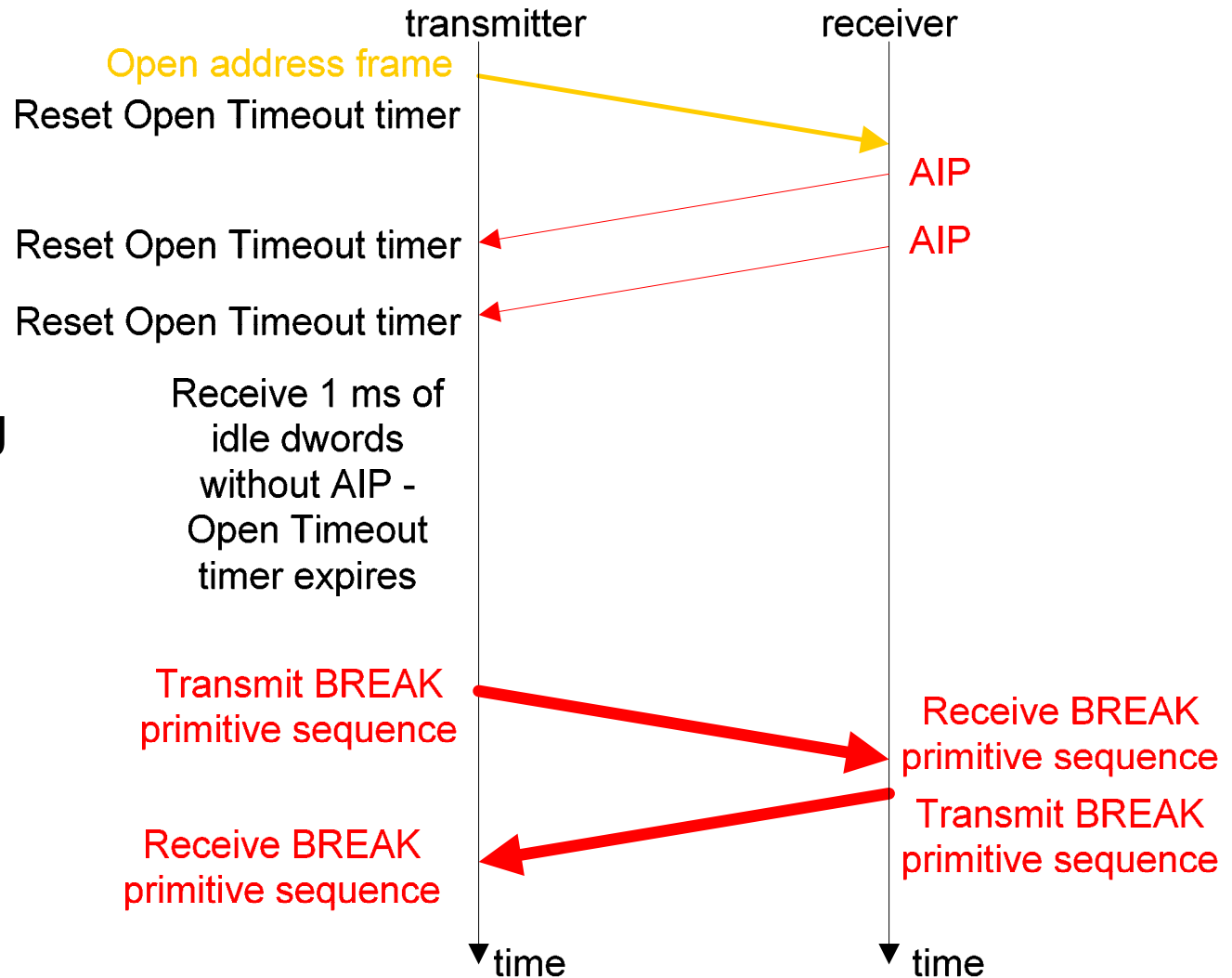
- After sending AIP, expander must not send an OPEN unless it needs to win
- This can happen more than once



