Improving Network Convergence Time and Network Stability of an OSPF-Routed IP Network

Amir Siddiqi[1]

amir@nortel.com

Nortel
3500 Carling Ave, Nepean, ON, K2H 8E9

Biswajit Nandy

bnandy@solananetworks.com

Solana Networks
120 Robertson Road Suite 210, Nepean, ON, K2H 5Z1

Abstract. The loss of routing protocol control messages due to network congestion can cause peer failures in routers, leading to routing failures and network instability. We attempt to decrease the network convergence time in response to a node or link failure in a network and study its implications on network stability. We develop an algorithm to dynamically tune OSPF failure threshold based on the network congestion level. The tunable OSPF protocol expedites failure detections as compared to standard OSPF protocol in lightly loaded networks by reducing the failure threshold parameter; whereas the failure threshold is increased during network congestion to sustain stable routing operation and avoid spurious failure decisions due to missed or delayed hello packets. Simulations performed with constant bit-rate (CBR) and FTP data traffic showed that the tunable OSPF protocol facilitated uninterrupted data transfer and resulted in shorter download times as compared to the standard OSPF protocol. Our findings demonstrate the importance of tunable failure threshold to achieve stable and robust network operation.

1 Introduction

The startling growth of the Internet and the flexibility of IP-based packet switched networks have expedited the convergence of data communications (packet switched) and voice-based communications (traditionally circuit switched) into a single IP-based core architecture. The Internet has become an interconnection of various types of heterogeneous networks and supports a wide range of applications. Aside from traditional web applications such as HTTP and FTP, the Internet is experiencing an exponential growth in audio and video streaming, and other real-time applications. However, the existing Internet only offers a single best-effort class of service that is mainly designed for traditional networks and applications, and can adversely impact the end-to-end performance of real-time applications and next-generation networks.

Decreasing Network Convergence Time (NCT) in response to a critical event in a network, such as a node or link failure, is significantly important for next-generation, real-time network applications. The NCT is defined as the difference in time when the critical event has occurred until the time the network is stabilized and all nodes have a loop-free path to every other node in the network domain. Network failures have been observed to be fairly common in the day to day operations of a network due to fiber cuts, hardware/software failures, faulty equipments, or router misconfigurations [1]. A link failure is usually detected immediately by lower layer protocols whereas a node failure is detected by the neighboring nodes after the expiration of a predetermined time without receiving any keep-alive message from the adjacent node. The failed node is declared down and the failure information is flooded to all nodes in the network domain. Each node, upon receiving the update, removes the failed node from its topological database and recalculates routing paths to all other nodes in its routing domain. As such, the NCT can be defined as the sum of failure detection time, information flooding time, update processing time, paths computation time and route installation time.

Guaranteeing a Quality of Service (QoS) for an application requires that the NCT of the network, where the application is running, is adequate for the operation of the application. The network is considered unstable after a node or link failure, which leads to routing failures and data packet drop, until the network has converged. During the transient period of network convergence, the application packets may be dropped

[1] This work is done as part of the Masters of Applied Sciences degree at the Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada with the financial support of *The Natural Science and Engineering Research Council of Canada (NSERC)*

due to invalid routes such as the ones transiting through the failed node or link, or the packets may get caught in routing loops. Any dropped application data packets need to be retransmitted once the network has converged. However retransmission, which is considered an acceptable option in traditional applications such as FTP, may not be feasible for the operation of most real-time applications due to an unacceptable delay incurred during retransmissions.

Emerging network technologies such as Wireless Mesh Networks (WMN) rely on routing protocols such as Open Shortest Path First (OSPF) to find optimal routes. These protocols are traditionally designed for wired networks and are mostly oblivious to the peculiar characteristics of a wireless network. Unlike wired links, the bandwidth of a wireless link can fluctuate frequently and considerably depending on variations in the radio environment. Factors such as interference, fading and shadowing affect the state of the radio link. These factors affect the stability of the network and need to be considered in the routing protocols to assure a stable and robust network operation.

In this paper, we investigate the concept of tunable failure threshold based on network congestion level. The work aims at (i) improving the NCT by reducing the failure detection time and (ii) improving the network stability in a varying network environment. Our implementation is based on OSPF routing protocol [2] and the results are based on simulations performed in *Opnet Modeler*. We analyze and compare the performance of the tunable and standard OSPF protocols in various network congestion scenarios and data traffic types.

The rest of the paper is organized as follows: In section II, we present background information on OSPF routing protocol emphasizing issues related to network convergence and present a study of the related past work done in the field of network convergence. In section III, we develop an algorithm to tune OSPF failure threshold based on the network congestion level. Section IV analyzes and compares the performance of the tunable and standard OSPF protocols. In section V, we discuss the applicability of the proposed protocol in WMN and finally in section VI, we conclude our findings.

