TCP Framing Discussion Slides

Palo Alto, CA

June 27-28, 2001

Jim Wendt

F2F Agenda – Wed 6/27/01

- Meeting Context
- TCP Shortcomings Summary
  - NLP Datagrams / Messages
  - Error Performance
  - Latency Performance?
- Direct Data Placement Discussion
  - System Goals
  - Memory-Based Solution Feasibility at 10Gbps
  - DDP Mechanisms
  - Discussion
  - Generate Recommendations & Actions
- TCP Framing Discussion
  - Discussion Goals
  - Need for Framing
  - Taxonomy / Categories / Alternatives
  - Requirements / Constraints / Cost Vectors
  - Correlation (Requirements / Costs / Alternatives)
  - Decision Tree
  - Generate Recommendations & Actions
- iSCSI Layering Evolution Discussion
  - iSCSI / RDMA / Framing Evolution Discussion
  - Generate Recommendations & Actions

F2F Agenda – Thur 6/28/01

- iSCSI Error Recovery and Data Integrity
  - Data Integrity / Error Recovery Goals
  - Crypto Hardware Feasibility at 10Gbps
  - Layering Alternatives
  - Architectural Implications of PDU-level Recovery
  - iSCSI Error Recovery Mid/Min/Max “Profiles”
  - Generate Recommendations and Actions

DDP & Framing Discussion Goals

- Direct Data Placement Discussion
  - Common understanding of alternatives / capabilities / costs
  - Recommend / reaffirm Direct Data Placement approach
- Framing Discussion
  - Common understanding of requirements
  - Common understanding of alternatives / capabilities / costs
  - Recommend Framing approach
- iSCSI Layering Discussion
  - Recommend iSCSI / RDMA / Framing layering approach & evolution

TCP Shortcomings Summary

- TCP Shortcomings
  - Byte-stream only / no NLP message support
  - Error Performance
  - Latency Performance?

Direct Data Placement (DDP) Discussion

DDP System Goals

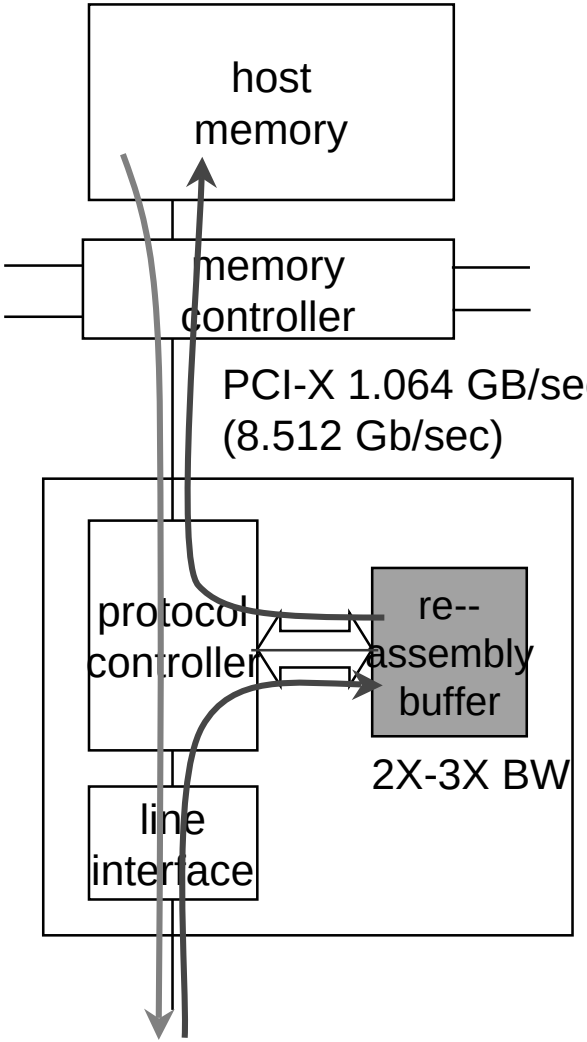
- Storage Access:
  - Is bursty
  - Will utilize full bandwidth of link
  - Priority is on minimizing response time
    - Data transfers just stretch response time

- Fibre Channel today:
  - Host OS executes Start-IO and IO-Completion paths only once per disk transaction
    - Nothing else requires host attention or processing between these
  - Upon IO-Completion the data is where SCSI expects it
  - Data in host memory is touched only once per disk transaction
    - Memory bandwidth is #1 problem / CPU is already starved of memory bandwidth
    - (Exception is that SCSI Write retries can access data again)

- iSCSI has to duplicate this behavior to be viable

Memory-Based Solution Feasibility

"Conventional"



