Minutes for 11/29/00 discussion of iSCSI Framing Issues

Attending:

Agilent
Matt Wakeley
Joe Steinmetz

Cisco
Dave Peterson
Costa Sapuntzakis

Giganet
Dave Wells
Alex Nicolson

HP
Randy Haagens
Marjorie Krueger
Pierre Labat

IBM
John Hufferd
Prasenjit Sarkar

Microsoft
Bernard Aboba

Quantum
Luciano Dalle Ore

Objectives:

It is a crucial business requirement for iSCSI to be cost and performance competitive with Fibre Channel. All technical discussion is held with this key requirement in mind.

Randy presented a model for iSCSI processing which illustrates that the challenge to iSCSI is one of multiplexing data from multiple applications onto a sending TCP stream, and demultiplexing iSCSI application data from a receiving TCP stream. We seek a framework to accomplish this with minimum data copies.

Direct Data Placement

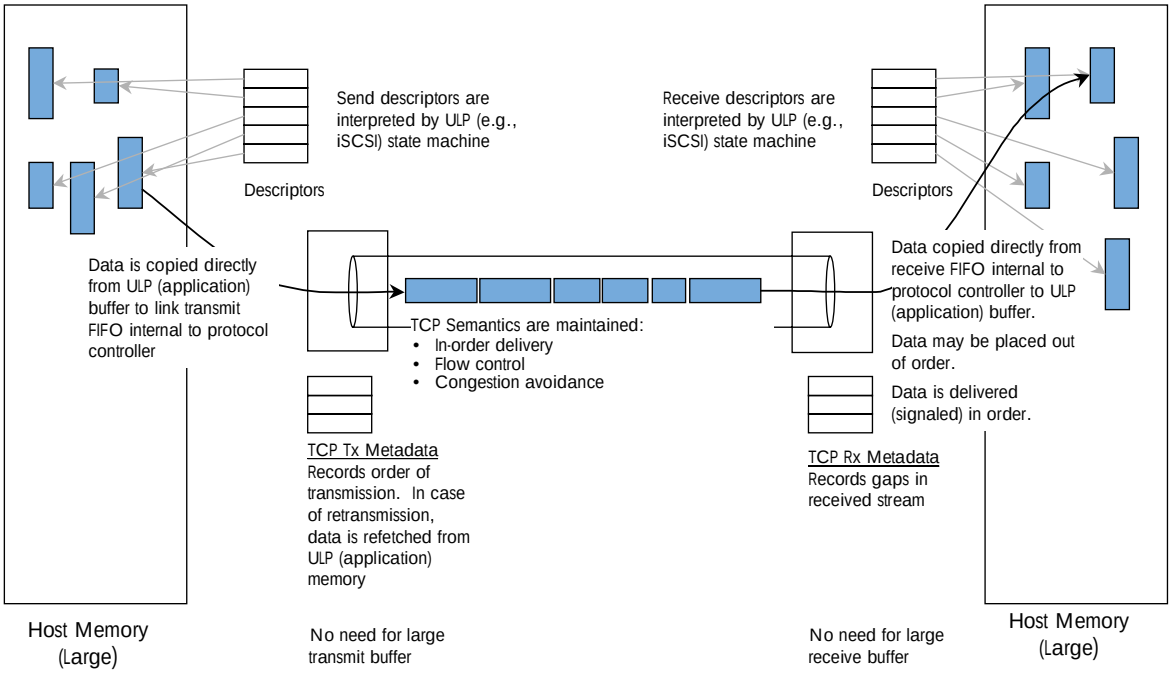


Figure 1 iSCSI's Challenge

In order to structure the day's discussion, Randy suggested the following flow chart:

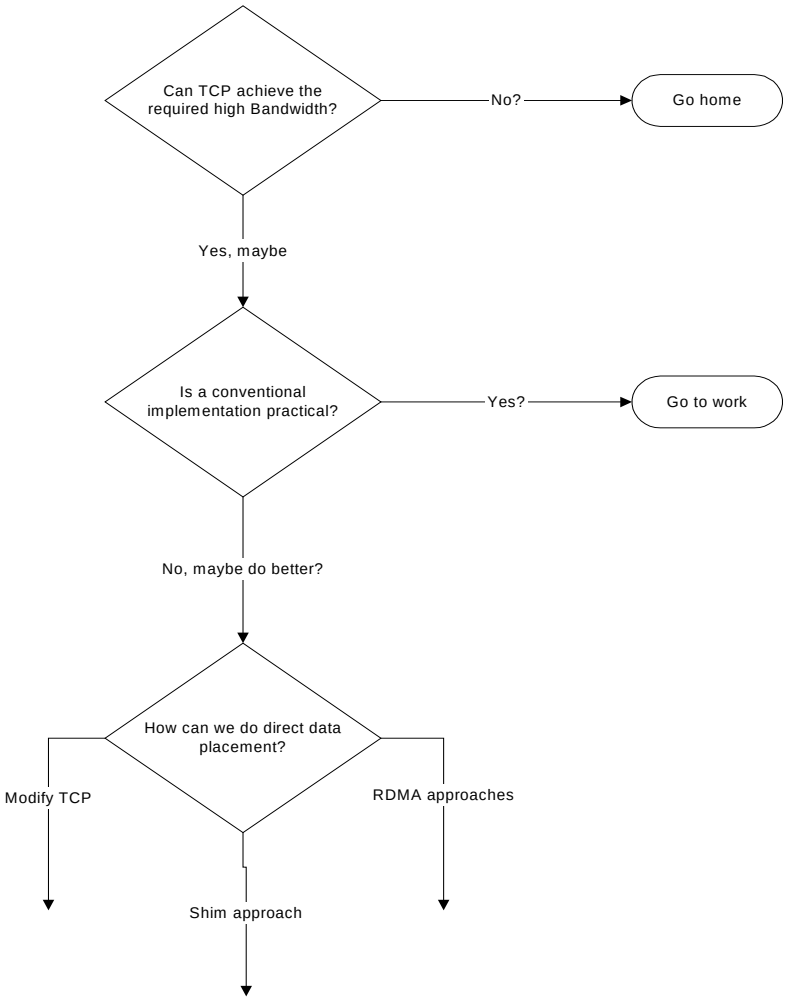


Figure 2 Structure of Discussion

Can TCP achieve high delay BW?

The first question is whether TCP can achieve the desired throughput on Gigabit media. If not, then TCP is the wrong transport for SCSI over the internet. The business case for iSCSI starts with 1 GbE, but to be interesting long term it is necessary to be able to run at near wire speed on 10 GbE, which is expected to be standardized by 2Q 2002.

There was a brief discussion of the impact of multiple TCP data connections for an iSCSI flow. iSCSI may use multiple connections as a method to achieve a sending rate equal to the full link bandwidth or greater. Multiple TCP connections also result in more rapid aggregate bandwidth usage ramp during slow start than would occur with a single connection.

The group speculated that the throughput profile over time of multiple TCP data connections through the same NIC may synchronize, with peak sending rates occurring at the same time. This would magnify the oscillations due to congestion. As a result, the available bandwidth would not be fully used.

It was noted that when window size is large, packet loss penalty is proportionately large. Assuming iSCSI windows will be large to try to take full advantage of bandwidth, there is a large penalty in terms of speed whenever there is a packet loss. It can take many minutes for the throughput rate to ramp back up after a loss. ECN and/or RED implementation in routers in conjunction with ECN TCP participation may alleviate this problem, as can using a single TCP connection with command pipelining.