2 OSPF Network Convergence

OSPF is the most commonly deployed link state Interior Gateway Protocol (IGP) in the current Internet infrastructure. An important characteristic of a link state protocol is that every node within a domain discovers and builds an entire view of a network topology. During initialization, an OSPF node sends hello packets to all its neighbors. The neighbors supporting the OSPF protocol recognize these packets and reply with their own OSPF packets. The neighbors then exchange their routing tables to build topological databases and are said to be adjacent. To keep the adjacency alive, the neighbors periodically send hello messages to all adjacent neighbors. Not receiving a hello packet from a particular adjacent neighbor for a predetermined fixed amount of time, called router dead interval (RDI), indicates that the neighbor is down. No state information is kept for the failed node and thus when the node recovers, the entire synchronization process is repeated.

The RDI value is a constant in the current OSPF standard and is usually set to four times the hello interval [3]. The default values for the hello interval and the RDI are 10 seconds and 40 seconds respectively. With these values, detecting a failure in a network can take up to 40 seconds. From the time of failure until the network has converged, the data traffic is continued to be forwarded to the failed node or link. The data packets, forwarded to the failed node or link, are dropped and need to be retransmitted. It has been suggested to decrease the hello and router dead intervals to expedite the failure detection time which consequently decreases the overall NCT. The values of 2 seconds for the hello interval and 8 seconds for the RDI are proposed in [4].

Routing protocol control packets usually share resources such as link bandwidth and router buffers with the data traffic. Congestion in the shared resources can thus, hinder the propagation of the control packets. The hello packets can get queued with the other data packets resulting in delayed transmission or packet drop. The adjacent neighbors will erroneously presume that the node is down if no hello packet has been received within the RDI expiration time and initiate the recovery process. Decreasing the hello and router dead intervals increases the probability of losing adjacencies and declaring spurious failures. As such, the RDI value is an important parameter in an OSPF network configuration. A small RDI value expedites real failure detections but may result in spurious failure decisions due to missed or delayed hello packets in a congested network whereas a large RDI value prolongs the real failure detection time.

There has been a growing interest in decreasing the NCT and improving network stability, and several techniques have been proposed in some recent publications. Choudhury et al. [5] explores the possibility of differentiated queuing that gives the hello control packets higher priority over other control and data packets. This proposal is backed by simulation results that show significant improvement in the network stability and scalability. The idea of keeping backup routes is investigated in [6] which proposes replacing slow path re-computation with faster path switching in case of failure. This proposal is more beneficial for link failures that can be detected almost immediately and the switching can be made fast enough to make the failure oblivious from the upper layers. Narvaez et al. [7] and Wu et al. [8] propose minimizing the flooding domain by performing localized restoration and running incremental shortest path first (SPF) algorithm. The unidirectional restoration algorithm is documented in [7] whereas [8] extends the algorithm for bidirectional links. Villamizar [9] proposes periodically caching the SPF results and using the cached information in case of failure and recovery for topology rediscovery. Caching allows transmitting minimal information for synchronization as only the updates that occurred since the last caching are required to be transmitted. The idea of reducing the hello interval to expedite the failure detections and its implications on network stability is investigated in [10]. The paper concludes that the selection of an optimal hello interval value is strongly influenced by the network congestion and the number of links in the topology.

The trade-off between reducing the RDI value to expedite the failure detection time and its effect on network stability is investigated in [11]. It is shown through simulations with constant bit-rate (CBR) and FTP data traffic that the optimal RDI value that lowers the failure detection time while assures a stable routing operation depends on the network congestion level. For improved network performance, [11] proposes tuning the RDI value based on the network congestion level, which is discussed in this paper.

3 State-Based Tunable Failure Threshold

In this section, we describe an algorithm to dynamically tune the RDI value based on network congestion level for a point-to-point OSPF network. The OSPF standard requires that the hello and router dead intervals must be the same on both ends of the point-to-point link and thus any change in the RDI value must be propagated and configured at both interfaces. Furthermore, we need to ensure that the algorithm can incorporate asymmetric traffic flows in opposite directions. To facilitate the consistency, we introduce the notion of master and slave in the connecting interfaces. The designation can be made simply by choosing the interface with a higher IP address as the master. Any update in the RDI value will be done only by the master interface. A slave, on the other hand, can suggest a new RDI value to the master and if the request is accepted, the new value is propagated back to the slave.

The steps involved in the tunable RDI algorithm are discussed in detail in the following sections:

3.1 Detecting Network Congestion

To determine the network congestion level, we maintain a history of hello packet arrivals at each interface. The frequency of updating the RDI value needs to be such that any short-lived traffic variations are ignored by the algorithm to minimize frequent changes in the RDI value and facilitate the network stability. On the other hand, the performance of the algorithm is measured by its response time which is the difference in time when the network congestion level has changed until the time the RDI value is updated to reflect the change in the network congestion. The optimal RDI update time that lowers the algorithm response time while assures the network stability depends on the traffic characteristics of the network [11]. In our algorithm, we determine the network congestion and update the RDI value on the arrival of every twentieth hello packet. Based on the history information, we determine the number of hello packets sent by the sending interface and calculate the hello packet drop rate. i.e.