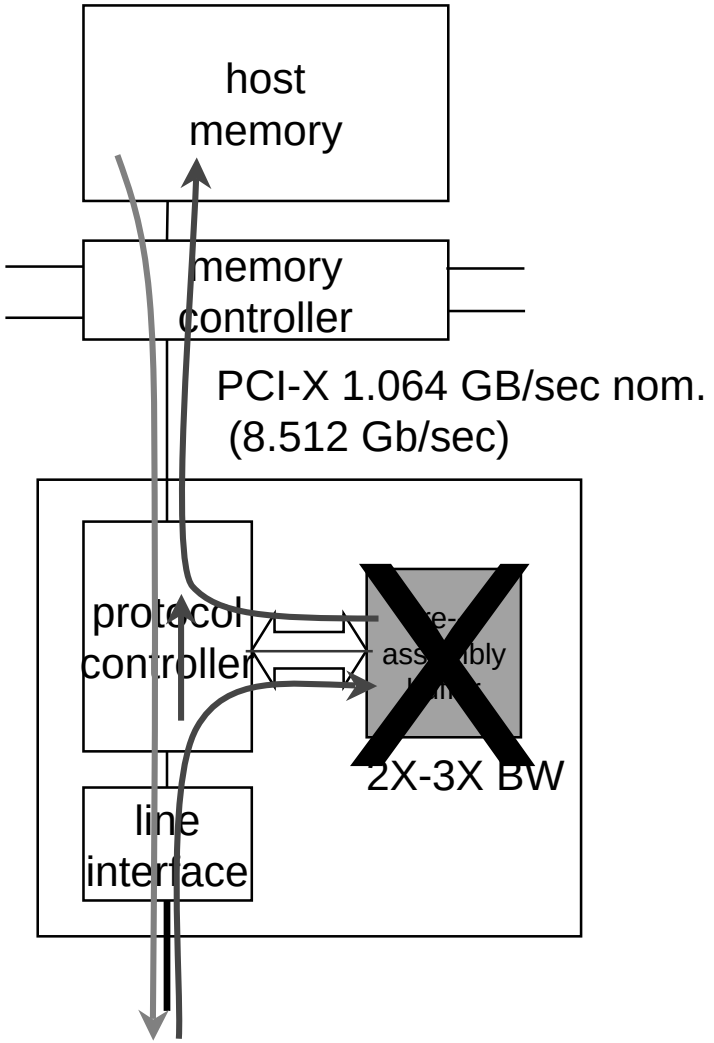
1 Gbps

- **Possible**
- **More expensive than FC**

10 Gbps

- **Questionable feasibility**
- **BWDP (100ms or 20K km) = 125 MB**

Desired

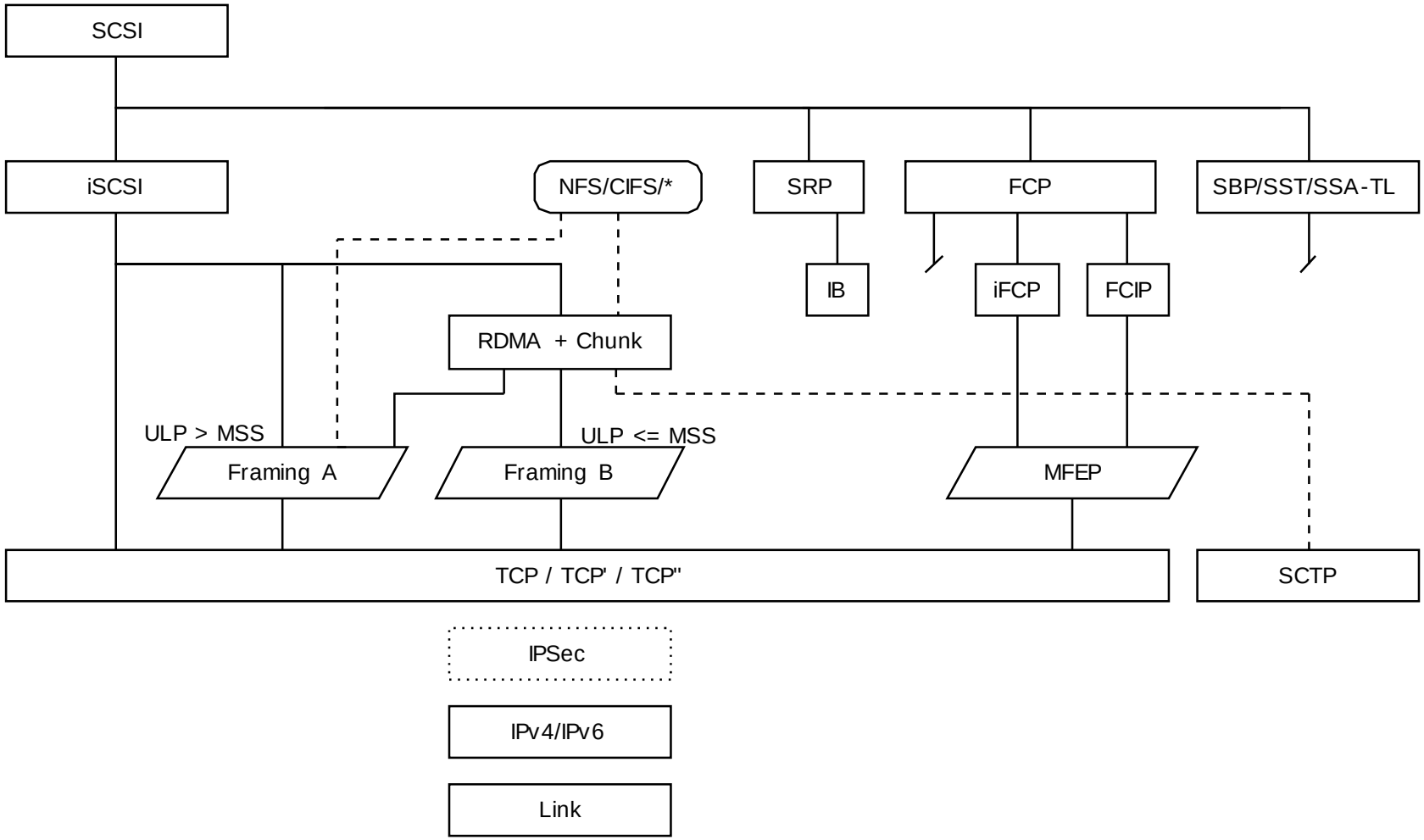


DDP Mechanisms

- ULP-Specific data placement / packet classification
  - Derive placement information from protocol headers and state info
  - Over TCP – get framing problem
  - Over SCTP – OK (message-based transport)

- General RDMA protocol
  - Explicit sender-to-receiver placement directives
  - Scatter across application and kernel buffers
  - RDMA can ride at:
    - RDMA above transport protocol
      - Over TCP – get framing problem
      - Over SCTP - OK
    - RDMA at transport
      - Example is: TCP options area holds RDMA info
    - RDMA below transport
      - TAF proposal (RDMA info bound to TCP segment payload - but TAF protocol header and trailer layered between TCP and IP)

Framing – iSCSI Layering Scenarios