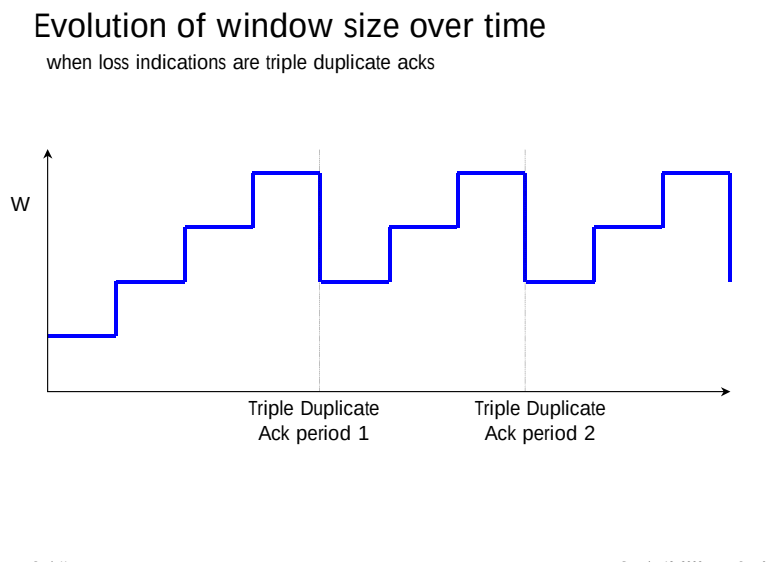


Figure 3 Window Size Profile

Vern Paxson and others have referred to formulas for TCP performance. These formulas have typically been derived based on a set of assumptions and covering a set of implemented TCP mechanisms, so it is important to understand what the formulas cover when applying them. The paper Vern referred to focused on performance within the congestion avoidance regime where lost packets are detected using triple duplicate ACKs. A more recent paper taking into account timeouts as well as triple duplicate ACKs is available in:

Padhye, J., Firoui, V., Towsley, D.F., and Kurose, J. F., "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation", IEEE/ACM Transactions on Networking, April 2000, Volume 8, Number 2, pp. 133.

The simplified formula derived in this paper is as follows:

Transfer rate (bps) =

$$MSS * \min\left\{ \frac{W_m}{RTT}, \frac{1}{RTT * \sqrt{2bp/3} + T_o * \min\left(1, 3\sqrt{3bp/8}\right) * p\left(1 + 32p^2\right)} \right\}$$

MSS = maximum segment size (bits)
RTT = round trip time (sec)
p = probability of packet loss
b = packets acknowledged by received ACK (typically 2)
T_o = timeout time
W_m = maximum receive window advertised by receiver

We then went through some calculations based on this formula, modified by some assumptions. The first assumption is that the receive window will be large enough that throughput will be governed by loss probability. The second assumption is that the Timeout portion of the formula will be negligible compared with the RTT portion. This is reasonable at low packet loss rates because p T_0 will typically be a lot smaller than RTT. It's also reasonable at higher p, because in those circumstances (WAN links) RTT will typically be a lot larger, and if RTT ≈ pT_0 then the packet loss is probably so high that iSCSI performance will be terrible anyway.

The equation simplifies to:

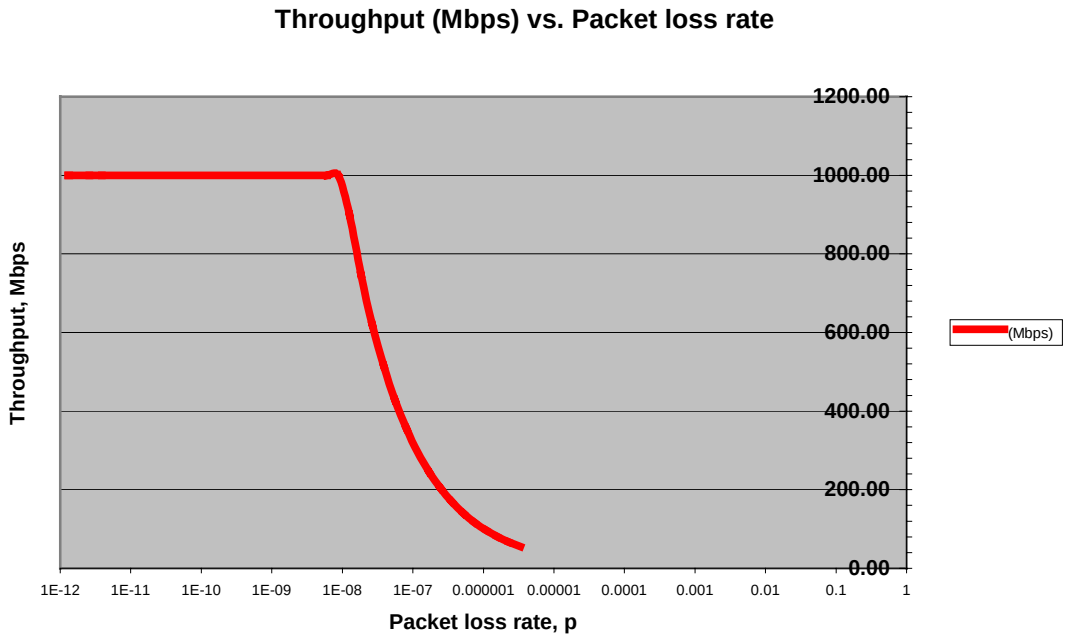
Transfer rate =

$$\frac{MSS\sqrt{\frac{3}{2}}}{RTT\sqrt{p}}$$

WAN scenario
Transfer Rate = 1 Gbps
MSS = 1460 octets ≈ 10^4 bits
A bit loss probability of 10^{-12} yields a packet loss rate of 10^{-8}

Plugging these values into the equation, we find a RTT of 100 ms which yields a round trip distance of 20000 km. This is a network diameter of 10000 km.

The graph of this formula for an RTT of 100ms is:



This formula implies that TCP can achieve the required performance (1 Gbps), assuming that the advertised window size is large enough and the packet loss rate is low enough ($<10^{-8}$). More importantly, this formula yields interestingly large network diameters. The formula is for performance dominated by the congestion avoidance phase. Even with fast retransmit, it will take a substantial amount of time for transfer rate to recover after a loss.

There was a discussion of iSCSI scenarios. These include LAN, MAN and WAN scenarios with 1 or 10 GbE. Current MAN technology is capable of 70 – 100 kM operation without EFDA amplification. It was noted that WAN scenarios are also likely, for cross-country backups, for example. Note that current packet loss rates over WAN links are typically considerably higher than on LAN links.

Is a conventional implementation practical?

The next discussion point is whether a "conventional implementation" of iSCSI over TCP can support the required transfer rates with reasonable hardware cost. A conventional TCP/IP implementation was defined as one which provides a conventional sockets interface as well as TPI/TDI level interfaces, and also implements zero copy on send, but not necessarily zero copy on receive. This represents more or less the state of the art as it exists today (BSD 4.4, Linux 2.3, Windows 2000).

Note that the group is assuming that the iSCSI/TCP/IP processing is occuring on some sort of hardware solution (NIC), not in a standard host software stack implementation. It is assumed that dedicated hardware is required to support the transfer rates.

At 1 Gbps, a NIC implementation with 16 MB of static RAM could potentially handle the memory buffering requirements for a WAN scenario (LAN scenario RTT is much less so WAN scenarios drive the memory requirements). At 1 Gbps data rate = 125 MB/sec x 100 ms RTT = 12.5 MB per connection of data in the pipe. The means 12.5 MB of memory is required for buffering of "anonymous data" (out of order packets) (assuming the window size is "wide open"). In terms of performance, a 32 MB static RAM running at 133 Mhz, and 32-bits wide can handle the required memory bandwidth (32 bits/clock cycle * 133 million clock cycles/second = 4256 Mbps > 2 trips * 1 Gbps.) In terms of cost, this part will go for

only $25 or so in volume. That will only modestly increase the cost of an iSCSI implementation. Thus, at 1 Gbps, putting memory on the NIC is a workable solution.

