International Journal of Industrial Electronics and Electrical Engineering, ISSN(p): 2347-6982, ISSN(e): 2349-204X
Volume-6, Issue-7, Jul.-2018, http://ijieee.org.in

METAL SHARED MEMORY INTERCONNECTS – MSMI

(High Frequency Dual System Communications Protocol on Massively Dense FPGA Cores)

[1]SOLOMON HARSHA, [2]KHALDOUN KHASHANAH

[1]Doctoral Candidate in School of Systems & Enterprises Stevens Institute of Technology
[2]School of Systems and Enterprises Stevens Institute of Technology, Hoboken, NJ 07030

Abstract- The TCP/IP protocol was deployed in 1983 and has not had an upgrade to keep pace with the increase in packet volume and trunk speed that today's applications require. In this paper authors propose a new protocol – Metal Shared Memory Interconnects (MSMI) algorithm and its design, which address all the above TCP problems and leverages the new multi-core processor technologies and also new protocol architecture on dual system to make networks intelligent. Such novel protocol makes use of the many cores architecture on FPGA's and metal algorithms to create many deeply dense pipelines working on hundreds of cores interconnected to their memory that are built on FPGA's to transfer payloads of the speeds from 10Gbps, 120Gbps and above. For e.g. this new protocol not only reaches transfer rates 9.5Gbps on 10Gbps but also makes the networks intelligent and can achieve 95-99% bandwidth utilization. All TCP flows within a trunk circuit accelerated to the maximum rates. As circuits are upgraded from existing 10Gbps to 100Gbps and above they work together to permit the utilization increase, lost packet elimination, and TCP speed gains across a whole network. The gateways built on this new protocol can be added incrementally to existing networks and the proposed protocol is compatible and transparent to existing traffic and protocols. First time in the history of communications in the lab and also on the live circuit of 2000miles we have successfully tested on 10Gbps line and achieved stateful transfer rates reaching 3.3Gbps with heavy losses induced both input side and output side of the trunk. The metal algorithms and the software is developed by creating reconfigurable FPGA cores (64) on Xilinx FPGA vertex 7, by creating 16 pipelines 8 in each direction.

Index Terms- MSMI, FPGA, Multi-Core, Low Latency, HPEC, HPC

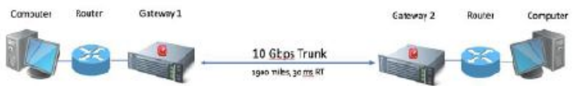
I. INTRODUCTION

MOOR's Law describes the performance increase for semiconductors and was estimated in 1971 as an 18-month doubling. If the last 30 years' experience were used to correct this rate it is possible that the 18 months would be slightly different, like 21 months, but clearly this trend has continued with little change. Computer performance is strongly influenced by the semiconductor trend but need not be identical. The very accurate results that obtained for computer performance is very similar to Moore's Law but not identical as might be expected. The trend for trunk speed was far less likely to be identical to semiconductor performance but somehow is very close to the computer performance trend. These trends and the communication cost trends are all important to understanding the trends of Internet cost/bit and then via elasticity, the current Internet traffic trend. The traffic trend then predicts the router/switch speed required.

<u>Trend Doubling Period Name</u>

- Semiconductor performance18 months Moore's Law
- Computer performance/dollar 21 months Roberts Law
- Communications- bits/dollar before 1995 79 months
- Communications- bits/dollar with DWDM 12 months
- Maximum Internet Trunk Speed in service 22 months
- Internet Traffic Growth 1969-1982 21 months
- Internet Traffic Growth 1983-1997 9 months
- Internet Traffic Growth 1997-2008 6 months Internet Router/Switch Max Speed until 1997 22 months
- Internet Router/Switch Max Speed after 1997 6 months.

From the above historical information, we can conclude that there is a great relation between hardware technology (Semiconductor Technology) and Communications traffic growth. In this paper we are proposing a new generation protocol which leverages the current semiconductor technology to build new dual system protocol algorithm. This dual system MSMI protocol will not demand to replace existing TCP/IP but increases the traffic speed on any trunk on which legacy protocols run such as TCP/IP etc. The two systems of the protocol are placed between two ends of any trunk as shown below:



A. New High Frequency Dual System Protocol - MSMI

In this paper the proposed new dual system protocol's algorithm and technology's is based on new generation of HPEC (High Performance Embedded Computing). The technology built on dual system architecture produce scale-invariant gateway systems connected across geographies can transfer data with extreme low latency and the new protocol developed is superior to TCP/IP. The resulting dual system protocol will truly perform parallel processing with highest speed attainable and minimum footprint.

MSMI - dual system protocol's technology is fully based on embedded computing on metal and does not have the burden of OS or the legacy heavy weight protocol stack.

Parallel processing in a cluster requires explicit message passing programming whereas shared memory systems can utilize compilers and other tools that are developed for multi-core processors. Parallel programming is a complex task and programs written for message passing normally contain 50% - 100% more code than programs written for shared memory processing. Since all programs contain errors, the probability of errors in message passing programs is 50% - 100% higher than for shared memory programs. A significant amount of software development time is consumed by debugging errors further increasing the time to complete development of an application. And they all require careful capacity planning to optimize cost and if compute requirements increase, it may be necessary to replace the entire server with a bigger and much more expensive one since the price increase is far from linear. For the most expensive servers, the price per CPU core is the range of USD 50,000 – 60,000.

Whereas with proposed new technology on a single FPGA chip we have 700-1500 cores and any sequential program can be parallelized programmed automatically on them. This drives the cost per CPU core down to 1/200th of any existing technology and Dual System Protocol technology represents a compelling proposition to get mainframe capabilities at the cost level of FPGA chips.

II. MSMI VS EXISTING WAN OPTIMIZATION/ACCELERATION SOLUTIONS

MSMI's solution targets the core issues that impact speed and bandwidth utilization on WAN network links. By placing intelligent dual systems/gateways on either end of the link and in front of routers MSMI is able to overcome the delays and packet losses that impact performance.

MSMI's intelligent dual systems/gateways are a "transparent bridge" on a WAN link that allow rapid ramps of TCP/IP data rates and eliminate lost packets thereby accelerating all traffic on the link. The results MSMI has demonstrated are TCP/IP flows at over 3.5 Gbps (as of today) and bandwidth utilization of over 95% on a 10GE trunk running between Santa Clara, CA and Austin, TX. These results can be scaled to both lower and higher speed trunks.

In contrast WAN Optimization/Acceleration appliances sit behind the router and employ a number of techniques to improve performance. Common techniques among the leading providers include: Compression, Data De-Duplication and Caching. The first two of these effectively reduce the number of bits being transferred in order to improve performance but do not address the fundamental performance limitations that MSMI is attacking. In fact, these techniques could be implemented in conjunction with MSMI's solution as MSMI passes all TCP/IP traffic transparently.

In addition, some WAN acceleration solutions rely on techniques that effectively pretend to be multiple users in order to provide more bandwidth for a single flow (lane stealing). The MSMI solution accelerates all flows independently and democratically.

WAN acceleration solutions attempt to attack the problem from the edge of the network whereas MSMI's solution adds intelligence inside the network thereby addressing some of the fundamental limitations of today's TCP/IP WANs.

III. LITERATURE REVIEW

A packet switched network only allocates bandwidth when a block of data is ready to be sent, and only enough for that one block to travel over one network link at a time. Depending on the nature of the data traffic being transferred, the packet-switching approach is 1000 times more efficient than pre-allocation techniques in reducing the wastage of available transmission bandwidth resources. To do this, packet systems require both processing power (faster CPU's) and buffer storage (memory) resources at each switch in the network for each packet sent. The resulting economic tradeoff is simple: if lines are cheap, use circuit switching; if computing is cheap, use packet switching. Although today this seems obvious, before packet switching had been demonstrated technically and proven economical, the tradeoff was never recognized, let along analyzed [3]. Thus, communication protocols performance relies on computers processing power and memory access speeds to develop faster protocols.

