Hardware Based Scalable Path Computation Engine for Multilayer Traffic Engineering in GMPLS networks

Sho Shimizu (1), Taku Kihara (1), Yutaka Arakawa (1), Naoki Yamanaka (1), and Kosuke Shiba (2)
1 : Keio University, 7-1 Shinkawasaki Saiwai-ku, Kawasaki, Japan
{shimizu, kihara, arakawa, yamanaka}@yamanaka.ics.keio.ac.jp
2 : IPFlex, Japan shiba@ipflex.com

Abstract
A parallel data-flow hardware based path computation engine that makes multilayer traffic engineering more scalable is proposed. The engine achieves 100 times faster than conventional path computation scheme.

Introduction
Multilayer traffic engineering or optical cut-through technique is an essential technique to utilize network resources efficiently in the next generation optical networks since the Internet traffic grows so rapidly compared with the growth of packet processing speed of a backbone router [1]. A lightpath is set up and torn down dynamically based on current network traffic condition, so path computation becomes a major issue in terms of efficient resource utilization.

There are two approaches to calculate an appropriate path. One is a per-layer basis approach and the other is a multilayer integrated approach. In terms of efficient resource utilization, the second approach is promising. However, optical cut-through creates many router network topology as shown in Fig. 1, so large computational time is needed since it requires solving a large number of combinational optimization problems where the number of nodes is large. Fig. 2 shows the execution time of multilayer traffic engineering is extremely larger than that of path computation in the conventional IP routing. In addition, QoS requirements from different clients need to be guaranteed. To cope with the complexity of path computation in multilayer traffic engineering, extremely scalable path computation mechanism is required.

In this paper, we propose a new path computation architecture that takes advantage of data-flow type parallel processing of dynamically reconfigurable processors. Our proposal shows high scalability compared with the conventional path computation scheme that uses Dijkstra’s algorithm.

Proposed approach
To achieve scalable path computation, we propose a new paradigm to calculate the shortest path. The proposed approach employs the hardware off-loading technique and takes advantage of parallel processing. The processor we use in this research is a data-flow type dynamically reconfigurable processor, which is different from popular sequential Neumann type processors, so we also propose a novel parallel shortest path algorithm called Multi-route Parallel Search Algorithm (MPSA), which is suitable for the processor we employ.

The basic idea of MPSA is that the shortest path is the first path reached to a node when we traverse all edges from the source node at the same pace. By assigning Processor Elements to calculation of each path search, all paths can be searched simultaneously.

MPSA can be expressed as matrix representation. Matrix representation is suitable for hardware implementation. In addition, matrix operations offer high parallelism since the operations of the elements are independent. For these characteristics, we express MPSA as matrix-based operations to...
implement it on DAPDNA-2 [2]. Details of the algorithm are described in [3].

Performance evaluation

Figure 3 shows the execution time of MPSA and Dijkstra’s algorithm, where the network topology is square-mesh. We assume the average cost is 3, and the reconfigurable processor has enough PEs and memories. Dijkstra’s algorithm is run on the PC whose processor is an Intel Pentium 4 3.0GHz, and which has 1024MB RAM. In the proposed algorithm, the execution time increases slowly as n increases because MPSA on a reconfigurable processor runs at $O(\sqrt{n})$. When $n=169$, 96.7% fewer time are used than with Dijkstra’s algorithm.

Figure 3: Execution time of MPSA and Dijkstra’s algorithm

Implementation and integration with software-based router

To validate our proposal, we implement MPSA on an actual dynamically reconfigurable processor, DAPDNA-2, which is developed by IPFlex Inc. The implementation works as a hardware off-loading engine to calculate a path. Figure 4 shows architecture of hardware off-loading and Figure 5 illustrates that the off-loading engine is plugged into a PCI slot of a PC. In this research, “application software” means routing daemon. We employ GNU Zebra, which is an open source as routing daemon. “Hardware” means DAPDNA-2 where we implement MPSA.

To be able to off-load the calculation, we modify Zebra version 0.94’s source code. We add the off-loading architecture in ospfd, which is a daemon processing OSPF included in Zebra. The modified ospfd usually collects link-state information, and it triggers the shortest path calculation on DAPDNA-2 when re-calculation is required. Before executing MPSA, link-state informations are transformed to the matrix-based data format as show used in MPSA. After executing MPSA, ospfd gets the result of the shortest paths from the hardware off-loading engine. The result is represented as matrix-based format, so ospfd transforms the result to Zebra’s internal data structure and makes a routing table.

Figure 4: Hardware off-loading architecture

Figure 5: Hardware off-loading engine is plugged into a PC and integrated with GNU Zebra

Conclusions

The computational complexity of path computation is a major issue to realize efficient multi-layer traffic engineering since it is a large-scale combinational optimization problem. We propose a new path computation approach that takes advantage of parallel processing of dynamically reconfigurable processor, and implement the hardware off-loading engine working with GNU Zebra software router. The evaluation shows our proposed approach is more scalable than Dijkstra’s algorithm, two orders of magnitude.

References


Acknowledgement

This work was supported in part by Global COE Program “High-Level Global Cooperation for Leading-Edge Platform on Access Spaces (C12)” and JSPS Grant-in-Aid for Scientific Research.