

Internet Draft
Expiration Date: May, 1998
File name: [draft-zhang-ospf-dvl-01.txt](#)

Z. Zhang
Bay Networks
November, 1997

Fixing Backbone Partition with Dynamic Virtual Links

Zhaohui Zhang

Bay Networks, Inc.

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as ``work in progress.''

To learn the current status of any Internet-Draft, please check the ``lidl-abstracts.txt' listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

This document proposes a Dynamic Virtual Link (DVL) algorithm to dynamically detect and fix OSPF backbone area partition.

In the DVL algorithm, each ABR periodically checks if it can reach via backbone area all other ABRs that it can reach via its transit areas. If not, it knows that a backbone partition has occurred and a spanning tree of Dynamic Virtual Links are created. When the DVLs are not needed, they are automatically deleted.

Internet Draft

Dynamic Virtual Links

November, 1995

1. Introduction

Open Shortest Path First (OSPF) protocol requires that all Area Border Routers (ABRs) be connected to a backbone area and that the backbone area be contiguous. However, the protocol does not try to detect and fix partitioned areas.

Virtual links can be used to provide or enhance backbone connectivity. However, a lot of redundant virtual links have to be used to ENSURE backbone connectivity and they introduce a lot of overhead both in traffic and to the routers.

The author proposes that virtual links be made dynamic: they are created when they are needed and destroyed when they are not needed.

Supporting Dynamic Virtual Links (DVL) is optional.

1.1 Differences from previous version

1. The "spanning tree center" is no longer "elected" and "maintained". The "center" is now always the router with the highest ID.
2. To detect redundant DVLs, routers now run a special Dijkstra in the backbone area, using tie-breakers similar to those in MOSPF, instead of depending upon random timers to avoid thrashing.
3. A configurable area parameter is added to control whether DVLs are allowed through a transit area.

2. Dynamic Virtual Links

2.1 Detect Backbone Partition

Since each router maintains a routing table in each area for all ABRs in that area, the backbone partition can be easily detected: if an ABR discovers that it can reach another ABR via a transit area, but it can not reach the ABR via the backbone area, then it knows that there is a backbone partition and DVLs need to be created to fix the partition.

2.2 Spanning Tree of DVLs

It is not necessary that a DVL be created between each pair of ABRs that can not reach each other via the backbone area.

In Figure 1, the ABRs in a transit area are divided into disjoint groups (shown as blocks): ABRs (shown as "o" in the blocks) in a group can reach each other via the backbone area, but they can not reach ABRs outside the group via backbone. A star-shaped spanning

Zhang

Expires May, 1998

[page 2]

Internet Draft

Dynamic Virtual Links

November, 1995

tree (see 5.2 of [2]) that connects the groups is enough: a chosen ABR in a leaf group creates a DVL to a chosen ABR in the center group B if it cannot reach the chosen ABR in the center via the backbone, and deletes the DVL when it is not needed to reach the ABR in the center. The chosen ABR in the center is passive and dynamically creates a DVL when it receives the first Hello packet from a new virtual neighbor with lower ID.

The chosen ABR in a center group is hereafter called a center.

Previously disjoint groups merge into one group after adjacencies on their DVLs to the center have been established. However, this does not change anything. A group can also split, and a chosen ABR in each subgroup will have a DVL to the center.

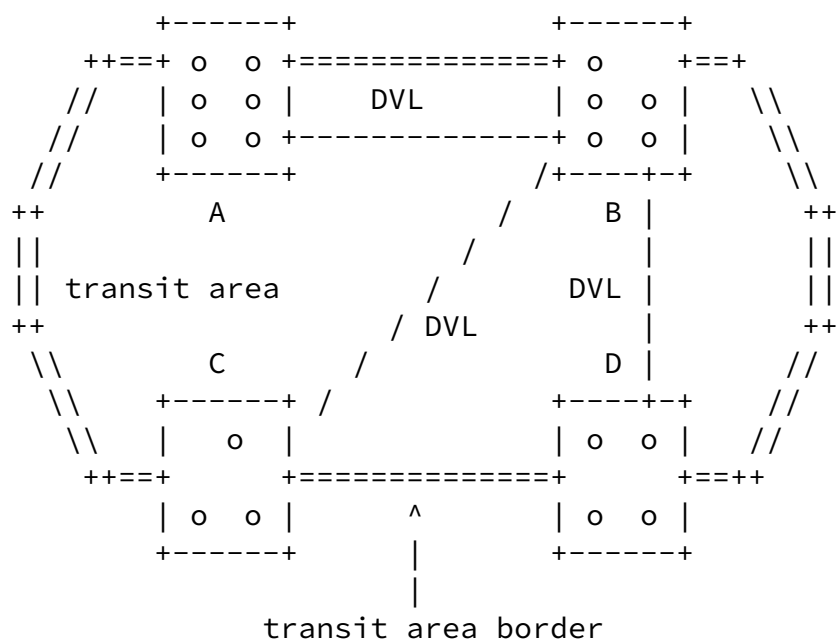


Figure 1. Disjoint groups of ABRs

The spanning tree needs not to be a Minimum Weight Spanning Tree (MST) (see 5.2.2 of [2]):