The references appended to this paper indicate the work done on multi-core processors, FPGA [5], [6] and systolic arrays [8] on ASIC processors are some of the evolving technologies. Of late Intel's multicore chips, and using XLINX/ALTERA FPGA's as co-processors, InfiniBand [7] (IB) Fabrics and NVidia's GPU's (Graphics Processing Unit) have established their presence in the High-Performance Computing (HPC) industry as High-performance interconnects and computing modules. Such solutions help increase computing power and communication speed but only independently. Acting independently needs special interconnecting hardware or unifying OS. In addition, acting independently has inherent problems such as programming these connected devices with different device driver software. As long as compute and communicate are independent functions, they do not produce a whole that is greater than the sum of the component functions.

If we look into some architectures of shared memory multiprocessors major portion of the communication latency is associated with the cache system and the interconnection network. An important factor that determines the latency of a cache system is the communication capabilities that the underlying interconnection network provides. All existing designs of the high-performance interconnects for shared memory multiprocessors, based on IB and NUMA (Non-Uniform Memory Access) technologies, build scalable OS based servers forming clusters and simultaneously suffer latency and compute power. And they all face the architectural challenge, which requires rigorous considerations of behavioral, topological and technological aspects of shared memory multiprocessors and interconnection networks. Most importantly for such cluster designs, system level influences of the cache system behavior on interconnection network design play an important role in facilitating high performance communication in system architectures.

A. InfiniBand

InfiniBand is the result of a merging of two different projects, Future I/O and Next Generation I/O. The projects aimed at creating new I/O technology for connecting systems with peripherals and eventually replacing other I/O interfaces like PCI (Peripheral Component Interconnect), Fiber Channel and even Ethernet and become the unified backbone of the datacenter. In short, InfiniBand entered the system scene from the I/O side (or the “outside”) of systems. In the InfiniBand architecture the main processors at each end of a connection cannot address each other’s memory or I/O devices directly.

This means that all communications on InfiniBand requires an external software driver to control the communication and handle buffers for the RDMA (Remote Direct Memory Access) engines. Thus, for InfiniBand, the sending program must set up the RDMA engine through a number of accesses to the IB adapter in the I/O system and then the RDMA engine will read the data from memory and send it across the interconnect fabric. And again, their communication is done on TCP/IP and InfiniBand inherits the TCP/IP constraints.

B. WAN Optimization through Routers, Network Adapters

Many co. in the past have tried to increase the speed of the TCP/IP by designing faster and much cheaper routers. For example, “Caspian Networks”, which built such routers, and the technique that that used was to look up the route once on the first packet and from then on use the packet’s unique flow ID to find the assigned route with much lower effort as the no. of flows is much less than the number of packets [9]. Based on this concept used a number of specialized ASIC’s which are designed to gain a large amount of the claimed speed/cost advantage. But they all failed keep when the no. flows increased over period of time.

One more network co. Anagran developed hardware-based solution to increase the network speed reducing P2P congestion which at that time was discouraging carries providing local access Internet service. P2P allows one user to receive 100’s of flows to download music or videos shared from 100’s of other users. The task was to recognize that all these flows were to one user paying one bill and adjusting the whole set of flows to the same total traffic rate as each other of the carrier’s users. Another feature was to allow customer control of traffic segments to be split by layer 2 labels for the various users [4]. This is a network edge function and it really does not address the problem with TCP/IP protocol as such.

Of late many successful co. such as Cisco, and Riverbed have WAN acceleration solutions [11] they all use some techniques to optimize the speed and they all customized to specific customers and they are not generic solutions and they all attack problem from the edge of the network. Some WAN acceleration solutions rely on techniques that effectively pretend to be multiple users in order to provide more bandwidth for a single flow (lane stealing). Whereas the MSMI solution accelerates all flows independently and democratically.

Some other solutions include accelerating specific flows only [10]based on the QoS techniques and again they all do not address the core TCP/IP problem. In fact, these techniques could be implemented in conjunction with MSMI’s solution as MSMI passes all TCP/IP traffic transparently.

Many recent companies have improved performance of the communication by using FPGA’s as co-processor to boost the traffic by using techniques like kernel by-passing and zero copy techniques. They all to certain extent work fine for short distances but they have not solved the real-problem of the TCP/IP.

Some even re-wrote the entire TCP/IP stack on FPGA’s and improved the performance [13] when compared conventional OS based TCP/IP performance, but they all carry the problems of TCP/IP on to the FPGA’s and they still cannot reach transfer rates more than 120Mbps for long-haul transmissions. They all provide either customized solutions which are not generic to all types of communication traffics or they are not stateful.

C. The current State of Communications Technology

In the last decade, **Moore's Law in Computing** has fueled three trends: The proliferation in the number of discrete microprocessor cores within a single device, the proliferation of multiple interconnected devices within close proximity to each, and the creation of vast "clouds" of interconnected servers, available on demand, residing on the Internet.

At the same time **Moor's Law in Communications** fuels demand for band-width in communication trunks, i.e. for every 18 months the band-width is being doubled in each trunk by adding extra circuits. This we can see in any bank or organization whose data-centers are inter-connected requires extra circuits to increase their band-width requirements.

Whereas if we look at the current state of communications, today's IP networks need improvement to handle the increasing demand for performance. The TCP/IP protocol was deployed in 1983 and has not had an upgrade to keep pace with the increase in packet volume and trunk speed that today's applications require. As a result, the current TCP/IP protocol response time and Quality of Service – QoS) suffer from random discards of packets due to congestion and high packet volume. Algorithmic Trading/HFT is slowed due to low flow rates and TCP/IP stalls. Video works today at HD but 4K, 8K, & 16K video do not work except 4K locally at the expense of other applications. While Moore's Law has increased computer speed and memory, there has been no sufficient improvement in TCP/IP, which tries to guess what transmission rate to send data over a long-congested path. What is required is rate control inside the network to assure rapid rate adjustment of all the flows.

Though technological developments made processors and memory access performance faster, the 60year old TCP protocol's design and architecture remained same till today and it cannot and there by achieve transfer rates more that 120Mbps for long distance transmissions such Trans-Atlantic and inter-continental long hauls. Many companies have provided higher transmission rates for content distribution but many of them or not stateful transmissions. Adding on to that current networks are dumb, and they are made intelligent enough to adjust the rates accordingly without packet drops.

Some Silicon Valley companies achieved semi stateful rates over TCP but they all use some techniques like caching, flow lane stealing, UDP tunneling, multi-threaded programming to increase transfer rates but they all have not changed the TCP protocol design as such and they can never reach more than 600Mbps and that too they are very expenive.

Whereas if we look at the chronological developments in processor designing, starting in the early 2000's silicon processor designs hit a "Power Wall", namely the inability to dissipate the heat produced when a single processor core ran at a faster clock rate, because of this the computer industry changed paradigms. Performance from that point on would be provided by adding more processor cores onto each piece of silicon and not by speeding up the clock rate of a single processor. The proposed new MSMI-protocol makes use of the many core architecture to implement its metal algorithms.

D. Gaps in Communications Models

When TCP/IP was deployed in 1983 it allowed the Internet routers to become much less complex than X.25 or NCP (the original ARPANET protocol. They need not manage flow rates and lost packets as TCP leaves this task to the edges (sender & receiver) **[12]**. This was good while computers were too slow and lacked sufficient memory to improve things inside the net. Recently however, computers with many fast cores and lots of fast memory have become capable of supporting the required functions at 10 Gbps and soon 100 Gbps. Thus, it is tsxime to add rate and congestion management inside the Internet. This is a major mind shift for the router companies and most network experts, as this has always been considered "unfeasible" and "incompatible". A complete solution plan which can isolate the solution to upgrade the network trunk by trunk increasing link efficiency, while at the same time saving Capex (capital expense.) is needed.

There is a great need for faster connections and a real-time response time demand that cannot be achieved with the legacy TCP/IP protocol **[2]**.TCP/IP utilization on a typical circuit (Trunk) must run at 50% packet throughput on average to avoid packet loss and network delay concerns. Which is a fundamental problem with existing communications.

III. MSMI - TECHNOLOGY DIFFERENTIATORS

As discussed in the literature review most of the WAN acceleration / optimization solutions available in the market are the network edge solutions, not the core like MSMI and the technology is largely different. The words about packet flows are common in any network product but the innovative concepts, the technical details, and the markets are very different between the prior company's solutions and MSMI's. Today the P2P problem has been largely eliminated due to the inexpensive availability of streaming video so not many people try and illegally share videos. Thus, that market was and is evaporating.

