



FocalPoint Hash Efficiency

Effective Flow-based Load Balancing

White Paper

April, 2009

Overview

Hash functions are used to distribute flows across multiple links in a LAG group or in a multi-stage fabric. Hash functions vary in both their load distribution efficiency and in their silicon implementation cost. FocalPoint devices use a modified Pearson's hash that is highly effective in load distribution while incurring a modest implementation cost. Various parts of the L2/3/4 header can be used as a source for the hashing function. The source is hashed to a 12-bit value (giving 4096 intermediate bins) and the result is distributed among the output links using modulo division. For purposes of load distribution, two hash values can be calculated from the header fields in each frame:

- Layer 3/4 Hash (36 bits)
- Layer 2/3/4 Hash (48 bits)

The keys to these hash functions are constructed in a configurable manner in order to provide the following features:

- Symmetry -- Hash value remains the same when source and destination fields are swapped.
- Static field dependence -- Support for including a specific set of header fields in the hash function.
- Dynamic field dependence, based on frame type -- Certain fields can be omitted or included when a frame is IPv4/IPv6.

The simulation results presented in this paper prove why designing a hash function is tricky. Some hash functions are not good when the sources are not that random.

The FocalPoint hash function is effectively an enhanced CRC, which is less susceptible to repeated bit patterns (we say "enhanced CRC" because typical CRCs can be less effective in some special rare flow relationships). The FocalPoint hash function was developed using mathematical analysis and was validated in silicon. These results showed that various bit patterns are working as expected.

LAG Simulation Results

LAG hashing is typically used to distribute flows across multiple egress ports in a LAG group. The basic operation of LAG hashing is to take flow related data (like SMAC, DMAC, SIP, DIP, L4SRC, L4DST), compute a hash key, and then compute the modulo of that key to distribute flows to the ports in the LAG by their index as shown in the equation below.

$$\text{LAG-Index} = \text{hash}(\text{smac}, \text{dmac}, \text{sip}, \text{dip}, \text{l4src}, \text{l4dst}) \% \text{num-links};$$

Hash functions are mathematical expressions. The quality of the distribution will depend on the quality of the hash function and the number of flows, along with correlation between the flows. When comparing FocalPoint's hash efficiency to other hash algorithms, it is important to understand that with every mathematical function, there will be corner cases that will favor one algorithm over another. The key objective of an efficient algorithm is to have the best hash distribution for all traffic conditions rather than selective corner case scenarios.

To illustrate this point, the FocalPoint hash algorithm was simulated under various scenarios and compared to a commonly used XOR hash algorithm. The two algorithms were tested under several different corners cases using a 4-port LAG group. The numbers represent the bandwidth percentage used in each port for fixed packet sizes.

The first two cases show examples where both hash functions perform about the same. They also show that if you have a very limited set of flows, the distribution will be very unbalanced, which is to be expected.

Case 1: 8 flows using random SMACs and random DMACs

Link	XOR Hash	Fulcrum Hash
0	12.50%	25.00%
1	25.00%	12.50%
2	25.00%	12.50%
3	37.50%	50.00%

Case 2: 1000 flows using random SMACs and random DMACs

Link	XOR Hash	Fulcrum Hash
0	22.60%	25.10%
1	24.50%	24.70%
2	25.90%	23.70%
3	27.00%	26.50%

The next two cases are based on flows with linear DMAC addresses. What is interesting here is that the XOR function hashes perfectly. This is because with the XOR function, the lower bits have the same effect as an index. This is a corner case showing that flows can be selected to make perfect use of a given hash function.

Case 3: 8 flows using random SMACs and linear DMACs from a random seed

Link	XOR Hash	Fulcrum Hash
0	25.00%	50.00%
1	25.00%	25.00%
2	25.00%	12.50%
3	25.00%	12.50%

White Paper: FocalPoint Hash Efficiency

Case 4: 1000 flows using random SMACs and linear DMACs from a random seed

Link	XOR Hash	Fulcrum Hash
0	25.00%	24.40%
1	25.00%	25.70%
2	25.00%	24.90%
3	25.00%	25.00%

The next two cases show that the XOR function can be ineffective, no matter how many flows in the LAG group. Scenarios like this can occur in real systems. For example, assume that all servers have dual-port NIC cards, and that one port has MAC X and second port as MAC X+1. If only first port is connected, addresses jumps by 2 from one server to the next.

Case 5: 8 flows using random SMACs and linear DMACs in steps of 2 from a random seed

Link	XOR Hash	Fulcrum Hash
0	50.00%	37.50%
1	0.00%	25.00%
2	50.00%	12.50%
3	0.00%	25.00%

Case 6: 1000 flows using random SMACs and linear DMACs in steps of 2 from a random seed

Link	XOR Hash	Fulcrum Hash
0	50.00%	24.20%
1	0.00%	24.90%
2	50.00%	25.80%
3	0.00%	25.10%

In all of the following cases, parts of the IP header are used in the hash calculation. When we include IP addresses in the hash equation, there are cases where the XOR function breaks down while the FocalPoint hash algorithm stays robust across all the scenarios.