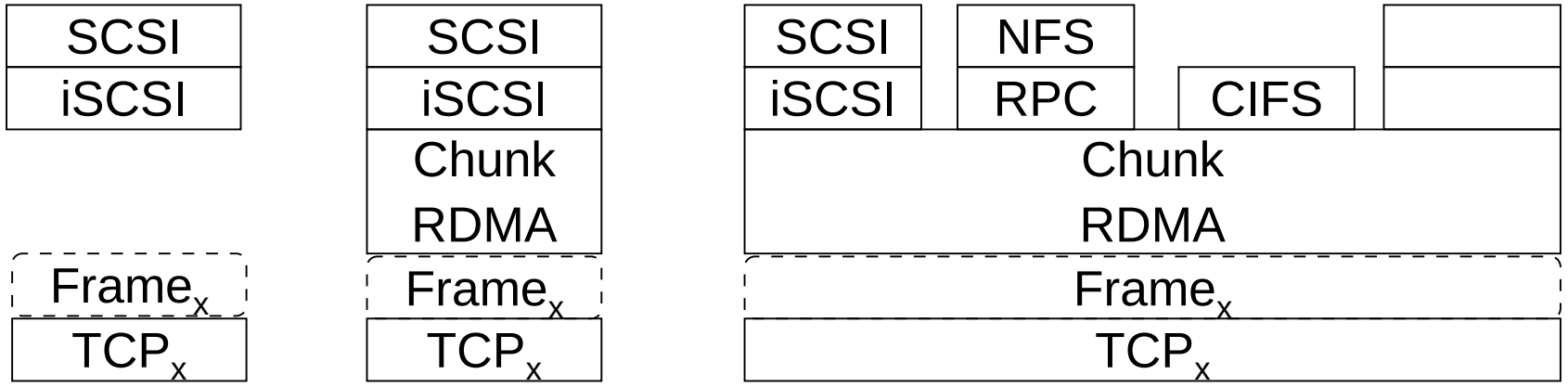
iSCSI Short Term Solution / 1Gbps

1) Use full reassembly buffer or use DDP+Framing ?
2) DDP via iSCSI-specific method
   - General RDMA protocol probably not available in timeframe
3) Framing over TCP provided by ?
   a) iSCSI specific mechanism (Markers, Periodic Alignment, etc)
   b) TCP-ULP-Framing I-D
   c) Other (byte stuffing, ?)

iSCSI Long Term Solution / 10Gbps

- Should we solve problem more generally within IETF?
  - Fully-collaborative IETF effort
  - Foster high-speed low-cost reliable-block-transport community
  - Future of RDMA in IETF ?
  - SCTP community involvement ?
  - Piggyback on the TCP-ULP-Framing work item ?
- iSCSI Long term solution
  - TCP or SCTP or other transport protocol?
  - DDP via generalized RDMA protocol or iSCSI-specific method?
  - TCP framing mechanism via ?
  - SCTP changes to support RDMA ?
- Design Team activity
  - Research & Analysis
  - Design Alternatives & Proposal
  - Prototypes

