IMPLEMENTATION AND PERFORMANCE EVALUATION OF PACKET FORWARDING BASED ON VS ROUTING SCHEME ON THE NETWORK PROCESSOR PLATFORM

Darshan V. Mujumdar, Himanshu P. Shringarpure and Girish P. Saraph
Department of Electrical Engineering
Indian Institute of Technology Bombay,
Powai, Mumbai 400076, INDIA.

A highly scalable dynamic routing algorithm can be helpful in reducing congestion hot-spots, balancing network loads, fast failure recovery and improve resource utilization. Many dynamic routing schemes have been put forward in the recent past. The main issues in a dynamic routing algorithm are the scalability and the computational complexity involved taking into account the real time network conditions, the operational overhead and the stability of the algorithm. Along with this, the algorithm should also preserve the key properties of conventional routing algorithms like loop free routing, distributed route calculations, statistical multiplexing and consistency. Here we are studying a dynamic Virtual Space (VS) Routing algorithm proposed in [1] and implementing it on the network processor platform. The main goal of the VS routing algorithm is to reduce the amount of information required for routing, thus making it more scalable and dynamic.

Traditionally, the implementation of routing algorithm is divided into control and forwarding planes. The control plane is responsible for the signaling, processing and updating routing information for taking routing decisions. Forwarding plane is assigned the task of per packet processing and routing table look-ups. For large networks under dynamic conditions, the control plane operations require heavy processing as compared to forwarding operations. For a highly dynamic and scalable algorithm, the control plane functionality needs to be simplified.

Under VS algorithm, the topological information of the network is expressed in a highly concise manner using the VS embedding scheme. The details of embedding the topological information into multidimensional VS configuration are given in [1]. The embedding scheme provides a directivity property to the multidimensional virtual space configuration, which enables simple geometric routing. The network topology information changes on a slower timescale and can be treated as quasi-static. Whereas, the link condition (or failure) and traffic loading can be treated as dynamic information. The quasi-static information is stored in concise form of VS configuration and is passed on to each node as VS coordinates. This information need not be exchanged between the nodes again. The dynamic information of link state updates and failures is expressed as link and node costs and is updated at fixed intervals. The dynamic information is exchanged locally for fast convergence and quick adaptation. Both the quasi-static and dynamic costs are combined into a single unified variable of *total cost* which is used for deciding the next hop.

The forwarding functionality can be implemented in two ways, through *fast* path and through *slow* path. In fast path, a hashing table is maintained for known data streams to enable fast packet processing. The hashing table is updated at fixed interval of 50ms so that the dynamic performance of the algorithm is maintained. While in slow path, calculations are performed for per-packet basis to find the best possible next-hop at that instant. We have implemented the *slow* path on the IXP1200 network processor from Intel. The aim is to evaluate its performance and determine what data rates it can support on multiple ports.

The processor provides a strongARM core processor to implement Control plane functionality and six microengines for forwarding plane operations. The control plane operations include the VS embedding, assignment of VS coordinates to nodes, calculations of dynamic link costs and exchanging this information. The forwarding plane is implemented on the IXP microengines. It includes per packet routing decisions based on VS algorithm. Thus the forwarding plane functionality differs from the traditional scheme. The information about the VS configuration required for routing decision at each node is minimal and is stored in the SDRAM memory provided by IXP. Thus, this information is shared, and can be accessed by all microengines. This gives the flexibility to assign any number of microengines for the required operation. IXP1200 provides multiple input-output ports and variable data rates can be achieved on each port. When a particular thread of a microengine is accessing memory, the next thread can continue with servicing the next packet. This hides the latency required for memory transfers and allows for higher data rates at the ports. Thus, IXP platform is a good choice for implementation of VS routing algorithm.

We implemented the VS functionality in a VS.c file and linked it to the microengine performing the receive operation. VS header was defined which contained the destination VS coordinates. After the packet is received, the VS header is extracted and given to the VS code. The VS algorithm requires calculating different vectors based on the VS coordinates, to be used in geometric routing [1]. For this calculation we needed floating point arithmetic functions. But IXP does not support floating point operations. We devised these microblocks using fixed point operations. We have implemented efficient computational algorithms of multiplication, division and square-root functions using Shift-Add techniques.

VS packets having fixed size of 64 bytes were generated using the simulator and performance was measured for 500000 clock cycles. Currently we have only implemented the forwarding plane of VS routing, so the routing information required is statically inserted into memory. This information is used by the VS forwarding plane for the cost calculations. In order to judge the throughput, simulator was run in the unbounded mode. The first implementation yielded a throughput of about 25 to 30 Mbps. Further attempts to increase the throughput were done by assigning more number of microengines to the VS code. This gave a linear rise in the throughput. Significant improvement in the performance was achieved by using different codes for 8-bit, 16-bit and 32-bit multiplication. Efficient and highly optimized algorithms *mul_8( )*, *mul_16( )* were developed along with *mul_32( )*. This implementation raised the throughput close to 100Mbps.

The size of packets was varied from 64 to 125, 250, 500 1000, and 1500 and the throughput was measured for each case. It was observed that as the packet size increases, the performance of the code improved and kept linearly increasing. This is because, as the packet size increases, the number of times the VS computations are done reduces for the same amount of data transferred across the network processor. At packet sizes of 64 bytes, 500 bytes and 1500 bytes throughput of around 100Mbps, 150Mbps and 300Mbps was achieved respectively. The above mentioned packet sizes span the spectrum of practical packet size distributions. So the above result can be considered as an average throughput of 175 Mbps for a typical variable packet size data flow.

In order to benchmark the performance, we did throughput analysis on the ethernet bridging code given in [2]. This code yielded a throughput of around 400Mbps. We have implemented the VS functionality using the slow path. On using the fast path with hashing function, the throughput can be expected to exceed 300Mbps, and thus comparable with any other packet routing scheme.

We are continuing to carry out detailed performance analysis of VS routing algorithm. The control plane functionality is being implemented on the StrongARM core. The performance analysis can be extended to the IXP2400 network processor. We intend to further show the dynamic adaptability of VS routing scheme on various network topologies.

References:

[1] G.P.Saraph and P.Singh, "New scheme for IP Routing and Traffic Engineering", *HPSR 2003 Conf., Torino, Italy, 2003.*

[2] E.Johnson and A.Kunze, "*IXP Programming – The Microengine coding guide for the Intel IXP1200 Network Processor family",* Intel Press.