$$\text{Number of hello packets received: } H_r = 20 \quad \textbf{(1)}$$

$$\text{Number of hello packets sent: } H_s = (t_2 - t_1) / \text{hello interval} \quad \textbf{(2)}$$

$$\text{Hello packet drop rate} = 1 - (H_r / H_s) \quad \textbf{(3)}$$

Where t_1 corresponds to the time when the first hello packet in the current sequence of twenty hello packets is received and t_2 corresponds to the arrival time of the twentieth hello packet. We divide the network congestion into five states (no congestion, low congestion, medium congestion, high congestion and very high congestion) based on the hello packet drop rate. Upon arrival of every twentieth hello packet at an interface, we calculate the hello packet drop rate and determine the desired RDI value from the look-up table given in Table 1 [11]. The current RDI value is then compared with the desired RDI value and the current value is incremented or decremented, as and if required, by one hello interval. These desired RDI values act as upper and lower bounds to prevent the RDI value grow or drop unboundedly. This approach of maintaining a history of hello packets to determine the network congestion level and updating the RDI value by one hello interval at a time circumvents abrupt changes in the RDI value due to sudden traffic burst or short-lived traffic variations and thus, facilitates the network stability.

Table 1. Network Congestion Lookup List

Hello Packet Drop Rate	Network Congestion State	Desired RDI
[0, 0.1]	No Congestion	6
(0.1, 0.25]	Low Congestion	8
(0.25, 0.5]	Medium Congestion	12
(0.5, 0.75]	High Congestion	16
(0.75, 1]	Very High Congestion	20

3.2 Propagating RDI Values

The hello packets are used for the propagation of new RDI values. An OSPF hello packet contains the hello interval and the RDI of the originating interface. In the current OSPF standard, these values contained in the hello packet must match the values stored locally at the receiving interface and the hello packet is discarded in case of any discrepancy. In the tunable RDI algorithm, any new RDI value from the master or suggested RDI value from the slave is contained in the hello packet. The receiving interface recognizes that the RDI contained in the packet is different from the local RDI and runs the tunable RDI algorithm. This approach of using the hello packets for the propagation of new RDI values allows the implementation of the tunable RDI algorithm without modifying the existing OSPF packet formats or adding any extra communication overhead.

3.3 Updating RDI Values

An increase in the RDI value requires signaling from only one interface to ensure that the RDI is increased when either interface is experiencing congestion. If the master interface determines that the local RDI value is less than the desired value, the master increments the local RDI value by one hello interval and sends the new value to the slave in the next hello packet. The slave updates its local RDI value upon receiving that hello packet. If the slave is experiencing congestion and wants to increase the RDI value, the slave sends a RDI value incremented by one hello interval in the next hello packet destined for the master. The slave however, does not modify its local RDI value. The master upon receiving the hello packet realizes that the slave is requesting to increase the RDI value and accepts the request unconditionally. The master then increments the local RDI value by one hello interval and sends the new value to the slave in the next hello packet resulting in the slave updating its RDI value.

Decreasing the RDI value requires acceptance from both interfaces to ensure that the RDI value is decreased only if both interfaces are experiencing low congestion. As such, only one interface needs to initiate the request while the other interface, upon receiving the request, determines if it is feasible to decrease the RDI value. In our implementation, the slave initiates the decrease RDI request. When the slave determines that the current RDI value is higher than the desired one, the slave sends a RDI value decremented by one hello interval in the next hello packet destined for the master. Upon receiving the request, the master consults its history to determine the possibility of reducing the RDI value. If the request is approved, the master lowers its local RDI value by one hello interval and sends the new value to the slave in the next hello packet resulting in the slave reducing its RDI value.

If the hello packet containing the slave suggestion is dropped, the slave will never find out that the master has not received the suggestion. The slave will presume that the request is rejected by the master. To handle this situation in our implementation, the slave sends the suggested RDI value to the master in up to three consecutive hello packets and if still the slave has not received the desired RDI value from the master, the request is presumed to be rejected.

The pseudo code for the tunable RDI algorithm when a hello packet is received at an interface is given in Fig. 1.

```
Event: hello packet received

/* Parse hello packet and extract RDI value from it (msg.rdi) */

If interface.rdi ≠ msg.rdi                  /* msg.rdi differs from the locally stored RDI value */
    If interface is a slave
        interface.rdi = msg.rdi                     /* Slave accepts new value from the master */
    Else if interface is a master                   /* msg.rdi is a suggestion from the slave */
        If msg.rdi > interface.rdi                  /* Suggestion is to increase the RDI value */
            interface.rdi = msg.rdi                 /* Always accept the increase RDI request */
        Else                                        /* Suggestion is to decrease the RDI value */
            /* Determine the desired RDI value for the current congestion level from lookup table */
            If desired_rdi < interface.rdi          /* Congestion level has decreased */
                interface.rdi = msg.rdi             /* Accept the decrease RDI request */
            End if
        End if
    End if
End if

/* Check if the RDI value can be increased or decreased. The adjacent interface will receive any
update in the next hello packet */

/* Calculate the hello packet drop rate and find the desired RDI value from the lookup table */
If interface.rdi < desired_rdi                      /* Congestion level has increased */
    If interface is a master                        /* Master increases its RDI value */
        interface.rdi = interface.rdi + interface.hello_interval
    Else if interface is a slave                    /* Slave suggests a higher RDI value */
        interface.suggested_rdi = interface.rdi + interface.hello_interval
    End if
Else if interface.rdi > desired_rdi                 /* Congestion level has decreased */
    If interface is a slave                         /* Slave suggests a lower RDI value */
        interface.suggested_rdi = interface.rdi – interface.hello_interval
    End if
End if
```

