# Improving Convergence Speed and Scalability in OSPF: A Survey

Mukul Goyal, Mohd Soperi, Emmanuel Baccelli, Gagan Choudhury, Aman Shaikh, Hossein Hosseini, Kishor Trivedi

# Improving Convergence Speed and Scalability in OSPF: A Survey

M. Goyal, M. Soperi, E. Baccelli, G. Choudhury, A. Shaikh, H. Hosseini, K. Trivedi

***Abstract*—Open Shortest Path First (OSPF), a link state routing protocol, is a popular interior gateway protocol (IGP) in the Internet. Wide spread deployment and years of experience running the protocol have motivated continuous improvements in its operation as the nature and demands of the routing infrastructures have changed. Modern routing domains need to maintain a very high level of service availability. Hence, OSPF needs to achieve fast convergence to topology changes. Also, the ever-growing size of routing domains, and possible presence of wireless mobile adhoc network (MANET) components, requires highly scalable operation on part of OSPF to avoid routing instability. Recent years have seen significant efforts aimed at improving OSPF's convergence speed as well as scalability and extending OSPF to achieve seamless integration of mobile adhoc networks with conventional wired networks. In this paper, we present a comprehensive survey of these efforts.**

***Index Terms*—OSPF, Fast Convergence, Scalability, MANET.**

## I. Introduction

*Open Shortest Path First* (OSPF) [1], [2] is a popular *interior gateway* routing protocol. Such protocols provide routing functionality within a domain, which is generally, although not necessarily, contained within an *autonomous system* (AS) [3]. OSPF belongs to the category of *link state* routing protocols that generally require each router in the network to know about the complete network topology. However, for scalability reasons, OSPF allows the routing domain to be split into multiple *areas* and a router needs to know the complete topology of only those area(s) to which its interfaces belong. Link state routing protocols have been in use now for more than 30 years. The first major deployment dates back to 1978 when a link state protocol, called *SPF*, replaced a *distance vector* approach in ARPANET [4], [5]. The OSPF protocol has been in existence now for over 20 years[1]. Today, link state routing protocols, OSPF and *IS-IS* [6], are the most deployed interior gateway protocols.

Wide spread deployment and years of experience, hence high comfort level, running OSPF has motivated continuous improvements in its operation as the nature and *quality of service* (QoS) needs of the routing infrastructures [7] changed over time. During the initial years of its existence, OSPF's prime objective was to provide robust and scalable routing functionality. Limiting the processing/bandwidth requirements of the protocol was the prime concern and the time required to recover from a failure in the network topology (*speed of convergence*) was of secondary importance. In the event of a device failure in the network, the protocol required several tens of seconds to recover from the failure. During this transient state, the network service would suffer serious deterioration in quality or breakdown completely. With the advent of real-time applications on the Internet (e.g., *voice over IP* [8]) over the last decade or so, a service deterioration/breakdown extending several tens of seconds can no longer be tolerated. The desire for quick failure recovery motivated extensive research to improve OSPF's speed of convergence as well as to develop other *proactive* approaches to protect the network traffic in the interim. In this paper, we present a comprehensive survey of these efforts.

Fast convergence to topology changes has emerged as a critical requirement for today's routing infrastructures, however limiting the processing/bandwidth overhead of the routing protocol continues to be as important as before. OSPF, being a distributed protocol, requires timely execution of certain operations, e.g., generation and processing of *hello* packets, by the participating routers. It is absolutely essential to ensure that routers are not so overloaded that they repeatedly fail to execute these operations. Such failures may quickly snowball into a complete meltdown of routing functionality. To avoid CPU overloads, modern routers typically have a distributed architecture with central processors executing routing protocols and linecards handling packet forwarding. The processing overhead of the routing protocols typically grows with the size of the routing domains they cater to. For example, a router's OSPF-related processing overhead depends to a large extent on the size of the areas to which the router's interfaces belong and the size of the router's local neighborhood. Although router CPUs are more capable than ever before, increasing size and complexity of routing domains make CPU overload in routers a real possibility. In this paper, we also present a detailed survey of various recent proposals to optimize OSPF operations to reduce its processing requirements and thus improve its scalability.

Traditionally, OSPF has been a routing protocol for *wired* networks with largely static topology. However, nowadays routing infrastructures increasingly include *wireless* components as well. These components consist of either *static* wireless mesh devices, or *mobile* devices, potentially moving in and out of each other's radio range, or a mixture of both. An example of such network is a wireless, mobile ad hoc network (MANET) of vehicles where some vehicles have (wireless) connections to one or more traditional wired network(s) running the OSPF protocol. Although a number of routing protocols have been designed for MANETs [9], using a different routing protocol for the MANET components would require a complex exchange of the routing information between OSPF and this other protocol, which may not be

[1]The first OSPF specification (RFC 1131) was published in October 1989.

able to avoid path suboptimality. Thus, there is a strong motivation to extend the OSPF protocol to provide routing functionality in MANETs and to seamlessly integrate the wired and wireless components of a routing domain. This paper includes a survey of the different proposals to extend OSPF for operation on MANETs. These proposals essentially enhance OSPF's scalability characteristics to suite the peculiar requirements of mobile ad hoc networking. Some of these proposals may be applied to the wired networks as well and can significantly improve both the scalability as well as the convergence speed of traditional wired OSPF networks.

Figure 1 illustrates the main steps discussed in this paper regarding improving OSPF's convergence and scalability. The rest of the paper is organized as follows. Section II provides an overview of the convergence process. In the subsequent sections, we describe each step in the convergence process in detail and also discuss various proposals to optimize the operations during the step. Section III describes the failure detection mechanisms used in OSPF networks: default *hello* protocol based failure detection as well as the hardware based failure detection mechanisms available in some link-layer technologies. This section also describes *bidirectional forwarding detection* (BFD), which is a light weight protocol to quickly detect path faults between two networked devices.

Section IV describes the process of adjacency establishment between two OSPF routers and important enhancements proposed for this process. This section also describes the protocol enhancements that reduce the number of adjacency establishments required in broadcast/NBMA (*non-broadcast multi-access*) LANs and mobile ad hoc networks (MANETs). Section V begins with a description of the generation and flooding of *link state advertisements* (LSAs), packets that carry topology information. Subsequently, this section describes factors that affect the LSA generation/flooding process: configuration parameters/delays, mechanisms like *DoNotAge* LSAs and *subnet aggregation* and various enhancements designed to reduce the flooding overhead especially in MANET environment.

Section VI describes the process of calculating the routing table following the receipt of a new LSA, the mechanisms used to avoid frequent routing table calculations and the algorithms used to create *shortest path trees* during a routing table calculation. Section VII describes the *graceful restart* mechanism that allows a planned *control plane* reboot in a router to proceed without requiring network-wide dissemination of information about the reboot. Section VIII describes non-OSPF *proactive* approaches to fast failure recovery: *MPLS fast reroute* and *IP fast reroute*. Finally, Section IX concludes the paper.

## II. Convergence to a Topology Change in OSPF: An Overview

OSPF is a *link state* routing protocol. In a link state routing protocol, each router in a network needs to know the complete network topology. For scalability reasons, OSPF divides the routing domain it is serving into multiple *areas*. As shown in Fig. 2, the OSPF areas in a routing domain are arranged

| Faster Failure Detection |
|---|
| Faster and Fewer Adjacency Establishments |
| Optimizing LSA Generation and Flooding |
| Optimizing Routing Table Calculations |

Fig. 1. Improving convergence speed and scalability in OSPF: main steps

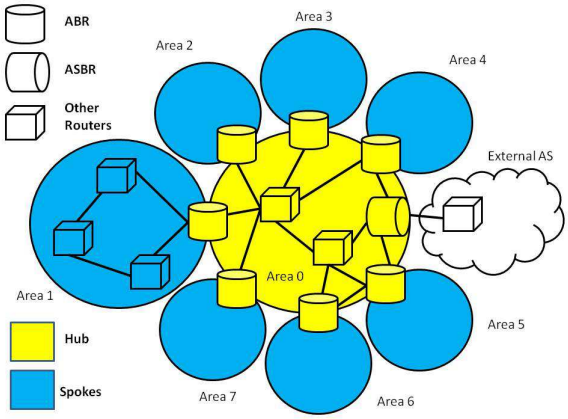



Fig. 2. *Hub and spoke* organization of OSPF areas

in a *hub and spoke* fashion with a special area, called *Area 0* or the *backbone* area, serving as the *hub* and other areas connected as *spokes* to the backbone area. All OSPF routes from a source in one area to a destination in another area need to pass through the backbone area. As shown in Fig. 2, a router may have interfaces in multiple areas. Such routers are known as the *area border routers* (ABRs). Also, some routers, known as the *autonomous system boundary routers* (ASBRs), may have links to routers in other autonomous systems (Fig. 2). Splitting a routing domain into multiple areas allows a router to require the complete topology information of only those area(s) to which its interfaces belong. In the following, we describe how a router comes to know about other routers in its immediate neighborhood and ultimately all the routers (and their interconnections) in the areas to which the router's interfaces belong. For detailed explanation of various aspects of OSPF operation, we refer the reader to [10] and [11].

An OSPF router, with interfaces on *broadcast* LANs or *point-to-point* links, comes to know about the routers in its immediate neighborhood via periodic exchange of *hello* messages. Each router *multicasts* a Hello message out of its interfaces after every *HelloInterval*. In its Hello, the router lists the other routers from which it has recently received a Hello message. When a router (say router $A$) finds itself listed in the neighbor's Hello message, it considers its adjacency with the neighbor (say router $B$) to be *bidirectional*. If router $A$ wants to establish *full* adjacency with neighbor $B$, it initiates

the process of synchronizing its *link state database* [2] (LSDB) with the neighbor's LSDB. The completion of the LSDB synchronization results in router $A$ considering its adjacency with neighbor $B$ to be *full*. At this point, router $A$ generates a new *router* LSA listing the adjacency state of all its interfaces that belong to the same area (as the link between itself and neighbor $B$) and sends the LSA out of these interfaces. When a neighbor router receives this LSA, it sends it out of all its interfaces in the area except the one on which the LSA was received. Thus, the LSA is *flooded* throughout the area. The flooding process achieves reliability by requiring a router to retransmit an LSA to a neighbor if it does not receive an acknowledgement of the LSA's receipt from the neighbor within a certain time interval (the *RxmtInterval*). Thus, each router in the area receives the LSA and comes to know about the neighbors with which router $A$ has established full adjacency.

The two routers stay adjacent to each other as long as they can periodically exchange the Hello messages. The adjacency breaks down when a router fails to receive a Hello message from the neighbor within the *RouterDeadInterval*. This happens if the link between the router and the neighbor fails or if the neighbor router is no longer functional. In some cases, the link layer protocol can inform a router about the failure of a link and thus allow the router to terminate adjacency without waiting for the *RouterDeadInterval* to expire. The breakdown of an adjacency causes a router to generate a new version of its router LSA. This LSA is flooded throughout the area thereby informing all the routers in the area about the adjacency breakdown. When a router receives a new LSA, it recalculates its routing table and updates the *forwarding information base* (FIB) on its line cards.

Overall, the convergence to a topology change in the OSPF protocol can be considered to consist of the following steps [1], [2]:

- Detection of a topology change by the routers in the vicinity.
- Adjacency establishment or breakdown by the routers affected by the topology change.
- The generation of new LSAs by the affected routers and their flooding throughout the OSPF area.
- Routing table calculations by each router on receiving the LSAs, followed by the distribution of the routing table updates to the line cards.

The overall convergence delay depends on the time required to complete each of the steps mentioned above. In the following sections, we describe each of these steps and survey recent research in reducing the delays or optimizing the processing associated with the step.

## III. Faster Failure Detection in OSPF

In this section, we first describe the nature of failures in IP networks. This is followed by a description of the default failure detection mechanism used in OSPF - the *hello* protocol, and recent proposals, summarized in Table I, to speed up the failure detection process including *bidirectional forwarding detection*.

[2]The collection of LSAs describing the network topology.

### A. The Nature of Failures in IP Networks

Failures are a common occurrence in an IP network. The failures at the IP layer may take place due to network maintenance operations, hardware/software failures in the routers, human errors (such as errors in configuring a protocol) or failures in the underlying optical fiber networks (such as a fiber cut or failure of an optical switch). The failure may manifest itself at the IP layer as the failure of a single/multiple links/routers. For example, a faulty line card would cause failure of a single IP link but a cut in an optical fiber would cause all the IP links travelling over the fiber to fail. Similarly, an OS reboot in a router would affect just that router but a power outage in a *point-of-presence* (PoP) may bring down all the routers located there. Sometimes, faulty hardware/software may result in *flapping* behavior, where one or more links in a router exhibit intermittent failures for extended time periods, resulting in a severe impact on the data traffic [12], [13].

Prescheduled or emergency maintenance operations, such as router reconfigurations, software upgrades and replacing of ageing hardware, account for moderate-to-significant fraction of failures in IP networks. Labovitz et al. [14] examined the failures on a medium size regional IP backbone in year 1998 and attributed 16% of observed failures to network maintenance operations. Markopoulou et al. [15] studied failures on Sprint's IP backbone in year 2002 and found 20% of the failures due to maintenance events. Medem et al. [16] analyzed year 2005-2007 failure data for Internet2, a network of 11 routers, and a large IP backbone, consisting of hundreds of routers, and found that 72% of failures on Internet2 and 25% failures on the large IP backbone were due to network maintenance operations.

Faulty router hardware has been reported as a major source of failures in IP networks [12]–[16]. Year 1998 study by Labovitz et al. [14] revealed that 40% of the router interfaces suffered a failure within an average of 40 days with 5% of the interfaces failing within 5 days on average. Year 2002 study by Markopoulou et al. [15] found that almost 70% of the unplanned (i.e., not maintenance related) failures were single link failures, presumably due to faulty/ageing interface cards. It was further noted that only 2.5% of the links accounted for more than half of these failures. Year 2005-2007 study by Medem et al. [16] attributed 8% of unplanned failure on Internet2 and 47% of unplanned failures on the large IP backbone to faulty router hardware.