Framing – Possible DDP Layerings



- ? DAFS / VI-TCP
- ? SRP / RDMA

Checkpoint

- q? Is Direct Data Placement is required
- q? Is TCP is the transport protocol
- q? How is Direct Data Placement performed:
  - q? At the iSCSI level
  - q? Via a generalized RDMA protocol
  - q? Via another mechanism (TAF?, TCP-options?, Other?)
- Can framing solution decision be made independently of iSCSI layering choice?
- Should framing solution be the same for iSCSI/TCP and iSCSI/RDMA/TCP?

TCP Framing Discussion

Framing – Possible Process

- Flush out potential requirements for Framing solution (?and weights)
- Flush out cost vectors and per-class or per-alternative costs
- Create Framing mechanism taxonomy
- Flush out classes of Framing alternatives and best-in-class
- Correlate potential requirements to Framing alts and features
  - (E.g. Interoperation with standard TCP stack precludes TCP header modification alternatives)
- Correlate potential requirements to specific costs per Framing alt
  - (E.g. ULP PDU not correlated to TCP MSS requires eddy buffer memory)
- Decision tree
- Recommendation

TCP Framing - Why?

- iSCSI must match or exceed FC
  - Work at 10 Gbps / 40 Gbps / beyond
  - Need high performance host architecture - comparable to FC
    - Very low overhead (host CPU only for I/O start and I/O complete)
    - Very low delay for comparable topologies
    - Minimized accesses to data in host memory
  - Must be cost competitive to FC (NIC CPU & memory / Host CPU & memory)
- Direct Data Placement is a solution
  - iSCSI data payloads in transport segments placed directly into SCSI request buffers
    - No (or little) transport reassembly buffering on NIC [after dropped segments]
  - Placement via iSCSI-specific mechanism or general RDMA protocol
- Need to locate NLP (iSCSI/RDMA) headers in transport segment stream to get direct placement info
    - If transport segment containing NLP header is dropped – need to find next NLP header in transport segment stream to resume DDP
    - Ideal is reliable, [ordered,] message-based transport protocol
- TCP requires Framing mechanism
    - To locate next NLP header (iSCSi/RDMA) in transport segment stream across segment drops (when dropped segment contains a NLP header)
    - Allows direct placement of TCP segments to resume starting with next NLP header following a dropped TCP segment containing an NLP header

Framing – Potential Requirements #1

1) Sustain link rate
    1) 1 Gbps / 10Gbps / 100 Gbps
2) Level of TCP changes acceptable?
    a) None
    b) Implementation changes
    c) Behavioral changes (window management, send segment sizes, etc)
    d) On-the-wire TCP header changes
3) SW-only and HW-accelerated solution interoperation (including 1GbE/10GbE aggregation)?
    a) No
    b) Yes & DDP works & modified TCP stack on SW-only side
    c) Yes & DDP works & standard TCP stack on SW-only side
    d) Yes & no DDP & standard TCP stack on SW-only side
4) Works across PMTU degrades? (upgrades?)
    a) No
    b) (Requires reconnect and recovery)
    c) Yes – but DDP stops
    d) Yes – DDP stops temporarily
    e) Yes – DDP works across PMTU changes
5) Works through TCP-level (resegmenting) middleboxes?
    a) No
    b) Yes – but DDP stops [?temporarily ?for duration of connection]
    c) Yes - DDP works with middlebox in path
6) Level of iSCSI or layering changes acceptable ? [or Support iSCSI PDU > MSS]
    a) None
    b) Chunking
    c) RDMA support (with chunking)
    d) Other ?

Framing – Potential Requirements #2

- TCP Friendly / good citizen / congestion avoidance
- TCP general-behaviors conformance
  - Zero window probes
  - Window management
- Framing selectable on per-connection basis
- Single operating mode (reduced complexity)
- Aesthetically acceptable

- Efficient on wire
- Efficient with host memory bandwidth
- Congestion avoidance - behavior / interoperation
- Performance not degraded compared to non-Framing TCP
- (portray as costs ?)