Fig. 1. Tunable RDI OSPF Algorithm Pseudo Code

4 Performance Evaluation

In this section, we analyze and compare the performance of the tunable and standard OSPF protocols. We developed two sets of test cases: (i) simulated hello packet drop and (ii) congestion-based hello packet drop. In the former case, the hello packets arriving at an interface are explicitly dropped with a known probability to simulate the network congestion. The simulated hello packet drop test cases allow us understand the dynamics of the algorithm in a controlled environment. In the latter case, we investigate the performance of the tunable and standard OSPF protocols in the presence of data traffic when both the data and control traffic flows are assigned equal priority. The congestion-based hello packet drop test cases give us an insight to the behavior of the applications in a network running the tunable OSPF routing protocol. To compare the performance of the protocols, we employ the method of *'Correlated Sampling'* in which the same set of random numbers is used to simulate both systems [12]. For each test case, we ran 30 replications and report the results with 95% confidence interval (CI).

The performance of the tunable and standards OSPF protocols is compared in terms of the number of adjacencies lost in each case. Since no real node or link failure is simulated in the network, all adjacency losses are spurious and thus, a lower number depicts superior performance. Each simulation is run for 1500 seconds and batch size is set to 100 seconds i.e. for every 100 seconds, we count the number of adjacencies lost in that particular batch. The hello interval and the RDI are configured to 2 seconds and 8 seconds respectively as recommended in [4] to expedite real failure detections in a network.

4.1 Simulated Hello Packet Drop Tests

In the simulated hello packet drop cases, we explicitly drop a predefined percentage of hello packets arriving at an interface to simulate network congestion for a two-node point-to-point OSPF network in the absence of any data traffic. As each link in a point-to-point OSPF network is independent, this simple topology is deemed sufficient for simulations in a controlled test environment.

4.1.1 Gradually Increasing Network Congestion: In this test case, we start the simulation with no congestion (0% packet drop) and increase the hello packet drop rate by 10% every 100 seconds on one interface until it reaches 70% at time=700 seconds. The congestion is then kept at 70% until the end of simulation. The hello packet drop rate at the other interface is kept at 0% throughout the simulation.

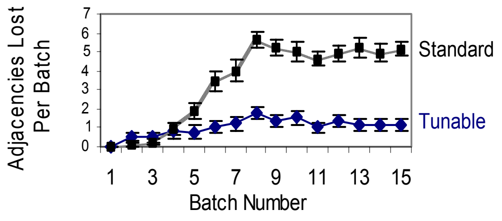
Fig. 2(a) shows the number of adjacencies lost per batch for the tunable and standard OSPF protocols. We notice that initially until batch 3, the tunable OSPF resulted in a slightly higher adjacency loss rate per batch; however after batch 3, the standard OSPF resulted in a much higher adjacency loss rate. The reason for the initial slightly higher loss rate in the tunable OSPF is that, in the beginning of the simulation, the hello packet drop rate of 0% allowed the tunable OSPF to reduce the RDI value from 8 to 6 seconds as shown in the RDI graph of Fig. 2(b). As such, when the congestion developed, the tunable OSPF declared adjacency down after 3 consecutive hello packet drops as compared to 4 consecutive hello packet drops in the standard OSPF protocol. However, the tunable RDI algorithm responded to the increasing hello packet drop rate and gradually increased the RDI value and thus, resulted in lower adjacency loss rate per batch later in the simulation. The standard OSPF is oblivious to the network congestion and continued to lose adjacencies at a higher rate until the end of the simulation.

4.1.2 Gradually Increasing then Decreasing Congestion: In this test case, we gradually build congestion on one interface, starting at 20% at time=100 seconds and increasing the hello packet drop rate by 10% every 100 seconds until the congestion reaches 50% at time=400 seconds. The congestion is then decreased by 10% every 100 seconds until it reaches 10% at time=800 seconds. The congestion is then kept at 10% until the end of simulation. The hello packet drop rate at the other interface is kept at 0% throughout the simulation.