In recent years, software and configuration related problems have also emerged as a major cause of failures in IP networks. Labovitz et al. [14] attributed only 1.3% of failures to software issues. However, Markopoulou et al. [15] attributed 16.5% of unplanned failures to router crashes, presumably due to software/configuration errors (although some router crashes could have been due to hardware failures as well). Medem et al. [16] attributed almost one third of all failures to software-related

| Mechanism | Advantage | Disadvantage |
|---|---|---|
| Hardware based failure detection | Failure discovery within tens of milliseconds. | Not always available. |
| Reduced HelloInterval | Can safely be reduced to half a second range. | Further reduction may lead to router overloads and false alarms. |
| Bidirectional forwarding detection | Protocol independent, light weight. Can be implemented in the line card's hardware/firmware. Can be used in association with reduced HelloInterval to significantly reduce the failure detection time. | Can't detect failures in control plane. |

TABLE I
MECHANISMS FOR FASTER FAILURE DETECTION IN OSPF

problems. Choi et al. [13] reported a staggering 1.8 million[3] link failure events over 9 months in 2006-2007 on a campus network of 40 routers and 373 switches and attributed most of these events to flapping links due to imperfect interaction among devices constituting the link.

Failures in the underlying optical fiber layer is the other major cause of IP-level failures. The fraction of unplanned failured attributed to optical network problems range from 10 to 15% in published studies [14], [15]. Ganjali et al. [17], in a year 2003 study on Sprint's IP backbone, observed that 84% of the link failures that had a significant impact on the network performance were caused by optical layer problems. A survey of various schemes to localize faults in optical networks can be seen in [18].

Finally, power outages were reported as being responsible for 16% of the failures in year 1998 study by Labovitz et al. [14], however, year 2005-2007 study by Medem et al. [16] suggests that it is no longer a major problem. Typical repair times for different failures have been reported to be between few tens of seconds (for individual link failures caused by recurring faults in old hardware), few minutes (for router/switch reboots) and few hours (for the fiber cuts) [15].

### B. The Hello Protocol

The *hello protocol* provides the default failure detection mechanism in OSPF. An OSPF router maintains an *inactivity timer* for each neighbor it has established full adjacency with. When a router receives a Hello from a neighbor, it resets the *inactivity timer* associated with the neighbor, scheduling it to fire after the *RouterDeadInterval*. The *RouterDeadInterval* is typically four times the *HelloInterval*. When the neighbor, or the link between the router and the neighbor, is no longer functional, the router will no longer receive the periodic hello from the neighbor and consequently the inactivity timer will fire *RouterDeadInterval* after receipt of the last hello from the neighbor. The firing of the inactivity timer causes the router to terminate its adjacency with the neighbor and generate a new router LSA to this effect. Depending on when the failure takes place after the receipt of the last Hello from the neighbor, a router may take anywhere between three to four *HelloIntervals* to break the adjacency and thus detect the failure. With default value of 10 seconds for the HelloInterval, the RouterDeadInterval would be 40 seconds and it would take anywhere between 30 and 40 seconds for a router to detect a failure. This time period typically constitutes the biggest chunk in the overall convergence delay.

Some hardware technologies, e.g., *packet over sonet* [19], allow the detection of a link failure within few tens of milliseconds by sending the routers at two ends of the link a *loss of signal* message. On receiving such a signal, the router waits for a *carrier delay* duration (few hundred milliseconds to few seconds) before letting OSPF act on it. The carrier delay allows the router to avoid false alarms and identify *link flapping*. However, the hardware-based failure detection is not always possible. For example, if a failure involves the central *route processor* but the router's line cards are functional, hardware detection of such a failure may not be possible.

There have been several proposals to reduce the *HelloInterval* and hence the *RouterDeadInterval* to reduce the failure detection time. Alaettinoglu et al. [20] proposed reducing the *HelloInterval* to millisecond range to achieve sub-second failure detection. There are multiple concerns with arbitrarily reducing the *HelloInterval* to very small values. One concern is that the need to send and receive the Hellos after every few milliseconds would cause the router CPU loads to shoot up. Another concern is that very small *RouterDeadInterval* may result in frequent false alarms, i.e., false adjacency breakdowns. As the *HelloInterval* becomes smaller, there is an increased chance that the network congestion will lead to loss or delayed processing of several consecutive Hello messages and thereby cause false breakdown of adjacency between routers even though the routers and the link between them are functioning perfectly well. The LSAs generated because of a false alarm lead to new routing table calculations, avoiding the supposedly down link, by all the routers in the network. A false alarm is soon corrected by successful Hello exchanges between the affected routers, which cause these routers to re-establish adjacency and generate new LSAs. These new LSAs force all the routers in the area to perform routing table calculations again. Thus, the false alarms cause temporary changes in the network traffic paths as well as unnecessary processing load on the routers. The changes in the traffic paths may have a serious impact on the traffic QoS since the changed paths may have significantly worse delay and loss characteristics, possibly due to congestion induced by the changes themselves, than the original paths.

Basu and Riecke [21] performed a simulations based analysis of the impact of sub-second *HelloInterval* values and reported that reducing the *HelloInterval* to 500ms or 250ms does not cause any significant increase in the router CPU loads. However, they did observe a six-fold increase in the number of route flaps (changes in the routing table), caused

[3]It is relevant to note that most commercial internet service providers treat *number* of failures in their IP networks as confidential information. So we do not know the extent of the problem in commercial networks besides that failures are *common*.

by false alarms, as the *HelloInterval* is reduced from 500ms to 250ms. Choudhury et al. [22], [23] observed that reducing the *HelloInterval* lowers the threshold (in terms of number of LSAs) at which an LSA burst will lead to generation of false alarms. Large LSA bursts can be caused by a number of factors such as simultaneous refresh of a large number of LSAs or several routers going down/coming up simultaneously. To avoid false alarms, they suggested prioritized generation and processing of Hello messages or, alternatively, resetting of inactivity timer on receiving any OSPF packet (e.g., an LSA) from the neighbor. Goyal et al. [24] observed that the frequency of false alarms in a network increases with the increase in the network congestion levels and with the increase in the number of links in the network. Thus, the optimal *HelloInterval* for a network depends on the network's tolerance for false alarm frequency, the expected congestion levels and the number of links in the network topology. In general, there seems to be a consensus that *HelloInterval* can safely be reduced to 500 milliseconds or so, which would result in failure detection times of around 2 seconds.

### C. Bidirectional Forwarding Detection (BFD)

Detecting the loss of connectivity between two networked devices quickly is a common requirement for many networking protocols [25]. Often the protocols do not have a native mechanism for this purpose or the native mechanism does not provide fast enough failure detection. For example, in case of OSPF, the native mechanism (Hello protocol) can not provide millisecond range failure detection. Another example is the LSP-Ping [26] mechanism to detect faults in a *label switched path* (LSP) in a *multi-protocol label switching* (MPLS) [4] network. The processing required for LSP-Ping messages is considered significant and hence the frequency of such messages can not be increased arbitrarily to achieve very fast detection of failures in an LSP. Some additional similar examples are described in [25].

*Bidirectional forwarding detection* (BFD) is a general purpose, light weight protocol to detect faults in the bidirectional path between two networked devices potentially very quickly [29]. BFD operates independently of other protocols and detects faults in the execution of the *packet forwarding* function, i.e., moving packets from one interface to another, of the networked devices. The packet forwarding function is typically performed by the processors in the line cards. To avoid fate sharing with the *control plane* (i.e., the CPU), which runs the routing protocols, BFD is intended to be implemented in the *data plane* (i.e., in the line cards) to the extent possible. BFD's ability to quickly detect data plane faults can be used in conjunction with a protocol's native ability to detect data/control plane faults. For example, an OSPF router can initiate a BFD session with a neighbor router and use it in conjunction with the Hello protocol to quickly detect the loss of connectivity with the neighbor [30]. Similarly, a BFD session between the ingress and egress routers of an MPLS LSP can be used in conjunction with the native LSP-Ping method to detect faults in the LSP [31].

A BFD session between two devices can operate in two different modes. In the *asynchronous* mode, the devices periodically send BFD control packets to each other and a device declares a failure when it does not receive any BFD packet from the other device for some pre-determined time. In the *demand* mode, there is no periodic exchange of messages between devices in a BFD session. Rather a short sequence of BFD control packets is exchanged when a device feels the need to verify the connectivity. BFD also supports an *echo* function, where a device sends control packets addressed to itself to the other device. These packets come back to the source device after travelling through the entire forwarding path in the other device. Thus, the Echo function allows a device to test only the forwarding path on the remote device and determine failures quickly [29].

BFD allows two devices establishing a BFD session to negotiate the time interval between successive BFD control packets. Thus, very fast detection times (around 50 ms [32]) can be obtained if devices in the BFD session can receive the control packets at a very fast pace. The time interval between successive control packets can be adjusted dynamically. The BFD protocol is well suited for implementation in the line card's hardware or firmware as a device in a BFD session expects to send and receive identical packets during the times of no fault [25].

## IV. Faster and Fewer Adjacency Establishments

The adjacency establishment process begins with neighboring routers exchanging Hello messages with each other and thus achieving *bidirectional* status. This is followed by the exchange of *database description* (DD) packets that describe the set of LSAs that the router has in its LSDB. With the examination of received DD packets, each router determines if the neighbor has newer instances of some LSAs and requests the neighbor (via *link state request* packets) to send these LSAs. The routers then send requested LSAs to each other in *link state update* packets. Thus, the two routers synchronize their LSDBs and generate new instances of their LSAs listing each other as fully adjacent. The area-wide flooding of these new LSAs ensures that the LSDBs of adjacent routers stay up-to-date and synchronized.

In the following subsections, we describe the proposed enhancements to the process of establishing adjacency between two routers as well as the enhancements that reduce the number of adjacency establishments required in broadcast/NBMA (*non-broadcast multi-access*[5]) LANs and mobile ad hoc networks (MANETs). Table II provides a brief overview of these enhancements.

[4]MPLS [27], [28] is a protocol-independent mechanism for forwarding packets based on the *label* they carry. See Section VIII-A.

[5]NBMA link layer technologies, such as *ATM* and *frame relay*, allow multiple devices on the same link but do not have inherent support for packet broadcast, i.e., a packet transmission does not inherently reach all the devices on the link. In contrast, *broadcast* LAN technologies, such as *Ethernet*, inherently allow all devices on the link to receive a packet transmission.

| Mechanism | Description | Pros/Cons |
|---|---|---|
| Database exchange summary list optimization [33] | DD packets do not include headers of LSAs that the neighbor does not need. | Simple. Can reduce the DD overhead by about 50% in large networks. IETF approved. |
| Exchange LSDB *signatures* rather than LSA headers in DD packets [34] | | The cost of database exchange no longer increases linearly with database size. |
| OSPF's interface state machine modifications [35] | | Reduces the time and processing requirements of DR/BDR election process. |
| OSPF-MANET extensions [36]–[38] | See Table III. | Seemless integration of MANETs with traditional wired networks. Significant reduction in the number of adjacencies required, size of hello messages and the overhead associated with LSA flooding in MANETs. IETF approved. |
| Smart adjacency establishment in OSPF [39] | Adjacency establishment by transitivity without database exchange. Similar to OSPF-OR. | Applicable to traditional wired networks. Significant speed up in the adjacency establishment process. |

TABLE II
OSPF ENHANCEMENTS FOR FASTER AND FEWER ADJACENCY ESTABLISHMENTS

### A. Optimizing the Database Exchange Process

Ogier [33] proposed *database exchange summary list optimization*, an extension to OSPFv2/v3 to speed up the database exchange process by minimizing the payload of DD packets. Upon receiving a DD packet from a neighbor, a router sends its DD packets as a response. In standard OSPF, the router sends DD packets that carry headers of the corresponding LSAs in its LSDB. In the extension, the router determines if there are LSAs in the received DD packet that are the same or newer instances of the LSAs in its own LSDB. Such LSAs, should they exist, are excluded from being listed into DD packets that will be sent to the neighbor as a response, decreasing the overhead due to the DD exchange. Baccelli et al. [34] proposed an alternative mechanism for database exchange. The basic principle, somewhat inspired by the one employed in IS-IS, is to exchange compact *signatures* (hashings of a partition of the LSDB) between neighbor routers, instead of the usual slew of DD packets, in order to detect differences in the router's LSDBs. When a discrepancy is detected between some signatures, the bits of information required to synchronize the LSDBs of the involved routers are then identified and exchanged.

### B. Reducing the Number of Adjacency Establishments on Broadcast/NBMA LANs

Upon starting up, an OSPF router, with an interface on a broadcast or an NBMA LAN, establishes bidirectional communication with its neighbors by exchanging Hello messages. In a broadcast/NBMA LAN environment, any other router can be considered a neighbor. The adjacency establishment with every neighbor may put a significant burden on a router. Hence, OSPF protocol requires that routers on a broadcast/NBMA LAN elect a leader among themselves known as the *designated router* (DR), and its backup, known as the *backup designated router* (BDR). The DR and the BDR establish full adjacency with all the routers on the LAN. The other routers that are neither DR nor BDR establish full adjacency only with DR and BDR. As a result, the number of adjacency establishments required on a LAN is reduced significantly. The DR originates a *network* LSA listing all the routers on the LAN. This LSA is flooded throughout the area and represents the LAN in the LSDBs of the routers in the area. The routers on the LAN, including the DR and the BDR, advertise an adjacency to the *network* (LAN) in their router LSAs. In the event of the DR's failure, the BDR can quickly take over the responsibilities of the DR, including the origination of a new network LSA, since it is already adjacent to all the other routers on the LAN.

Goyal et al. [35] analyzed OSPF's *interface state machine* to determine the time required to settle on the final identity of the DR/BDR as the routers on a LAN come up and the number of *DR elections* performed by the routers in the process. Here, the DR election refers to the algorithm used by a router to identify the current DR/BDR in the LAN. They further proposed modifications to the OSPF's interface state machine in order to reduce the time and processing requirements of the DR/BDR election process.

