Distributed Online LSP Merging Algorithms for MPLS-TE

Li Lei and Srinivas Sampalli

Faculty of Computer Science, Dalhousie University
Halifax, NS B3H 1W5, Canada
{llei, srini}@cs.dal.ca

Abstract. Merging of Label Switched Paths (LSPs) saves label space and reduces processing time in routers. We introduce two distributed merging algorithms for online LSP merging.

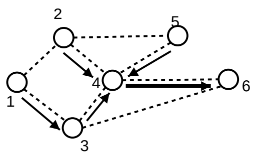

Fig. 1. LSPs merging example

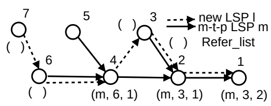

Fig. 2. Example of on the fly merging

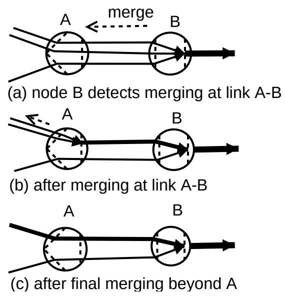

Fig. 3. Example of upstream wave merging

1 Introduction

As the size of the MPLS network [1] increases, the large label space becomes a big performance concern [2]. Labels can be saved by merging conventional point-to-point (p-t-p) LSPs to form Multipoint-to-Point (m-t-p) LSP trees [1], as shown in Figure 1.

The optimization of LSP merging problem is NP-hard [5]. Previous proposed merging schemes [4] [5] require a central control and global route information and suffer from performance degradation in online use [5]. In this paper, we describe two fully distributed online LSP merging algorithms.

2 Distributed LSP Merging Algorithms

We abstract the LSP control and management as general message passing processes. We also assume only local information is available at each route.

The *on the fly merging algorithm* requires two messages, *request* and *resv*. The procedure *REQUEST* collects merging information along the route, as shown in Figure 2. The procedure *RESV* assigns to the new LSP l the same label as that of selected LSP m rather than a new one. The normal label distribution process resumes after the $refnode$ specified in the reference entry. This algorithm rapidly merges a new LSP into an existing LSP but unable to merge all possible LSPs.

```
procedure REQUEST(reflist)
    for all entry i ∈ reflist do
        if port_out(i) = port_out(l) then
            hop(i) + +
        else
            remove entry i
        end if
    end for
    N ← {LSP m|port_out(m) = port_out(l) ∧ port_in(m) ≠ port_in(l)}
    for allLSP j ∈ N do
        if qos(j) = qos(l) then
            reflist ← reflist ∪ {j}, hop(j) ← 1, refnode(j) ← itself
        end if
    end for
end procedure
procedure RESV(reflist)
    if reflist ≠ ϕ then
        m ← reference LSP in reflist
        if refnode = itself then
            clear reflist
            assign l a new label
        else
            assign l the same lables as m
            modify the bandwidth reservation of m
        end if
    end if
end procedure
```

Algorithm 1: On the fly merging algorithm

The *upstream wave merging algorithm* detects and merges all possible LSPs starting from the egress nodes. It requires two messages *merge* and *release*. The procedure *DETECT* finds merging opportunities starting from the egress nodes. The procedures *REMAP* and *RELEASE* illustrate the merging operation, as shown in Figure 3.

```
procedure DETECT
    for every output port r do
        OUT ← {all outgoing labels to r}
        for every input port s do
            for every label i ∈ OUT do
                IN_i ← {all LSPs from s to outgoing label i}
            end for
            if(|IN_i| > 1) send message merge(IN_i) to node s
        end for
    end for
end procedure
procedure REMAP(LSP set M, message source node s )
    l ← LSP which has the minimal label in M
    for all LSP j ∈ M − {l} do
        label_out(j) ← label_out(l)
        bandwidth(l) ← bandwidth(l) + bandwidth(j)
    end for
    send message release(M) to s
end procedure
procedure RELEASE(LSP set M)
    remove NHLFEs for all LSP l ∈ M
end procedure
```

Algorithm 2: Upstream wave merging algorithm

3 Conclusion and Future Work

In this paper, we propose two distributed LSP merging algorithms for MPLS-TE. Currently, we are in the progress of simulating our algorithms. LSP merging may affect other fields of traffic engineering, such as preemption. Integration of tess algorithms with ours previously proposed preemption scheme [3] is a work for further study. Extending MPLS signaling protocols for LSPs merging also requires more attention.

References

1. E. Rosen, A. Viswanathan, and R. Callon, *Multiprotocol Label Switching Architecture*, IETF, RFC-3031, January 2001.
2. H. Hummel and J. Grimminger, *Hierarchical LSP*, IETF Internet Draft, draft-hummel-mpls-hierarchical-lsp-00.txt, March 2002, work in progress
3. L. Lei and S. Sampalli, *Backward connection preemption in multiclass QoS-aware networks*, Proceeding of 12th IEEE IWQOS, page(s):153-160, June 2004
4. H. Saito, Y. Miyao, and M. Yoshida, *Traffic Engineering using Multiple Multipoint-to-Point LSPs*, Proceeding of IEEE INFOCOM 2000, Page(s): 894-901 vol.2, March 2000.
5. S. Bhatnagar, S. Ganguly and B. Nath, *Creating Multipoint-to-point LSPs for Traffic Engineering*, Proceeding of IEEE HPSR Workshop 2003, page(s): 201-207, June 2003