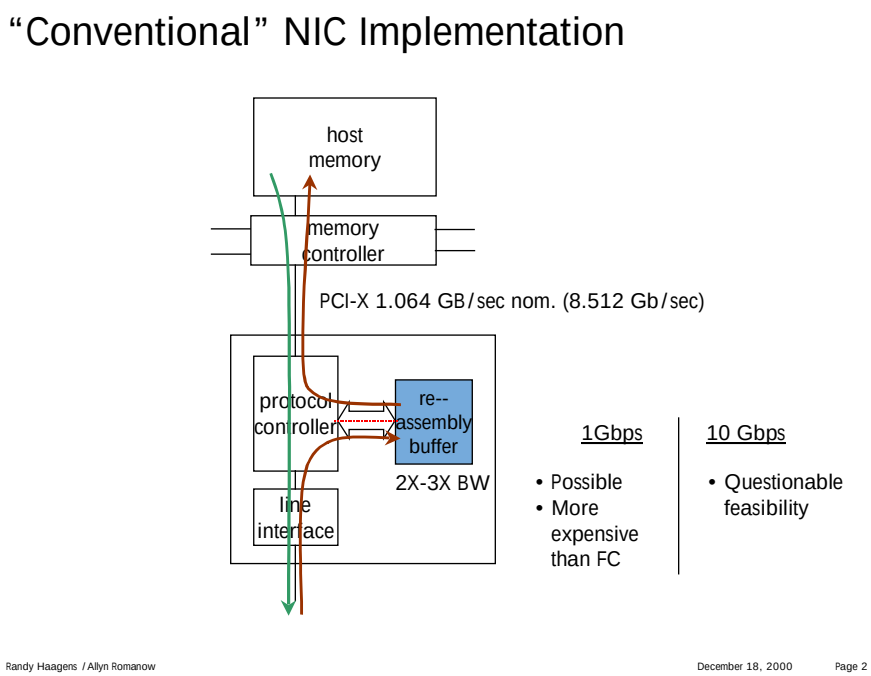


Figure 4 Flow of Data thru "conventional" implementation

The group then discussed whether this approach would scale to 10Gbps. Here you might need 125 MB/connection of static RAM to handle the required buffering at 100 ms RTT. The quantity of SRAM, while substantial is not necessarily prohibitive, given that we won't have 10 GbE until 2Q 2002 anyway, and 10 Gbps iSCSI might not be required until 2003-2004, so that you have several years of cost declines to take into account.

The question is whether you can deliver 20 Gbps memory bandwidth at a reasonable price. This is the more difficult problem. At 10 Gbps, we assume that if you start to hold data in a buffer, waiting for a retransmitted segment, when you get that segment, the memory will have to be emptied while still receiving incoming data packets. So data is flowing in at 10 Gbps, out at 10 Gbps, and the buffer is being emptied at 10 Gbps which necessitates 20 Gbps of aggregate bandwidth.

PCI bus or its successors (PCI-X, PCI-4X) won't cut it anymore; you'd need Infiniband. The bottom line is that you would need wide data path memory (128 bits) running at high clock rates (266 Mhz) to have a shot at this (128 bits/clock * 266 million clocks/second = 34048 > 2 trips * 10 Gbps). This part (if available) is likely to be expensive. So the answer is that the conventional approach may be theoretically feasible in the 2004 time frame, but would probably need to be restricted to high end markets. Shipping mass market implementations (desktop cards, for example) will require addressing framing/DMA issues.

Even if 10 Gbps feels possible, what about 100 Gbps? IEEE has been churning the next order of magnitude in LAN technology every 3 years or so. By ~2006, get ready for 100 GbE – and since memory technology follows Moore's law (18 month doubling) versus Gilder's law (9 month doubling), memory speeds fall further and further behind wire speeds. We'll eventually have to bite the bullet and solve the problem by accelerating the transport data movement without buffering the whole window.

In the TCP model, there is one application per TCP stream. A possible "0 copy" TCP implementation would have the application supplying the buffers and the application is delivered a scatter gather list from TCP. For the iSCSI application, we explored possible scenarios for movement of data from the wire to the application buffers. Figure 5 Data Movement Scenario shows a possible design where the iSCSI buffer receives the header only, then tells the TCP NIC where to put the rest of the data. With this model, only 5% of this memory must be on a NIC solution, the other 95% can be on the host. This model leads to 6 MB per connection for buffer memory, which seems workable. However, this number assumes a drop rate of 10**-12; if you change that to 10**-10 you change the memory requirements by a factor of 100.