Framing – Potential Cost Vectors

Hardware Implementations

- Hardware Resources
  - *Buffer memory size*
    - ? How much buffer memory on NIC ASIC is reasonable?
    - Does this eliminate certain alternatives or put them into low-performance mode under some circumstances?
  - *Buffer memory bandwidth*
  - CPU
- Performance
  - *Host memory bandwidth*
  - On-wire efficiency
    - Short segments
    - Protocol/padding overhead
    - Multiple ULP in a segment
  - Packet loss probabilities
  - Congestion avoidance
- Reliability
  - Complexity (also perf & cost)
    - Eddy buffer mgmt
    - IP fragmentation

SW-Only Implementations

- Host Resources
  - Buffer memory size (if not zero copy)
- Performance
  - On-wire efficiency
  - *Processing overhead*
    - *Per byte processing*
    - *General processing*
  - Packet loss probabilities
  - Congestion avoidance
- Reliability
  - Complexity

Framing – Categories / Alternatives

- I**ntervalic**
  - "*Periodic Marker*" – Periodic marker at X byte stride - holds byte offset to start of next NLP header
  - "*Periodic NLP Alignment*" – Periodic alignment point at X byte stride – appropriate NLP PDU is aligned to that point
  - "*Fixed Length NLP*" – All NLP PDUs are padded to a fixed length – periodic NLP header points
- **TCP Payload Alignment**
  - "*Quantized TCP*" [or "*Segment-Aligned TCP*"] – Every TCP header followed by a NLP header – TCP window management is quantized
  - "*ULP Framing Header with Mod-4 Flag*" – Framing header follows TCP header – TCP segment length not mod-4 indicates partial NLP PDU in segment (or NLP hdr doesn't follow TCP hdr)
  - "*ULP Framing Header with Key/Len Match*" - "" - Framing header key mismatch indicates non-alignment)
  - "*ULP Framing Header with IP Frag*" - "" - IP fragmentation handles PMTU degrade
  - "*ULP Framing Header with Pipe Drain*" - "" - If sender can't send key-matching Framing hdr aligned to TCP hdr then sender waits for all data to be ACKed before sending
- **TCP Message Boundary Indication**
  - "*TCP Reserved Flag*" – TCP header reserved bit indicates NLP header follows TCP header
  - "*TCP Option Flag*" – TCP option indicates NLP header follows TCP header
  - "*TCP Option Offset*" – TCP option indicates byte offset in segment data to NLP header start
  - "*TCP-Message-Boundary-Other*" - (Urgent Pointer, PSH Bit, Urgent Pointer as extended flags, etc)
- **Data Stuffing**
  - "*COBS*" – Consistent Overhead Byte Stuffing
  - "*7B/8B*" – 7B/8B code expansion on byte stream – 8th bit indicates control info in other 7 bits
  - "*Stuffing-Other*" – (special character, magic number, etc)
- **Stream Scanning**
  - "*Running NLP Check*" – keep running check for NLP frame structure and consistency

<cleaner slide or multiple slides – perhaps a table>

Framing – Taxonomy #1

- Intervalic
- Data Stuffing
- TCP Payload Alignment
  - PMTU degrade via IP Frag
  - PMTU degrade via flag
    - Explicit flag
    - Encoded flag
- TCP Message Boundary Flag

Framing – Taxonomy #2

- Above TCP
  - Independent of seq #
    - Data Stuffing
  - Tied to seq #
    - Fixed interval reference pointer to ULP PDU start
    - PDU aligned/padded to fixed interval
- At TCP
  - ULP > MSS (ULP Span Segments)
    - Explicit Message Boundary Flag
    - Implicit Message Boundary Flag
    - Header reference pointer to ULP PDU start
  - ULP <= MSS (ULP Align to Segment)
    - PMTU Degrade via IP Frag
    - PMTU Degrade via Flag
      - Explicit Message Boundary Flag
      - Implicit Message Boundary Flag

Category Comparison Table

- Level of TCP changes / asthetics
- Level of iSCSI changes
- SW-only implementation over stock TCP
  - (interoperate with HW implementation)
- SW-only performance considerations
- NIC buffer memory requirements
- HW implementation complexity
- Handle PMTU degrade
- Work through resegmenting middleboxes
- ?

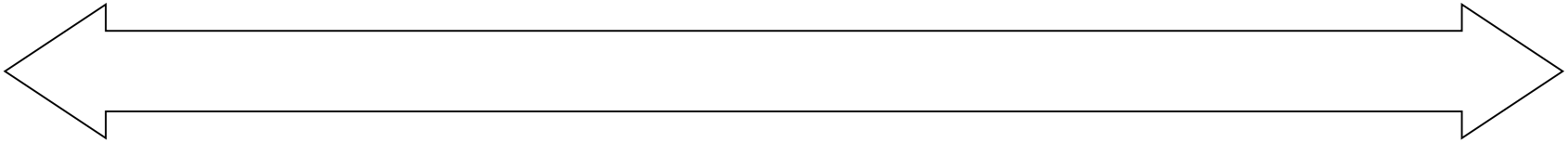
Framing – Category Comparison Table

TBD

Framing - Comparisons Summary

- Markers
  - ULP > MSS
  - Stock TCP sender
  - Eddy buffer
  - Receiver complexity
- Periodic Alignment
  - ULP > MSS
  - Stock TCP Sender
  - Eddy buffer
  - Receiver complexity
- Byte Stuffing
  - ULP > MSS
  - Stock TCP Sender
  - Eddy buffer
  - Receiver complexity
  - SW overhead
- TCP Alignment
  - ULP <= MSS
  - Chunking required
  - Modified TCP sender
  - No eddy buffer (?PMTU change)
  - Receiver simplicity