Connection request timeout – aborting a connection requests



- If Open Timeout timer expires, transmit a BREAK
- Should never happen
- After transmitting BREAK, expect to receive a BREAK within a 1 ms Break Timeout time

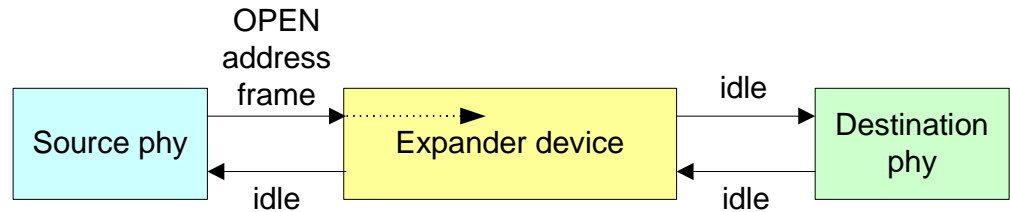


Connection request BREAK forwarding in an expander

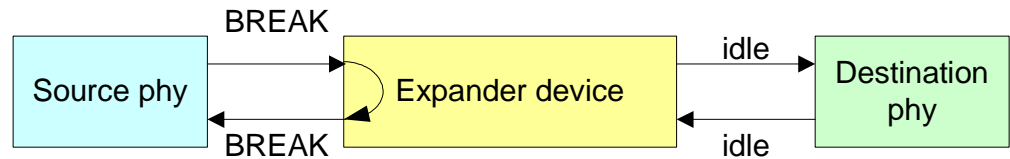


- BREAK is handled separately on each physical link
- Expander replies to BREAK directly
- Then, it forwards a BREAK only if necessary (if the OPEN address frame has been transmitted)

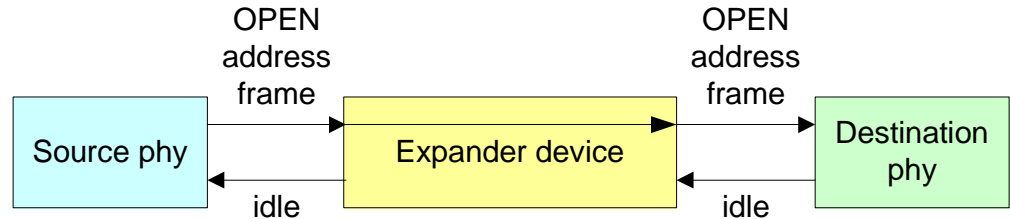
Case 1: OPEN address frame has not propagated through the expander device:



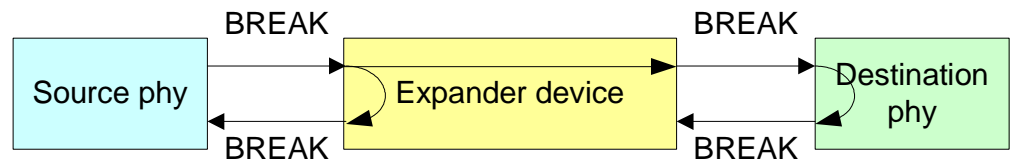
Case 1 result: BREAK only on Source device physical link



Case 2: OPEN address frame has propagated through the expander device:



Case 2 result: BREAK on Source device's physical link, then on destination device's physical link

