

Avoiding Instability during Graceful Shutdown of OSPF

Aman Shaikh, Rohit Dube, Anujan Varma

Abstract—Many recent router architectures decouple the routing engine from the forwarding engine, so that packet forwarding can continue even when the routing process is not active. This opens up the possibility of using the forwarding capability of a router even when its routing process is down, thus avoiding the route flaps that normally occur when the routing process goes down. Unfortunately, current routing protocols, such as BGP, OSPF and IS-IS do not support this behavior. In this paper, we describe an enhancement to OSPF, called the *IBB (I'll Be Back) capability*, that enables other routers to use a router whose OSPF process is inactive for forwarding traffic for a certain period of time. The IBB capability can be used for avoiding route flaps that occur when the OSPF process is brought down in a router to facilitate protocol software upgrade, operating system upgrade, router ID change, AS and interface renumbering, etc.

When the OSPF process in an IBB-capable router is inactive, it cannot adapt its forwarding table to reflect changes in network topology. This can lead to routing loops and/or black holes. We provide a detailed analysis of how and when loops or black holes are formed and propose solutions to prevent them. Using the GateD platform, we have developed an IBB extension to OSPF incorporating these solutions. Using this system in an experimental setup, we demonstrate that the overhead of the IBB extension is modest compared to the benefit it offers, and has good scaling behavior in terms of network size and the number of routers with inactive OSPF processes.

I. INTRODUCTION

While data networks have been increasing in size and complexity over the last few years, little attention has been paid to mechanisms that could ease software maintenance on the switches and routers that make up these networks. The current practice for maintenance tasks such as software upgrades is to withdraw the router undergoing maintenance from forwarding service. To minimize the impact of the withdrawal, network operators perform upgrades at a time when traffic is low, and stagger upgrades of different routers. Nevertheless, upgrades cause at least some disruption in the forwarding service provided by the network. With society's increasing reliance on data networks, even minor disruptions in the forwarding service may be unacceptable in the future. Moreover, upgrades can also lead to network outages in extreme cases. There have been several cases of serious network outages in recent years, lasting for several days in some cases. Two of these outages are known to have been triggered while performing software upgrades on the routers in the network [1]. Since routing software is frequently upgraded due to bug-fixes and enhancements, it is of great interest to providers and vendors to make these upgrades as non-intrusive and non-disruptive as possible.

Aman Shaikh is at the University of California, Santa Cruz, CA 95064 E-mail: aman@cse.ucsc.edu.

Rohit Dube is with Xeboo Communications, Inc, South Plainfield, NJ 07080 E-mail: rohit@xeboo.com.

Anujan Varma is at the University of California, Santa Cruz, CA 95064 E-mail: varma@cse.ucsc.edu.

In most of the current-generation core routers, the routing engine (software) is decoupled from the forwarding engine (hardware). Thus, one can potentially continue using these routers for forwarding packets even when its routing software is made temporarily inactive for maintenance. Supporting such a behavior in a router requires (i) allowing use of the forwarding engine even when its routing engine is inactive, and (ii) developing extensions to the routing protocol to avoid routing loops and black holes resulting from changes in the network topology that the inactive router is incapable of responding to. Current routing protocols such as BGP [2], OSPF [3], [4] and IS-IS [5] do not support this capability. The current definitions of these protocols implicitly assume that when the routing process of a router goes down, the router is incapable of forwarding packets, forcing other routers to avoid the inactive router for forwarding. To avoid this behavior, the protocol must refrain from re-routing around the router undergoing upgrade at least for a period of time. Several extensions have recently been proposed in the IETF to add such a capability to routing protocols [6], [7]. These proposals are known by various names such as *IBB (I'll Be Back)*, *graceful restart* and *hitless restart*; we will use the term IBB in general to refer to all such proposals.

In this paper, we focus on an IBB proposal for OSPF protocol. The aim of the IBB proposal is to enable a router undergoing software maintenance to forward packets for a certain period of time even if its OSPF process is down. Before going down, the OSPF process on the router informs other routers that it is temporarily being shut down, but its forwarding engine remains active. The process also specifies a time interval within which it expects to be back in operation. We will call this interval the *IBB Timeout period*. Other routers can continue using the router for forwarding packets during the IBB Timeout period. Hence, if the routing process returns to operation within the IBB Timeout period, the scheme avoids the route flaps and instability normally associated with such a disruption. Other routers stop using the affected router for forwarding only when the routing process in the router remains inactive after the IBB Timeout period.

A key issue with IBB-like proposals is how to handle topological changes while a router's OSPF process is inactive. Since the process is inactive, it cannot update its forwarding table to reflect these changes. Under these circumstances, continued use of the router for forwarding can lead to problems such as loops or black holes. Recently, Moy has proposed an IBB-like proposal called *hitless restart* for OSPF [7]. This proposal takes a conservative approach for handling topological changes, by withdrawing the inactive router from forwarding service as soon as the first topological change is detected by its neighbors. Thus, in the case when the topology does not change within the upgrade interval, the *hitless restart* proposal avoids the with-

drawal completely. However, in a large network running OSPF, changes in network topology occur often, limiting the usefulness of the proposal.

Our IBB scheme is based on the observation that not every topological change leads to forwarding loops or black holes even when one of the routers is unable to modify its forwarding table in response to the change. Therefore, it is possible to continue forwarding packets through a router whose routing process is inactive even after a topological change if we ensure that loops or black holes do not form. Thus, our IBB scheme avoids packet forwarding to a destination through such an inactive router only when using the router as the next hop to the specific destination leads to a routing loop or black hole. In this paper, we systematically analyze how loops and black holes are formed when a router does not respond to topological changes, and based on that propose algorithms to avoid them as a part of our IBB scheme. Using the GateD platform, we have developed a prototype implementation of the IBB scheme and used it to evaluate the overhead of our algorithms. The results show that the computational overhead of the algorithms is modest, and that the overhead scales well in terms of the network size and the number of inactive routers.

The paper is organized as follows: In Section II, we provide a brief overview of OSPF. In Sections III and IV, we present the analysis and procedure for avoiding loops and black holes while one or more of the routers are inactive. In Section V, we describe the prototype implementation of IBB extensions developed using the GateD OSPF implementation. Section VI presents the performance analysis of the loop and black-hole avoidance algorithms through experiments we carried out using the prototype. Section VII comments on the deployment of our IBB scheme. Finally in Section VIII, we present our conclusions and provide directions for future work.