1. A distributed MST algorithm requires exchange of cost information among the ABRs, which can cause extra routing traffic and longer convergence time.
2. A MST needs to be reformed whenever there is a topology change in the transit area, which again leads to more traffic and longer convergence time.
3. Even if a nonoptimal spanning tree of virtual links is used, the route calculation will still result in shortest path trees for the ABRs (see 16.3 of [1]).

The spanning tree is formed per transit area, but some tie-breakers make sure that there will be no redundant DVLs formed or kept in all areas (see 3.2.2 & 3.3).

Zhang

Expires May, 1998

[page 3]

Internet Draft

Dynamic Virtual Links

November, 1997

The center could change when more ABRs join the transit area. It is always the router with the highest ID in a transit area. However, existing DVLs to old centers are not affected by a new center.

Note that if a DVL is already established for a group, when another ABR with higher Router ID joins the group, it does not create another DVL because it can reach the center via the existing DVL.

[2.3](#) Detect/delete Redundant DVLs

Some DVLs become redundant when the original backbone connection failures are fixed. Sometimes redundant DVLs could be created due to the distributed nature of the algorithm. Redundant DVLs should be deleted.

Note that the deleting of a redundant DVL is only initiated by the router that initiated the DVL when the router was a leaf.

To find out if a DVL initiated by the router is redundant, the router runs a Dijkstra in the backbone area, rooting the tree at the router with the highest ID of those reachable in the backbone area. Tie-breaks similar to those in MOSPF are used to ensure that all ABRs get the same shortest path trees rooting at the highest ID router, so that they agree which DVLs are redundant.

3. Implementation Details

3.1 A Functionality Bit -- D-bit

A new functionality bit, D-bit, is used in Router LSAs to indicate that a router supports DVL in the area, as shown in Figure 2.

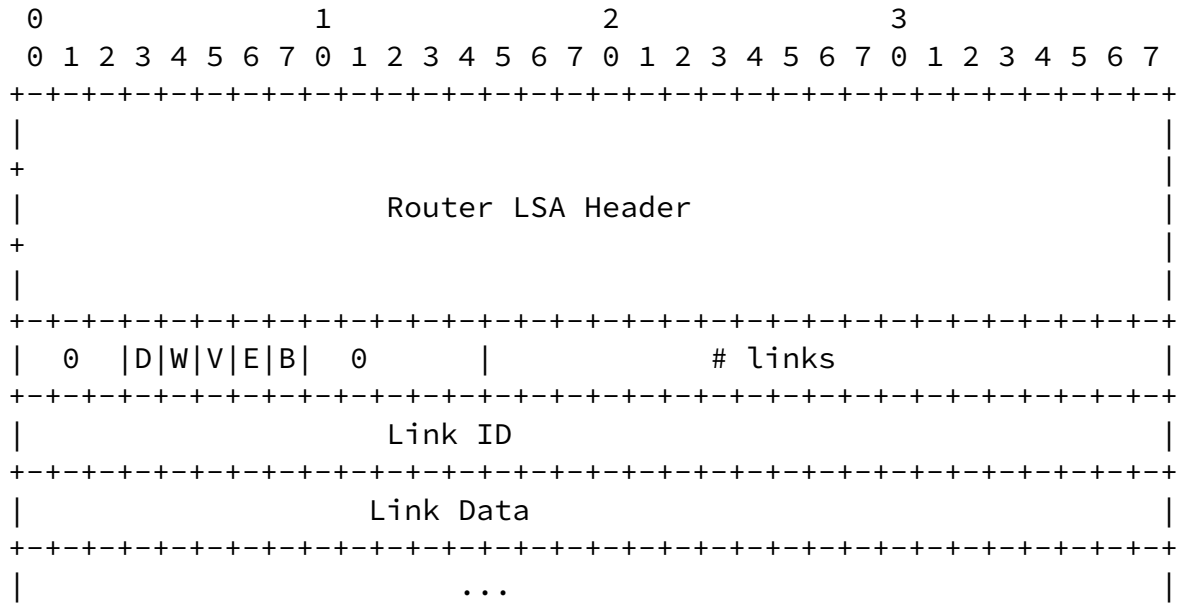


Figure 2. Functionality bits

3.2 Detect and Fix Possible Backbone Partition

The following procedure is taken by an ABR for a transit area whenever:

- a. there is a topology change (indicated by new Router/Net LSAs) in the backbone area.
 - b. there is a topology change in the transit area.
1. Find out current center C, which is the ABR with the highest ID of those with the D-bit set in their Router LSAs and reachable in the transit area.