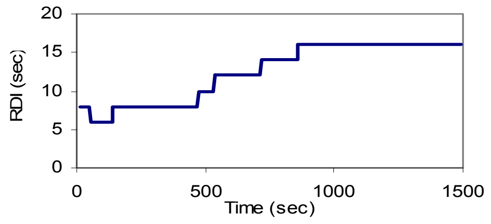
Fig. 3(a) shows a graph of the number of lost adjacencies per batch by the tunable and standard OSPF protocols. We notice that the standard OSPF results in higher adjacency loss rate in the first half whereas the tunable OSPF resulted in a slightly higher adjacency loss rate in the latter half of the simulation. The reason for the first half has already been explained in the first test case. For the latter half, the hello packet drop rate of 10% allowed the tunable OSPF to decrease the RDI to 6 seconds as shown in the RDI graph of Fig. 3(b). With the RDI of 8 seconds for the standard OSPF, the tunable OSPF has a higher probability of declaring the adjacencies down in response to hello packets drop as compared to the standard OSPF protocol. However, on the other hand, the tunable OSPF protocol will detect a real failure in 6 seconds as opposed to 8 seconds by the standard OSPF and thus, decrease the overall network convergence time.

4.1.3 Asymmetric Network Congestion: In this test case, we start with no congestion and increase the hello packet drop by 10% every 100 seconds on both interfaces. However after time=600 seconds, the packet drop at one interface is decreased by 10% every 100 seconds until the hello packet drop rate at that interface reaches 10% at time=900 seconds and then is kept at that level. On the other interface, the hello packet drop rate keeps increasing by 10% every 100 seconds until the congestion level is raised to 80% at time=800 seconds and then is kept at that level. The intent of this test case is to analyze the algorithm for asymmetric traffic flows. As the interface where the hello packet drop rate is decreasing will request to decrease the RDI value, the interface with high congestion will reject the decrease RDI request and keep the RDI value high.

Fig. 4(a) shows the number of lost adjacencies per batch and Fig. 4(b) shows the corresponding RDI graph for the tunable RDI case. It is evident that the RDI is increased gradually and is kept high until the end of simulation. The interface with decreasing network congestion periodically requests to decrease the RDI value but the request is rejected by the interface with persistent high network congestion. The adjacencies loss rate graph depicts the same pattern as in the first test case and has already been explained.

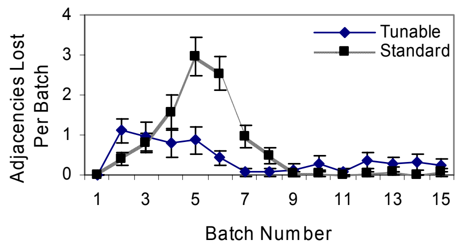


(a) Number of adjacencies lost (95% CI)

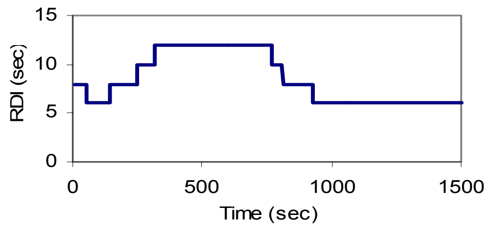


(b) RDI graph for tunable OSPF

Fig. 2. Simulated Hello Packet Drop Test Case 1 (Gradually increasing network congestion)

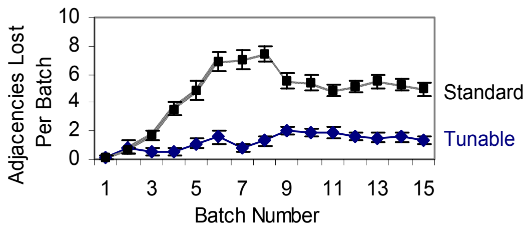


(a) Number of adjacencies lost (95% CI)

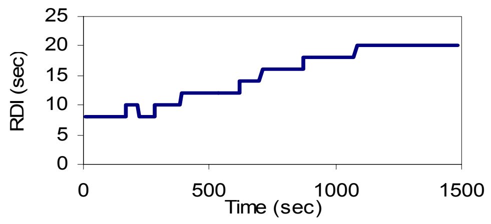


(b) RDI graph for tunable OSPF

Fig. 3. Simulated Hello Packet Drop Test Case 2 (Gradually increasing then decreasing network congestion)



(a) Number of adjacencies lost (95% CI)



(b) RDI graph for tunable OSPF

Fig. 4. Simulated Hello Packet Drop Test Case 3 (Asymmetric network congestion)

4.2 Congestion-based Hello Packet Drop Tests

In the congestion-based hello packet drop test cases, we analyze the performance of the standard and tunable OSPF protocols when the network resources are shared between the data and control traffic. We experimented with UDP-based CBR and TCP-based FTP data traffic flows.

4.2.1 CBR Data Traffic

In this section, we analyze the performance of the standard and tunable OSPF protocols in the presence of CBR data traffic. The CBR traffic is unreliable and unidirectional. It is predictable as the packet size, packet interval and stream duration are known. Most audio and video streams are based on UDP and thus, are also unreliable and unidirectional. Due to the absence of flow control in UDP, the source transmits the traffic at a constant and predictable rate. Due to these similarities, we can approximate the real-time UDP streaming traffic with the CBR traffic.