II. OSPF OVERVIEW

OSPF is a widely used intra-domain routing protocol [4]. It is a link-state routing protocol, meaning that each router within the AS discovers and builds an entire view of the network topology. This topology view is conceptually a directed graph. Each router represents a vertex in this topology graph, and each link between neighbor routers represents a unidirectional edge. Each link also has an associated weight that is administratively assigned in the configuration file of the router. Using the weighted topology graph, each router computes a shortest path tree with itself as the root, and applies the results to build its forwarding table. This assures that packets are forwarded along the shortest paths in terms of link weights to their destinations [4]. We will refer to the computation of the shortest path tree as an *SPF (Shortest Path First) computation*, and the resultant tree as a *Shortest-Path Tree (SPT)*.

The OSPF topology can be divided into areas determining a two level hierarchy. Area 0, known as the *backbone area*, resides at the top level of the hierarchy and provides connectivity to the non-backbone areas (numbered 1, 2, ...). OSPF assigns each link to exactly one area. The routers that have links to multiple areas are called *border routers*. A router maintains the complete topology graph of each area in which it has links. The router does not get to know about the entire topology of remote

areas, but knows about the total weight of the path from one or more border routers of its areas to each node in a remote area. OSPF allows routing information to be imported from other routing protocols like BGP. The router that imports routing information from other protocols into OSPF is called an *AS Border Router (ASBR)*.

OSPF performs SPF calculation in three stages. In the first stage, it calculates routes to intra-area destinations by computing SPTs for the topology graph of each area. This stage is termed as the *intra-area stage*. In the second stage, it calculates routes to each remote node by selecting an appropriate border router as an intermediate node based on the path weight information. This stage is termed as the *inter-area stage*. In the final stage, it calculates routes to each external node by selecting an appropriate ASBR as an intermediate node. This stage is termed as the *external stage*.

Every router running OSPF is responsible for describing its local connectivity in a *Link-State Advertisement (LSA)*. These LSAs are *flooded* reliably to other routers in the network, which allows them to build the view of the topology. The set of LSAs in a router's memory is called a *link-state database* and conceptually forms the topology graph for the router.

III. HANDLING TOPOLOGICAL CHANGES: SINGLE AREA CASE

When router R goes down¹, its forwarding table gets frozen. Thus, if a topological change occurs while R is inactive, it cannot update its forwarding table to respond to the change in the topology. This can lead to problems like loops and black holes. Figure 1 illustrates how a topological change occurring while R is inactive can lead to a loop.

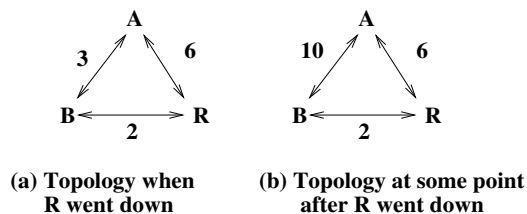


Fig. 1. An example of a loop being formed while R is inactive.

In the figure, A , B and R are routers. As can be seen from the link weights in Figure 1(a), R 's shortest path to A goes via B . Assume that R goes down at this point, thereby freezing its database image and forwarding table. After a while, the weight of the $A \leftrightarrow B$ link increases (Figure 1(b)). Under normal circumstances, B could have started using R as the next hop for reaching A because R would have started using the direct link between itself and A to reach A . Since R 's forwarding table is frozen, however, a loop is formed if B tries to use R as the next hop for reaching A . To avoid the loop, when router R is inactive, other routers such as B cannot calculate their forwarding tables as they do normally; they need to keep in mind that R 's forwarding table is frozen, and make sure no loops or black holes are formed.

¹From now on, when we state that a router is going down or is inactive, we mean only its OSPF process is being shut down, unless specified otherwise.

To formulate the problem, we assume that the entire OSPF domain is a single area in this section; in the next section we will deal with multiple areas. Let us pick an arbitrary router S as a source and D as a destination. Assume that S sends a packet to D . We will denote the actual path traversed by a packet as $P_{S \rightarrow D}$. Path $P_{S \rightarrow D}$ has a *loop* if it has a cycle. Note that if path $P_{S \rightarrow D}$ has a loop, it never terminates. On the other hand, if $P_{S \rightarrow D}$ terminates at a node other than D , we say it has a *black hole*. Note that node S also has a path to D in its SPT². Packets may or may not actually travel along this hypothetical path, but the path can never have a loop or a black hole. We will denote this hypothetical path by $P'_{S \rightarrow D}$.

before proceeding further, we note the following properties of SPTs:

- 1) If all the nodes calculate their SPTs from the same image of the graph, $P_{S \rightarrow D}$ is same as $P'_{S \rightarrow D}$.
- 2) If all the nodes calculate their SPTs from the same image of the graph, and if $P'_{S \rightarrow D}$ exists (in S 's SPT), $P_{S \rightarrow D}$ cannot have a loop or a black hole.

A. Single Inactive Router

We start with the simplest case of a single inactive router at a given time. We will denote this inactive router by R . Moreover, we assume that all the routers other than R are using the same image of the topology graph for calculating their SPTs and forwarding tables. R 's SPT and forwarding table are based on the graph it had at the time it went down. Let us suppose that R is using Y as the next hop for reaching D . We first analyze when $P_{S \rightarrow D}$ has a loop and how to avoid it. Then we address the black hole problem.

1) *Loop Formation:* Since all the routers except R compute their SPTs from the same image of the topology, router R must be part of any loop that gets formed. Proposition 1 formalizes this.

Proposition 1: If $P_{S \rightarrow D}$ has a loop, R must lie on this loop. Moreover, R must be a part of the cycle.

Proof: The proof is by contradiction. Assume R does not lie on $P_{S \rightarrow D}$. Therefore, all the nodes on $P_{S \rightarrow D}$ have the same image of the graph. Hence, by the virtue of property 2 of the SPT, $P_{S \rightarrow D}$ cannot have a loop. This is a contradiction. Hence, R must lie on $P_{S \rightarrow D}$. Now assume that R lies on the path, but is not part of the cycle. Then, $P_{S \rightarrow D}$ is of the form $\{S, \dots, R, \dots, C_1, C_2, \dots, C_n, C_1, C_2, \dots, C_n, \dots\}$. Therefore, $P_{C_1 \rightarrow D} = \{C_1, \dots, C_n, \dots\}$. This means that path $P_{C_1 \rightarrow D}$ has a loop. But all the routers along $P_{C_1 \rightarrow D}$ have the same image of the graph since R does not lie on it. This violates property 2 of the SPT and hence is a contradiction. Therefore, $P_{C_1 \rightarrow D}$ must have R on it. ■

The next proposition proves that if a packet sent from S to D reaches R , then S must have R on the path to D in its SPT.