MSMI, on the other hand is in a new growing market where all packet networks lack intelligence and do not try and control the traffic volume, just drop packets if they build too large a queue. MSMI protocol addresses a totally different issue, that of adding intelligence to the network so it can control its own load and not drop packets. Without intelligence there is a fixed upper bound for file transfer rates which cannot be fixed at the edge. To fix this MSMI technology uses new techniques to ramp up flow rates dramatically. This task requires very high rate adjustments in about 1 millisecond and even in micro seconds. Many leading company solutions reorganize the traffic with routing and QoS rules, whereas MSMI does not do this.

So, the techniques are very different in MSMI and MSMI dual system protocol does this without replacing the network infrastructure and saves money in the process buy improving the trunk utilization. MSMI is adding inexpensive gateways built using FPGA's to the network trunks and not touching the router infrastructure. They can accelerate flows to 10-50 times the rates possible with TCP today as TCP's rate is inversely proportional with distance. With MSMI's two gateways working together the impact of the trunk distance is eliminated. The technology only became possible as computers became fast enough to keep up with fiber rates in the last two years. MSMI has produced software to speed up packet networks from the inside using newly conceived concepts for accelerating TCP flows and increasing the utilization of the trunks. MSMI on the other hand eliminates packet drops and aims at speeding the flows to fill the trunk and allow a priority flow or flows to speed up greatly and the solution is very generic.

IV. NEW PROTOCOL MSMI'S DESIGN GOALS

The problem being solved is that TCP today cannot achieve data transfer rates desired and potentially possible over distances greater than 10 miles. TCP rate is proportional to 1/RTT and 1/(Loss)^.5. Loss is when a packet is lost in transit which is mainly due to overloads in routers and switches and for wireless or wire lines, noise.

The first goal is to greatly reduce the time it takes sender to deliver data to the receiver. This requires greatly increasing the operating rate of the flow and also reducing the time to get to the operating rate. TCP can achieve much higher rates if the RTT between it and a receiver which acks it packets is very small. Thus, the goal can be achieved if the two local loops are short with low RTTs and the GWs can somehow maintain this high speed across the trunk. Until the trunk saturates the rate will be limited by the longer local loop. If the sender or receiver is in a Data Center that local loop RTT will be very low and the user's local loop at the other end will be the controlling factor. The maximum rate of his connection and its distance will control the max transfer rate, not the trunk (or trunks) or the data center local loop or the server speed.

As a majority of users have local loops less than 64 miles (1ms RTT) from an originating Trunk Head End and many are connected at 100 Mbps to 10 Gbps, this is the primary target market for major improvements. In these cases, the goal is the rapidly increase a TCP flow's data rate to their local Max Rate (usually the connection to their building) subject to the total trunk capacity. If both local loops are in Data Centers and connected at 10 Gbps, then their Max Rate will be determined by the Trunk load with a maximum for a 10 Gbps trunk of 8 Gbps.

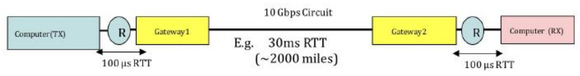
If the trunk is loaded today and each flow is increased in speed by 10:1, the trunk will still have the same load as all flows finish in 1/10 the time leaving capacity for the other flows (same bits moved/sec). As trunks, today are usually loaded to an average of 50% to avoid synchronization overload, if the GW's also manages the flows to be smooth, not saw toothed, and manages their rates more rapidly to control the total load, safe utilization of ~95% can be achieved. This saves the trunk operator the cost of a second trunk and thus more than offsets the cost of the GWs producing a major cost saving. Thus, the second goal is to smooth the flow rates [4]and achieve rapid rate correction times so as to permit ~95% trunk utilization.

V. HIGH FREQUENCY MSMI-PROTOCOL ALGORITHM GOALS

This paper proposes high speed communication protocol algorithm and also its implementation to produce stable scale-invariant reconfigurable computing architecture on a dual FPGA based gateway systems connected with any trunk speed from 1Gbps – 100Gbps. To that end, we introduce a new high performance architectural design wherein the software is programed on the "metal" (Silicon) using many processing cores and their memory built on FPGA's (Field-Programmable Gate Arrays). The length of the trunk i.e. end-to-end WAN links between these gateways can be up to 20thousand miles.

All the components of the computing model cores/memory on the metal can create deeply dense pipelines to process 20 million flows connecting vast pools of memory and cores.

A. The typical deployment of the new MSMI-protocol gateways will as below:

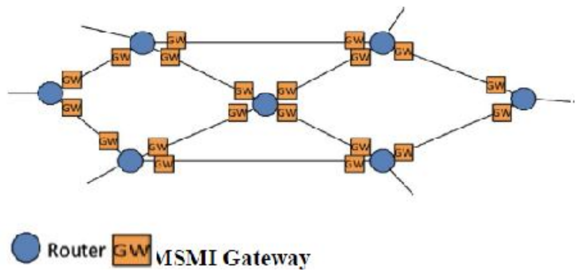


Gateway1 & 2 are FPGA based devices programmed with new protocol and Computer1 (TX) traffic generator & Computer (RX) receiver.

Design Goals:

1. The design should allow the gateways to be commissioned at the customer owned WAN links at the two ends of the trunk without disturbing the existing network traffic over a long-haul 10Gbps distance of 1000 – 2000 miles.
2. The protocol should accelerate and quickly saturate the network traffic to its 95-99% capacity.
3. The protocol should acknowledge packets as they enter Gateway1 instead of waiting for RX computer, ensuring the delay now is 100μs instead of actual RTT (30 milliseconds).
4. The protocol should accelerate data rate and ramp up the same.
5. The protocol should speed up and achieve traffic transfer rates to 30-100X i.e. 1500 - 9500Mbps.
6. Protocol's load measurement algorithms and communication between gateways synchronization is key to insuring results.
7. The protocol should rapidly manage and speed up of all flows within the circuit to ensure that the traffic remains within the circuit capacity, this eliminates overloads and packet losses in the routers

B. Typical deployment environment in geographically distributed networks will be as below:

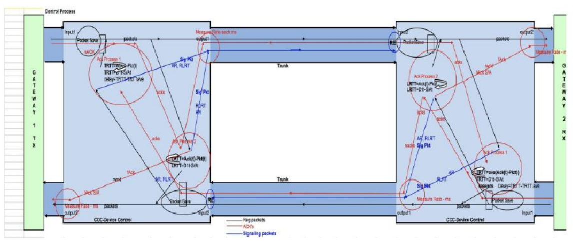


8. The protocol in dual gateways to form a system and make network highly intelligent efficient between every segment of the network.
9. The network capacity should near double at 95+% utilization
10. The routers to run smoothly due to the protocol and its unique buffering technique
11. Any TCP/IP flows to maintain acceleration across the network between routed nodes to 2.x Gbps
12. The new protocol in gateways to maintain state-full flows between ingress/egress interfaces between routers.
13. Entire algorithm is based on metal, is implemented on FGPGA multi-core using deeply danced pipelines using hardware programming language Verilog.

VI. HIGH FREQUENCY MSMI-PROTOCOL CONTROL PROCESS DESIGN

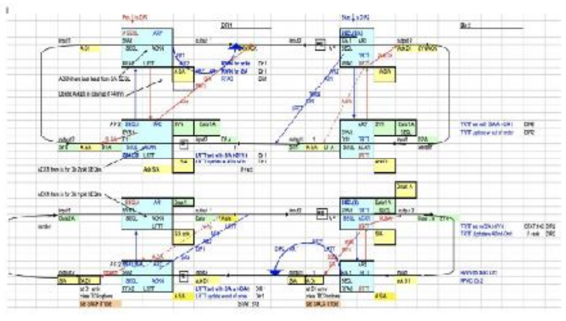
A. Top-level control process design:

1. Loss or Delay in remote Local Loop: 1st loss & delay detect at Ack Process 2 in 2nd GW - if loss, & have packet, replace & reduce rwnd to slow. If delay>MIND reduce rwnd to slow. Loss & don't have pktforwarded Ack only (Slow at 1st GW).
2. Loss on trunk: Detect atPacket Process 2 in 2nd GW by missing packet SEQ#. Send nACK to Ack Process1, 1st GWW and let it determine where loss occured.
3. Delay in trunk: Detect in Ack Proc.1, 1st GW as Ack received, pkt there. Resend packet & slow here. Also if TRTT-RLRT-smoth Trunk>MIND - slow here. Next auto SigPkt will update RtEng and 2nd GW.
4. Loss in 1st Local Loop: Detect Loss in Packet Proc.1 in 1st GW, send nACK. If SACK on, compute SACK. On 3rd nAck record time and compute LRTT upon receipt of lost packet.
5. All Sigpkts, auto or #1, incluse RLTT to other side. Auto SigPkts sent from Ack Proc. 2, 1st GW & second GW with AR, RLRT.