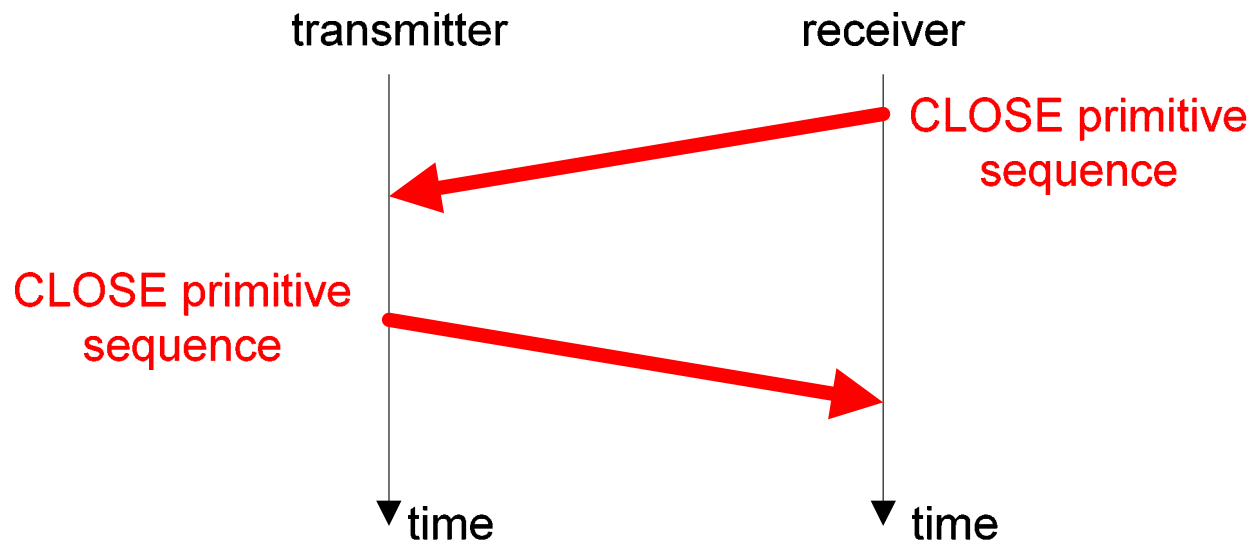


Closing a connection

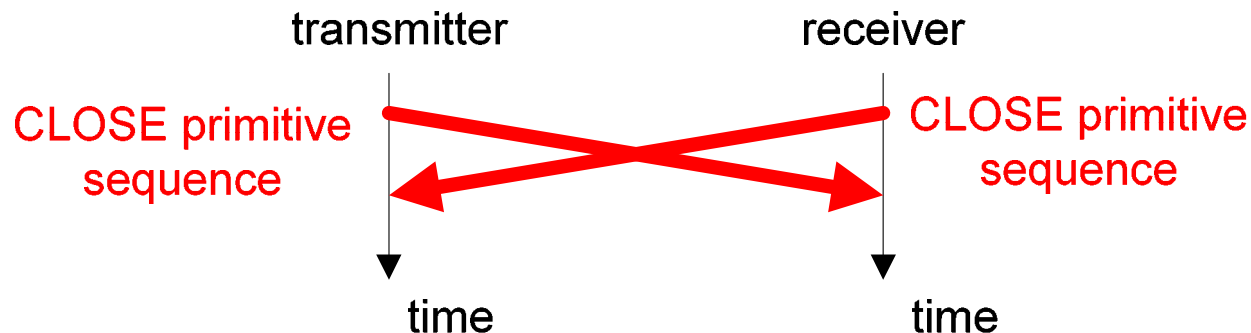


- Exchange CLOSE primitive sequences end-to-end to close a connection
- SSP and SMP require traffic be stopped first
 - CLOSE ignored by SAS phy state machines if not at the correct time
- STP connection should be idle before closing
 - Not enforced by SL state machine
- Expanders always honor CLOSE regardless of connection state

Example 1: CLOSEs sent one at a time



Example 2: CLOSEs sent simultaneously



Breaking a connection



- If CLOSE fails, BREAK can be used
- CLOSE is send-3, receive-3; BREAK is send-6 receive-3
 - Single bit error invalidates a CLOSE, but not a BREAK
- BREAK is not end-to-end
 - Each expander replies to it locally, then forwards to the outbound phy
 - Same as for aborting connections with BREAK (but here the connection was open, so it always gets forwarded)
- After transmitting BREAK, start a 1 ms Break Timeout timer looking for a BREAK in response
 - If timer expires, assume the physical link is idle
 - might be prudent to perform a phy reset sequence