Proposition 2: If R lies on $P_{S \rightarrow D}$, then R must lie on $P'_{S \rightarrow D}$.

Proof: The proof is again by contradiction. Assume R does not lie on $P'_{S \rightarrow D}$. Thus, every router on $P'_{S \rightarrow D}$ has the same image of the graph. Therefore, by virtue of property 1 of

² S may have more than one shortest paths to D . For ease of exposition, we will assume that only one such path exists.

SPT, $P_{S \rightarrow D}$ should not have R on it, which is a contradiction. Hence, R must lie on $P'_{S \rightarrow D}$. ■

We now prove that if path $P_{S \rightarrow D}$ has a loop, router Y which is the next hop used by R for D , will find R on its path to D when it calculates its SPT.

Proposition 3: If path $P_{S \rightarrow D}$ has a loop, R must lie on $P'_{Y \rightarrow D}$.

Proof: Since R uses Y as the next hop for reaching D , $P_{S \rightarrow D} = \{S, \dots, R, Y, \dots\}$. Since $P_{S \rightarrow D}$ has a loop, by virtue of Proposition 1, $P_{S \rightarrow D} = \{S, \dots, R, Y, \dots, R, Y, \dots, \dots\}$. Thus, R lies on the path $P_{Y \rightarrow D}$. Hence, by virtue of Proposition 2, R must lie on $P'_{Y \rightarrow D}$. ■

The implication of Proposition 3 is that if a loop exists, Y will detect it once it determines that R lies on the path to D in its SPT. The following proposition proves that if Y determines that R lies on the path to D in its SPT, it means that a loop exists for every router that has R on the path to D .

Proposition 4: If R lies on $P'_{Y \rightarrow D}$ and $P'_{S \rightarrow D}$, $P_{S \rightarrow D}$ has a loop.

Proof: Let us say $P'_{Y \rightarrow D} = \{Y, \dots, R, X, \dots, D\}$. By virtue of property 1 of the SPT, a packet from Y will follow $P'_{Y \rightarrow D}$ until it reaches R ; R will then forward the packet to Y since R uses Y as the next hop for reaching D . Therefore, $P_{Y \rightarrow D} = \{Y, \dots, R, Y, \dots, R, \dots\}$. Thus, $P_{Y \rightarrow D}$ has a loop. Since R lies on $P'_{S \rightarrow D}$, by property 1 of SPT, a packet from S will reach R , and then Y . In other words, $P_{S \rightarrow D} = \{S, \dots, R, Y, \dots, R, Y, \dots\}$. Hence, $P_{S \rightarrow D}$ has a loop. ■

The essence of Propositions 3 and 4 is that any loop for destination D is detected if R lies on Y 's path to D in the SPT of Y . Avoiding the loop is straightforward. Every router (other than R) has to calculate an alternate path to D such that R does not lie on it. A router can achieve this by removing R and its incident edges from the graph, and by calculating the shortest path to D using the modified graph. If every router selects its next hop in this manner, the two properties of the SPT guarantee that $P_{S \rightarrow D}$ cannot have a loop.

2) *Black Hole Formation:* Path $P_{S \rightarrow D}$ can have a black hole if the topology graph is not fully connected, and S and D do not belong to the same connected components of the graph. In such a case, D cannot exist in S 's SPT. Thus, if D is part of S 's SPT, it means S and D belong to the same connected component of the graph. As a result, if $P_{S \rightarrow D}$ has a black hole, it must be due to R . Proposition 5 given below proves this result. Before we proceed, we assume that we have already performed loop detection and avoidance. Thus, R 's next hop Y either has a loop-free path to D or does not have a path to D in its SPT.

Proposition 5: If a black hole exists on $P_{S \rightarrow D}$, R must be part of $P_{S \rightarrow D}$. Moreover, R or Y must be the last node on $P_{S \rightarrow D}$.

Proof: The proof is by contradiction. Assume R does not lie on $P_{S \rightarrow D}$. Then since every router other than R calculates its path from the same image, there cannot be a black hole on $P_{S \rightarrow D}$ by property 2 of the SPT. Hence, R must lie on $P_{S \rightarrow D}$. Now assume that R or Y is not the last node. Then, $P_{S \rightarrow D} = \{S, \dots, R, Y, \dots, A\}$. Thus, $P_{Y \rightarrow D} = \{Y, \dots, A\}$. Since all the nodes on $P_{Y \rightarrow D}$ have the same image of the graph, $P'_{Y \rightarrow D} = \{Y, \dots, A\}$. This is not possible since Y either has

- 1) For each node D in the SPT, perform the following steps:
 - a) For each of the minimum weight paths the router has to reach D , perform the following steps:
 - i) If the path does not go through any of the R_i s, skip the next step.
 - ii) The path goes through some R_i s. For each such R_i , “cancel” this path if either of the following holds true.
 - Some router (this includes the local router also) has sent an $avoid(R_i, D, L)$ message.
 - The router checks the next hops used by R_i for reaching D and at least one of them turns out to be the local router.
 If the path was “canceled” due to the second condition above, originate an $avoid(R_i, D, L)$, provided no other router has done it so far.
 - b) If all the paths to D in the router’s SPT were “canceled” above, mark D . Marking of D means that the router needs to find an alternate path to D that does not have any of the n inactive routers on it (this is done in step 2 below).
- 2) If at least one such D was marked in the previous step, calculate a new SPT with all the R_i s ($1 \leq i \leq n$) removed.
- 3) For each unmarked destination, determine the next hop(s) from the first SPT (the one with all R_i s in it). For each marked destination, determine the next hop(s) from the second SPT (the one with all R_i s removed).

Fig. 2. Procedure for loop avoidance. We assume that R_1, R_2, \dots, R_n denote the inactive routers.

a loop-free path to D or does not have D in its SPT. Therefore, the path $P_{Y \rightarrow D}$ must end with R or Y . ■

Path $P_{S \rightarrow D}$ can end with R or Y under the following circumstances:

- 1) R ’s forwarding engine is not functioning.
- 2) D does not exist in R ’s forwarding table.
- 3) The link between R and Y is not functional.
- 4) Y ’s forwarding engine is not functioning.
- 5) Y does not have a loop-free path to D in its SPT.