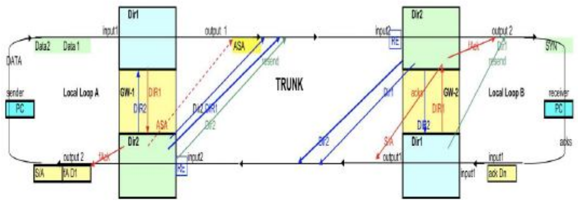
B. Top Level TCP flow control between the gateways will be as below:

1. Sender: There will be a limit for a sender rate based on his feed link, let it saturate the link.
2. Receiver: Only info is FAILs at similar rate.



VII. HIGH FREQUENCY DUAL SYSTEM MSMI-PROTOCOL DESIGN

The system requires two Gateways (GW) built on FPGA boards which are at either end of an Internet trunk. The trunk has a RTT of TRNK seconds with a supported range from 2 ns (8”) to .999 sec (64 K miles) although no use is expected less than 100 µs (few miles). In each GW, there are two major processes, one for packets (pkts) entering from outside, called Direction 1 (Dir1) and one for packets entering from the trunk, called Direction 2 (Dir2).



In above figur the sender and receiver can reverse any moment and there can be hundreds to thousands of PC’s or servers, each with their own flows (packet streams with the same ID). A flow ID for Internet traffic is having the same addresses, protocol, and ports. To identify a flow we hash the addresses, protocol, and ports into one 64-bit number. In release 1 we only hash and process in detail TCP flows (protocol 6) as they are the primary traffic and are rate controllable. All other traffic is passed through unchanged. The diagonal lines are GW constructed traffic of three types: Signaling packets (Sigpkts) shown as blue lines or dashed red line, fAcks which are fake Acks (red lines) generated by the GW, or retransmissions of lost packets (green lines). Sigpkts are a based on the TIA 1039A standard with a modified payload. They convey information between the two GWs about a flows local loop rate, local loop delay, packets sent, and error information.

A. Dual System Max rate Controlling

The low RTT for the sender allows a much faster rate increase ramp than TCP over the Trunk RTT. The gain is proportional to the Round-Trip Times (RTT’s). So, if the Trunk is 10ms and the sender loop is 1ms that is a 10:1 potential speedup. Cross country would be 40:1. In most cases this is too fast and must be controlled to be somewhat slower but still the rise time to Max Rate will be much faster. Second, the rate is not reduced by distance except for the Local loop distance.

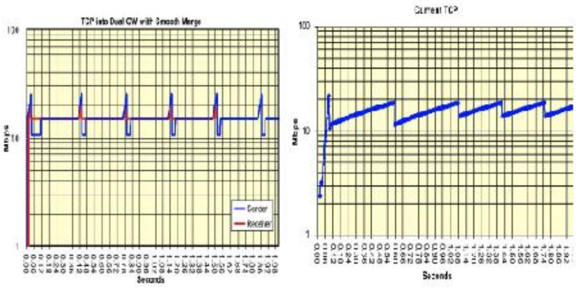
Once packets are acknowledged by the 1st GW it has the copies if needed. The trunk itself has no switch or router to drop packets so there is no distance slowdown effect. The second GW keeps another copy of packets so that losses on the final local loop can be quickly resent. Thus, it is basically possible to quickly ramp up to the maximum sender rate and maintain that rate. If one is sending to a faster or equivalent server the max rate of the sender is easy to achieve. For backups to a data center server this allows the sender’s highest rate subject to trunk capacity.

B. Dual System Receiver Overload

As TCP operates today it does a SYN and in a RTT receives a SYN-ACK to confirm. Then it sends 2 packets and in a RTT receives an ACK allowing it to send 4 packets. As long as no error occurs this continues to double the packets sent and thus the rate every RTT.

When the net or the receiver overloads a packet will be lost or delayed which signals the sender to cut the rate in half, change to a slower increase pace and keep this process up creating a saw tooth wave shape, one cycle every error. The constant feedback end-to-end helps limit overloading the receiver with too many packets too fast.

This limitation is missing for the dual GW system as it could ramp up extremely fast with a low delay sender local loop, well in advance of the receiver seeing any data packets. If the receiver or its local loop has a lower rate limit than the sender, this creates a problem where the overload could be thousands of packets too many in way too short a time. Thus, a new technique we call “Smooth Merge” has been developed to make this work perfectly, even better than normal TCP.



The above figure shows the Smooth Merge where the Sender peaks above the receiver, then drops, and when merged returns to the receiver rate. The receiver is at rate in 30ms. On the right with standard TCP the sender peaks in 92ms and starts its saw tooth behavior getting the receiver to rate in 100ms. Both trunks have 15ms RTTs.

Smooth Merge involves two ramp rates for the server end and holding a fixed rate to the receiver after a receiver loop error while signaling the 1st GW to slow the sender to 50% of that rate. This signaling is a critical feature and will be expanded on later. Packets are saved in the 2nd GW as they arrive too fast for the receiver rate. This process is mathematically complex but allows the 1st GW to determine exactly when to raise the rate up to the receiver rate so that the packets received by the 2nd GW just merge into the receiver as the saved packets have all been sent.

This insures that the receiver never loses a whole bunch of packets as happens in normal TCP when the senders double speed spray of packets arrives and cannot be slowed for the full RTT delay. In the dual GW system, the sender's rate at the receiver is only modestly above the rate causing an error and is slowed quickly to 80% of that rate based on the small local loop RTT. Thus, error recovery is fast, and the receiver gets a fixed rate stream after the error.

When the streams have merged, and sender and receiver are both held at the same rate the system waits a period and tries higher rates again every so often until fixed at the best rate obtainable. As a result, the performance where the receiver or receiver local loop are the rate limiting factor gets up to the max RATE fast, and smoothly adapts to the receive rate limit with minimal error recovery problems, much less than occur in normal TCP.

C. Dual System Rapid Rate Control

Using lost packets or delay to slow the sender is what TCP expects. When done from a receiver 15ms away it lets the rate grow for 30ms from when it went too high. If done from the 1st GW 1ms away it is much faster but still causes a 50% rate drop and a packet recovery cycle. A much more precise and efficient (no packet recovery) method is to use the receiver window plus delay to control the rate. The sender is mandated to not send more packets than the receiver window says can be received per RTT.

The rate it then sends depends on the packets allowed, times their size divided by the round-trip delay. As the receiver window has limited range the RTT can also be adjusted by delaying the fAck thus expanding the RTT. This provides a powerful ability to set the sender rate precisely to any rate (after the first few ms). Thus, when a rate reduction is needed for the trunk overload, all flows can have their rate directly dropped as required a few ms after their next packet arrives. So, the trunk load can be controlled rapidly in a few ms. Thus, it can be held at 95% with ease.

Similarly, when a receiver on a flow has an error, the input rate can be dropped by 4:1 rapidly with one fAck. Another benefit is that there needs to be no saw tooth to find the receiver max rate, the sender rate can be increased smoothly at any rate slope desired. Then if an error occurs, the rate that caused it is known and the rate need not be dropped in half but 10-20% can work. Precise rate control allows dramatically smoother rates which largely eliminate router and switch packet losses for the flows. This applies to all routers in each flows path. Applied to all trunks in a network, router losses should go to zero.

D. Dual System Flow Synchronization

A major problem limiting the ability of routed networks to about 50% utilization is flow synchronization. When a router has a brief overload, it drops packets from its queue. Those flows that were caught in the discard drop their rate in half and restart. Unfortunately, they all restart at the same time. When their saw tooth's peak at the same time they catch other flows in a router discard. This multiplies fast until most all flows have synchronized peaks.