The network topology is shown in Fig. 5. The links connecting the routers are PPP-DS3 supporting a maximum bandwidth of 44.7Mbits/second. Routers ‘B’ and ‘D’ are connecting to traffic sources that are generating data traffic at a constant rate of 22Mbits/second (50% of the link bandwidth) from time=100 seconds to time=1500 seconds in all the simulation scenarios. The CBR traffic generated by a source connected to router ‘A’ is varied during the experiment and we observe the performance of the tunable and standard OSPF protocols when all traffic flows are destined for router ‘E’. As both the data and control packets are assigned equal priority and thus share the router FIFO queue, increasing the data traffic results in the control traffic being delayed or dropped. Under normal conditions, the data traffic originating from router ‘A’ takes path ABE which we refer to from here on as the primary path. However, as we increase the traffic rate from router ‘A’, congestion develops at router ‘B’ resulting in both the control and data packets being delayed and dropped. Not receiving a hello packet within the RDI from router ‘B’ results in router ‘E’ advertising router ‘B’ down, and all nodes are notified about the router ‘B’ spurious failure. Router ‘A’, upon receiving the update, re-computes its routing table resulting in redirecting the data traffic to path ACDE. The re-routing gradually alleviates the congestion at router ‘B’ and eventually a hello packet reaches router ‘E’ resulting in the rediscovery of router ‘B’. The rediscovery of router ‘B’ results in redirecting the router ‘A’ data traffic back to the primary path. Frequent rerouting causes network instability and data packets drop. Each update (failure or rediscovery) is flooded to all nodes in the network domain resulting in excessive bandwidth usage as well as inconsistent network view and excessive CPU usage at the node level.

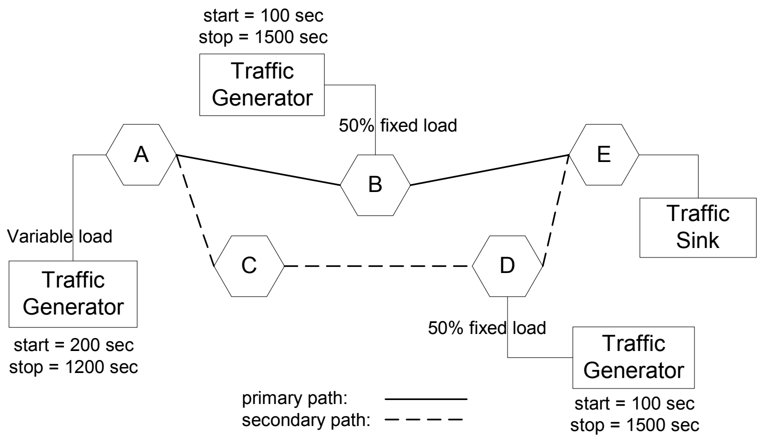
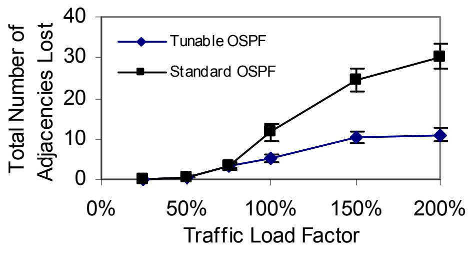


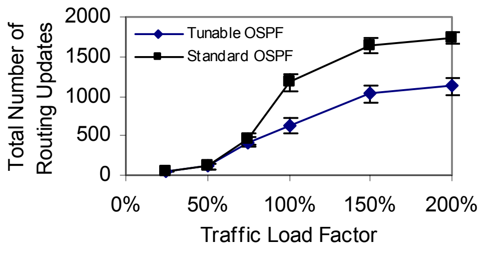
Fig. 5. Simulation Network Topology for Congestion-based Hello Packet Drop Test Case with CBR Data Traffic

Fig. 6 shows the results of overloading traffic on the standard and tunable OSPF protocols when the traffic source connected to router ‘A’ is generating a CBR data traffic of 25%, 50%, 75%, 100%, 150% and 200% of the link bandwidth. Fig. 6(a) and Fig. 6(b) compare the number of adjacencies lost and the number of routing updates sent in the network respectively. Up to 50% traffic load, there is no congestion at router ‘B’ and thus, the two protocols behave comparably. However, the performance of the standard OSPF protocol deteriorates considerably as the traffic load is increased beyond 50%, due to the congestion at router ‘B’. The tunable OSPF protocol resulted in about 3 times lesser number of adjacency losses as compared to the standard OSPF protocol when the traffic load is increased beyond 150%. Also, it is evident that the number of adjacency losses in the standard OSPF protocol increased linearly with the increase in the traffic load. The tunable OSPF, however, is adaptable to the applied traffic load and thus, the number of adjacency losses by the tunable OSPF protocol is almost the same for both the 150% and 200% traffic load cases. Each spurious failure and rediscovery is flooded in the network as an update and thus, as expected, the number of routing updates increases with the increase in the number of lost adjacencies as shown in Fig. 6(b).