### C. Strategies for Optimizing Adjacency Establishment on MANETs

In mobile ad hoc networks (also called MANETs), routers can dynamically join or leave the network frequently, which causes standard OSPF to trigger a large number of adjacency establishments and break down. Thus, new strategies have been proposed to minimize the number of adjacency establishments that will be triggered by OSPF in that kind of environment. The Internet Engineering Task Force (IETF) has developed several proposals extending OSPF for efficient operation on MANETs:

- *OSPF-MPR* [36] and *OSPF-OR* [37], based on *multi-point relays* (MPR),
- *OSPF-MDR* [38], based on *MANET designated router* (MDR).

The commonality between the different OSPF extensions for MANET is that they propose a new OSPF interface type, tailored for the characteristics of multi-hop wireless networks, while letting OSPF run unaltered on usual networks and existing interfaces. They use alternative mechanisms to reduce overhead and speed up convergence time, which can be classified into the following categories [40]:

- *Adjacency selection:* Rather than establishing adjacency with all its neighbors, a router becomes adjacent with only selected neighbors.

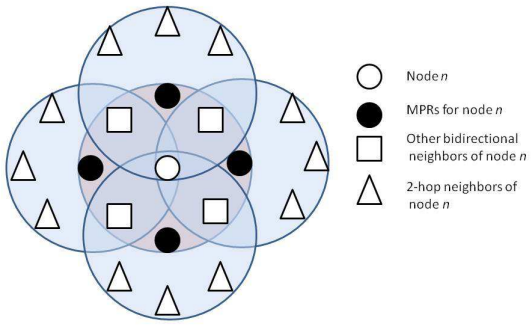



Fig. 3. Multi-point relaying (MPR). Node $n$ selects MPRs, from its bidirectional neighbors, to cover every neighbor 2 hops away. The circles show the radio range of the nodes in their center.

- *Flooding optimizations* to reduce redundant retransmissions.
- *Topology reduction:* Rather than listing all adjacent neighbors, a router reports only a subset of its adjacencies in its LSAs.
- *Hello redundancy reduction:* Rather than carrying full neighborhood information, some Hello messages report only changes in the router's neighborhood.

Table III provides an overview of different OSPF extensions for MANET. In this section, we discuss the adjacency selection mechanisms in these extensions. The other categories of alternative mechanisms mentioned above are discussed later in this paper.

OSPF-MPR [36] uses the *multi-point relaying* (MPR) technique introduced by a MANET routing protocol called *Optimized link state routing* (OLSR) [41]. In OSPF-MPR, each router selects a number of *multi-point relays* from the set of its bidirectional neighbors. The MPR neighbors are selected by the router so that any other "neighbor" 2 hops away is reachable through at least one MPR (Fig. 3). Each router thus maintains a set listing neighbors it has currently selected as MPR, as well as a set listing neighbors that have currently selected it as their own MPR (these neighbors are called *MPR selectors*). A router establishes full adjacency only with its MPRs and its MPR selectors, thereby reducing the total number of adjacency establishments needed in the MANET. In order to cope with the rare pathological case where the resulting set of adjacencies is not connected network-wide, one router in the network (the *sync router*) establishes adjacency with all its neighbors. Heuristics to select the MPRs and Sync routers can be found in [36].

OSPF-OR [37] (*overlapping relays*) uses the *smart peering* technique. The underlying idea is that two routers need not establish adjacency if they can already reach each other in the *shortest path tree* (SPT). In OSPF-OR, when a router receives a Hello message from a new neighbor, the LSDB is examined to look for the neighbor's router LSA. If none exists, it means that the neighbor is not reachable in the SPT and the adjacency is established via database exchange. Otherwise, the database exchange is typically not performed and the neighbor is optionally listed in the router's LSA as an *unsynchronized adjacency*[6]. In OSPF-OR, an unsynchronized adjacency can be used in routing table calculation but the two ends of such an adjacency must perform explicit database exchange if they can not reach each other in the SPT built after excluding all the links with unsynchronized adjacencies. Smart peering aims to reduce the database exchange overload in OSPF operation in MANET environment. However, the underlying concept can also be used in conventional OSPF networks.

Venkatesh [39] proposed an extension to OSPF operation on conventional networks where adjacency establishment via database exchange takes place only along the links of a *spanning tree* maintained in a dynamic fashion by the routers in the network. If a router can reach a new neighbor via the links on the spanning tree, an *unsynchronized* adjacency is declared without any database exchange. Otherwise, the two routers establish adjacency via database exchange. They further conclude that they must have belonged to two hitherto unconnected parts of the network. Hence, the two routers merge their spanning trees into a larger spanning tree that also includes the link between the two routers. The rest of the nodes in the network are informed about the new spanning tree by flooding this information along the links on the tree. The breakdown of an adjacency along the current spanning tree may trigger database exchange on an unsynchronized adjacency and the inclusion of this link in the spanning tree so as to avoid its partition. As in OSPF-OR, the unsynchronized adjacencies are used in route calculations with no distinction.

OSPF-MDR [38] uses the *connected dominating set* (CDS) technique. This mechnanism forms a connected backbone of routers, called *MANET designated router*s (MDRs). Each router in the network is either an MDR or a neighbor of an MDR. Similar to OSPF operation on a broadcast/NBMA LAN, routers also form a backup backbone consisting of *backup MDR*s (BMDR). Again, each router in the network is either a BMDR or a neighbor of a BMDR. Routers then become adjacent only with their MDR and BMDR neighbors. Heuristics to identify the backbone and the backup backbone are given in [38].

## V. LSA Generation and Flooding

In OSPF, the topology information is carried in LSAs. A *router* LSA describes the state of the router's interfaces to an area. A *network* LSA represents a broadcast/NBMA LAN and describes the set of routers connected to the LAN. Additionally, *area border routers* (ABRs), i.e., the routers that have interfaces to multiple areas, may originate in an area the *summary* LSAs that describe the originating ABR's cost to destinations outside the area but inside the AS. Finally, *AS border routers* (ASBRs), i.e., the routers that have links to routers in an external AS, may originate *AS external* (ASE)

[6]Such an adjacency is termed *unsynchronized* since reachability in SPT does not guarantee synchronization of databases. This is because a router's LSDB may not contain the latest LSAs at all times and hence the router may consider a neighbor reachable in the SPT even though it is not so. In fact, assuming that two routers have synchronized databases because they are reachable in SPT is a common pitfall that must be avoided.

| | Multi-point Relays (MPR) | MANET Designated Routers (MDR) | Overlapping Relays (OR) |
|---|---|---|---|
| **Key Terms** | MPR set: Set of neighbors of a router that provide reachability to all its 2-hop neighbors.<br>MPR Selector: A neighbor that selects the router as an MPR. | MDRs: The set of routers that form a connected backbone and provide reachability to all other routers in the network. | Smart Peering: Two routers need not establish adjacency if they can already reach each other in the SPT.<br>OR: A neighbor that provides reachability to one or more 2-hop neighbors of the router.<br>Active ORs: Set of neighbors of a router that provide reachability to all its 2-hop neighbors. |
| **Adjacency Selection** | Adj establishment only with MPRs and MPR selectors. | Adj establishment only with MDR and backup MDR neighbors. | No need to establish adj with neighbor already reachable in SPT. |
| **Flooding Optimization** | Only a router's MPRs relay back the LSA, received from the router, on their MANET interface. | An MDR always relays back a received LSA on its MANET interface. A backup MDR relays back a received LSA on its MANET interface only if necessary. | An active OR of a router always relays back an LSA received from the router on its MANET interface. A non-active OR of a router relays back an LSA received from the router on its MANET interface only if necessary. |
| **Topology Reduction** | LSAs report only adjacencies between MPRs and their MPR selectors. | LSAFullness value determines the extent of topology reported in LSAs. | LSAs optionally report only adjacencies established through smart peering. |
| **Support for *delta* hellos** | No | Yes | Yes |

TABLE III
AN OVERVIEW OF OSPF-MANET EXTENSIONS

LSAs that describe the originating ASBR's cost to destinations outside the AS. Table IV provides a brief overview of different LSAs used in OSPF networks.

A topology change within the area results in the generation of new instances of router/network LSAs by the affected routers. Similarly, the topology change events outside the area may result in generation of new summary/ASE LSAs. A new router, network or summary LSA is flooded throughout the area to which it belongs while a new ASE LSA may be flooded throughout the AS. In other words, the flooding scope of a router, network or a summary LSA consists of a single area whereas that of an ASE LSA may consist of the entire AS. Each router receiving the new LSA takes part in the flooding process by sending the new LSA across all interfaces within the flooding scope except the one on which the LSA arrived[7]. Eventually, all routers in the LSA's flooding scope receive the new LSA, update their LSDB and perform recalculation of their routing tables to reflect the current topology. A router also generates a new instance of its LSA when the old instance reaches the age specified by the *LSRefreshTime* parameter (30 minutes by default). This process, called *LSA refresh* helps increase the protocol's robustness.

In this section, we first describe various configuration parameters that affect LSA generation/flooding process. This is followed by a description of the *DoNotAge* LSAs and the *subnet aggregation*, the mechanisms that significantly reduce the flooding overhead. Subsequent subsection describes various proposals aimed at optimizing the process of flooding an LSA throughout its flooding scope. Finally, we describe the flooding overhead reduction mechanisms used in OSPF extensions for MANET environment. Table V provides a brief summary of the OSPF enhancements described in this section.

[7]As discussed later in Section V-E, an LSA received on a MANET interface may need to be resent along that interface as well.

### *A. Configuration Parameters Affecting LSA Generation and Flooding*

In the following, we describe various standard and vendor-specific configuration parameters that have a significant impact on the LSA generation and flooding process:

- The *minLSInterval* parameter, with a default value of 5 seconds, limits the frequency with which a router can originate new LSAs. A router can not originate a new instance of an LSA if the previous instance was originated less than *minLSInterval* ago.
- The *minLSArrival* parameter, with a default value of 1 second, limits the frequency with which a router can accept new LSAs transmitted by other routers. A new instance of an LSA arriving at a router is discarded if the previous instance was received less than *minLSArrival* time ago.
- The *RxmtInterval*, with a default value of 5 seconds, parameter specifies the time interval after which a router should retransmit an LSA if no acknowledgement was received for the previous transmission.
- Routers increase the *age* of LSAs in their database at regular intervals.[8] A router refreshes a self-originated LSA (i.e., an LSA originated by the router itself) when it reaches the age specified by *LSRefreshTime* parameter (30 minutes by default). If the originating router fails to refresh an LSA, the routers in the network will continue to age this LSA further. When a router determines that an LSA, irrespective of whether it is self-originated or not, has reached the *MaxAge* (default value: 1 hour), it refloods this LSA throughout its scope. The receipt of a *MaxAge* LSA causes all instances of this LSA to be deleted from the receiving router's LSDB. Thus, an LSA that has reached the *MaxAge* in any router is quickly deleted from the LSDBs of all the routers in the network. Deleting LSAs in this manner allows OSPF to "garbage collect" LSAs of dead routers.

[8]Unless the LSA has *DoNotAge* bit set [54].

| LSA Type | Originating Router | Information carried | Flooding Scope |
|---|---|---|---|
| Router LSA | Any router | Adjacency status on the router's interfaces in the area | Area wide |
| Network LSA | Designated Router (DR) | Describes the set of routers on a broadcast/NBMA network | Area wide |
| Type 3 Summary LSA (OSPFv2 [1])/ Inter area prefix LSA (OSPFv3 [2]) | Area Border Router | Describes an IP network or a range of IP addresses in the AS but external to the area in which the LSA is flooded | Area wide |
| Type 4 Summary LSA (OSPFv2)/Inter area router LSA (OSPFv3) | Area Border Router | Describes an ASBR external to the area in which the LSA is flooded | Area wide |
| AS-external LSA | AS Boundary Router | Describes a destination external to the AS | AS wide except in stub areas and not-so-stubby areas (NSSA) [42] |
| Group Membership LSA | Any router | Describes the originating router's directly attached networks that contain members of a particular multicast group [43] | Area wide |
| Type 7 NSSA LSA | NSSA AS Boundary Router | Describes a destination external to the AS | Within the originating NSSA |
| Link LSA (OSPFv3) | Any router | Informs other routers on the link about the originating router's link-local address and IPv6 prefixes associated with the link | Link local, i.e., not flooded further by routers receiving the LSA |
| Intra area prefix LSA (OSPFv3) | Any router | Associates a list of IPv6 prefixes with the originating router or the transit network for which the originating router is the DR | Area wide |
| Opaque LSA | Any router | Provides a general mechanism to distribute information via OSPF | Link local for type 9 opaque LSAs; Area wide for type 10 opaque LSAs; AS wide for type 11 opaque LSAs except in stub areas and NSSA |

TABLE IV
OSPF LINK STATE ADVERTISEMENTS [1], [2]

| Mechanism | Description | Pros/Cons |
|---|---|---|
| Dynamic minLSInterval [44], [45]. | The minLSInterval increases with LSA generation frequency. Available in commercial routers [46]. | Speeds up convergence for many topology changes. |
| Dynamic RxmtInterval and pacing delay [22]. | Dynamically increase the RxmtInterval and pacing delay for a congested neighbor. | Helps avoid exasperating congestion at a neighbor. |
| Group pacing delay [47]. | LSA refreshes in groups so as to reduce the number of LS update packets and avoid LSA storms. Available in commercial routers [47]. | |
| Setting DoNoAge bit in LSAs to avoid periodic refresh [48]. | | Significant reduction in LSA processing overhead of routers. IETF approved. |
| Algorithms for smart subnet aggregation [49], [50]. | Subet aggregation refers to an ABR generating a single type 3 summary LSA for multiple subnets in an area. | Helps reduce the number of summary LSAs while minimizing suboptimality in path selection. |
| Extended reverse path forwarding [51]–[53]. | An LSA is forwarded only along a spanning tree rooted at the LSA's source. | Can significantly reduce the LSA flooding overhead. |
| OSPF-MANET extensions for topology reduction and flooding optimization [36]–[38]. | LSAs forwarded only along a common subgraph irrespective of their source. See Table III. | Significant reduction in LSA flooding overhead in MANETs. |

TABLE V
OSPF ENHANCEMENTS TO OPTIMIZE LSA GENERATION AND FLOODING

- The *LSA pacing delay* is a non-standard parameter that specifies the minimum time interval between consecutive transmissions of link-state update packets by a router. This delay limits the link capacity consumed by LSA flooding/retransmission operations and causes *batching* together of the LSAs possibly originated by different routers into few link-state update packets.