The router now must buffer packets during the huge peaks to smooth the load being passed to the next trunk. As a result, the router delay increases toward 1 RTT. It can now operate smoothly at high utilization, but the added delay hurts every flow, doubling their round-trip time. This effect is sufficiently harmful to customer performance that capacity is normally added to hold the average utilization closer to 50% where synchronization is unlikely.

In the dual gateway system, this problem is avoided by treating each flow separately with no common queue. Also, with short local loops at either end of the trunk and using the receiver window to precisely control flow rates with no saw tooth, the control is so rapid that any overload can be controlled quickly. As packets are not discarded to control rate, there is no synchronization. Utilization can then be increased to ≥95% without adding delay, either in the gateways or in the adjacent routers.

VIII. HIGH FREQUENCY DUAL SYSTEM PROTOCOL MSMI RATE CONTROLLING

A. Receiver Window

To control the rate of a flow the gateway receiving data packets from a sender sends an acknowledgment packet called herein a fAck for each 1 or 2 packets received. The fAck is sent locally back to the sender with a receive window called RWND and an added delay of DLAY. If the rate desired is RATE, the receive window scale factor is SSCL, the local loop RTT is LRTT, the data size is SMSS, and the packet size is LEN (in bits including Ethernet overhead) then:

$$RATE = (((2^{SSCL} * RWND * LEN)/SMSS)/(LRTT + DLAY))$$

RWND must be between SMSS and 2^16-1 by increments of SMSS so in general there are 44 values. DLAY expands the dynamic range to 1719 with 40ms and to 16,813 with 400ms. Normally >40ms is not used but where a very low rate like 500 Kbps is needed and SSCL is set to 4 to allow 9 Gbps maximum rate then large delays are needed. We define RFAC=SMSS/LEN. Then from the RATE equation above but leaving out DLAY to start we find:

$$RWND = Min\left(44, Max\left(1, int\left(2^{-SSCL} * \frac{LRTT * RFAC * RSTP}{SMSS}\right)\right)\right) * SMSS$$

This makes RWND a multiple of SMSS as required. Then we can use DLAY to make the rate precise:

$$DLAY = Min(0.4, Max\left(0, \frac{2^{SSCL} * RWND}{RATE * RFAC} - LRTT\right))$$

These calculations are done periodically upon packet receipt the greater of (every millisecond (ms) or every packet) to minimize compute time and are used for sending fAcks. A controlled ramping of the RATE is used when a flow starts up to insure it does not overshoot the receiver's maximum local loop rate limit by too much. Given a trunk RTT of TRNK (measured) the ramp is set to double the sender's rate smoothly every TRNK, similar to normal TCP over the trunk, after an initial fast rise to a starting rate (the senders ramp with the short local loop). For this ramp with the time since the last update of t, the rate ramp RSTP is computed incrementally from the last rate:

$$RSTP = Min(AR, RSTP * (1 + RSTX * (\frac{t}{TRNK}) + RSTY * \left(\frac{t}{TRNK}\right)^2)$$

Where $RSTX = \ln(2)$, $and\ RSTY = (\ln(2)\text{^}2)/2$

This comes from the first 2 terms of the series for an exponential increase. The result is quickly computed and RSTP is used instead of RATE in the prior equations for RWND and DLAY. If RWND is unchanged, as it often is, then only one fAck with its DLAY is needed per packet received. If RWND changes then TCP protocol requires a first fAck without RWND changed and a second fAck with the RWND change. Each fAck also contains an acknowledgement # (ACKN), or packet sequence #'s received without holes (lost packets). This is carefully computed so that the sender is informed about packets received and if there were lost packets. As the Local loop RTT is small, lost packets are quickly resent. The ramp continues until one of 3 events:

1. The rate increases to AR which is the maximum rate currently allowed on the trunk (Capacity-uncontrolled traffic)/ (# of controlled TCP flows) or a lower local loop rate.
2. The sender's local loop starts losing packets: the rate is lowered by 30% each time and then continues to ramp up. But after a number of errors with no significant rate gain, the rate is frozen.
3. The receiver's local loop loses packets or incurs delay increases > 30ms at which a process called SLOW is invoked whereby the sender's rate is reduced 50% for a short period and then equalized with the receiver's rate. This is the Smooth Merge process.

All this is done per flow based on its own events. Global trunk overload feedback (load exceeding 95%) is applied to the fAcks which tell the sender the rate to send however as this is using a receive window there is no sender reset to ½ or zero, just a small (0-5%) rate decrease.

B. Scale Factor SSCL

TCP sets a scale factor for its receiver window at startup on SYN and SYN/ACK. Thereafter there is no way to change the scale factor, SSCL. Thus, in the dual GW system SSCL must be set before the maximum local loop rates are known. SSCL is a number between 0 and 14 and the total data bytes that can be sent by the sender is the receiver window RWND times 2^SSCL. Therefore, it is very important. When s SYN or SYN/ACK is processed the final GW changes SSCL to a value determined from experience with past flows over this trunk.

This is a fairly good guess at a good value considering the trunk RTT is fixed and the local loops are likely to have similar RTT's for all flows. The value of SSCL determines the minimum rate a flow can be controlled to with the receiver window as well as the maximum rate the flow can be controlled to. Using a maximum DLAY delay of 400ms allows the ratio of max/min rates allowed of 16,813. With a 10 Gbps trunk SSCL values of 0-4 are all that are needed; SSCL=0 allows 32 Kbps to 546Mbps whereas SSCL=4 allows 519 Kbps to 8.7 Gbps. Most all CPUs and their local loops can support 519 Kbps for SSCL=4 is the likely choice but some deployments like feeding a cellular head end may require lower SSCLs.

To determine the best SSCL through history there are cleanup processes which periodically cleanse the flow records and they collect the average RATE and Local Loop RTT (LRTT) for each direction of flow. Then they compute the optimal SSCL as follows:

$$SSCL = Min(14, Max(0, int\left(\frac{\ln\left(\frac{RATE * RFAC * LRTT}{65535}\right)}{\ln(2)} + 1\right))$$

To eliminate LN computation the following is faster;

$$SSCL = Min(14, SR(int((RATE * LRTT * \text{RFAC})/65535)))$$

Where SR () means Shift Right an integer 'n' times until 0 left, then SR () =n

This process adjusts the SSCL used for new flows whereas the initial value of SSCL is set after an in-place test.

C. SLOW Process

When the receiver loop has a lost packet or extra delay >30ms then the process sets SLOW=1, lowers the output rate to the receiver to 80% of the RATE at the failure and sends a signaling packet (sigpkt) to the other GW. Note that the sender has been raising the rate every 1ms by 1.7% so the failure rate is very precisely known. The sigpkt carries the total # of packets sent since flow start, the new rate R=.8*RATE, the SLOW level, and the local RTT (LRTT). When SLOW=1 the receiver holds the rate at R by saving packets received from the trunk and only sending them at the new fixed rate.

The other GW upon receiving the sigpkt TRNK/2 later sets its SLOW=1 and cuts its rate with a fAck to half the new receiver rate (R/2). TRNK/2 later this lower arrival rate at the receiver GW port starts reducing the # of saved packets. The maximum # saved may be 2 to 7000 packets. The receiver easily determines when there are no saved packets left and resets SLOW=0 and reverts to sending packets as they are received. The sender side GW must revert on its own to R at the right time to insure the receiver GW is cleanly merged to that rate. This is a somewhat precise calculation so that the merge happens precisely when the sender runs out of saved packets. The calculation the receiver makes is:

$$t = \left(\frac{INSM - OTSM}{\frac{R}{LEN}} - TRNK \right) * 2$$

Where INSM=total packets, the sender has sent since flow start

OTSM= the total packets the receiver GW has sent from the sigpkt, TRNK=trunk RTT, LEN=total bits/packet including overhead

The concept is that there is a INSM-OTSM packets difference at a packet rate of R/LEN minus the trunk RTT time from the receiver fault to when the receiver GW should merge. Thus, the sender GW computation provides the time (t) it should revert to R. The sender GW then sets a time when it should try again to do this over again by raising its rate. After a number of tries it stops and fixes the flows rate as uncontrolled which allows other flows to use the capacity reserved for this flow's increase.
Also, provision is made to complete the transfer of saved packets if the sender completes and sends a FIN in the middle of the process. Without sigpkts this SLOW process would be impossible and a receiver on a limited capacity local loop would lose potentially thousands of packets which is extremely hard to recover from. The merge happens in less than 2*TRNK even when the saved packets number 7000 and thus there is no need to touch the packets which are still in the GW's input buffer.