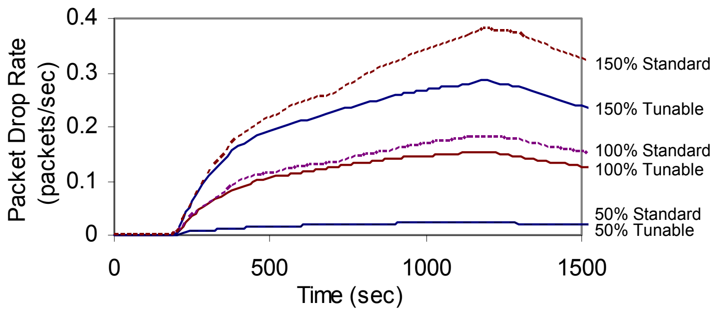
To demonstrate the effect of number of spurious lost adjacencies on the network stability, Fig. 6(c) shows average packet drop rate for selected traffic loads with the tunable and standard OSPF protocols. The two protocols behave comparably for 50% traffic load whereas the standard OSPF dropped 15% and 25% more packets as compared to the tunable OSPF for 100% and 150% loads respectively, due to the spurious lost adjacencies and frequent re-routing, causing routing failures and network instability.



(a) Total number of adjacencies lost



(b) Total number of routing updates sent in the network



(c) Average packet drop rate for selected loads

Fig. 6. Comparison of Tunable and Standard OSPF Protocols Performance with CBR data Traffic

(buffer size=1K bytes, packet size=64 bytes, 30 replications, 95% CI)

4.2.2 FTP Data Traffic

In this section, we compare the performance of the standard and tunable OSPF with heavy FTP traffic load. The simulation network topology is shown in Fig. 7. The users in LAN1 and LAN2 request files located at the FTP server. The requests from LAN1 start at time=200 seconds and continue until time=1300 seconds whereas the users in LAN2 requests files from time=500 seconds until time=1000 seconds. The size of each file located at the server is 500K bytes whereas the user requests are made with an inter-arrival time distributed exponentially with λ=50.

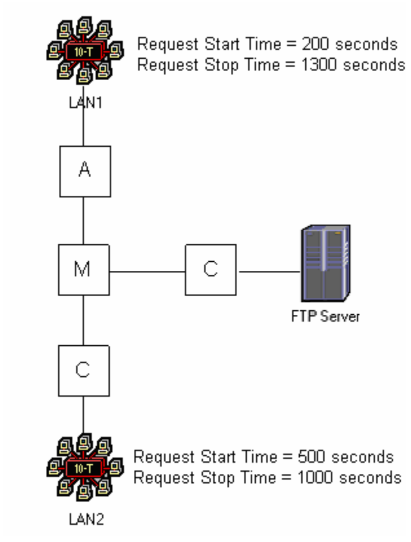

Request Start Time = 200 seconds
Request Stop Time = 1300 seconds
LAN1
A
M
C
FTP Server
C
Request Start Time = 500 seconds
Request Stop Time = 1000 seconds
LAN2

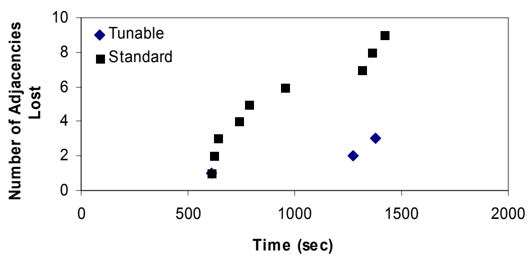
Fig. 7. Simulation Network Topology for Congestion-based Hello Packet Drop Test Case with Heavy FTP Data Traffic

Fig. 8 compares the performance of the standard and tunable OSPF protocols with the heavy FTP traffic load. We see that the tunable OSPF protocol resulted in a total of 3 lost adjacencies as compared to 9 adjacency losses by the standard OSPF protocol. In the standard OSPF protocol, most of the adjacencies are lost between time=500 seconds to time=1000 seconds when the users in LAN1 and LAN2 are making simultaneous requests. Both the standard and tunable OSPF protocols lost their first adjacency at time=600 seconds due to increase in the network congestion. However, the tunable OSPF lost its second adjacency at time=1270 seconds. The standard OSPF protocol lost 5 more adjacencies until time=1000 seconds. The reason for the difference is that the tunable OSPF protocol algorithm adjusted to the increase in the network congestion level and accordingly increased the RDI value. As such, there are no more adjacency losses until the high congestion level persisted in the network. The standard OSPF protocol is oblivious to the network congestion and continued losing adjacencies at regular intervals.

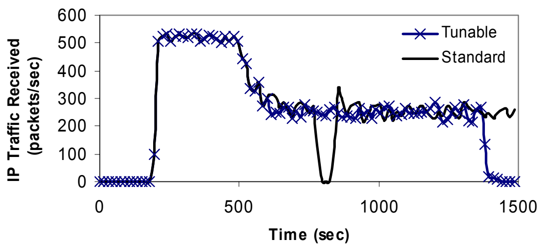
After the users in LAN2 stopped making the file requests at time=1000 seconds, the network congestion level started decreasing. From time=1000 seconds to time=1500 seconds, the standard OSPF protocol lost 3 more adjacencies as opposed to 2 more adjacencies by the tunable OSPF protocol. The adjacency losses by the tunable OSPF protocol are caused by the varying network congestion level. With the decreasing network congestion level, the tunable RDI algorithm gradually decreases the RDI value. As the RDI value is decreased, it becomes more likely to lose the adjacencies due to delayed or dropped hello packets.