Link layer – Connection management state machines

SAS SL state machines

- In the SAS phy object
- Establishes connections
- Two state machines
 - SL_CC – Connection Control
 - SL_RA – Receive OPEN address frame
- Timers
 - 1 ms Open Timeout timer
 - 1 ms Close Timeout timer
 - 1 ms Break Timeout timer
- **SL_CC7:Connected** state hands over dword transmission and reception to the SSP, STP, or SMP link layer state machines

SAS XL state machine



- In the expander phy object
- Establishes connections between expander phys
- Timers
 - Partial Pathway Timeout timer
 - Initial value reported in the SMP DISCOVER function and set via the SMP PHY CONTROL function
 - Recommended value $7 \mu s$
 - This allows time for OPEN_REJECT (PATHWAY BLOCKED) to propagate through the domain
 - 1 ms Break Timeout timer
- **XL7:Connected** state means a full duplex connection is established between two phys



Wrap up

Serial Attached SCSI tutorials



- General overview (~2 hours)
- Detailed multi-part tutorial (~3 days to present):
 - Architecture
 - Physical layer
 - Phy layer
 - Link layer
 - Part 1) Primitives, address frames, connections
 - Part 2) Arbitration fairness, deadlocks and livelocks, rate matching, SSP, STP, and SMP frame transmission
 - Upper layers
 - Part 1) SCSI application and SSP transport layers
 - Part 2) ATA application and STP/SATA transport layers
 - Part 3) Management application and SMP transport layers, plus port layer
 - SAS SSP comparison with Fibre Channel FCP

Key SCSI standards

- Working drafts of **SCSI** standards are available on <http://www.t10.org>
- Published through <http://www.incits.org>
 - Serial Attached SCSI
 - SCSI Architecture Model – 3 (SAM-3)
 - SCSI Primary Commands – 3 (SPC-3)
 - SCSI Block Commands – 2 (SBC-2)
 - SCSI Stream Commands – 2 (SSC-2)
 - SCSI Enclosure Services – 2 (SES-2)
- **SAS connector** specifications are available on <http://www.sffcommittee.org>
 - SFF 8482 (internal backplane/drive)
 - SFF 8470 (external 4-wide)
 - SFF 8223, 8224, 8225 (2.5", 3.5", 5.25" form factors)
 - SFF 8484 (internal 4-wide)

Key ATA standards



- Working drafts of **ATA** standards are available on <http://www.t13.org>
 - Serial ATA 1.0a (output of private WG)
 - ATA/ATAPI-7 Volume 1 (architecture and commands)
 - ATA/ATAPI-7 Volume 3 (Serial ATA standard)
- **Serial ATA II** specifications are available on <http://www.t10.org> and <http://www.serialata.org>
 - Serial ATA II: Extensions to Serial ATA 1.0
 - Serial ATA II: Port Multiplier
 - Serial ATA II: Port Selector
 - Serial ATA II: Cables and Connectors Volume 1

For more information



- International Committee for Information Technology Standards
 - <http://www.incits.org>
- T10 (SCSI standards)
 - <http://www.t10.org>
 - Latest SAS working draft
 - T10 reflector for developers
- T13 (ATA standards)
 - <http://www.t13.org>
 - T13 reflector for developers
- T11 (Fibre Channel standards)
 - <http://www.t11.org>
- SFF (connectors)
 - <http://www.sffcommittee.org>
- SCSI Trade Association
 - <http://www.scsita.org>
- Serial ATA Working Group
 - <http://www.serialata.org>
- SNIA (Storage Networking Industry Association)
 - <http://www.snia.org>
- Industry news
 - <http://www.infostor.com>
 - <http://www.byteandswitch.com>
 - <http://www.wwpi.com>
 - <http://searchstorage.com>
- Training
 - <http://www.knowledgetek.com>



i n v e n t