Thus, if a black hole exists on $P_{S \rightarrow D}$, at least one of these conditions must hold. Moreover, it is easy to see that if at least one of these conditions hold, and if R is part of $P_{S \rightarrow D}$, there will be a black hole on $P_{S \rightarrow D}$. To avoid the black hole, every router must calculate an alternate path to D such that R does not lie on it.

3) *Procedure for Loop and Black Hole Avoidance:* We observed in Section III-A.1 that $P_{S \rightarrow D}$ has a loop if and only if R is part of $P_{Y \rightarrow D}$. If a router such as S has to detect loops for every destination D on its own, it needs to know the next hop used by R for D when it went down. When this condition is iterated over all possible D , the implication is that S has to know R ’s entire forwarding table at the time it went down. One way S can learn about R ’s forwarding table is to construct R ’s forwarding table based on its local image of the graph at the time it receives R ’s message. This approach may not produce a correct version of R ’s forwarding table at S . The other approach is to have R send its forwarding table to S before it goes down. This approach does not scale well. Instead, we propose the following approach which produces a correct version of R ’s forwarding table and is scalable.

R sends its forwarding table to its neighbors. Thus, if R was using Y as the next hop to reach D , Y gets to know R ’s forwarding table and the fact that R is using it as the next hop for D . During the IBB Timeout period of R , if Y detects that R exists on the path to D in its SPT, it recalculates its path to D such that R no longer remains on the path. Moreover, it also sends a message to other routers asking them not to use R for reaching D . We will represent this message as an $avoid(R, D, L)$ message which means router R is to be avoided on the path to D because of a possible loop. A router recalculates its path to D upon receiving the avoid message if its current path to D has R on it, and ignores the message otherwise. Note that this pro-

cedure is independently applied to every destination in the area. Essentially, Y — the next hop used by R for D — becomes *in-charge* of detecting loops involving R and D and informing other routers about it. Generalizing this, every neighbor of R becomes *in-charge* of detecting loops for all those destinations for which R uses the neighbor as the next hop.

In a similar fashion, neighbors become in charge of detecting the black hole cases we identified in Section III-A.2. For the third and the fifth case, the next hop used by R (Y) becomes in charge of detecting the black hole. For the second case, all the neighbors become in charge of detecting the case since R does not have a next hop in this case. In the fourth case, we can assume that Y disappears from the topology graph since the failure of the forwarding engine results in the failure of the OSPF adjacencies Y has. This means that one of the other neighbors of R becomes in charge of detecting the case. Finally, the first case where R ’s forwarding becomes non-functional during its IBB shutdown is difficult to detect. It requires an out-of-band support in the form of an operator withdrawing R from forwarding service.

Once a neighbor detects that using an inactive router R leads to a black hole for D , it originates an $avoid(R, D, BH)$ message which means that the receiving router should avoid R for reaching D because of a possible black hole. The receiving router then selects next hops for D from the SPT which does not have R on it as it does in the case of a loop. In other words, the receiving router follows the same procedure whether it receives an $avoid(R, D, L)$ or an $avoid(R, D, BH)$ message.

B. Multiple Inactive Routers

So far we have assumed that R is the only inactive router and all the other routers have the same image of the database. We now relax this assumption and allow more than one router, say n of them, to be inactive at the same time. Let us denote these n routers by R_1, R_2, \dots, R_n . As in the single router case, we can have the next hop used by a router R_i to the destination D in charge of making sure that using R_i to reach D does not create a loop or a black hole. If the router in charge detects a loop or a black hole, it asks other routers not to use R_i for reaching D . Other routers then calculate a new path to D by removing R_i from the path if it is part of the path. Note that with n inactive routers, other routers can get messages asking

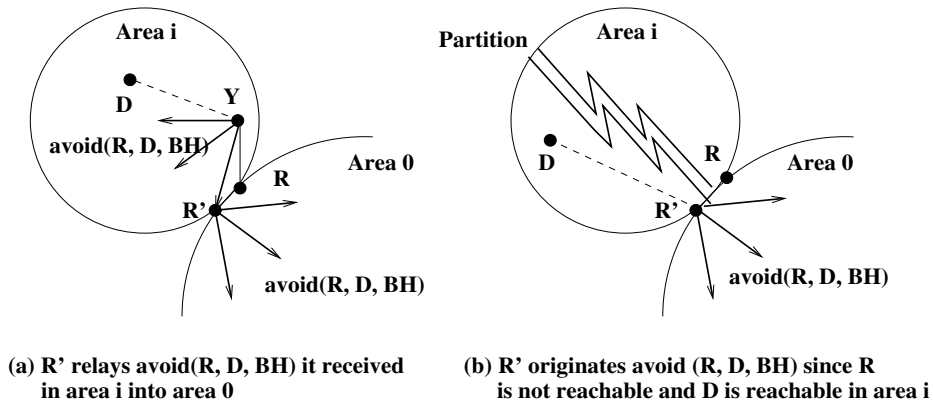


Fig. 3. Cases where border router R' originates an $avoid(R, D, BH)$ message into area 0.

- 1) R divides its forwarding table into n parts with each part describing the entries whose next hops belong to one particular area. R then sends each part to its neighbors in that area.
- 2) The neighbors of R in each area generate $avoid$ messages as and when required as described in Section III.
- 3) If an ABR R' that shares an area i with R sees an $avoid(R, D, BH)$ message for a destination D in area i , it originates an $avoid(R, D, BH)$ message in all those areas where R had originated a Summary LSA for D . Moreover if R' cannot reach R in area i , it generates an $avoid(R, D, BH)$ for all those D it can reach in those areas where R had originated a Summary LSA for D .

Fig. 4. Procedure that needs to be followed when a border router R goes down. We assume that R was connected to n areas when it went down.

them not to use j ($1 \leq j \leq n$) of these n routers for reaching D . In that case, these routers can calculate a new path to D without the j forbidden R_i s on it. Unfortunately, this approach can lead to a combinatorial explosion because of the possibility that the set of forbidden routers can be different for different destinations.