A large value (e.g., default value 5 seconds) for the *minLSInterval* parameter limits the LSA origination by a router and hence acts as a stabilizing factor when large scale topology changes take place (e.g., a PoP-level router reboots) or in face of pathological conditions such as link flaps. On the other hand, large *minLSInterval* causes delays in LSA generation and hence delays in convergence to a topology change. Hence, Katz [44] suggested that important LSAs (e.g., LSAs describing a failure) may be flooded without enforcing *minLSArrival*, *minLSInterval* or LSA pacing delays. Choudhury [45] reported significant speedup in convergence times if the *minLSInterval* parameter is set to a small value (1 second) but is allowed to double (up to a maximum value, say 5 seconds) whenever the router attempts to originate a new instance of its LSA before the expiry of current *minLSInterval*. The parameter returns to its initial small value when router does not attempt to originate a new LSA within the current *minLSInterval*. Such dynamic adjustment in *minLSInterval* has been implemented in Cisco IOS (Release 12.2(27)SBC onwards) and is known as *LSA throttling* [46].

Cisco IOS (Release 12.2(14)S onwards) provides three types of LSA pacing delays: *retransmission pacing*, *flood pacing* and *group pacing* [47]. The retransmission pacing delay is another name for *RxmtInterval* while the flood pacing delay is same as the LSA pacing delay described above, i.e., it is the minimum

time interval that must elapse between transmission of two link-state update packets by a router. The default value of the flood pacing delay is 33 milliseconds, although it can be set to any value in the range from 5 milliseconds to 100 milliseconds.

The *per-link* pacing delays can add up quickly, thus slowing down the convergence process and causing large variance in the arrival times of the LSAs at different routers in the network. This may cause the transient routing loops following a topology change to last longer. On the other hand, the pacing delays serve a very important purpose by regulating LSA flooding/retransmissions to a 'congested' neighbor. Choudhury et al. [22] suggested that a router should dynamically adjust the *RxmtInterval* and pacing delays for a neighbor based on its perception of whether the neighbor is facing congestion or not. To avoid exasperating congestion at the neighbor, they suggest that a router should exponentially increase the *RxmtInterval* for an LSA if the neighbor repeatedly fails to acknowledge this LSA (presumably due to congestion). Additionally, the router should try to mitigate the congestion at the neighbor by adjusting the pacing delay based on the number of LSAs that have not been acknowledged by the neighbor. If the number of unacknowledged LSAs is more than a high-water mark, the pacing delay for the neighbor should be multiplicatively increased (up to a certain maximum) with time. The pacing delay for the neighbor can be rapidly reduced when the number of unacknowledged LSAs falls below a low-water mark.

Cisco's *group pacing* delay [47] allows the LSA refreshes to be grouped together in a desired manner. Consider a router that originates multiple LSAs, e.g., an area/AS border router originating several *summary* LSAs. In order to reduce the flooding overhead due to LSA refreshes, it is important to pack as many LSAs in a single link-state update packet as possible. On the other hand, the router should not refresh all its LSAs simultaneously as it may lead to *LSA storms* especially if the router originates a large number of LSAs. Thus, the number of LSAs that are refreshed together should be neither too small nor too large. When the group pacing delay timer fires, the router increases the age of LSAs in its database and if some self-originated LSAs have reached the *LSRefreshTime* age, the router refreshes them. Thus, the group pacing delay specifies the time granularity with which a router ages the LSAs in its database and also the minimum time interval between two batches of LSA refreshes.

### B. DoNotAge LSAs

OSPF allows a link to be categorized as a *demand circuit* [54], which means that the operational cost of the link depends on its usage. Some legacy technologies, such as ISDN and X.25, fit this description. OSPF control traffic due to periodic Hello exchange and LSA refreshes may prove expensive on such demand circuits. Hence, OSPF allows Hellos and LSA refreshes to be suppressed on the demand circuits. LSA refreshes are avoided by setting the *DoNotAge* bit in the LSAs. As their name indicates, the *DoNotAge* LSAs are not aged and hence there is no need to refresh them after every *LSRefreshTime* interval.

Periodic LSA refreshes can result in a significant processing overhead for the routers in a large network. Hence, OSPF now allows a more general use of *DoNotAge* LSAs to avoid this overhead for large but stable network topologies [48]. A router may set the *DoNotAge* bit in its self-originated LSAs before flooding thereby making it unnecessary to refresh them after every *LSRefreshTime* interval. A new instance of the LSA needs to be generated only when the contents of the LSA change.

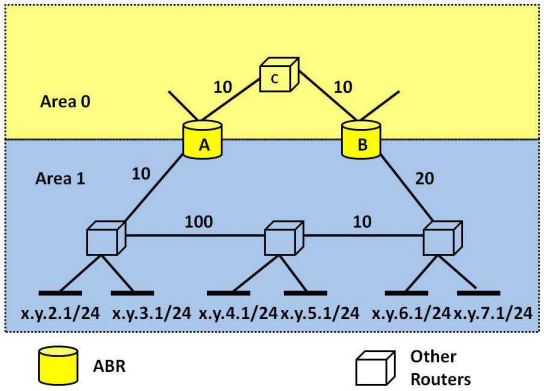



Fig. 4. An example topology to illustrate the suboptimal routes caused by subnet aggregation

### C. Subnet Aggregation

In general, each OSPF area in a routing domain is made up of links connecting routers and subnets. The standard OSPF supports *subnet aggregation*, which allows an *area border router* (ABR) to aggregate several subnets in one area and describe them as a single *type 3 summary* LSA in a different area. Route summarization leads to a much smaller size of link-state database and hence significant reduction in flooding and database synchronization overhead. However, these advantages come at the expense of optimality in routing. Depending on how the ABRs perform the aggregation, some information may be lost which may cause a router to choose a sub-optimal (longer than necessary) path to a subnet in the remote area. Consider the example shown in Figure 4. In this figure, routers $A$ and $B$ are ABRs with interfaces in both *area 0* and *area 1*. *Area 1* contains six subnets as shown in the figure. In the absence of any subnet aggregation, routers $A$ and $B$ would send an individual *type 3 summary* LSA in *area 0* for each subnet in *area 1*. Thus, router $C$ in *area 0* would correctly choose router $B$ as the next hop on its shortest path to subnet *x.y.7.1/24*. On the other hand, if routers $A$ and $B$ choose to aggregate all six subnets as one prefix *x.y.0.0/21* with advertized cost being the maximum of all the subnets, router $C$ would incorrectly choose router $A$ as the next hop on its shortest path to subnet *x.y.7.1/24*. This is because router $A$ would advertise a cost *max(10, 110, 120) = 120* for prefix *x.y.0.0/21*, which is better than the cost *max(20, 30, 130) = 130* advertised by router $B$ for the the same prefix.

Such path selection errors due to aggregation can be minimized by careful selection of aggregates and their advertized costs. Rastogi et al. [49] presented a *dynamic programming* based algorithm to determine the given number of aggregates

*for all OSPF areas* such that the cumulative error in path selection for all source-destinations pairs is minimized. They also presented heuristics to determine the costs to be assigned to the aggregates. Shaikh et al. [50] observed that the aggregates for one area can be determined solely based on the information about that area. Thus, the aggregates for one area can be determined independently of the aggregates for other areas. They present an algorithm to determine the minimal set of aggregates for a given area given the upper limit on the acceptable path selection error.

### D. Optimizing the Flooding Process

As described earlier, new instances of LSAs are disseminated throughout an area to ensure the routers have the same view of the network. The LSA dissemination takes place via a reliable *flooding* algorithm, where a router floods an LSA received on one interface out of all the other interfaces in the same area.[9] Reliability is achieved by retransmitting the LSA out of an interface if an acknowledgement is not received for the previous transmission within the *RxmtInterval*.

The main disadvantage of this algorithm is that a router may receive multiple copies of a new LSA from its neighbors during the flooding process. Only one of them is actually needed by the receiving router to update its view of the network (i.e., its LSDB). Other copies of the LSA that are being forwarded to the receiving router (and the acknowledgements that it has to send back) are redundant. As the network becomes larger in size, the number of redundant packets being generated during the flooding procedure also increases. The overhead of processing these packets can have a significant impact on network stability. This is especially true when OSPF LSAs are used to spread not only the topology information but also the information about link-level *QoS* parameters such as available bandwidth, delay and jitter [56]. Such QoS parameters change much more frequently than network topology and hence LSAs carrying this information would be originated and flooded much more frequently than regular LSAs carrrying topology information [57].

Although not yet adapted in OSPF standard (except in the context of MANETs as discussed in Section V-E), optimizing the flooding process in link state routing protocols has been a topic of research for a long time. In 1978, Dalal and Metcalfe [51] proposed *reverse path forwarding*, where a node forwards a packet to its other neighbors only if the packet was received from the node's next hop neighbor on the "best" route from the node to the source of the packet. The redundant transmissions can be further avoided if a node forwards a packet to a neighbor only if the node is the next hop on the best route from the neighbor to the source of the packet. This approach, referred to as the *extended reverse path forwarding* (ERPF) [51], ensures that a broadcast packet is forwarded along a spanning tree rooted at the source of the packet.

Bellur and Ogier [52] proposed *topology broadcast based on reverse-path forwarding* (TBRPF), an ERPF based approach, where dissemination of topology information takes place along a *minimum hop* tree rooted at the source of the information. In this approach, a node $i$ calculates its *parent*, $p_i(j)$, on the minimum hop route to each node $j$ in the network and lets the parent know about it. When a node receives topology information originated by node $j$, it forwards this information to only those nodes that have selected it as the parent on their minimum hop route to node $j$. The topology information travels along the minimum hop tree and is also used to modify the tree itself.

Humblet and Soloway [53] proposed an alternative approach for topology broadcast, where a node, based on the topology information it has, calculates its *children*, rather than its parent, on the spanning tree along which the topology information would spread. Again, each source of the topology information has its own spanning tree to spread the information it originates. Alternatively, the nodes in the network can calculate a common subgraph along which the dissemination of topology information takes place irrespective of its source. This subgraph could simply be a minimum spanning tree or a richer structure that stays connected even in face of some failures [58]. As discussed next, OSPF extensions for MANET [36]–[38] perform LSA forwarding along a common subgraph irrespective of the LSA's source.

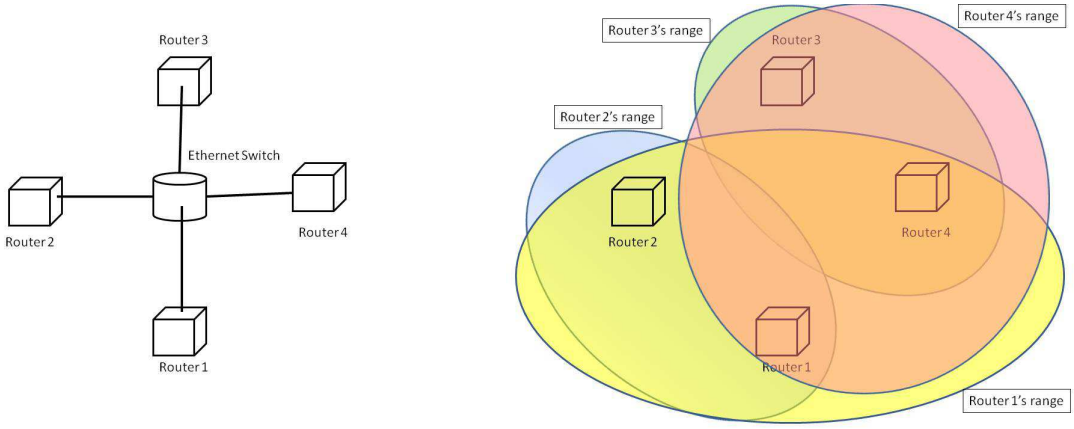
### E. Reducing Flooding Overhead in MANETs

On conventional *wired* networks, a router does not need to send an LSA out of the interface on which it was received. However, on multi-hop wireless networks, if a router receives an LSA on its MANET interface, it may need to send the LSA out of the same interface to ensure that all the routers on the network do receive the LSA [59]. Figure 5 presents an example illustrating such a case. If routers $1$ through $4$ are connected over an Ethernet, as in Fig. 5(a), router $1$ can expect all other routers to receive an LSA it sends on the Ethernet and these routers need not send this LSA out of their interface on the Ethernet. However, if these routers constitute a multi-hop wireless network with radio ranges as shown in Fig. 5(b), an LSA sent by router $1$ on its MANET interface would be received only by routers $2$ and $4$. Thus, router $4$ would need to forward the LSA out of its MANET interface to ensure that router $3$ receives it.

Whether a router should relay an LSA received on a MANET interface out of the same interface or not requires careful consideration. Blindly relaying all LSAs received on a MANET interface out of the same interface is not advisable because:

- The frequency of topology changes, and hence that of LSA generation, is expected to be much higher in MANETs than in conventional networks because of node movements and the *on/off* nature of wireless connectivity among MANET nodes.
- Most wireless communication protocols used by MANET nodes are based on *carrier sense multiple access* (CSMA) [60], [61] protocol, where a node competes with other nodes in its radio range for access to transmission channel. Only one node, among the set of competing nodes, may transmit at a given time. The performance of

[9] Dalal and Metcalfe [51] characterized this scheme as *hot potato forwarding* and attributed it to Baran et al. [55].




(a) An LSA sent on a wired broadcast LAN is received by all the routers on the LAN

(b) An LSA sent on a MANET interface may not reach all the routers in the MANET

Fig. 5. An LSA received on a MANET interface may need to be sent out of the same interface

CSMA protocol tends to breakdown, i.e., the number of successfully delivered packets decreases, with increased contention for channel access.