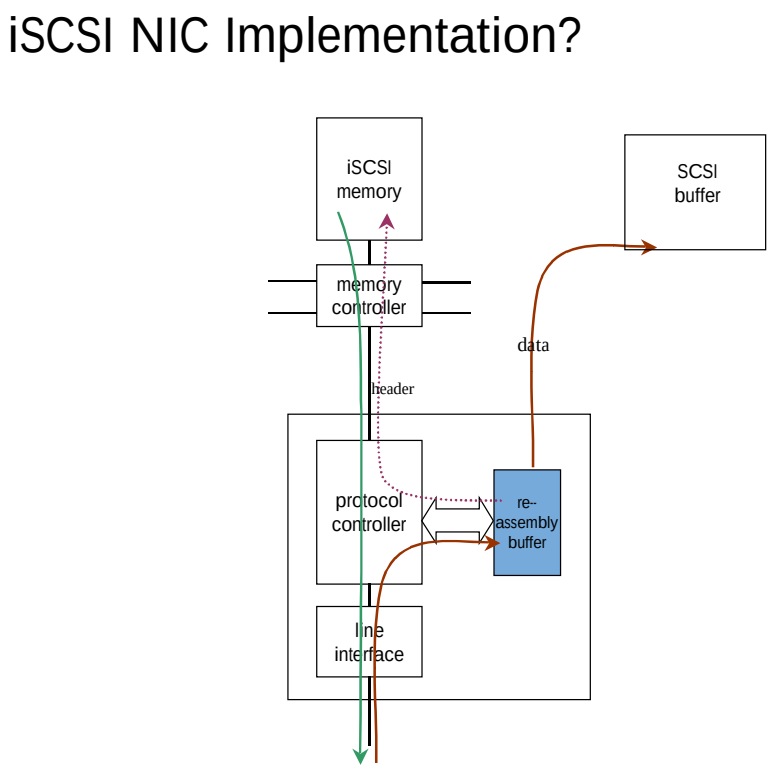


Figure 5 Data Movement Scenario

How can we accomplish direct data placement?

So we've tenatively answered "yes" to the question of whether TCP can acheive high delay bandwidth, but we have doubts regarding whether conventional TCP implementations can handle emerging line speeds. This brings us to the question of what improvements would accelerate data placement (minimize buffering requirements) in iSCSI over TCP?

Randy discussed his philosophy of optional capabilities. Making a capability optional is a worse alternative than either making it mandatory or omitting it. If you are trying to interoperate well with others, then you implement the optional capabilities (be liberal in what you accept). However, you can't guarantee that the work will generate any benefits, because the capabilities are optional, so that another vendor might not implement them. The situation is particularly onerous with respect to acceleration/direct placement alternatives. If you can't assume a mandatory to implement framing/DMA approach, then you may have to add memory to your NIC implementation – thus preventing the very cost savings that framing/DMA is attempting to provide. So Randy's advice is choose one and only one framing/DMA solution, and make it mandatory to support it in both accelerated and non-accelerated solutions. That way you'll be able to have backward compatibility between 1 Gbps and 10 Gbps solutions. In talking between an accelerated and non-accelerated solution, you might be slower, but at least you'll always be able to communicate. This in turn implies that the framing/DMA solution needs to be negotiable.

We then categorized the the solutions in two groups: In-stream (requiring no modifications to TCP) and Out-of-Stream (modifications to TCP). **Table 1 Possible Data Solutions** outlines the solutions that have been proposed under these catagories:

In-Stream

- special characters
- fixed length PDUs
- periodic marker
- "magic number"

Out-of Stream

- PSH bit
- URG pointer
- Align iSCSI header with TCP header
- TCP option
    - 1 bit in hdr
    - 16-32 bits in options

Table 1 Possible Data Solutions

Special characters = mark the end of a frame with an escape sequence. Problems: a lot of work for software implementations, requires scan of all data to replace occurrances of the escape sequence in user data.

Fixed length PDUs=Make PDU length = N*MSS so each TCP segment. Problems: wastes bandwidth, fails in packet loss situation if lost packet contains iSCSI header, how does iSCSI know when TCP MSS changes?

Periodic Marker = place an 8 byte marker every N bytes that points to the next PDU header in the stream (Figure 6 Periodic Marker)

"magic number" = ?? Works in conjunctions with aligning an iSCSI header with the TCP header. You can then protect the iSCSI header with a CRC to verify that you in fact have an iSCSI header. Problems: requires scanning the data stream, may not be unique

PSH bit = set PSH bit at the end of every iSCSI data transfer. Problems: TCP behavior for PSH bit is not mandated, some implementations ignore PSH bit, PSH bit is more for sender behavior, doesn't control receiver behavior predictably.