To overcome this combinatorial explosion, we propose the following. When a router receives a message asking it to avoid one or more R_i s on its path to D , the router attempts to calculate a path to D that does not have *any* of the n inactive routers, not just the forbidden ones. In order to calculate this path, the router simply calculates a new SPT which does not have any of the n routers on it. A big advantage of this is that the router can use the same SPT (that is, the one without any of the n inactive routers in it) for finding paths to all those destinations for which it needs to find a path without *some* R_i s on it. In other words, the router first calculates its SPT with all the inactive routers in it. After that for each destination D , the router marks the destination if the path to the destination has at least one forbidden router on it. For all the unmarked D s, the router selects next hops from the SPT it calculated. For all the marked D s, the router computes a second SPT ignoring all the inactive routers and selects next hops from this newly calculated SPT.

One problem with the simplified approach is that some routers may use the first SPT (one with all the inactive routers in it) for a destination D because they may not have any forbidden routers on their paths to D . On the other hand, some other routers may use the second SPT for D since they may have at least one forbidden router on their paths. The question is whether this disparity in how routers select next hops for D can itself lead to loops or black holes. Proposition 6 in the expanded version of this paper [8] proves that this disparity does not lead to loops or black holes.

Figure 2 summarizes the procedure each router has to follow

after calculating its SPT in order to avoid loops when one or more of the routers are inactive.

IV. HANDLING TOPOLOGICAL CHANGES WITH MULTIPLE AREAS

The procedure developed in Section III for loop and black-hole avoidance assumed that the entire OSPF domain is a single area. Since OSPF allows a domain to be divided into multiple areas, we will extend the procedure to cover the case of multiple areas in this section. For simplicity, we will concentrate on a single inactive router R here. If router R belongs to a single area, the procedures developed in Section III are sufficient to solve the loop and black-hole problems, provided that they are applied to the inter-area and external stages of the SPF calculation to take care of loops or black holes involving remote nodes.

Next, consider the case where R belongs to multiple areas, that is, R is a border router. Hence, it must be connected to area 0 (backbone area) [3] and at least one other area. Let i be one of the non-backbone areas that R is connected to. Since OSPF employs hierarchical routing between nodes belonging to different areas [3], loops do not get formed due to R . On the other hand, new cases that lead to black holes can arise if R is a border router.

To illustrate, let us focus on a destination D that belongs to area i . Assume that R knew how to reach D when it went down. As a result, routers outside area i may select R as an intermediate node to reach D . If a black hole is created for D due to R subsequently, routers in area i resolve it by calculating alternate paths to D such that R does not lie on them (Section III-A.2). For routers outside area i , if R is the only border router between area 0 and i , no alternate path to D exists. As a result, packets go into a black hole, which is unavoidable under any scheme.

LSA Name	Purpose	Opaque information part
IBB Cease LSA	To inform other routers that the originating router is going to be shutdown	IBB Timeout
Path Info LSA	To inform neighbors about originating router's forwarding table	Set of destinations and next hops used for them
Avoid Router LSA	To ask routers not to use an inactive router for a set of destinations	Router ID of the inactive router and a set of destinations
ICB (I Came Back) Open LSA	To inform other routers that the originating router is back	Router ID before the router went down

TABLE I
THE FOUR TYPES OF LSAs USED FOR IBB-RELATED COMMUNICATION.

Note that routers outside area i would have known about this problem if they received an $avoid(R, D, BH)$ message; however, since the $avoid$ message is originated by a router in area i , only R can propagate it into area 0 which is not possible when R 's OSPF process is inactive.

If another router R' exists between areas 0 and i , it can relay the $avoid(R, D, BH)$ message it receives in area i into area 0 (see Figure 3(a)). In that case, remote routers can use R' as an intermediate node to reach D provided R' has a path to D in area i . Additionally, R' has to originate an $avoid(R, D, BH)$ message in another case which takes care of area i partition. Specifically, let us say R' calculates its SPT for area i and realizes that it cannot reach R . This suggests a partition of area i with R and R' in different partitions. R' then should make sure that all the routers in area 0 use it as an intermediate node for destinations in its partition. It does so by originating $avoid$ messages for these destinations. Thus, if D is reachable from R' , R' concludes that D is in its partition of the area i and hence cannot be reached from R . As a result, R' then originates an $avoid(R, D, BH)$ message into area 0 as shown in Figure 3(b).

Figure 4 summarizes the key steps of the procedure that needs to be followed when R is a border router connected to n areas.

V. PROTOTYPE IMPLEMENTATION

Based on the concepts described in the previous sections, we have developed a prototype implementation of an IBB extension to the OSPF protocol. We used GateD (Gate Daemon) [9], a popular, public-domain routing software platform³ for developing the prototype. Adding the IBB capability increased the code size by a relatively small amount, about 4%.

The enhancements we made to the GateD code to introduce the IBB capability can be categorized into four distinct parts:

- 1) IBB-related communication.
- 2) Mechanisms to signal IBB shutdown and returning from shutdown.
- 3) Storing information about inactive routers.
- 4) Implementation of the procedure in Figure 2 for the SPF calculation.

We describe each of these parts briefly.

A. IBB-related Communication

The prototype adds four new types of Opaque LSAs [10] for IBB-related communication. These LSAs are defined in Table I.

B. Shutdown and Returning from the Shutdown

When the prototype is shutdown in IBB mode, it has to inform other routers by flooding an IBB Cease LSA. The LSA contains the IBB Timeout value in its Opaque information part. After originating the IBB Cease LSA, the prototype sends its forwarding table in one or more Path Info LSAs to its neighbors.

When the prototype returns back to operation, the procedure it follows depends on whether the IBB Timeout interval has expired or not. In the former case, it follows the normal start-up procedure followed by GateD. If the IBB Timeout interval has not expired, however, it originates an ICB Open LSA to announce that it is operational again. The prototype can originate other types of LSAs (including its Router LSA) only after it has originated the ICB Open LSA.

C. Storing Information about Inactive Routers

The prototype stores information about each inactive router in an *ibb-info* block. If R is an inactive router, the associated *ibb-info* block stores the router ID of R , the IBB Timeout value, its forwarding table and the set of destinations for which it needs to be avoided. The block remains active until IBB Timeout occurs even if an ICB Open LSA is received from R before the timeout. While the block is active, the prototype retains all LSAs originated by R even if they need to be deleted because of their age reaching MaxAge [3].

D. Modifications to the SPF Calculation

We have modified the SPF calculation part of GateD code to implement the procedure in Figure 2. These modifications are activated only if there is at least one inactive router, ensuring that no overhead is incurred in the normal case. At present, the procedure of Figure 2 is applied only to the intra-area stage of SPF calculation. As suggested in Section IV, the procedure should be applied to the inter-area and external stages also; we believe that there are no fundamental difficulties in modifying the inter-area and external stages. Moreover, the prototype currently deals only with loop problem; we are working on extending the prototype to cover black-hole cases.