Hence, uncontrolled relay of received LSAs out of MANET interfaces may turn out to be problematic. This is especially true in MANET topologies consisting of a large number of densely deployed nodes. Hence, OSPF extensions for MANETs, introduced in Section IV-C, specify mechanisms to reduce the flooding overhead. These mechanisms fall in two categories: *flooding optimization* and *topology reduction*. Note that these mechanisms may also be used beneficially in conventional wired networks.

*1) Flooding Optimization in MANETs:* Flooding optimizations in OSPF MANET extensions commonly reduce the number of routers participating in the flooding process, while ensuring that all the routers still receive the LSA. As discussed earlier, in OSPF-MPR [36], each router maintains a set of *multi-point relay* (MPR) routers, selected from its bidirectional neighbors, such that all 2-hop neighbors of the routers can be reached via one of the MPRs (Fig. 3). Each router also maintains the set of *MPR selectors*, i.e., the routers that have selected this router as an MPR. In OSPF-MPR, an LSA is flooded only along the MPR tree rooted at the node originating the LSA. In other words, a router floods an LSA further only if it has been received from an MPR selector.

OSPF-OR [37] also uses the MPR technique although MPRs are now called active *overlapping relays* (OR). Each router selects *active* ORs from the set of its OR neighbors, where a neighbor is considered an OR if it can reach a router that the router can not reach directly, i.e., a 2-hop neighbor of the router. As in OSPF-MPR, the active ORs are determined such that all 2-hop neighbors can be reached via the active ORs. Similarly to OSPF-MPR, if a router receives an LSA from a neighbor for which it is an active OR, the router immediately relays the LSA out of the same MANET interface on which the LSA was received. However, unlike OSPF-MPR, a router still has a role to play in the flooding of the LSA if it is OR, although non-active, for the neighbor that sent the LSA. A non-active OR does not immediately relay the received LSA. Rather, it starts a timer and listens for the relay of this LSA or its ACK by the neighbors. If all neighbors have relayed the LSA or its ACK before the timer's firing, there is no need for the router to relay the LSA itself. Also, the router may choose not to relay this LSA if it hears a relay that must have reached all its neighbors that are 2-hop neighbors of the router from which it received the LSA. Otherwise, the router relays the LSA when the timer fires. The timer duration is randomly selected from a given range so that the timer fires at different times at different non-active ORs receiving the LSA.

As discussed in Section IV-C, routers under OSPF-MDR scheme [38] select a *bi-connected dominating set* of MDRs and BMDRs among themselves. Only MDRs and BMDRs participate in LSA flooding. An MDR immediately relays back the received LSAs on its MANET interface. A BMDR waits for a certain time interval before deciding whether to relay the LSA or not. During this interval, the BMDR actively monitors the LSA/ACK relays over the MANET. At the conclusion of this interval, the BMDR relays the LSA only if it is certain that one or more of its bidirectional neighbors have not received the LSA yet.

*2) Topology Reduction in MANETs:* The topology reduction mechanisms used by OSPF extensions for MANETs, propose to report only partial topology information in LSAs, while still ensuring that LSDBs contain enough information to connect the network, thus reducing both LSA size and the number of LSAs that need to be flooded.

OSPF-MPR [36] reports only adjacencies between MPRs and their MPR selectors in LSAs. This reduces the number of links that needs to be reported, while ensuring that the shortest paths can still be computed network-wide, and that paths use adjacencies only.

OSPF-MDR [38] proposes several options regarding links that are listed in LSAs, depending on the value of the *LSAFullness* parameter. With the minimum *LSAFullness* value, LSAs report only a minimal number of links, so that the network is still connected, but computed paths may then be longer than necessary. With a higher *LSAFullness* value, LSAs report more links, ensuring that the computed paths are shortest at the

expense of more overhead. With the maximum *LSAFullness* value, LSAs report all the links. However, one downside of OSPF-MDR in that respect, is that computed paths may use links that are not adjacencies.

OSPF-OR [37] optionally proposes that LSAs report only adjacencies established through *smart peering* (see Section IV-C), whereby a router becomes adjacent only with new neighbors that are not already reachable in their *shortest path tree* (SPT). This reduces the number of links that need to be advertized in LSAs, but typically yields longer paths.

OSPF extensions for MANET also use *hello redundancy reduction* mechanisms. *Incremental* hellos [37] and *differential* hellos [38] allow the routers to report only the changes noticed in the neighborhood over the last *HelloInterval*, instead of complete neighborhood information. Thus, if the topology is stable, most Hello packets will be significantly smaller in size. However, in doing this, transmission failures may cause loss of Hello synchronism and may take away node's ability to track neighborhood changes properly. In order to detect these cases, additional mechanisms check sequence number gaps in received Hello packets. Differential Hellos use a proactive synchronism recovery mechanism, while incremental Hellos make the receiver responsible for synchronism management. These mechanisms are also able to track less stable topologies, but do not offer significant overhead savings in such context [40].

## VI. Routing Table Calculation

On receiving a new router or network LSA, a router needs to rebuild its routing table from scratch [1], [2]. This process involves calculating

1) the *intra-area* routes for all OSPF areas to which the router belongs (typically using Dijkstra's *shortest path tree* (SPT) algorithm [62], [63] on the contents of router and network LSAs) and
2) the *inter-area* routes by examining the contents of all summary LSAs.
3) the *AS-external* routes by examining the contents of all ASE LSAs.

Typically, a *backbone* router may have up to a few hundred router/network LSAs and up to a few thousand summary/ASE LSAs in its link state database. Calculating intra-area routes using Dijkstra's algorithm (with a time complexity $O(n \times log(n))$) takes a few tens of milliseconds on modern routers [64], [65]. This time can be further reduced by using *dynamic* SPT algorithms (Section VI-B) rather than Dijkstra's algorithm. The examination of the summary/ASE LSAs may potentially take more time based on the number of such LSAs. If the routing table calculation results in change in the next hops for some destinations, this information needs to be conveyed to the line cards. Modern routers allow the routing table calculation and distribution of the next hops to the line cards to take place concurrently. Additionally, the order in which the next hops are installed can also be prioritized so that the important next hops (e.g., to VoIP gateway destinations) are installed first and made available for forwarding much earlier than the less important ones [64], [66]. Francois et al. [64] report the delay between the calculation of a next hop and intimation of this information to a line card to be of the order of 50ms on modern routers. Thus, a routing table calculation may keep the router CPU busy for a long time ($\sim$ 100ms).

In the rest of this section, we first describe the mechanisms used to avoid frequent routing table calculations in the event of a topology change. Subsequently, we describe Dijkstra's algorithm as well as the *dynamic* algorithms used to create the *shortest path trees* during a routing table calculation. Mechanisms described in the rest of this section are summarized in Table VI.

### A. Delays in Scheduling A Routing Table Calculation

A typical topology change, such as the failure of a router, may cause generation of several new LSAs (one for each router with which the failing router had an adjacency). The nature of the topology change (e.g., whether a link or a router goes down or comes up), the failure detection method used (e.g., Hello-based or hardware-based) and the values of OSPF parameters like the *HelloInterval* and *minLSInterval* determine the time range over which the affected routers would generate new LSAs. Assuming standard OSPF flooding, these LSAs will travel over the current shortest routes from their originating routers towards a particular router. The time range over which these LSAs arrive at a particular router depends on the time range over which these LSAs are generated, the exact routes followed by the LSAs on their way to the target router, the use of any pacing/flood delays by routers and the traffic loads on the links traversed by these LSAs (which determine the queueing delays and the probability of loss for these LSAs). Thus, there is a strong likelihood that the LSAs resulting from a topology change arrive at a target router over a significant time range.

If a router were to perform a routing table calculation immediately on receiving a new LSA, it may end up doing several such calculations in quick succession that may keep the router CPU busy for several hundred milliseconds and prevent it from doing other important tasks such as timely generation and processing of Hello messages. Such scenarios are undesirable as they may lead to network-wide routing instablity [22]. Hence, commercial routers typically do not perform a routing table calculation immediately on receiving a new LSA. Cisco routers, with older IOS releases, used a fixed value parameter *spfHoldTime*, henceforth called the hold time, to limit the frequency of routing table updates to once per 10 seconds. Additionally, there was an *spfDelay* (5 seconds) in doing a routing table calculation after receiving the first new LSA since the previous routing table calculation.

While fixed *spfDelay* and *spfHoldTime* parameters limit the number of routing table calculations and hence help avoid routing instability, they also slow down the router's convergence to the new topology. With their default values (5 seconds for *spfDelay* and 10 seconds for *spfHoldTime*), a router may take anywhere between 5 to 15 seconds to converge to a topology change after receiving a new LSA. To balance the needs for fast convergence and routing stability, Cisco routers with post 12.2(14)S release IOS use a simple exponential

| Mechanism | Description | Pros/Cons |
|---|---|---|
| Fixed hold time | Fixed delay (the hold time) enforced between successive routing table calculations. | Avoids too many routing table calculations following a topology change. Good for routing stability but delays delays the convergence process. |
| SPF throttling [67] | Hold time initially small. Receipt of one or more LSAs during a hold time causes the hold time following the next routing table calculation to double up to a certain maximum. If no LSA received during a hold time, it is reset to its initial small value. | Fast convergence to topology changes generating a small number of LSAs. Avoids too many routing table calculations for topology changes generating a large number of LSAs. Susceptible to premature reset of hold time if no LSA received during the current hold time duration. |
| SPF throttling with quiet period [68] | Similar to SPF throttling except that the hold time is reset to its initial small value only if no LSA received during a large quiet period. | Retains the pros of SPF throttling. No premature reset of hold time. |
| Juniper scheme [69] | First few routing table calculations done with a small hold time. Subsequent calculations done with a large hold time. If no LSA received during a large hold time, it is reset to the small value. | Fast convergence to most topology changes with a few routing table calculations. Limits frequency of calculations for large scale topology changes. |
| LSA correlation [68] | LSAs correlated to identify the underlying topology change. Routing table calculation done on identifying the topology change. | Allows fast convergence to a topology change with minimal number of routing table calculations. |
| Dynamic SPT algorithms [5] [71]–[74] | Based on correcting the existing SPT rather than creating a new SPT from scratch. Often available in commercial routers [70] | Moderate improvement in routing table calculation times since SPT calculation is not the most time consuming step in the routing table calculation. |

TABLE VI
OPTIMIZATIONS IN ROUTING TABLE CALCULATIONS

backoff scheme to adjust the hold time between successive routing table calculations [67]. In this scheme, referred to as *SPF throttling* in Cisco literature, the hold time between successive routing table calculations is initially set to a small value. However, receipt of one or more LSAs during a hold time causes its value to double up to a certain maximum. Thus, quick convergence can be achieved for topology changes that lead to generation of only a small number of LSAs (e.g., individual link failures). For topology changes leading to generation of a large number of LSAs that arrive at a router over an extended time interval, the hold time is expected to quickly reach its maximum value and thus limit the number of routing table calculations at the expense of some convergence delay. However, this scheme is susceptible to undesirable resets in the hold time to its initial small value if no LSA arrives for a time duration equal to the current hold time value. Such undesirable hold time resets can be avoided by requiring a relatively large *quiet* period, during which no new LSA must arrive, before the hold time is allowed to return to its small initial value [68]. Rather than doubling the hold time value for continuous LSA arrivals, Juniper routers use two fixed values for the hold time [69]. A certain number (by default 3) of routing table calculations are performed with a small hold time value (by default 200 ms). If new LSAs continue to arrive, the subsequent routing table calculations take place with a large hold time value (by default 5 seconds). The hold time is reset to the small value if the router does not receive a new LSA during the large hold time following a routing table calculation. The underlying assumption behind this scheme is that, for most topology changes, the new LSAs would arrive at a router within few hundred milliseconds. Fast convergence to such topology changes can be achieved by using a small hold time value. Large scale topology changes, that result in continuous arrival of new LSAs over a large time interval, would cause the large hold time value to come in effect and limit the frequency of routing table calculations.

For most networks, configuring a hold time scheme to achieve a good tradeoff between the convergence delay and the number of routing table calculations is not trivial. A particular hold time configuration may result in either too much convergence delay or too many routing table calculations for some topology changes. These considerations have motivated an alternate method to schedule routing table calculations. In this method, called *LSA correlation* [68], an individual LSA does not trigger a routing table calculation. Rather, the individual LSAs are *correlated* to determine the topology change that caused their generation. A routing table calculation can be performed as soon as all the LSAs, generated following a topology change, have arrived at the router and the topology change has been identified. Thus, the LSA Correlation approach avoids not only the hold time related delays in convergence but also the unnecessary routing table calculations performed with only a partial set of LSAs. Large scale topology changes, e.g., simultaneous reboot of a large number of routers, can be dealt by additionally enforcing a dynamic wait time between successive routing table calculations.

### B. Dynamic SPT Algorithms

Dijkstra's SPT algorithm can be categorized as a *static* algorithm since it builds the shortest path tree (SPT) *from scratch* whenever it is executed. Initially, the SPT contains just the root node, i.e., the node executing the algorithm. The nodes directly connected to the root are assigned a *distance* (from the root) same as the cost of their link to the root while other nodes are assigned infinite distance. In each iteration, the node with minimum distance from the root is added to the SPT and the distance associated with its out-of-tree neighbors is updated to be the smaller of their original distance and their distance from the root via the node being added to the tree. The iterations continue until all the nodes have been included in the SPT.

Generally, the shortest path trees before and after a topology change have a significant overlap. For example, when a link *X:Y* goes down or increases in cost, only the nodes in the subtree connected to the existing SPT via link *X:Y* need to be re-attached to the SPT. The position of other nodes in the SPT

is not affected by this topology change event. Even within the subtree connected to the existing SPT via link *X:Y*, it may be possible to attach many *branches* to the new SPT without any modification. Figure 6 shows one such example where the subtree rooted at the failed link requires very little modification for reattachment to rest of the SPT. When a link *X:Y* decreases in cost, the subtree connected to the existing SPT via this link would not be affected.