Framing – Alternatives Continuum End-points



ULP chunking
ULP never span TCP seg
PMTU via IP Frag

No ULP Chunks
Data Stuffing
PMTU via TCP reseg

Requirements vs Costs & Alts Correlation

TBD

Framing Alternatives Descriptions

Framing – Intervalic Category

- Periodic Marker
- Periodic NLP Alignment
- Fixed Length NLP PDUs

Periodic Marker

- Markers occurs at fixed intervals in byte stream (fixed stride)
- Marker is 8B field
  - Two 32-bit offset fields
  - Offset field indicates byte offset in TCP stream from current point to start of next NLP header
  - Offset is counted from marker end to beginning of next NLP header
- If TCP segment holding NLP header is dropped
  - Subsequent segments are placed into eddy-buffer
  - Until next marker segment is received and then next NLP header segment is received
- If TCP segment holding marker is dropped
  - DDP continues unless that segment held a NLP header
- Can have multiple outstanding eddy-buffers in progress
- NLP PDU vs marker stride:
  - Large NLP PDUs may have multiple markers all pointing to same NLP header
  - Small NLP PDUs may have multiple NLP headers before next marker
- Use of markers is negotiable
  - Initiator & Target indicate readiness to use markers during login (per connection)

Periodic NLP Alignment

- Alignment points occur at fixed intervals in byte stream (fixed stride)
- If NLP header would span an alignment point
  - Then zeros are inserted in byte stream up to alignment point
  - NLP header is sent out starting at alignment point
- If TCP segment holding NLP header is dropped
  - Subsequent segments are placed into eddy-buffer
  - Until next alignment point segment is received and NLP header is obtained
- If TCP segment holding alignment point is dropped
  - Same as NLP header segment drop
- Can have multiple outstanding eddy-buffers in progress

Fixed Length NLP

- NLP PDUs occur at fixed intervals in byte stream (fixed stride)
  - All NLP PDUs are padded out to be a fixed length
- If TCP segment holding NLP header is dropped
  - Then subsequent segments are placed into an eddy-buffer
  - Until next NLP PDU segment is received
- Can have multiple outstanding eddy-buffers in progress

Framing – TCP Payload Alignment Category

- Quantized TCP
- ULP Framing Header with Mod-4 Flag
- ULP Framing Header with IP Frag
- ULP Framing Header with Key/Len Match
- ULP Framing Header with Pipe Drain

Quantized TCP

- ULP PDU is chunked into self-describing NLP PDUs that fit into TCP segments
- NLP PDU (chunk header) always follows TCP header
  - Every TCP segment is self-describing (relative to a NLP PDU chunk)
  - Multiple whole chunks can be packed into a TCP segment
- TCP sender/receiver window management is modified:
  - Sender
    - Only send whole (or multiple whole) NLP PDUs in a segment
    - Don't send if receive window doesn't allow whole chunk to be sent
    - Send zero-window probes with whole-chunk segment
    - ? Is it necessary to probe a receive window smaller than chunk to send?
  - Receiver
    - Don't accept any bytes in a segment unless receive window allows whole segment to be accepted (including zero window probes)
    - ? What about send-ACK-every-2-MSS-received policy and self-clocking behavior?
- If TCP segment is dropped
  - Then subsequent TCP segments can still be placed since they are self describing
  - No eddy-buffer is required
- PMTU degrade can be handled as follows:
  - xxx
- PMTU upgrade can be handled as follows:
  - Xxx
- Middlebox impact
  - Xxx
- << Slide showing std TCP behaviors and differences in Q-TCP (793, 1122, congest RFCs, HS RFCs, etc) >>

TCP ULP Framing - General

- ULP PDU is (may be?) chunked into self-describing NLP PDUs
- NLP PDU is wrapped with 8B Framing header
  - 6B key (randomly selected) & 2B length
- Framing header always follows TCP header
  - Every TCP segment is self-describing (relative to a NLP PDU chunk)
  - ? Multiple whole chunks can be packed into a TCP segment

- Various schemes for handling when sender can't send full segment (or alignment has been broken):
  - Resegmenting middlebox breaking TCP/NLP header alignment
  - PMTU degrade – retransmission of sent segments
  - Sending a zero window probe
  - Sending into window that is smaller than full PDU

- Zero-window probe handling
  - Sender can send full PDU in zero-window probe
  - Is receiver's behavior specialized?
    - Does it ignore data in zero-window probe and sender retransmits when window can take full segment?
    - Does it only accept data in zero-window probe if it can accept entire segment?
  - What if a receiver accepts (ACKs) only partial data from zero-window probe?
    - Does this force sender to resend second half of NLP PDU in a separate TCP segment?