Case 7: 1000 flows using random SIP/SMAC and random DIP/DMAC

Link	XOR Hash	Fulcrum Hash
0	24.50%	24.10%
1	25.20%	26.90%
2	25.80%	23.70%
3	24.50%	25.30%

Case 8: 1000 flows using fixed SIP/SMAC and random DIP/DMAC

Link	XOR Hash	Fulcrum Hash
0	24.80%	24.00%
1	26.80%	24.60%
2	23.90%	27.40%
3	24.50%	24.00%

Case 9: 1000 flows using fix SMAC/SIP/DMAC and incremented DIP

Link	XOR Hash	Fulcrum Hash
0	25.90%	29.00%
1	25.20%	22.80%
2	24.70%	24.80%
3	24.20%	23.40%

Case 10: 1000 flows using incremented DMAC and incremented DIP

Link	XOR Hash	Fulcrum Hash
0	50.00%	24.70%
1	0.00%	24.90%
2	50.00%	25.60%
3	0.00%	25.80%

White Paper: FocalPoint Hash Efficiency

Case 11: 1000 flows using random SIP/SMAC

Link	XOR Hash	Fulcrum Hash
0	25.20%	23.10%
1	25.60%	23.90%
2	26.00%	26.00%
3	23.20%	27.00%

Case 12: 1000 flows using fixed SMAC and incremented SIP

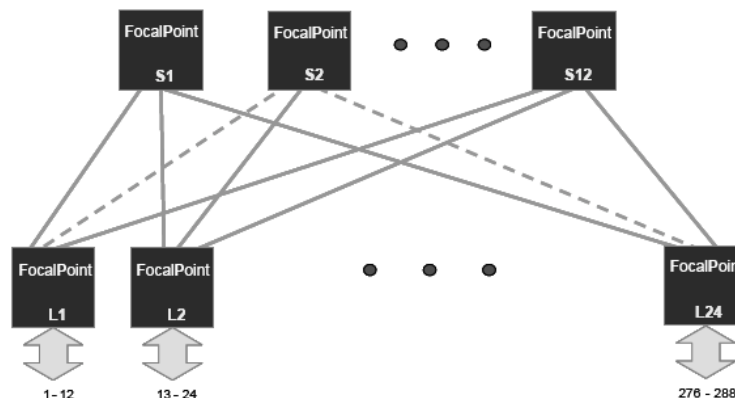
Link	XOR Hash	Fulcrum Hash
0	25.00%	23.50%
1	25.00%	21.30%
2	25.00%	26.40%
3	25.00%	28.80%

Case 13: 1000 flows using incremented SMAC and incremented SIP

Link	XOR Hash	Fulcrum Hash
0	0.00%	25.60%
1	100.00%	25.90%
2	0.00%	24.00%
3	0.00%	24.50%

Fat Tree Simulation Results

The FocalPoint hash function can be used to distribute flows across spine switches in a fat tree fabric configuration as shown in the figure below.



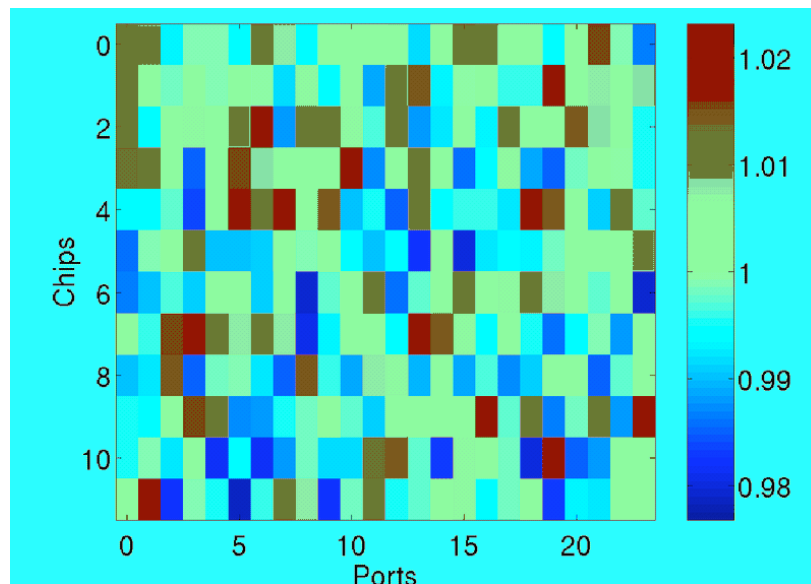
Here, multiple line switches shown across the bottom distribute traffic across multiple spine switches shown across the top.

Simulation results presented here highlight the performance of the hash function by using highly stressful traffic patterns in a large system with 288 ports across 12 line switches. The simulation was created using a byte-accurate switch simulator and is cycle-based, with one byte of data being transferred per cycle.

The switch topology and data traffic patterns were simulated as follows:

- 288-port system composed of 24-port 10GE switches.
- 9216 MAC addresses connected to the 288 ports (P1-P288), 32 per port, randomly assigned
- Mesh traffic pattern:
 - 1st round: P1 sends to P2, P2 sends to P3...P288 sends to P1
 - 2nd round: P1 sends to P3, P2 sends to P4...P288 sends to P2
 - Repeated 287 times or more so every port transmits to every other port
- Each round produces 288 ports x 287 frames – 82656 total frames.
- Frame size is 4KB

The load distribution simulation results are shown in figure 4 below. In this simulation, the total number of frames directed to each of the 288 ingress line ports from 9216 random MAC addresses during the whole simulation is counted. For good statistics, the cycle of mesh transmissions was repeated so that 50 frames were transmitted from each port to every other port. The system port utilization rates are irrelevant in this simulation. The loads are normalized so that differences in loading can be easily interpreted in terms of percent. The result is an extremely even load across the 288 line switch ports (12 line switches, 24 ports each).



Conclusion

Hash functions are important for distributing flows across multiple links in a LAG group or in a multi-stage fabric. Traditional hash functions use an XOR to calculate the hash value. Although there are cases where this provides good uniformity, there are also several common cases where this function is inefficient. FocalPoint devices contain a modified Person's hash algorithm that provides much more uniform results over a wide variety of traffic conditions.

Fulcrum Microsystems, Inc.
26630 Agoura Road
Calabasas, CA 91302
818.871.8100
www.fulcrummicro.com