Rather than calculating the new SPT from scratch following a topology change, the *dynamic* SPT algorithms *correct* the existing SPT. Many such algorithms are generalizations of Dijkstra's algorithm [5], [71]–[73] in the sense that the next node added to the SPT is the one with smallest distance to the root. Another strategy, shown to result in fewer computations [74], is to pick the node with most decrease (or least increase) in its distance to the root following the topology change. Many commercial OSPF implementations (e.g., [70]) now offer the use of a dynamic algorithm for SPT calculation as a configurable option.

## VII. Graceful Restart

Modern routers have a distributed architecture with the central CPU running the routing protocols such as OSPF while the line cards handle the task of packet forwarding (i.e., moving the packets from one interface to another). This clear separation of the control and forwarding planes allows a router to continue the packet forwarding functionality even if the routing plane is being restarted due to a planned activity (e.g., a software upgrade). Normally, the control plane reboot would cause the neighbor routers to break adjacency with this router. The neighbor routers would generate new LSAs that would be flooded throughout the area and all routers in the area would need to perform (multiple) routing table calculations on receiving these LSAs. A few minutes later, once the control plane reboot has completed, the neighbor routers would re-establish adjacency with this router and the whole LSA flooding/routing table calculations sequence would be repeated. Since the forwarding plane functions normally, even though the control plane reboots, it is not necessary for the network to undergo this commotion.

With graceful restart [75], a router, whose control plane is about to restart and whose forwarding plane is functioning normally, sends a *grace* LSA to its adjacent neighbors, declaring its intention to perform a graceful restart within a specified grace period. Such a router is called the *initiating* router. The grace LSAs are *link-local* in scope, i.e., the adjacent neighbors do not flood these LSAs. The adjacent neighbors, also known as *helpers*, continue to list the initiating router as fully adjacent in their LSAs during the grace period. Thus, the helper neighbors hide the control plane reboot of the initiating router from rest of the network during the grace period. Once the control plane restarts, the initiating router goes through normal adjacency establishment procedure with all the helpers, at the end of which the initiating router and the helpers regenerate their router/network LSAs. The graceful restart mechanism is available in commercial routers [76], [77].

Any change in the network topology during the grace period would cause the helper routers to abandon graceful restart and generate their router/network LSAs showing the breakdown of adjacency with the initiating router. This is done because the forwarding tables in the initiating router may no longer be consistent with the new network topology. It may be possible to continue using the initiating router for packet forwarding in the changed topology as long as any packet loss and/or routing loops are avoided [78], [79]. The graceful restart may optionally be used for unplanned control plane reboots as well. In such cases, the Grace LSAs are sent soon after the restart of the control plane (rather than before the control plane restart as in case of planned reboots). However, the graceful restart is primarily designed for planned control plane reboots.

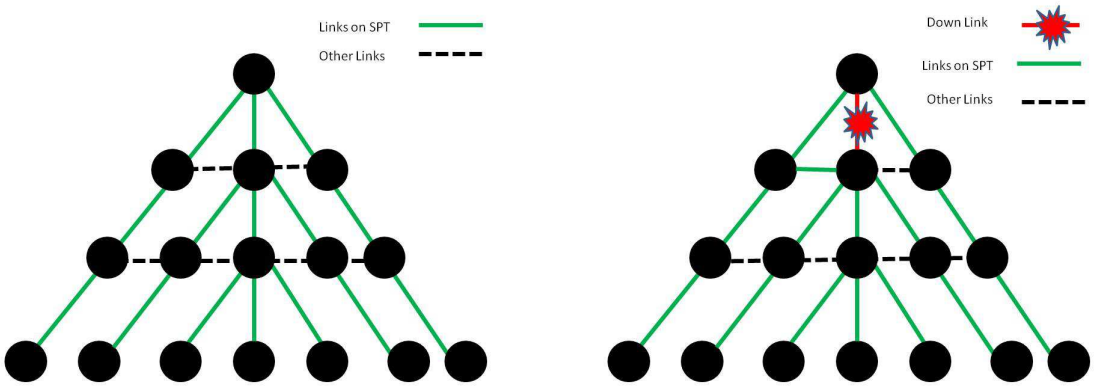
## VIII. Proactive Approaches to Failure Recovery

OSPF failure recovery is *reactive* in nature, i.e., the new routes to be used in case of a failure event are determined only after the failure has occurred. A reactive approach is intrinsically much slower than a *proactive* approach where the new route is computed in advance before the failure happens. Hence, given the need for fast failure recovery, there has been a strong motivation to develop proactive failure recovery mechanisms in the Internet. Proactive approaches aim to achieve very fast (sub-50ms) failure recovery. In this section, we describe two such mechanisms: *multi-protocol label switching fast reroute* (MPLS FRR) and *IP/label distribution protocol fast reroute* (IP/LDP FRR or simply IP FRR). Key proposals under these two categories are summarized in Table VII. A more comprehensive survey of proactive failure recovery mechanisms can be found in [80]. These mechanisms are an integral part of various *traffic engineering* approaches [81]–[83] that aim for traffic load distribution in a desired fashion across the network. See [84] for a discussion of failure recovery mechanisms above the network layer.

### A. Multi-protocol Label Switching Fast Reroute (MPLS FRR)

MPLS [27], [28] is a *connection-oriented* fast forwarding mechanism based on *labels* instead of the *longest prefix match* forwarding principle. MPLS resides between layers 2 and 3 of the protocol stack. It does not replace IP routing but provides enhanced services such as *traffic engineering* (TE), which includes balancing traffic loads across links in a network, and *fast reroute* (FRR). An OSPF extension [91] has been standardized to support MPLS services by flooding relevant link information in special TE LSAs. This information is used by MPLS to establish tunnels called *label switched paths* (LSPs).

MPLS FRR is based on the use of the *backup* LSPs to protect the *primary* LSPs in case of failures. It is a *local* protection mechanism, where a *point of local repair*, adjacent to the failure, switches the traffic from the primary LSPs affected by the failures to the backup LSPs [85]. A backup LSP allows the traffic to bypass the failure and merges with the primary LSP at a *merge node*. Several protection schemes, such as 1+1, 1:1, 1:n and m:n, are possible. In both 1+1 and 1:1 configurations, a separate backup LSP is dedicated to




(a) The shortest path tree before the link failure

(b) The shortest path tree after the link failure

Fig. 6. When a link fails, only the nodes in the sub-tree rooted at the failed link in the original shortest path tree need reattachment. In this example topology, all links carry equal weight.

| **Mechanism** | **Description** | **Pros/Cons** |
|---|---|---|
| MPLS FRR [85] | Based on the use of backup LSPs to protect the primary LSPs in case of failure. | Allows sub 50ms failure recovery. Does not scale well. Needs MPLS infrastructure. |
| MPLS FRR Mechanisms | | |
| 1+1<br>1:1<br>1:n<br>m:n | Data traffic sent along both primary and backup LSPs.<br>Data traffic sent along the backup LSP only after failure detection.<br>One backup LSP shared by n primary LSPs.<br>m backup LSPs shared by n primary LSPs. | |
| IP FRR [86] | Based on a router directing traffic to a precomputed backup path in case of a local failure. | Fast failure recovery as in MPLS FRR. Pure IP solution. |
| IP FRR Mechanisms | | |
| Equal cost multi path | The router sends the packet along an alternate path with same cost as the failed path. | Part of standard OSPF. |
| Loop free alternate [87] | The router sends the packet to an immediate neighbor that has a *safe* path to the packet's destination. | |
| Multihop repair paths [86] | The router sends the packet to a neighbor, two or more hops away, that has a safe path to the packet's destination. Used when ECMP/LFA not available. | Difficult to determine and invoke. Typically needed for only a few destinations. |
| Multihop repair path mechanisms | | |
| Not-via address [88] | A special address assigned to a router prohibiting reaching the router via a particular neighbor. When a router detects failure to reach the next hop, router $B$, for a packet, it forwards the packet to "not-via B" address of router $C$, the next hop from router $B$. | |
| U turn alternates [89] | A neighbor that considers the router as the next hop for the packet's destination and has an LFA for the destination that does not lead the packet back to the router. On detecting a failure to reach the next hop, the router sends the packet to a U turn alternate. | |
| Tunnels [90] | The router *tunnels* the packet towards a router from where it can reach its destination via normal IP forwarding. | |

TABLE VII
MPLS/IP FAST REROUTE MECHANISMS

protect each primary LSP. The difference between 1+1 and 1:1 configurations is that, in the absence of failures, both primary and backup LSPs carry identical traffic in 1+1 configuration whereas in 1:1 configuration, the backup LSP does not carry any traffic or carries low priority traffic. In 1:n configuration, a dedicated backup LSP is shared by $n$ primary LSPs. The m:n configuration is the general case where $m$ dedicated backup LSPs are shared by $n$ primary LSPs ($m <= n$).

Details on how the LSPs are pre-established and mechanisms that are used to redirect traffic to the backup LSPs upon failure detection can be found in [85]. Being a local recovery mechanism, MPLS FRR avoids the need to convey the failure notification to the source router of the primary LSP. Hence, the main delay in failure recovery is the failure detection time and thus it is possible to achieve very fast (less than 50ms) failure recovery times.

### B. IP Fast Reroute (IP FRR)

MPLS-FRR requires IP networks to have MPLS infrastructure. Further, MPLS FRR does not scale well as the routing domain grows in size. In the worst case, MPLS-FRR uses $O(nk)$ and $O(nk^2)$ LSPs to handle link and node failures respectively, where $n$ is the number of nodes and $k$ is the number of links in the network [92]. These concerns led to

a renewed interest in proactive ways to mitigate the impact of large IGP failure recovery times in pure IP networks. Such schemes, broadly classified as *IP fast reroute* (IP FRR) [86], [93], try to avoid loss of data caused by inconsistencies in router FIBs during the time IGP convergence is taking place.

IP FRR is similar to MPLS FRR in the sense that both sets of schemes depend on pre-computed backup routes that allow local failure recovery by routers detecting the failure without the need to immediately inform other routers about the failure. IP FRR differs from MPLS FRR since it does not use MPLS LSPs as backup routes. In IP FRR, a router pre-calculates the repair paths to be used for each possible local failure. If the router knows about multiple *equal cost multi-paths* (ECMP) for a destination and some of these paths do not traverse through the failure, such paths can trivially be used as the repair paths. In the absence of such paths, the router looks for a directly connected neighbor that has a safe path, i.e., a path that does not travel through the failure, to the destination. Such a path through a directly connected neighbor is referred to as the *loop free alternate* (LFA) path [87]. Figure 7 shows the *reverse* SPT rooted at a particular destination in an example topology and the ECMP, LFA and U turn alternate (discussed later) based backup paths available to two nodes in the network.

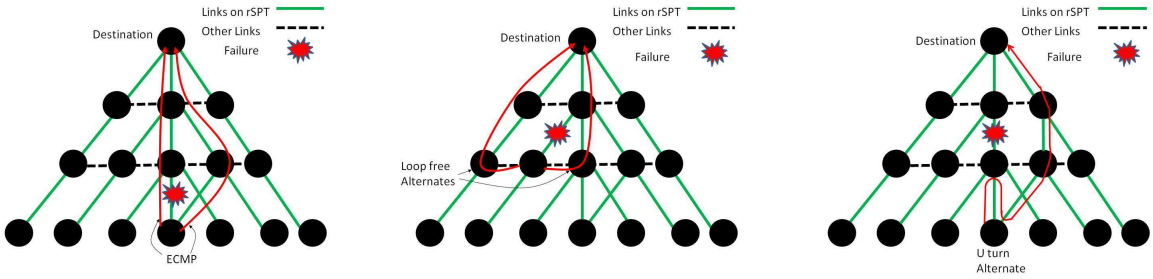
If ECMP/LFA paths are not available, the router looks for a neighbor, two or more hops away, that has a safe path to the destination. Repair paths through such neighbors are called *multi-hop repair* paths [86]. Several IP FRR mechanisms, using multi-hop repair paths, have been proposed recently:

- One proposed mechanism involves the use of *not-via* address [88], which is a special address assigned to a router prohibiting reaching the router via a particular neighbor. Suppose router $A$ considers router $B$ and router $C$ to be the next hop and the next-to-next hop for destination $D$ of a packet. If router $A$ notices that router $B$ is unreachable, it can forward the packet to the "not-via B" address of router $C$. A packet forwarded to the "not-via B" address of router $C$ must not pass through router $B$ on its way to router $C$ and thus can avoid any failure involving router $B$.
- Under another mechanism, called *U turn alternates* [89], on detecting the failure of the primary next hop for a packet going to destination $D$, router $A$ sends the packet to a neighbor $B$ (the U turn alternate) that considers router $A$ as the primary next hop for destination $D$ and has a loop free alternate for destination $D$ that does not cause the packet to pass through router $A$. Router $B$ identifies such *u-turned* packets implicitly (since they arrived from router $A$, the primary next hop for destination $D$) or explicitly (via a special label in the packet header) and forwards them to the loop free alternate it has for destination $D$. Figure 7(c) shows an example failure scenario and a U turn alternate based backup path to deal with this failure.
- Nelakuditi et al. [94] suggest a similar approach, where a router infers the possible failure of certain links on receiving a packet from an *unusual* interface. For example, if router $B$ is not a part of the shortest path from neighbor $A$ to destination $D$, router $B$ can infer the possible failure of certain links on receiving a packet destined for $D$ from $A$. Router $B$ can then forward the packet along a path that avoids these possibly down links.
- Cicic et al. [95] suggest that the routers maintain *multiple routing configurations* for the topology, i.e., multiple sets of link weights, such that the routes calculated using a particular configuration avoids the use of a particular set of links/routers. Each router also maintains multiple FIBs corresponding to each configuration. When a router notices the failure to reach a neighbor, it *marks* the packets, that would otherwise be forwarded to this neighbor, so that these packets are forwarded using the FIB/configuration that does not use the failed link/router. OSPF now supports the existence of multiple routing configurations, referred to as *multi-topologies*, in the network [96].
- Xi and Chao [97] present an integer linear programming (ILP) formulation as well as a sequential search based algorithm for the problem of finding backup next hops to be used by routers in case of a link/router failure along the primary path such that the number of routers using the backup next hops is minimized. In this proposal, a router decides that a failure has occurred along the primary path when it receives a packet from its primary next hop for the packet's destination.
- Bryant et al. [90] suggest that, on detecting the failure to reach the next hop for a packet, the router should *tunnel* the packet towards a router from where it can continue towards its destination via normal IP forwarding. Any tunneling mechanism, such as IP-in-IP [98] or GRE [99], can be used for this purpose.