Find out the ABR with the highest ID of those with the D-bit set in their Router LSAs and reachable via the backbone area, calling it L.

2. create a DVL to the center if all the following three conditions

are met:

- a. this router is L but not the current center C
- b. it can not reach the center C via the backbone
- c. there is not another transit area with a higher area ID, via which the center can be reached

[3.3](#) Detect/Delete Redundant DVLs

If a router has active DVLs (with neighbors fully adjacent over the DVLs) initiated by itself, the following procedure is taken in the backbone area when there is a topology change in the backbone area and there is no partition in the backbone area after the change.

1. Find out the router H with the highest ID of those reachable in the backbone area, whether the D-bit is set in their Router LSAs or not.
2. With H's Router LSA as the root, do a Dijkstra in the backbone area with the following special process:
 - a) Use tie-breakers similar to those in MOSPF (see step (2)(d) in [Section 16.1](#), [RFC 2178](#)).
 - b) When a vertex is moved from the candidate list to the SPF tree, if the link between its parent and itself is a DVL initiated by this router, mark it as "needed".
3. Delete those DVLs that are initiated by the router and not marked as "needed".

Because of the tie-breakers all DVL routers will build the same tree so they will agree on which DVLs are redundant. Note that DVLs that are not absolutely needed but do lead to optimal paths are not deleted.

Zhang Expires May, 1998

[page 5]

Internet Draft

Dynamic Virtual Links

November, 1997

Before a DVL is deleted, if the associated neighbor is in state N2WAY or higher, the router should send a Hello packet with no listed neighbor over the DVL, so that the neighbor can immediately drop its adjacency with this router and delete the DVL from its end.

[4.](#) Discussions

[4.1](#) Performance

The backbone partition detecting/fixing procedure is taken for a transit area only when there is a topology change in the backbone area or in the transit area; the procedure for detecting/deleting redundant DVLs is taken only for the backbone area when there are DVLs initiated by this router and there is a topology change in the backbone area and there is no backbone partition after the change.

Therefore, the algorithm does not introduce much extra load.

[4.2](#) Spanning tree center/leaf determination

The algorithm relies on the routers that have the highest IDs of all ABRs in a transit area or of all ABRs reachable via backbone. If each ABR, instead of only the one with the highest ID of all ABRs reachable via backbone, creates a DVL to the center, there will be some redundant DVLs being created and then deleted, causing extra traffic and routing oscillation.

[5.](#) Configurable Area Parameters

DVLEnabled

Control whether DVLs are allowed through this transit area.

DVLRxmtInterval

DVLInfTransDelay

DVLHelloInterval

DVLRouterDeadInterval

DVLAuType

DVLAAuthKey

[6.](#) Acknowledgement

Special thanks goes to John Moy. This document and related work could not have been finished without help from him.

In fact, John had his DVL idea in his mind too when the author's was brought up to him. He also pointed out some holes and gave some suggestions, such as on spanning tree, eliminating center election in version 0, and the logic for deleting DVLs, which are very important in the DVL algorithm.

A special thanks also goes to Rob Coltun, whose OSPF simulator made it possible to implement and test the DVL algorithm in draft version 0.

The initial work and versions of this document were finished while the author was working in the IP/Routing Consortium, InterOperability Laboratory (<http://www.iol.unh.edu>), University of New Hampshire.

7. Author's Address

Zhaohui Zhang
Bay Networks, Inc.
600 Technology Park
Billerica, MA 01821

Email: zzhang@baynetworks.com

Phone: (978) 916-8225

Fax: (978) 670-8760

Reference

- [1] John Moy, "OSPF Version 2", [RFC2178](#), July 1997
- [2] Dimitri Bertsekas and Robert Gallager, "Data Networks", Prentice-Hall, Inc., 1992.
- [3] John Moy, "Multicast Extension to OSPF", [RFC 1584](#), March 1994.

This document expires in May, 1998.