D. Signalling Packets

One need for sigpkts was shown in the preceding SLOW process. SLOW also requires the most information to be transferred of all the uses (RATE, OTSM, and LRTT) plus control fields. The basic structure of a sigpkt is based on the Telecommunications Industry Association's TIA 1039-A standard. It was designed to allow a sender using an app to create sigpkts and send them along with his other TCP packets to request a rate and priority. It required routers (or attachments to routers) to process the sigpkts to manage the flow rates.

The receiver also needed to add an app to process and return sigpkts. As neither the user apps nor the router attachments have been produced, it has had no commercial use. However, the packet structure was designed to allow encapsulate a TCP Ack so that it would pass harmlessly through an IP network and allow the processors in the network to recognize a flow from the fields in the Ack and receive a short information block providing rate information about the flow. This same standardized structure is ideal for the dual GW system although the data required is slightly different and there is no need for a user app. Adapting this structure inside the dual GW for sending information between GWs was done in a way that it does not interfere with the original use and can cleanly be compatible.

The encapsulation of the Acks uses an IP standard GRE encapsulation with a IEEE Ether type created for this type of use. Thus, the packets are totally standard and cannot confuse any system. They are passed cleanly across any IP network but no user system today would understand the IEEE Ether type so they would be discarded if ever received.

In the dual GW system, they are used between GWs to keep the two ends aware of individual flow failures, the local loop delays and rates seen at the other end, and the SLOW process. If no errors occur the Rate and RTT of a flow seen at one end is transferred to the other end at flow startup (at the SYN/ACK Ack) and periodically thereafter every 20ms or so. This allows each end to know the end-to end status. Also, upon three types of local loop failures (lost packet or excess delay) they are immediately sent so the other end knows the new local loop rate and can retransmit packets if somehow lost on the trunk.

The three types are:

- Receiver lost or delayed packets as described in the SLOW process.
  Data packets missing upon receipt from the trunk. The Ack identifies the serial # of the packet and thus the sending end can find its copy and resend the packet.

- If the Sender local loop lost the packet the sending GW would have requested a copy already and most likely received it.
- Packet loss in the incoming sender's data stream reporting the slowdown of that local loop rate.
  The sigpkt has two parameters which identify which type this sigpkt is reporting. This type of information transfer is critical to a dual GW system.

E. Parameters

STAT

STAT is the state of the connection. FB1_STAT is used for levels 1-4, levels 5, 7, 8 are specific to the direction and kept in FB1 and FB2. When a SYN is received STAT=1, when a S/A is received STAT=2, when an Ack of S/A (ASA) is received STAT=3 the operating level.

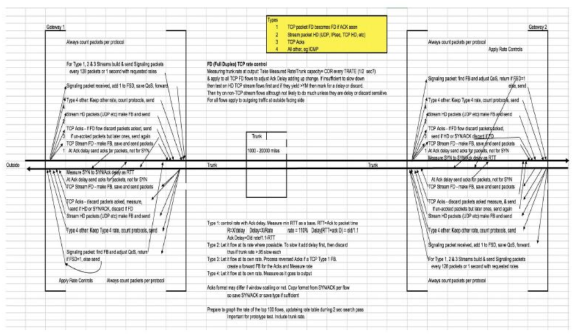
When a RST (reset) is received it is ignored if STAT<3 as we stay listening. But if STAT>2 then the connection is to be aborted and STAT=8. When a FIN is received then the directions STAT=5 and FB1_STAT>=4 to indicate a close is in process.

Then when an Ack of a FIN packet occurs, STAT is increased to 7 in that direction indicating the connection is closed in that direction. When one STAT=7 and the other directions STAT<7 then the connection is still operating. Cleanup will delete any connection with both STATs=7 or 8.

IX. HIGH FREQUENCY MSMI-PROTOCOL DESIGN IMPLEMENTATION

A. Following are the set of activities that are carried on in Dir-1 and Dir-2 of each gateway:

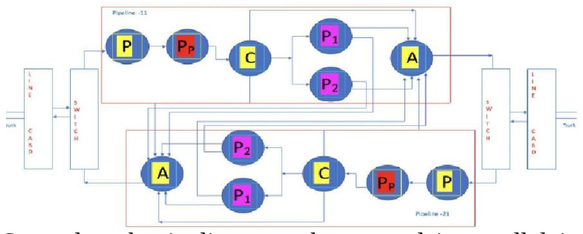
All the activities mentioned below are carried out in various processes in several pipelines.Multiple cores will be used in single pipeline. On each core, particular activity of the logic will be run.



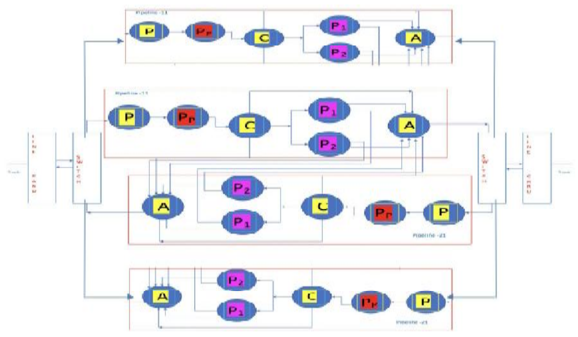
B. Following figure shows one pipeline in both Dir-1 and Dir-2 (Full-Duplex) as red box:

On each pipeline, several cores are being used to execute several activities on each core.

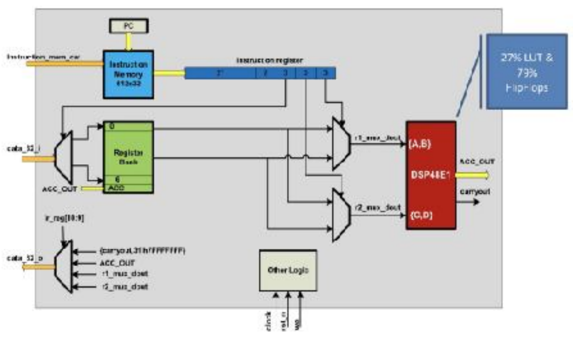
P---Packet Procuring Engine
Pp ---Pre-Processing Engine
C--- Parallel Process Exchange (Runs Packet processing on Multiple Parallel process)
P_1 ----Parallel Process 1
P_n ----Parallel Process n
A ----Process to collect the processed packets and send them



Several such pipelines can be created in parallel in each direction on a single FPGA as below:



C. Following figure shows single core design on a Xilinx vertex-7 FPGA with available DSP's and LUT's on the die:

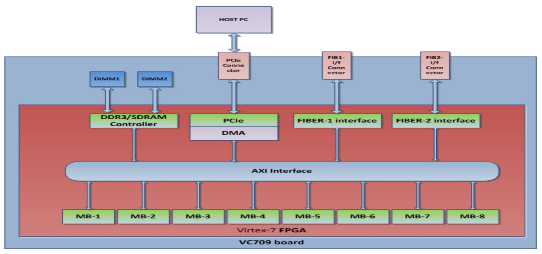


<u>Description of the system/logic in the FPGA is:</u>

1. Xilinx MicroBlaze processor area optimized flavor core is used. There are sixty-four, (64), such cores, hooked onto AXI bus fabric. Each core has individual instruction and data memory, each 512 deep implemented with the BRAM blocks. There is no cache and the core does supports floating point arithmetic.
2. A flash interface for the on-board flash memory is system peripheral used to boo the cores.

3. A DDR3 controller as slave for these sixty-four, (64), cores are hooked onto the AXI bus fabric.
4. A PCI express IP core along with a third-party DMA is hooked onto AXI bus as another peripheral in the system.
5. Custom designed RTL logic is used to as a slave peripheral of the system, through which the SFP+ connectors are accessed by the system.
6. GPIOs for on board LEDs/switches etc. and a JTAG UART for debug purpose are other peripherals.
7. A PLL on FPGA is used to generate the required frequency (156.25MHz) from the external clock supplied by on board 200MHz fixed frequency oscillator.
8. The VC709 board setup is controlled from a host PC. A GUI running on the host PC communicates with the VC709 board through PCI express.
9. The GUI controls following operations of the FPGA based system.
10. Algorithm to be loaded or to run on the multi-core system.
11. After loading the Algorithm, the GUI also controls the Start and Stop of the algorithm running as application on the processor cores.
12. The GUI is also responsible for setting up control parameters for the Algorithms. The GUI also acquires and displays results for both the Algorithms.