³GateD consortium has now become a company called NextHop Technologies and is the commercial provider of GateD.

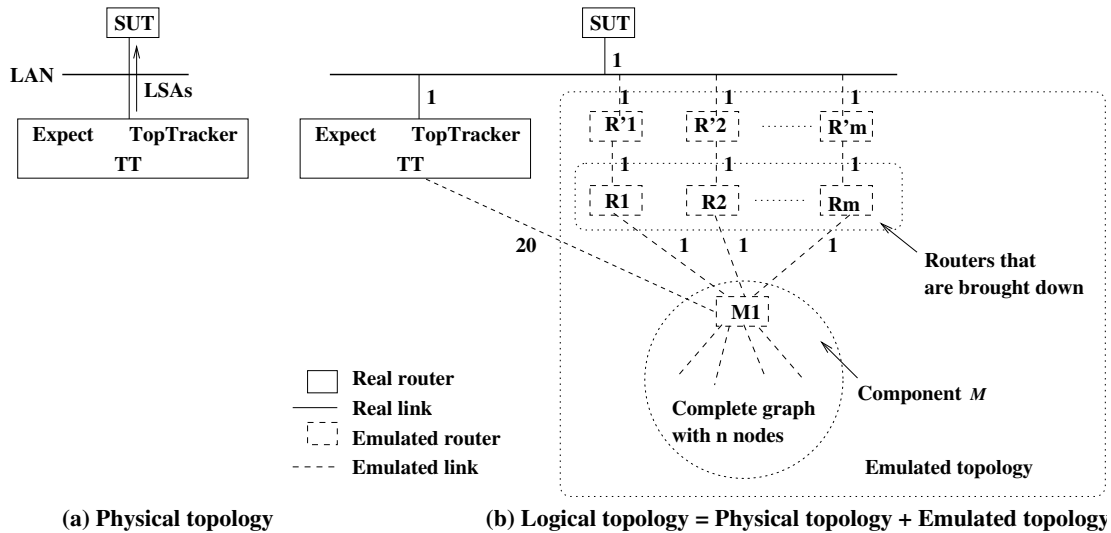


Fig. 5. Testbed used for performance evaluation.

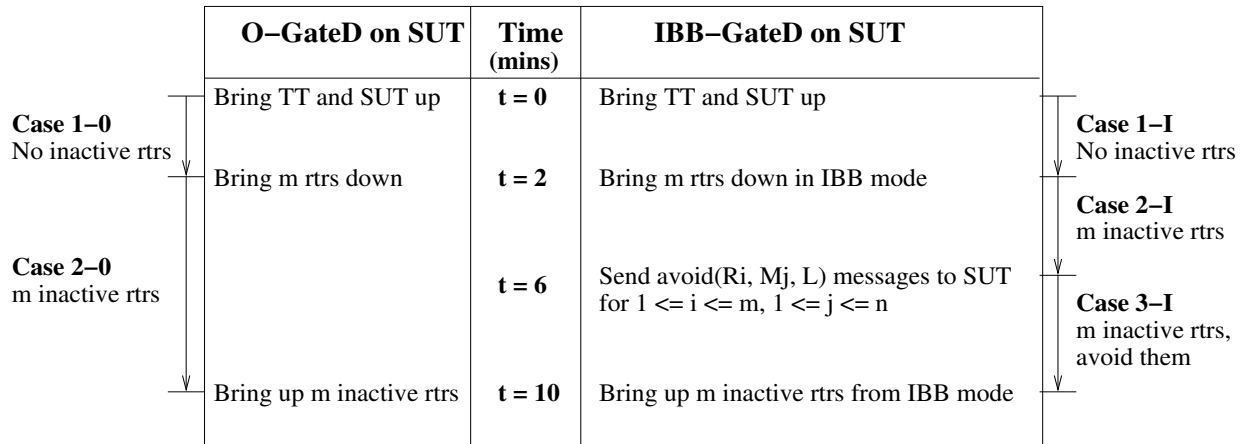


Fig. 6. Sequence of events performed with O-GateD and IBB-GateD running on the SUT to measure SPF calculation time for the 5 cases of interest.

Based on the procedure of Figure 2, the prototype keeps track of the inactive routers on every path leading to a destination D . Moreover, the prototype implements the second pass of the procedure which involves calculating the SPT using a graph which has all the inactive routers and incident edges removed from it.

VI. PERFORMANCE EVALUATION

In this section, we evaluate the performance of our IBB scheme, with focus on the overhead on SPF calculation due to the procedure of Figure 2. We perform experiments to measure the overhead, and study how it scales with the size of the topology and the number of inactive routers.

A. Setup and Methodology

To measure the overhead on the SPF calculation introduced by our IBB scheme, we measured the mean SPF time for GateD with and without our IBB extensions. In this section, we will refer to the GateD with IBB capability as IBB-Gated (the prototype we described in Section V), and the original unmodified GateD as O-GateD. We will use the term GateD generically to

refer to both. We instrumented both IBB-GateD and O-GateD to measure the time it takes for each SPF calculation.

The testbed used for experiments is shown in Figure 5. The SUT (System Under Test) is the router undergoing measurements. The basic idea is to send various LSAs including IBB-related LSAs to the SUT, and measure the SPF time with and without the enhancements of Figure 2. Since we wanted to measure overhead for large topologies which are impossible to setup and manage in a lab, we developed an OSPF topology emulator, termed *TopTracker*, which runs on the machine termed as TT in Figure 5. TopTracker is derived from Moy's OSPF implementation and simulator [11]. Its main function is to emulate any desired OSPF topology and events related to the topology. It does so by generating required LSAs and flooding them to the SUT so that the SUT believes that the emulated topology actually exists. Figure 5(b) shows the logical topology used in the experiments. The logical topology is a combination of the physical topology shown in Figure 5(a) and the emulated topology generated by the TopTracker.

Although the emulated topology shown in Figure 5(b) is not a typical intra-area OSPF topology, it allowed us to measure the overhead and scaling behavior of our scheme very conveniently.