URG ptr = use URG pointer to point to the end of a frame. We've discussed this one to death on the reflector, hopefully all are aware of the problems with this one.

TCP option, 1 bit in header = assign an option bit to be set to mean "this segment contains at least one ULP PDU header"

TCP option, 16-32 bits in options data = set bit, pointer = byte where header starts. Problems: potentially conflicts with SACK use of options data area.

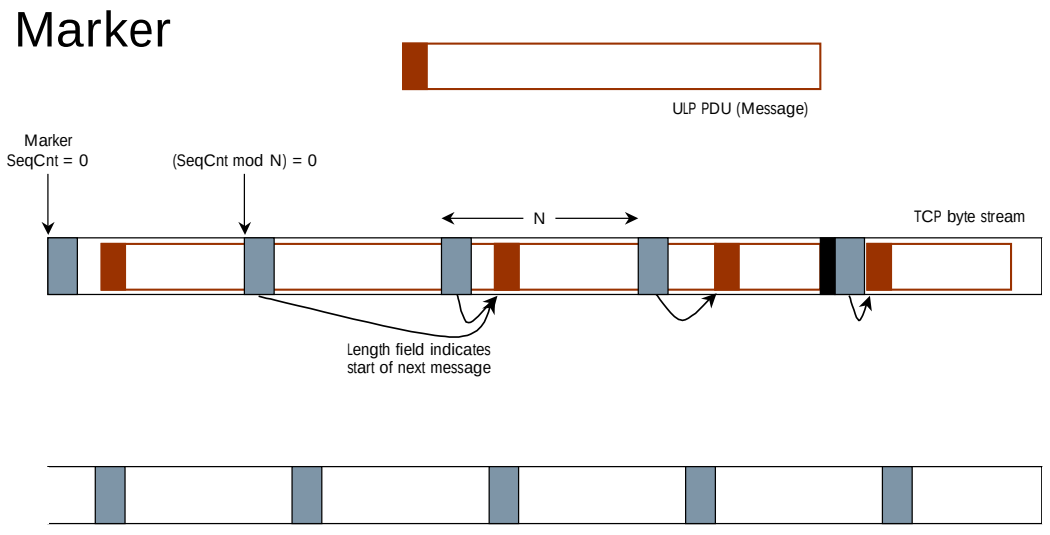


Figure 6 Periodic Marker

In general, the out-of-stream approaches have the following advantages:

- cleaner and don't require any removal of bytes from the stream.
- can appear in every packet

The in-stream approaches have the following advantage:

- can be implemented as a "shim layer" between TCP and iSCSI
  - therefore no mods to TCP API or stack.

It was pointed out that it takes the same number of instructions to negotiate options as it does to implement some of the in-stream approaches. However, modifications to TCP will benefit many other things that run over TCP (RPC, SNMP, etc).

Fibre Channel discussion

There was a discussion of how Fibre channel handles the direct placement issue. Fibre Channel can handle direct placement with an ASIC and no external memory because all the data necessary for direct placement is in the frame header. It was also noted that FC does not handle the issue of packet loss well. It can take 10 minutes to recover from a loss. However, since we are talking about a Fibre interface, packet loss is very low so this is not that frequent. So with respect to loss resiliency, whatever we do with iSCSI over TCP, it will be better than FC.

Conclusion:

At speeds > 1 Gbps, it's doubtful whether a conventionable implementation of TCP will be able to drive the bandwidth given memory bandwidth limitations. Therefore it is necessary to devise a way to allow direct copy of data from upper layer protocol PDU's contained within the TCP stream even in when packets are dropped.

There are a number of ways to do this, but the two most promising ideas from the two catagories are the "periodic marker" treatment, and the bit signalling ULP header presence in the TCP header. Matt Wakely, Agilent Techologies is preparing a draft of the "periodic marker" idea. Costa Sapuntzakis is preparing a draft of the TCP header bit idea.