The multi-hop repair paths are intrinsically more difficult to determine as well as invoke. However, it is expected that, on a typical IP network, only a small fraction of destinations would require such repair paths. Note that the repair paths are only used temporarily while the IGP (OSPF) convergence is taking place. When IGP convergence ends, packets will be forwarded using the new route produced by the IGP.

Redirection of affected traffic along alternate routes following a link failure may create significant overload in certain parts of the network. Careful assignment of link weights may help to avoid this situation. However, changing the link weights throughout the network *after* a failure is not feasible if the failure is transient in nature. Hence, Nucci et al. [100] suggest that the set of link weights for a network should be determined in such a manner that the link overloading can be avoided both during normal operation and during a transient single link failure. They also present a tabu-search heuristic [10] that finds a good set of link weights by evaluating the impact of *all* possible link failures for each sampled set of link weights. For very large networks, the computational complexity of this task can be significanly reduced by evaluating the impact

[10]Determining the optimal set of link weights in an OSPF network so as to achieve specific traffic engineering objectives is an NP-hard problem [101]. Hence, the search for good link weights is typically based on heuristics of different sorts. Ghazala et al. [102] presents a survey of some of these heuristics.




(a) Equal cost multi-paths (b) Loop free alternates (c) U turn alternate

Fig. 7. Equal cost multi-paths, loop free alternates and U turn alternate used in IP FRR. In this example topology, all links carry equal weight.

of only *critical* links. Sridharan and Guerin [103] suggest a technique for identifying such critical links in a network and report significant reduction in the time required to determine good link weights especially for large networks.

Additionally, there are several alternate proposals to prevent transient routing loops. Francois et al. [104] suggest incrementing the routing cost of a failed link in steps, rather than immediately advertising an infinite cost, in order to prevent transient routing loops. The underlying assertion is that a cost increment of $x$ for the failed link can create transient routing loops in only that region of network that has a cyclic cost of less than or equal to $x$. Since transient routing loops are caused by routers closer to the failed link/router updating their FIBs before the routers farther away, such loops can also be prevented by requiring a router to update its FIB only after its children routers in the shortest path tree rooted at the failed device [105]. Another similar approach, applicable in situations where FIB update times are significant, is to require that all routers switch to the new FIB corresponding to the new topology at exactly the same time once all routers have calculated the new FIB [106]. Such an approach requires precise time synchronization among all routers in the network.

## IX. Conclusion

OSPF is one of the most widely deployed protocols in the Internet. In twenty years of its existence, this protocol has proved to be remarkably flexible in meeting the changing demands of the routing infrastructures. The protocol's longevity, to a large extent, is a tribute to its sound design principles. The original protocol design focussed on scalability and robustness against failures. These objectives were achieved by dividing the routing domain into multiple areas and limiting the processing overhead of the protocol. These features allowed large OSPF networks to exist even with not-so-powerful routers and to avoid routing meltdowns even in face of multiple and frequent topology changes.

Although failure recovery within few tens of seconds was considered sufficiently fast in the beginning, the situation has changed with increasing commercial use of the Internet. Any service deterioration/outage lasting more than few seconds can no longer be tolerated. In fact, real time applications, e.g., voice over IP, require sub-second recovery times. However, the need for fast convergence times has to be met without increasing the processing overhead of the protocol, which adversely impacts the routing stability. Although the routers are much more powerful and reliable today than before, the tremendous increase in the scale of the networks (in terms of the number of routers and their interconnectivity) means that the need to limit the processing overhead of the protocol continues to be critical even today. In fact, there is a clear need to further reduce the protocol overhead to make it work in new environments such as MANETs.

While optimizing OSPF's performance, either in terms of the convergence speed or the processing overhead, one has to avoid making simplistic assumptions about the protocol's operation. For example, as pointed out in Section IV-C, reachability to a neighbor in the shortest path tree can not be assumed to imply synchronized databases. Any modification to OSPF operation must not compromise the protocol's correctness in any scenario, even if such a scenario is highly improbable. In particular, one has to pay close attention to all possible race conditions that may arise in the distributed operation of the protocol.

In this paper, we surveyed recent efforts to improve the convergence speed of the OSPF protocol and eliminate the redundancies in its operation to enhance its scalability. The needs for fast convergence and scalability in link state routing protocols continue to challenge the research community as the routing domains grow larger and more complex. One possible avenue for reconciling these two often conflicting needs may be to examine the area-level organization of the routing domains. Small area sizes are good from convergence speed and scalability perspectives but may be difficult to achieve in a large routing domain given the constraint to organize the areas in the *hub and spokes* fashion. Clearly, there is a case for examining the possibility of removing this constraint and allowing arbitrary inter-connection among areas, similar to the way *autonomous systems* are interconnected. Another related direction for future research is to examine *dynamic organization* of routers into areas, where the constituent routers would run a distributed algorithm to decide whether to merge two areas into one or split an area into two.

Link state routing protocols are an essential component of today's Internet and will continue to serve this role in the foreseeable future. However, it will be interesting to observe how link state protocols will influence tomorrow's Internet, particularly in view of the Internet's next anticipated phenomenal expansion in scale: from millions of *computers* to billions

of energy/CPU/memory constrained *sensors* communicating with each other wirelessly. In a large network of wireless sensors, a hybrid approach is conceivable where a link state routing protocol is used inside small routing domains, which in turn are inter-connected using a distance vector approach. Clearly, to be useful in such context, the link state routing protocols will need further improvement in terms of scalability.

## REFERENCES


[1] J. Moy, "OSPF version 2," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 2328, April 1998.

[2] R. Coltun, D. Ferguson, J. Moy, and A. Lindem, "OSPF for IPv6," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 5340, July 2008.

[3] J. Hawkinson and T. Bates, "Guidelines for creation, selection and registration of an autonomous system (AS)," Internet Engineering Task Force, Request For Comments (Best Current Practice) RFC 1930, March 1996.

[4] J. McQuillan, "The birth of link-state routing," *IEEE Annals of the History of Computing*, vol. 31, no. 1, January/March 2009.

[5] J. McQuillan, I. Richer, and E. Rosen, "The new routing algorithm for the ARPANET," *IEEE Transactions on Communications*, vol. 28, no. 5, May 1980.

[6] "ISO 10589: Intermediate system to intermediate system routing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service," 2002.

[7] J. Soldatos, E. Vayias, and G. Kormentzas, "On the building blocks of quality of service in heterogeneous IP networks," *IEEE Communication Surveys and Tutorials*, vol. 7, no. 1, 2005.

[8] G. Scheets, M. Parperis, and R. Singh, "Voice over the internet: a tutorial discussing problems and solutions associated with alternative transport," *IEEE Communication Surveys and Tutorials*, vol. 6, no. 2, 2004.

[9] L. Hanzo and R. Tafazolli, "A survey of QoS routing solutions for mobile ad hoc networks," *IEEE Communication Surveys and Tutorials*, vol. 9, no. 2, 2007.

[10] J. Moy, *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley Professional, 1998.

[11] T. Thomas, *OSPF Network Design Solutions, Second Edition*. Cisco Press, 2003.

[12] C. Boutremans, G. Iannaccone, and C. Diot, "Impact of link failures on VoIP performance," in *Proc. The 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video, Miami, USA*, 2002.

[13] B. Choi, S. Song, G. Koffler, and D. Medhi, "Outage analysis of a university campus network," in *Proc. 16th International Conference on Computer Communication and Networks (ICCCN)*, August 2007.

[14] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of internet stability and backbone failures," in *Proc. The 29th International Symposium on Fault-Tolerant Computing*, 1999.

[15] A. Markopoulo, G. Iannaccone, S. Bhattacharya, C. Chua, Y. Ganjali, and C. Diot, "Characterization of failures in an operational IP backbone network," *IEEE/ACM Transactions on Networking*, vol. 16, no. 4, August 2008.

[16] A. Medem, R. Teixeira, N. Feamster, and M. Meulle, "Determining the causes of intradomain routing changes," University Pierre and Marie Curie," Technical Report, 2009, http://www-rp.lip6.fr/ medem/inmworkshop_tech_report.pdf.

[17] Y. Ganjali, S. Bhattacharyya, and C. Diot, "Limiting the impact of failures on network performance," Sprint ATL Tech. Res. Rep., Tech. Rep. RR04-ATL-020666, 2003.

[18] B. Wu, P. Ho, K. Yeung, J. Tapolcai, and H. Mouftah, "Optical layer monitoring schemes for fast link failure localization in all-optical networks," *Accepted for publication in IEEE Communication Surveys and Tutorials*.

[19] J. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical SONET-SDH and MPLS*. Morgan Kaufmann Publishers, Inc., 2004.

[20] C. Alaettinoglu, V. Jacobson, and H. Yu, "Towards millisecond IGP convergence," in *NANOG 20*, October 2000.

[21] A. Basu and J. Riecke, "Stability issues in OSPF routing," *Computer Commununication Review*, vol. 31, no. 4, October 2001.

[22] G. Choudhury, "Prioritized treatment of specific OSPF version 2 packets and congestion avoidance," Internet Engineering Task Force, Request For Comments (Best Current Practice) RFC 4222, October 2005.

[23] G. Choudhury, G. Ash, V. Manral, A. Maunder, and V. Sapozhnikova, "Prioritized treatment of specific OSPF packets and congestion avoidance: algorithms and simulations," Technical Report, AT&T, Tech. Rep., August 2003.

[24] M. Goyal, K. Ramakrishnan, and W. Feng, "Achieving faster failure detection in OSPF networks," in *Proc. IEEE International Conference on Communications (ICC 2003)*, 2003, pp. 296–300.

[25] R. Aggarwal, "Applications of bidirectional forwarding detection (BFD)," May 2004, presented at RIPE-48.

[26] K. Kompella and G. Swallow, "Detecting multi-protocol label switched (MPLS) data plane failures," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 4379, February 2006.

[27] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 3031, January 2001.

[28] B. Davie and Y. Rekhter, *MPLS: Technology and Applications*. Morgan Kaufmann, 2000.

[29] D. Katz and D. Ward, "Bidirectional forwarding detection (BFD)," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 5880, June 2010.

[30] ——, "Generic application of bidirectional forwarding detection (BFD)," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 5882, June 2010.

[31] R. Aggarwal, K. Kompella, T. Nadeau, and G. Swallow, "BFD for MPLS LSPs," INTERNET-DRAFT, draft-ietf-bfd-mpls-07.txt, June 2008, (work in progress).

[32] J. Doyle, "Reducing link failure detection time with BFD," Network World, December 2007, http://www.networkworld.com/community/node/23380.

[33] R. Ogier, "OSPF database exchange summary list optimization," Internet Engineering Task Force, Request For Comments (Informational) RFC 5243, May 2008.

[34] E. Baccelli, T. Clausen, and P. Jacquet, "OSPF database exchange and reliable synchronization in mobile ad hoc networks," in *Proc. IASTED Conference on Wireless Networks and Emerging Technologies (WNET), Banff, Canada*, 2004.

[35] M. Goyal, W. Xie, S. H. Hosseini, K. Vairavan, and D. Rohm, "Improving OSPF dynamics on a broadcast LAN," *Simulation*, vol. 82, no. 2, pp. 107–129, 2006.

[36] E. Baccelli, P. Jacquet, D. Nguyen, and T. Clausen, "OSPF multipoint relay (MPR) extension for ad hoc networks," Internet Engineering Task Force, Request For Comments (Experimental) RFC 5449, February 2009.

[37] A. Roy and M. Chandra, "Extensions to OSPF to support mobile ad hoc networking," Internet Engineering Task Force, Request For Comments (Experimental) RFC 5820, March 2010.

[38] R. Ogier and P. Spagnolo, "Mobile ad hoc network MANET extension of OSPF using connected dominating set CDS flooding," Internet Engineering Task Force, Request For Comments (Experimental) RFC 5614, August 2009.

[39] S. Venkatesh, "Smart adjacency establishment in OSPF routing protocol," Master's thesis, University of Wisconsin - Milwaukee, 2006.

[40] E. Baccelli, J. Cordero, and P. Jacquet, "Multi-point relaying techniques with OSPF on ad hoc networks," in *Proc. IEEE International Conference on Systems and Networks Communications (ICSNC)*, September 2009.

[41] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Internet Engineering Task Force, Request For Comments (Experimental) RFC 3626, October 2003.

[42] P. Murphy, "The OSPF not-so-stubby area (NSSA) option," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 3101, January 2003.

[43] J. Moy, "Multicast extensions to OSPF," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 1584, March 1994.

[44] D. Katz, "Why are we scared of SPF? IGP scaling and stability," in *NANOG 25*, October 2002.

[45] G. Choudhury, "Models for IP/MPLS routing performance: Convergence, fast reroute and QoS impact," in *Keynote Speech at ITCOM Conference on Performance, QoS and Control of Next Generation Communication Networks, Philadelphia, USA*, October 2004.

[46] Cisco, "OSPF link-state advertisement (LSA) throttling," http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/fsolsath.html.

[47] ——, "OSPF update packet-pacing configurable timers," http://www.cisco.com/en/US/docs/ios/12_2t/12_2t4/feature/guide/ftospfct.html.

[48] P. Pillay-Esnault, "OSPF refresh and flooding reduction in stable topologies," Internet Engineering Task Force, Request For Comments (Informational) RFC 4136, July 2005.