As Figure 5(b) shows, the emulated topology has a fully connected component M with n nodes in it. The cost of each link in M was a randomly-chosen integer between 1 and 10 (inclusive). The fully connected component M stressed the SPF calculation the most and allowed us to measure the performance in a worst-case scenario. The emulated topology also has m routers (R_i s) which we assume need to be brought down for maintenance. Under normal circumstances, i.e., when R_i s are active, the SUT has to choose paths passing through them for reaching all the destinations in M . The reason is that the cost of paths through R_i s is lower than the cost of the path through TT. In other words, the SUT has m paths to reach each router in component M , and each path has exactly one R_i on it. Once the R_i s go down, the behavior of the SUT depends on whether O-GateD is running on it or IBB-GateD. With O-GateD, once the R_i s go down, the SUT starts using the alternate path through TT to reach the routers in component M . On the other hand, with IBB-GateD, the SUT keeps using paths through R_i unless it receives an *avoid* message. At some point let us say Top-Tracker generates *avoid* messages asking the SUT to avoid each R_i for reaching all the n routers in component M . In this case, the SUT is forced to cancel all the m paths leading to every router in M , and instead use the path through TT. This forces the IBB-GateD running on the SUT to run the second pass of the Dijkstra algorithm for all the n routers of M . By varying n and m , we determine the scaling behavior of our enhancements of Figure 2.

Figure 6 shows the sequence of events we carried out with O-GateD and IBB-GateD to measure the SPF time under various circumstances. Figure 6 also identifies five different cases of interest for which we measured the mean SPF time. By comparing mean SPF times in case 1-I with that in case 1-O, we can evaluate how IBB-GateD performs in comparison of O-GateD when no routers are inactive. Similarly, comparison of cases 2-I and 3-I versus case 2-O allows us to compare the SPF performance of IBB-GateD in comparison of O-GateD when m routers are inactive. To achieve statistical accuracy, we collected several samples of SPF calculation time for each case by repeating the sequence in Figure 6 for every value of n and m . An Expect/Tcl [12] script running on TT controlled the events in each sequence.

The next section presents results we obtained when a Linux PC with a 550 MHz AMD-K6-2 CPU and 128 MB of RAM was used as the SUT.

B. Results

First, we compare SPF calculation time in the five cases of interest identified in Figure 6. Figure 7 shows how mean SPF time varies with the size of the fully connected component (n). The results are for m equal to 1; similar results were obtained for other values of m . As can be seen from Figure 7, the SPF computation incurs significant overhead due to the IBB enhancements when a router is inactive (cases 2-I and 3-I). This is because IBB-GateD employs two passes if at least one router is inactive even when none of the destinations actually requires the second pass. Moreover, the overhead increases after the *avoid messages* are sent to the SUT. This is due to the fact that with the topology change, IBB-GateD has to deal with the *avoid*

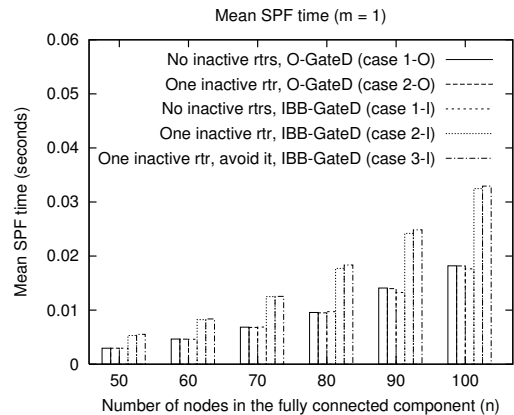


Fig. 7. Comparison of mean SPF calculation time for the 5 cases of interest.

messages when it cancels the path calculated at the end of the first pass for each destination in component M . It is also worth noting here that SPF computation in IBB-GateD incurs negligible overhead compared to O-GateD when all the R_i s are functioning normally.

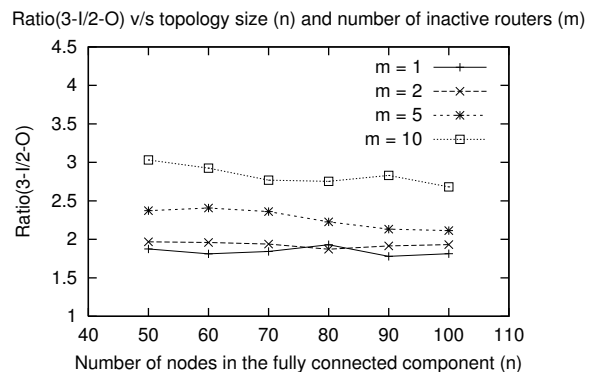


Fig. 8. Overhead of IBB extension on mean SPF time versus the number of nodes (n) in the fully connected component of the emulated topology.

Figure 7 shows that the IBB extension introduces maximum overhead to SPF computation under case 3-I. Therefore, we compute the ratio of mean SPF time in case 3-I to that in case 2-O and use it as a metric for characterizing the overhead of our scheme; we will denote this ratio as $R(3-I/2-O)$. Figure 8 shows how $R(3-I/2-O)$ varies with the number of nodes (n) in component M for four different values of m . The ratio $R(3-I/2-O)$ stays nearly constant as n increases for all the values of m , which demonstrates the scalability of our scheme in terms of network size. On the other hand, as m increases, $R(3-I/2-O)$ increases linearly. The reason is that as m increases, the number of paths canceled by IBB-GateD for each router in component M increases linearly. Each path contains one R_i on it. Therefore, canceling a path involves looking for the destination in the set of destinations for which R_i has to be avoided. Since this set is currently implemented as a linked list in IBB-GateD, the operation takes on the average $O(n)$ time per path per router in M . This overhead can be reduced if the set of to-be-avoided destinations is stored in a data structure which is optimized for searching, such as a Patricia tree.

Another factor that contributes to the increase is the way the logical topology is structured. Note that the mean SPF time under case 2-O has very little dependence on m since once the m routers are brought down, the remaining part of the topology becomes invariant under m . This factor also contributes to the increase in $R(3-I/2-O)$ as m increases. To illustrate this point, we also computed the ratio $R(3-I/1-O)$ since SPF calculation in case 1-O is equivalent to that in case 3-I minus the overhead of the steps in Figure 2. Figure 9 shows how $R(3-I/2-0)$ compares with $R(3-I/1-O)$ as m increases for n equal to 50; similar results were obtained for other values of n . The figure clearly demonstrates that part of the increase in ratio $R(3-I/2-0)$ is due to the difference in topology for cases 3-I and 2-O and has nothing to do with the additional steps of Figure 2.