TCP ULP Framing – Variations

Methods for handling non-alignment of Framing Header with TCP header:

- Sender can't place full NLP PDU into TCP segment – and subsequent segments wouldn't have Framing Header:
  - Resending of data after dynamic PMTU reduction
  - Sending of data in a zero window probe
  - Sending into a window that is smaller that full segment (or NLP chunk + Framing header)
- When resegmenting middlebox breaks alignment of Framing Header to TCP Header

<describe re: Sender Actions / Receiver Actions>

<separate slide per variation>
<comparison between variations? - in XLS as separate columns also?>

A. TCP segment length Mod-4 Flag
   - TCP segment length not Mod-4 indicates that TCP segment doesn't contain whole NLP PDU
   - (could also indicate that NLP PDU doesn't immediately follow TCP header)

B. Key/Length Mismatch
   - Rely on key/length mismatch to indicate that first 8B in TCP segment payload are not a Framing Header

C. IP Fragmentation
   - Rely on IP fragmentation to delivery TCP segments until ULP starts chunking at smaller MTU

D. Sender waits for all data to be ACKed
   - Sender waits for all data to be ACKed before sending segment that doesn't contain Framing Header – Receiver is assured to have received small segment containing Framing Header and know that more data associated with NLP PDU is forthcoming

Framing – TCP Message Start Indication Cat.

- TCP Reserved Flag
- TCP Option Flag
- TCP Option Offset
- TCP Other

TCP Message Start Indication

- General
  - NLP PDU can span multiple TCP segments
  - Multiple NLP PDUs can be packed into a TCP segment
  - (Or ULP PDU can be chunked into self-describing NLP PDUs)

- TCP Reserved Flag
  - Occassionally NLP PDU header is aligned to follow TCP header
    - TCP Reserved Flag bit is allocated to indicate that this TCP segment has NLP header following after TCP header
- TCP Option Flag
  - Occassionally NLP PDU header is aligned to follow TCP header
    - TCP Option used as Flag bit to indicate that this TCP segment has NLP header following after TCP header
- TCP Option Offset
  - TCP Option 2B value hold byte offset to start of NLP header in that TCP segment
- TCP-Message-Start-Indication-Other
  - Urgent Pointer, PSH Bit, Reuse Urgent Pointer field as extended flags, others?

Framing – Data Stuffing Category

- COBS
- 7B/8B Code Expansion
- Data-Stuffing-Other (SLIP, PPP, simple special char, etc)

Data Stuffing

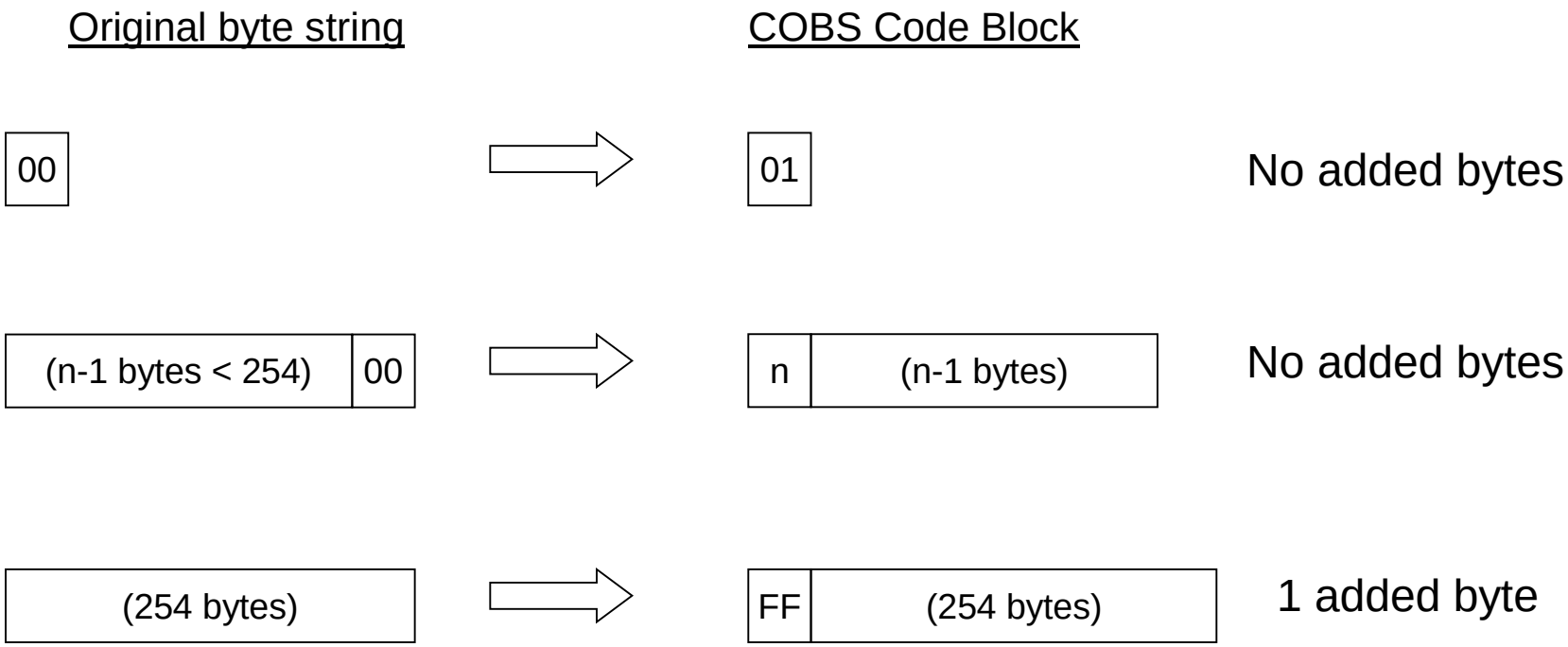
- Data Stuffing – General
  - Use special bit-sequence or byte value to indicate packet boundaries (frame marker)
  - Uses encoding to eliminate this value from regular data stream