[49] R. Rastogi, Y. Breitbart, M. Garofalakis, and A. Kumar, "Optimal configuration of OSPF aggregates," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 181–194, 2003.

[50] A. Shaikh, D. Wang, G. Li, J. Yates, and C. Kalmanek, "An efficient algorithm for OSPF subnet aggregation," in *Proc. International Conference on Network Protocols (ICNP)*, November 2003.

[51] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communications of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[52] B. R. Bellur and R. G. Ogier, "A reliable, efficient topology broadcast protocol for dynamic networks," in *Proc. IEEE INFOCOM*, 1999, pp. 178–186.

[53] P. A. Humblet and S. R. Soloway, "Topology broadcast algorithms," *Computer Networks and ISDN Systems*, vol. 16, no. 3, pp. 179–186, 1989.

[54] J. Moy, "Extending OSPF to support demand circuits," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 1793, April 1995.

[55] P. Baran, S. Boehm, and P. Smith, "On distributed communication," Rand Corp., Santa Monica, California," Technical Report, 1964.

[56] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, and T. Przygienda, "QoS routing mechanisms and OSPF extensions," Internet Engineering Task Force, Request For Comments (Experimental) RFC 2676, August 1999.

[57] G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi, "Quality of service based routing: a performance perspective," *SIGCOMM Comput. Commun. Rev.*, vol. 28, no. 4, pp. 17–28, 1998.

[58] M. Takashi, K. Takashi, and A. Michihiro, "Enhancing the network scalability of link-state routing protocols by reducing their flooding overhead," in *Proc. IEEE Workshop on High Performance Switching and Routing (HPSR)*, June 2003, pp. 263–268.

[59] E. Baccelli, T. Clausen, U. Herberg, and C. Perkins, "IP links in multihop ad hoc networks?" in *Proc. IEEE International Conference on Software Telecommunications and Computer Networks (SOFTCOM)*, September 2009.

[60] "Part 11: Wireless LAN medium access control and physical layer specifications," *IEEE Std 802.11-2007 (Revision of IEEE Std 802.11-1999)*, 12 2007.

[61] L. Kleinrock and F. Tobagi, "Packet switching in radio channels: Part 1 - carrier sense multiple-access modes and their throughput-delay characteristics," *IEEE Trans. on Communications*, vol. 23, no. 12, pp. 1400–1416, December 1975.

[62] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[63] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009.

[64] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving subsecond IGP convergence in large IP networks," *Computer Commun. Rev.*, vol. 35, no. 3, July 2005.

[65] A. Shaikh and A. Greenberg, "Experience in black-box OSPF measurement," in *Proc. 1st ACM SIGCOMM Workshop on Internet Measurement*, November 2001, pp. 113–125.

[66] Juniper, "Applying policies to OSPF routes," http://www.juniper.net/techpubs/en_US/junos9.6/information-products/topic-collections/config-guide-routing/routing-applying-policies-to-ospf-routes.html.

[67] Cisco, "OSPF shortest path first throttling," http://www.cisco.com/en/US/products/sw/iosswrel/ps1838/products_feature_guide09186a0080134ad8.html.

[68] M. Goyal, W. Xie, M. Soperi, H. Hosseini, and K. Vairavan, "Scheduling routing table calculations to achieve fast convergence in OSPF protocol," in *Proc. IEEE Broadnets*, September 2007.

[69] Juniper, "Configuring SPF options for OSPF," http://www.juniper.net/techpubs/en_US/junos9.6/information-products/topic-collections/config-guide-routing/routing-configuring-spf-options-for-ospf.html.

[70] Cisco, "OSPF incremental SPF." [Online]. Available: http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/ospfispf.pdf

[71] P. Franciosa, D. Frigioni, and R. Giaccio, "Semi-dynamic shortest paths and breadth-first search in digraphs," in *Proc. The 14th Annual Symposium on Theoretical Aspects of Computer Science*, 1997, pp. 33–46.

[72] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni, "Fully dynamic output bounded single source shortest path problem," in *Proc. ACM-SIAM Symposium on Discrete Algorithms*, January 1996.

[73] G. Ramalingam and T. Reps, "An incremental algorithm for a generalization of the shortest-path problem," *Journal of Algorithms*, vol. 21, pp. 267–305, 1996.

[74] P. Narváez, K.-Y. Siu, and H.-Y. Tzeng, "New dynamic SPT algorithm based on a ball-and-string model," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 706–718, 2001.

[75] J. Moy, P. Pillay-Esnault, and A. Lindem, "Graceful OSPF restart," Internet Engineering Task Force, Request for Comments (Standards Track) RFC 3623, November 2003.

[76] Juniper, "Configuring graceful restart for OSPF," http://www.juniper.net/techpubs/en_US/junos9.6/information-products/topic-collections/config-guide-routing/routing-configuring-graceful-restart-for-ospf.html.

[77] Cisco, "NSF–OSPF (RFC 3623 OSPF graceful restart)," http://www.cisco.com/en/US/docs/ios/12_0s/feature/guide/gr_ospf.html.

[78] A. Shaikh, R. Dube, and A. Varma, "Avoiding instability during graceful shutdown of OSPF," in *Proc. IEEE INFOCOM*, June 2002.

[79] ——, "Avoiding instability during graceful shutdown of multiple OSPF routers," *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, June 2006.

[80] A. Raj and O. Ibe, "A survey of IP and multiprotocol label switching fast reroute schemes," *Comput. Netw.*, vol. 51, no. 8, pp. 1882–1907, 2007.

[81] N. Wang, K. Ho, G. Pavlou, and M. Howarth, "An overview of routing optimization for internet traffic engineering," *IEEE Communication Surveys and Tutorials*, vol. 10, no. 1, 2008.

[82] Y. Lee and B. Mukherjee, "Traffic engineering in next-generation optical networks," *IEEE Communication Surveys and Tutorials*, vol. 6, no. 3, 2004.

[83] O. Younis and S. Fahmy, "Constraint-based routing in the internet: basic principles and recent research," *IEEE Communication Surveys and Tutorials*, vol. 5, no. 1, 2003.

[84] N. Ayari, D. Barbaron, L. Lefevre, and P. Primet, "Fault tolerance for highly available internet services: concepts, approaches, and issues," *IEEE Communication Surveys and Tutorials*, vol. 10, no. 2, 2008.

[85] P. Pan, G. Swallow, and A. Atlas, "Fast reroute extensions to RSVP-TE for LSP tunnels," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 4090, May 2005.

[86] M. Shand and S. Bryant, "IP fast reroute framework," Internet Engineering Task Force, Request For Comments (Informational) RFC 5714, January 2010.

[87] A. Atlas and A. Zinin, "Basic specification for IP fast reroute: Loop-free alternates," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 5286, September 2008.

[88] M. Shand, S. Bryant, and S. Previdi, "IP fast reroute using not-via addresses," INTERNET-DRAFT, draft-ietf-rtgwg-ipfrr-notvia-adresses-06.txt, October 2010, (work in progress).

[89] A. Atlas, "U-turn alternates for IP/LDP fast-reroute," INTERNET-DRAFT, draft-atlas-ip-local-protect-uturn-03.txt, March 2006, (work in progress).

[90] S. Bryant, C. Filsfils, S. Previdi, and M. Shand, "IP fast reroute using tunnels," INTERNET-DRAFT, draft-bryant-ipfrr-tunnels-03.txt, November 2007, (work in progress).

[91] K. Kompella and Y. Rekhter, "OSPF extensions in support of generalized multi-protocol label switching (GMPLS)," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 4203, October 2005.

[92] M. Gjoka, V. Ram, and X. Yang, "Evaluation of IP fast reroute proposals," in *Proc. COMSWARE*, January 2007.

[93] A. Atlas, G. Choudhury, and D. Ward, "IP fast reroute overview and things we are struggling to solve," North American Network Operators Group (NANOG) 33, January 2005.

[94] S. Nelakuditi, S. Lee, Y. Yu, Z.-L. Zhang, and C.-N. Chuah, "Fast local rerouting for handling transient link failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 2, pp. 359–372, 2007.

[95] T. Cicic, A. Hansen, A. Kvalbein, M. Hartmann, R. Martin, M. Menth, S. Gjessing, and O. Lysne, "Relaxed multiple routing configurations: IP fast reroute for single and correlated failures," *IEEE Transactions on Network and Service Management*, vol. 6, no. 1, pp. 1–14, March 2009.

[96] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault, "Multi-topology (MT) routing in OSPF," Internet Engineering Task Force, Request For Comments (Standards Track) RFC 4915, June 2007.
[97] K. Xi and J. Chao, "IP fast rerouting for single-link/node failure recovery," in *Proc. IEEE Broadnets*, September 2007.
[98] W. Simpson, "IP in IP tunneling," Internet Engineering Task Force, Request For Comments (Informational) RFC 1853, October 1995.
[99] S. Hanks, T. Li, D. Farinacci, and P. Traina, "Generic routing encapsulation (GRE)," Internet Engineering Task Force, Request For Comments (Informational) RFC 1701, October 1994.
[100] A. Nucci, B. Schroeder, S. Bhattacharyya, N. Taft, and C. Diot, "IGP link weight assignment for transient link failures," in *in Proc. International Teletraffic Congress*, 2003.
[101] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *Proc. IEEE INFOCOM*, March 2000.
[102] A. Ghazala, A. El-Sayed, and M. Mousa, "A survey for open shortest path first weight setting (OSPFWS) problem," in *Proc. The 2nd International Conference on Information Security and Assurance (ISA 2008)*, April 2008.
[103] A. Sridharan and R. Guerin, "Making IGP routing robust to link failures," in *Proc. Networking 2005*, May 2005.
[104] P. Francois, M. Shand, and O. Bonaventure, "Disruption free topology reconfiguration in OSPF networks," in *Proc. IEEE INFOCOM*, May 2007.
[105] P. Francois, "Loop-free convergence using oFIB," INTERNET-DRAFT, draft-ietf-rtgwg-ordered-fib-02.txt, February 2008, (work in progress).
[106] M. Shand and S. Bryant, "A framework for loop-free convergence," Internet Engineering Task Force, Request For Comments (Informational) RFC 5715, January 2010.

**Mukul Goyal** is an Assistant Professor in Computer Science department at University of Wisconsin Milwaukee. He received a Ph.D. degree in Computer and Information Science from Ohio State University in 2003. His research interests lie in the performance analysis of networking protocols, especially those related to routing in wired and wireless networks. More details about his research can be seen at *https://pantherfile.uwm.edu/mukul/public*.

**Mohd Soperi** received a Ph.D. degree in Computer Science from University of Wisconsin Milwaukee in 2009. He is currently a faculty member in the Faculty of Computer Science and Information System, Universiti Teknologi Malaysia. His research interests include Internet routing protocols and delay/disruption tolerant networks.

**Emmanuel Baccelli** is a Scientific Researcher at INRIA, Paris, France. He received a Ph.D. degree from Polytechnique, Paris, France in 2006. His main research interests involve mobility, ad hoc networks, design and analysis of network protocols and algorithms. Emmanuel Baccelli is a frequent contributor to the standardization efforts at the Internet Engineering Task Force. More information about his research can be found at *http://www.emmanuelbaccelli.org*.

**Gagan Choudhury** received the Ph.D. degree in electrical engineering from the State University of New York, Stony Brook. He is a Lead Member of Technical Staff with AT&T Labs. His research interests include deterministic and stochastic performance analysis of wireless and wired networks, analysis and robust design of large IP/Optical networks, data plane and control plane modeling of unicast and multicast protocols, queueing theory, and algorithms for multidimensional transform inversion. Gagan is an IEEE Fellow (2009) and an AT&T Fellow (2009).

**Aman Shaikh** is a Senior Member of Technical Staff at AT&T Labs - Research. He obtained his Ph.D. and M.S. in Computer Engineering from the University of California, Santa Cruz in 2003 and 2000 respectively. He also holds a B.E. (HONS) in Computer Science and an M.Sc. (HONS) in Mathematics from the Birla Institute of Technology and Science, Pilani, India. His current research interests include IP routing, and network management and operations. He has published several research and technical papers in these areas.

**Hossein Hosseini** received his Ph.D. in Electrical and Computer Engineering from University of Iowa in 1982. He has been a faculty member with the Department of Electrical Engineering and Computer Science at the University of Wisconsin Milwaukee (UWM) since 1983. Currently, he is a Professor and Chairman of the Computer Science Program at UWM. Dr. Hosseinis expertise is in the areas of Computer Networks, Computer Architecture, Fault-Tolerance, Distributed and Parallel Computing. He is the founder and Co-Director of Computer Networks Laboratory at UWM. Dr. Hosseini has published over 120 research papers in journals and conference proceedings, has received funding from NSF and industry, and has supervised nine Ph.D. and over 60 M.S. students.

**Kishor S. Trivedi** holds the Hudson Chair in the Department of Electrical and Computer Engineering at Duke University, Durham, NC. He has been on the Duke faculty since 1975. He is the author of a well known text entitled, Probability and Statistics with Reliability, Queuing and Computer Science Applications, published by Prentice-Hall; a thoroughly revised second edition (including its Indian edition) of this book has been published by John Wiley. He has also published two other books entitled, Performance and Reliability Analysis of Computer Systems, published by Kluwer Academic Publishers and Queueing Networks and Markov Chains, John Wiley. He is a Fellow of the Institute of Electrical and Electronics Engineers. He is a Golden Core Member of IEEE Computer Society. He has published over 420 articles and has supervised 42 Ph.D. dissertations. He is on the editorial boards of IEEE Transactions on Dependable and Secure Computing, Journal of Risk and Reliability, International Journal of Performability Engineering and International Journal of Quality and Safety Engineering. He is the recipient of IEEE Computer Society Technical Achievement Award for his research on Software Aging and Rejuvenation. His research interests are in reliability, availability, performance, performability and survivability modeling of computer and communication systems. He works closely with industry in carrying out reliability/availability analysis, providing short courses on reliability, availability, performability modeling and in the development and dissemination of software packages such as SHARPE and SPNP.