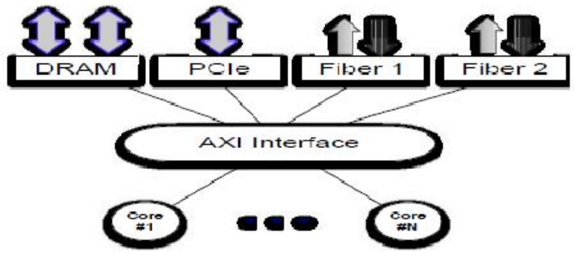
<u>Multiple such processors are connected as below, showing 8-core as below:</u>



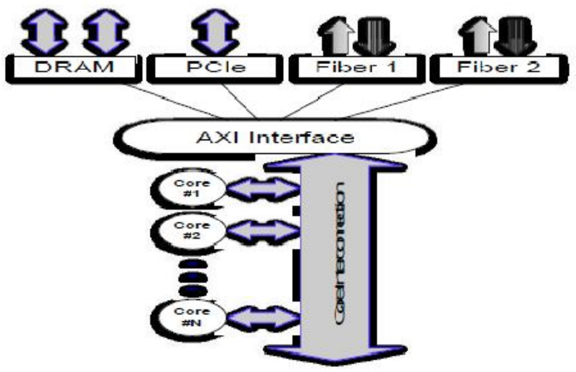
<u>FPGA Bit File Design:</u>

1. FPGA part number: Xilinx XC7VX690T-2FFG1761C
2. Architecture Used:
   - To minimize the time required to get the demonstration systems running we reused as much as possible of the existing IP infrastructure of the Xilinx Connectivity Kit (see block diagram)
   - No Ethernet MAC layer IP required –Need a direct connection to the fiber.
   - Modified the AXI IP block to attach the “Core Interconnection”
   - Modified the IP block to allow “Read” and “Write” operations from the “Core Interconnection” to the DRAM IP, and the Fiber Connections
   - 2 DRAM channels/blocks are used.
3. New Blocks
   - Array of Cores
   - The number of cores will be specified by algorithm
   - Estimate the number of cores from Algorithm A & B pseudo code
   - Core Interconnection
   - Determine if “Core Interconnection” is required vs. using the existing AXI interfaces
   - Specify the inter-core connection micro-architecture.
   - Assume AXI IP blocks

4. High-Level Architecture if AXI can support the number of core

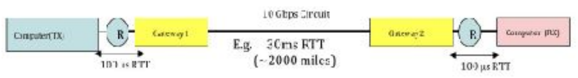


5. High Level Architecture with “Core Interconnection” as below:



X. LABORATORY TEST SETUP

The test setup being used in Santa Clara, CA consists of two 10Gbps Servers and two gateways as shown below are connected in a chain, the same configuration that the product will be used in except that the distance and thus delay of the middle link and the packet losses normally experienced in packet networks is simulated.



Typically, in tests, Computer 1 sends a TCP file to Computer 2, traversing the two Gateways. As the two links between the Computers and their adjacent Gateway will in practice have routers in the middle which will in general cause packet losses to control load, these links have simulated losses. The trunk is expected to be a dedicated circuit which is expected to have very low packet errors or losses. In the la b tests are then virtually identical to a field test with a long haul dedicated trunk and short distance (0 - 30 miles) edge links through routers. The only difference is perception as the performance will be identical for the same error rates and Trunk delay.

A. Current State of the Software

The current software includes all the functions necessary to pass all IP traffic except TCP flows transparently. For TCP flows, the Gateway receiving data packets acknowledges each incoming packet locally over the short delay between the sender and the Gateway, thereby allowing the sender to ramp up sending speed 10 - 60 times as fast as would be possible over the Trunk delay. The receiving Gateway keeps a copy of the packet and forwards it to the remote Gateway which keeps another copy and forwards the packet to the receiver computer. If a packet is lost on input, the Gateway acknowledgement causes the sender to resend the packet in microseconds, rather than 15 - 30 milliseconds. If a packet is lost on output to the receiver, the 2nd Gateway resends the packet in microseconds, again instead of milliseconds. Also, if a packet is lost or damaged over the trunk, Gateway 2 requests a resend from Gateway 1 over the trunk but meanwhile the following packets continue to be sent. Thus, throughput can continue at high speed. The speedup of each TCP flow can then be increased 10:1 or more subject to the remaining capacity of the trunk and the local access limits.

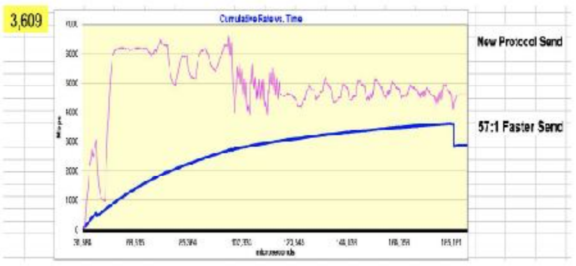
A second mode exists for the Gateways: a transparent mode that allows TCP flows to be passed transparently with only the packet losses and delays to be simulated. This allows direct comparisons between standard TCP and New Protocol's processed TCP.

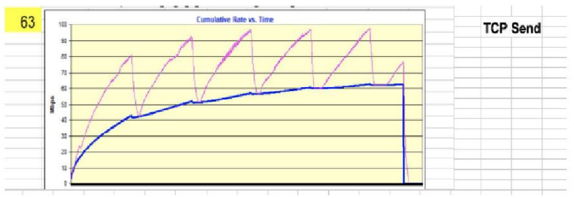
The fact that the New Protocol technology allows the Gateway s to signal the sending computer as to rate it is allowed and that this can be reduced in a few microseconds if needed means that the Gateway s can quickly adjust the rate of all existing flows so fast that it can maintain extremely high utilization on the trunk (95% - 99%) as opposed to today's typical 50%. This feature however is not ready to be tested as that software is not yet complete. However, the speed demonstrated by the flow speed increase provides assurance that this can be achieved as the software is completed.

B. Performance Tests– Live Circuit 2000miles

To date most experiments have been done with a 30ms (2000 mile) trunk delay and 2 μs (700 feet) local delays. All links are 10 Gbps links. There are 3 types of packet loss error rate cases tested; Output link only, Input Link only and losses on both Input and Output links. The packet loss rates tested have been 10^ - 3 (1 in 10^3 packets), 10^ - 4, and 10^ - 5 covering the range of typical router losses.

The TCP used is TCP New Reno. Each Gateway keeps a log that allows easy analysis of the dynamic rate over time and to produce an accurate average rate for the entire file transfer. Although various file sizes have been tested the tests below were al l for a 54 MB file. By this size the performance difference between Standard TCP (run transparently) and New MSMI protocol on TCP traffic has generally stabilized.





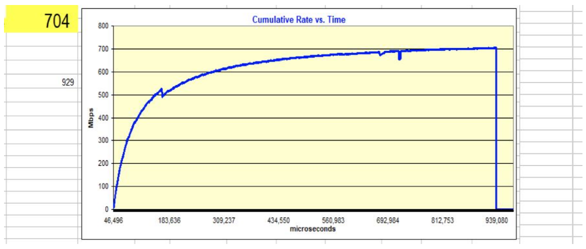
As can be seen in the test results above, New MSMI Protocol's processing speeds up Standard TCP from 63Mbps to 1.6 Gbps on the average, a 57:1 speed increase. For trunks, less than 56Mbps New Protocol's processing would not be able to speed up the flows but it still could increase the utilization toward 100%. However, for faster trunks the gain keeps improving until at 10 Gbps it is as shown. At 100 Gbps (same code, faster FPGA's) the gain per should improve based on the faster computers reducing New Protocol's internal delay. Also, we expect the New MSMI Protocol speed to increase with additional code tuning and current generation servers.