- COBS
  - Byte-stuffing mechanism
  - Transforms byte stream to eliminate a byte value
  - Uses code blocks
- 7B/8B Encoding
  - ULP PDU treated as bit stream
  - Every 7bits of ULP are encoded as 8bits of TCP payload data (with high-bit = 0)
  - High-bit = 1 indicates that lower bits hold control value
  - Use control value for packet boundary indication
- Data-Stuffing-Other
  - Special bit string
  - Special byte string
  - ”Magic Number”

COBS

- Performs reversible transformation on packet to eliminate a single byte value from it
- Byte value then used as unambiguous framing marker
- Overhead is maximum of 1 byte per every 254 bytes
- Operation (using 0x00 byte as framing value)
  - Logically append 0x00 to packet
  - Divide packet into 0x00 terminated chunks
    - Every chunk has one 0x00 byte / 0x00 is last byte in chunk / chunk length can be one byte
  - Encode each 0x00-terminated chunk as one or more variable-length COBS code blocks / chunks longer that 254 bytes use multiple code blocks
  - COBS code block = single code byte followed by zero or more data bytes
  - Code byte indicates number of data bytes and presence of implicit zero byte
    - 0x01= no data bytes + 0x00 byte
    - n = (n-1) data bytes + 0x00 byte
    - 0xFF = 254 data bytes (not followed by 0x00 byte)
- COBS/ZPE (Zero Pair Encoding) variation frees additional byte values
  - Some code byte values used to indicate data block terminated with pair of 0x00
    - 0x01-0xDF = (n-1) data bytes + 0x00 byte [Example frees 31 byte values]
    - 0xE0 = 223 data bytes (not followed by 0x00 byte)
    - 0xE1-0xFF = (n-225) data bytes + 0x00 byte + 0x00 byte

COBS - Transformation



Framing – Stream Scanning Category

- Running NLP Check
- ?Nuke this category altogether?

iSCSI Layering Evolution Discussion

iSCSI Layering Evolution - Questions

- RDMA now or later?
- Framing optional or mandatory?
- RDMA optional or mandatory?
- Chunking layer (or within RDMA)?
- Common framing mechanism or separate framing for iSCSI/TCP and iSCSI/RDMA/TCP?
- 1 Gbps and 10 Gbps interoperation mode?

Slide Dumpster

Other Framing Examples (vs Encapsulation)

- Transport protocols
  - SCTP (message-based)
  - xxx
- Byte stuffing algorithms
  - SLIP [RFC1055]
  - PPP [RFC1662]
  - AX.25 (Amateur Packet Radio) [ARRL84]
- Bit stuffing
  - HDLC

Related Work

- iSCSI
- iSCSI markers
- ULP TCP framing
- SCSI mappings (SRP, SBP-3, SST, SSA)
- FCIP/iFCP Common Encaps
- FCIP
- IFCP
- TAF
- SCTP
- COBS
- RTP
- VI
- VI/TCP
- InfiniBand
- TCP-RTM
- RTMP
- DAFS
- ALF
- HTTP / CIFS / NFS – encaps
- TCP - PMTU / High-speed / TCP friendly / partial order service
- xxxx

SCSI Transport Protocols

- Serial Storage Architecture Transport Layer 1 (SSA-TL1)
- Serial Storage Architecture Transport Layer 2 (SSA-TL2)
- SCSI-3 Fibre Channel Protocol (FCP)
- SCSI-3 Fibre Channel Protocol – 2 (FCP-2)
- Serial Bus Protocol – 2 (SBP-3) [1394]
- Serial Storage Architecture SCSI-2 Protocol (SSA-S2P)
- Serial Storage Architecture SCSI-3 Protocol (SSA-S3P)
- SCSI on Scheduled Transfer (SST)
- SCSI RDMA Protocol (SRP)