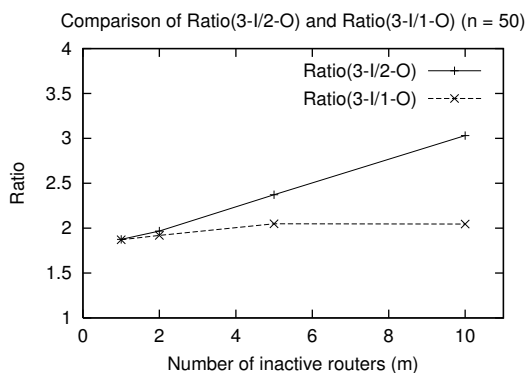


Fig. 9. Comparison of ratios $R(3-I/2-O)$ and $R(3-I/1-O)$ as m increases ($n = 50$).

The SPF calculation in case 3-I also incurs the overhead of the second pass of the procedure in Figure 2. This nearly doubles the mean SPF time for IBB-GateD (see the plot for $m = 1$ in Figure 8). The overhead of the second pass can be reduced by employing an incremental SPF algorithm [13], [14] for the second pass.

We have also performed experiments in which the SUT was actually responsible for originating the *avoid* messages. The results showed similar characteristics.

VII. DEPLOYMENT CONSIDERATIONS

In this section, we make a few general comments on the potential deployment of our IBB scheme in a real internetwork, and the procedure for loop and black-hole avoidance. One difficulty with our scheme is that it requires support from all the routers within the network, which makes its incremental deployment impossible. We are currently exploring ways of making the scheme incrementally deployable by limiting the support requirement to a small subset of routers.

Another deployment-related issue is that the overhead of our scheme may be difficult to justify in certain situations. One example of such a situation is when the shutdown period is small. In this situation, one can either stop using the inactive router upon a topological change as Moy's hitless restart proposal does [7], or continue using the router without being concerned about loops or black holes. In fact, a deployed version of IBB extension can provide all the three options to handle topological

changes. The operator can decide which option to use at the time of the shutdown which is then propagated to other routers along with the message that announces the shutdown of a router (e.g., the IBB Cease LSA). Furthermore, the deployed version can allow using the loop and black hole avoidance procedures selectively for certain destinations, since the procedures are applied to each destination independently. The advantage is that the overhead of the procedures is incurred only for the subset of destinations chosen. Finally, the deployed version should allow the operator to manually intervene and stop packet forwarding through the inactive router(s) in the event of unrecoverable errors or malfunctions.

VIII. CONCLUSIONS

In this paper, we introduced and evaluated an IBB extension to OSPF. The IBB capability helps avoid routing instabilities that arise when the OSPF process is brought down in a router for maintenance tasks such as hardware or software upgrade, configuration update, change of router ID, etc., provided the forwarding engine is still functioning. When a router is shut down in IBB mode, it informs other routers that its routing process is going to be inactive for a while and specifies time interval (IBB Timeout) within which it expects to be back. Other routers treat the originating router as capable of forwarding packets during that time period. If the routing process on the originating router returns to operation within the specified time period, the network behaves as if the process never went down with respect to packet forwarding.

Upon a topological change, the inactive router cannot update its link-state database and forwarding table. There are three alternatives to deal with the resulting consistency problem: do nothing, stop using the inactive router for forwarding, or keep using the inactive router but make sure no loops or black holes are formed. We have chosen the third approach, where we attempt to use the forwarding capability of the inactive router as much as possible, except in cases where it can lead to routing loops or black holes. Based on a systematic analysis of how loops and black holes are formed, we developed procedures for avoiding them. The key idea behind these procedures is to make the neighbors of the inactive routers in charge of detecting loops and black holes. The neighbors, in turn, ask other routers not to use the inactive router for such destinations. Although we have developed the procedures for OSPF, we believe that they are fairly general and can be applied to other link-state protocols, such as IS-IS.

We also developed a prototype implementation of the IBB extension using the GateD implementation of OSPF and used it to evaluate the overhead of the procedure for loop avoidance. Our results indicate that the procedure introduces only a modest overhead on the SPF calculation and exhibits good scaling behavior in terms of the network size and the number of inactive routers. In the future, we plan to incorporate the remaining procedures in our prototype and analyze their performance. We also plan to improve the efficiency of the current implementation in several ways, for example, by using an incremental algorithm for the second pass of the SPF computation.

ACKNOWLEDGMENT

We would like to thank Jennifer Rexford and Albert Greenberg for several helpful discussions, and feedback on the earlier draft of the paper. We are also grateful to the anonymous reviewers for their comments and feedback.

REFERENCES

- [1] North American Network Operators Group (NANOG), mailing list archives, <http://www.nanog.org>.
- [2] Y. Rekhter and T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC1771, Mar. 1995.
- [3] John T. Moy, "OSPF Version 2," RFC2328, Apr. 1998.
- [4] John T. Moy, *OSPF : Anatomy of an Internet Routing Protocol*, Addison-Wesley Publishing Company, Reading, Massachusetts, Jan. 1998.
- [5] R. Callon, "Use of OSI IS-IS for Routing in TCP/IP and Dual Environments," RFC1195, Dec. 1990.
- [6] Srihari Ramachandra, Yakov Rekhter, Rex Fernando, John Scudder and Enke Chen, "Graceful Restart Mechanism for BGP," Work in progress, Internet Draft, Jul. 2001.
- [7] John T. Moy, "Hitless OSPF Restart," Work in progress, Internet Draft, Aug. 2001.
- [8] A. Shaikh, R. Dube, and A. Varma, "Avoiding Instability During Graceful Shutdown of OSPF," Technical Report, University of California at Santa Cruz, Dept. of Computer Engineering, July 2001 (www.cse.ucsc.edu/research/hsnlab/publications.html).
- [9] "The GateDaemon (GateD) Project," Merit GateD Consortium, <http://www.gated.org>.
- [10] Rob Coltun, "The OSPF Opaque LSA Option," RFC2370, July 1998.
- [11] John T. Moy, *OSPF : Complete Implementation*, Addison-Wesley Publishing Company, Sept. 2000.
- [12] Don Libes, *Exploring Expect*, O'Reilly & Associates, Inc., Dec. 1994.
- [13] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Incremental Algorithms for Single-Source Shortest Path Trees," in *Proceedings of Foundations of Software Technology and Theoretical Computer Science*, Dec. 1994, pp. 113–224.
- [14] Paolo Narvaez, Kai-Yeung Siu, and Hong-Yi Tzeng, "New Dynamic SPT Algorithm based on a Ball-and-string Model," in *Proceedings of the IEEE Infocom*, 1999.