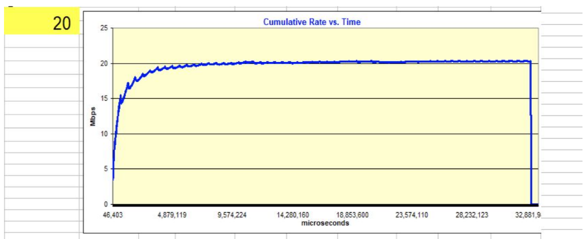
C. More tests with variety of losses

The initial tests are a series of 52 file transfer tests over a live 10 Gbps circuit from Santa Clara, CA to Austin TX. The Round-Trip Time (RTT) over this circuit is 30ms. As TCP slows as 1/RTT the 30ms delay causes TCP to only transfer files at 9 to 115Mbps. The new Dual System/Gateway MSMI-Protocol allows file transfers 11 - 57 times this fast on the average based on these initial tests. Briefly, the Dual Gateways sit at either end of the circuit and locally acknowledge TCP packets, eliminating the impact of the Circuit's RTT and allowing much faster file transfer rates.

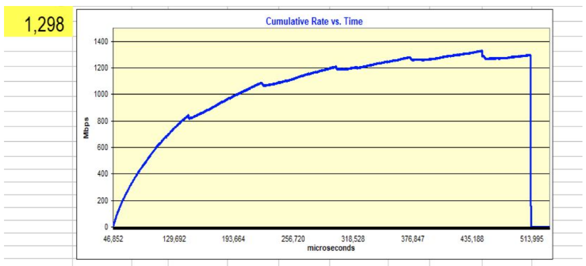
In all the initial tests send an 84 MB file from Santa Clara to Austin with various input and output line rates to evaluate the utility of Dual System MSMI-Protocol technology for various line speeds. At input line speeds below 100Mbps TCP can usually run about as fast as new Dual System MSMI-Protocol enhanced transfers. But as the line rate expands to 10,000Mbps Dual System MSMI-Protocol has a major advantage averaging 24 – 57 times as fast as normal TCP. These initial tests results were predicted when this project was started over 2 years ago and one of the initial test goals is to measure how well the Dual System Protocol technology performs vs. the predicted gain over TCP. The live tests performed with 112% better gains than the prediction.

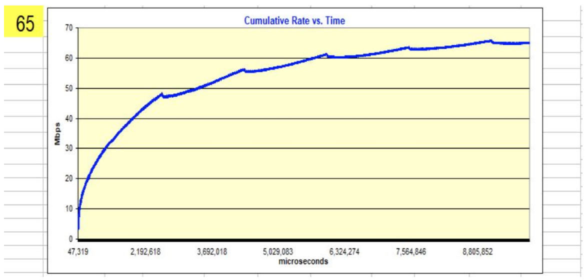
<u>Test – 1: Input link loss = 1/1000; Output link Loss= 0, 1GB trunk</u>



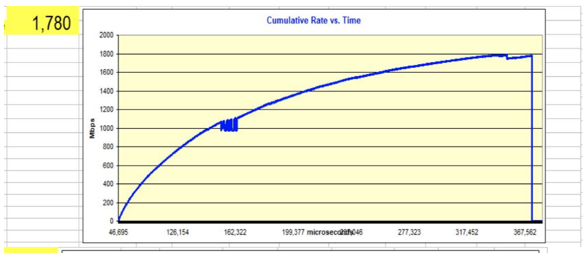


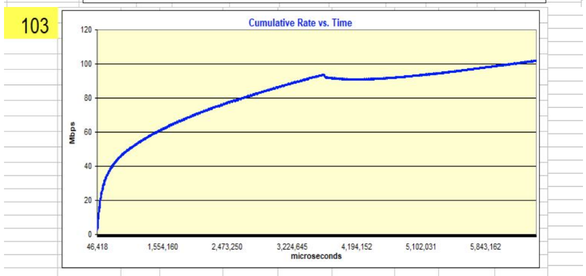
<u>Test – 2: Input link loss = 1/1000; Output link Loss= 0, 10GB trunk</u>





<u>Test – 3: Input link loss = 1/10000; Output link Loss= 0, 10GB trunk</u>





XI. DUAL SYSTEM MSMI-PROTOCOL TECHNOLOGY – SUMMARY

The new dual system MSMI-protocol developed will be applicable to the Banking and Capital markets, Media, Oil and Gas, US Defense and emerging IoT (Internet of Things) applications, including hyper-scale deployments. This protocol will ultimately improve cost efficiencies. These efficiencies will benefit the US consumer and all parties that deliver or depend on improvements in the above said markets. Both social and economic improvements will be derived from his work across all segments of the United States, from the largest enterprises to the individual consumer.

The current MSMI-protocol can be deployed wherever the TCP/IP is used and make the transfer rates faster by 30-100times and with 95-99% trunk utilization. This protocol, can be used in all enterprise communications and also in very mission critical operations which require high speed communications. The final outcome of this research, the algorithm and the technology is fully based on embedded computing on metal and unlike the traditional systems; systems built on this protocol technology do not have the burden of OS or the legacy heavy weight protocol stack. The computing and communication are fundamental in the architecture of the gateways that provide high speed data transfers across the geographically distributed gateways enabling cost savings on resource utilization, their management and programmer productivity.

The new MSMI-protocol's transfer rates are very high and can connect multiple compute nodes can be interconnected forming geographically distributed computes nodes to form a fabric and deliver distributed supercomputing.

Thus, the products/systems built on this protocol can be deployed in markets to do specialized low latency communication intensive jobs like Enterprise Systemic Risk Management for globally distributed corporation OR on networks working in an already existing datacenter doing specialized tasks such as replication/synchronization of several DB's in real-time.

Finally, the MSMI-protocol's technology makes it a unique i.e. no OS burden, fully on Metal with reconfigurable hardware programming model and fully deterministic in its performance, providing super communications at lowest cost.

REFERENCES

[1] L. G. Roberts, "Beyond Moore's law: Internet growth trends," in Computer, vol. 33, no. 1, pp. 117-119, Jan 2000.

[2] Paula Doyle, "Introduction to Real-Time Ethernet", Industrial Ethernet University, 2004

[3] L. G. Roberts, "The evolution of packet switching," in Proceedings of the IEEE, vol. 66, no. 11, pp. 1307-1313, Nov. 1978.

[4] Roberts, Flow Rate Management – Anagran Inc., Sept. 2008.

[5] N. M. Khalilzad, F. Yekeh, L. Asplund and M. Pordel, "FPGA implementation of real-time Ethernet communication using RMII interface," 2011 IEEE 3rd International Conference on Communication Software and Networks, Xi'an, 2011, pp. 35-39.

[6] D'Amour, Michael R., Chief Operating Officer, DRC Computer Corporation. "Standard Reconfigurable Computing". Invited speaker at the University of Delaware, February 28, 2007.

[7] R. Rex, F. Mietke, W. Rehm, C. Raisch and H. n. Nguyen, "Improving Communication Performance on InfiniBand by Using Efficient Data Placement Strategies," 2006 IEEE International Conference on Cluster Computing, Barcelona, 2006, pp. 1-7.

[8] D. Haule and A. S. Malowany, "High-speed 2-D Hardware Convolution Architecture Based on VLSI Systolic Arrays", IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, p 52-55, Jun 1989.

[9] Lily Yu and Jennifer Liscom, Gartner, Inc. and/or its Affiliates. All Rights Reserved, TELC-WW-DA-0154, April 2003.

[10] L. D. Circiumarescu, G. Predusca, N. Angelescu and D. C. Puchianu, "A New Approach to Improve Performance of Routing Protocols for Voice, Video Conferencing and FTP Services," 2017 21st International Conference on Control Systems and Computer Science (CSCS), Bucharest, 2017, pp. 392-398.

[11] Y. Zhang, N. Ansari, M. Wu and H. Yu, "On Wide Area Network Optimization," in IEEE Communications Surveys & Tutorials, vol. 14, no. 4, pp. 1090-1113, Fourth Quarter 2012.

[12] S. L. Mathison, L. G. Roberts and P. M. Walker, "The history of telenet and the commercialization of packet switching in the U.S.," in IEEE Communications Magazine, vol. 50, no. 5, pp. 28-45, May 2012.

[13] A. Nakanishi, K. Hatayama, T. Onoduka and T. Kimura, "An embedded system of TCP/IP communication by using FPGA," 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), Osaka, 2015, pp. 489-492.

★★★