When we compare the IP traffic received (packets/seconds) by the users in LAN1 and LAN2 with the standard and tunable OSPF protocol as shown in Fig. 8(b) and Fig. 8(c) respectively, we notice that the tunable OSPF protocol provides a more stable network operation. Due to 6 adjacency losses by the standard OSPF protocol between time=500 seconds until time=1000 seconds, we notice a sudden drop in the IP traffic received by both LAN1 and LAN2 at around time=800 seconds. The reason for the drop is that losing a number of adjacencies in a short interval results in inconsistent network view at the nodes which leads to routing failures and consequently data traffic being dropped. There is no such drop in the received traffic when the tunable OSPF is the employed routing protocol. As such, the deployment of the tunable OSPF protocol facilitates a more stable network environment.

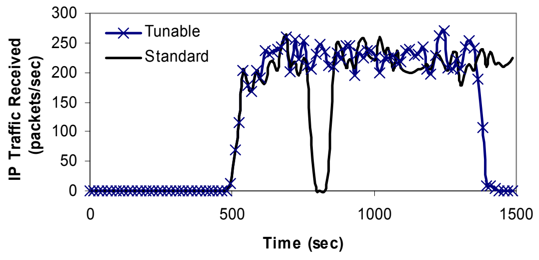
When we analyze the IP traffic received pattern from time=1000 seconds to time=1500 seconds, we note that both protocols are showing comparable performance. However we see that in the case of the tunable OSPF protocol, the file transfer finished at time=1425 seconds whereas the file transfer with the standard OSPF protocol continued until the end of simulation at time=1500 seconds. The reason for this difference is that the tunable OSPF protocol provided a more stable network environment with no interruptions and routing failures. As such, the file transfer completed early with the tunable OSPF protocol as compared to the standard OSPF protocol where the routing failures resulted in dropped packets. The dropped data packets need to be retransmitted after the network has stabilized which prolonged the overall file transfer time.



(a) Number of adjacencies lost



(b) IP traffic received on LAN1



(c) IP traffic received on LAN2

Fig. 8. Comparison of Tunable and Standard OSPF protocols with Heavy FTP Traffic (File size=500K bytes, λ=50)

5 Applicability of Tunable OSPF Protocol in WMN

Wireless mesh networks (WMN) is an emerging broadband Internet access technology that is drawing significant attention [13]. The WMN backhaul consists of fixed 802.11 wireless nodes connected to each other via point-to-point radio links as shown in Fig. 9. The mesh network ensures the availability of multiple paths for each node in the network and relies on routing protocols, such as OSPF, to find an optimum route. Unlike wired links, the bandwidth on a wireless link can fluctuate frequently and considerably due to variations in the radio environment. To keep an acceptable signal-to-noise ratio (SNR), most WMN supports multi-rate bandwidth [14]. As the radio link quality deteriorates, the supported bandwidth is decreased to reduce the transmission rate, resulting in a decreased packet error rate. The decrease in bandwidth results in increasing the network congestion level and raises the probability of dropping the hello packets and losing the adjacencies due to spurious failure decisions. The standard OSPF protocol, which is oblivious to the network congestion, starts declaring adjacencies down after the expiration of the fixed RDI. The tunable OSPF protocol, on the other hand, adapts to the varying network congestion level. As the network congestion increases due to the decrease in the link bandwidth, the tunable OSPF protocol increases the RDI value to avoid the spurious failures and facilitates network stability. As the radio link quality improves, the link bandwidth increases resulting in a decrease in the network congestion level. The tunable OSPF protocol gradually decreases the RDI value to expedite the failure detection time and thus, reduces the overall NCT.

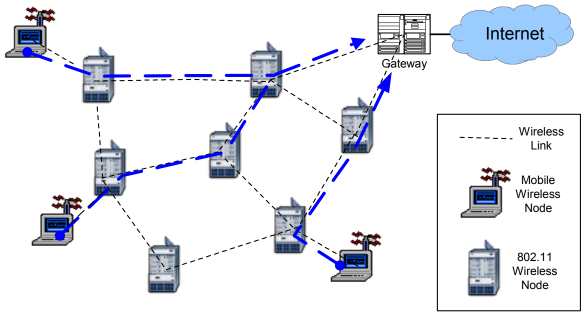


Fig. 9. Wireless Mesh Networks

6 Conclusion

This paper shows that network convergence time and number of spurious lost adjacencies in an OSPF network can be reduced by tuning the router dead interval (RDI) based on network congestion level. An algorithm is proposed for a point-to-point OSPF network to dynamically tune the RDI value based on the network congestion level. Results obtained from simulations of the proposed algorithm showed reduced numbers of spurious adjacency losses under high network congestion as compared to standard OSPF while improving the real failure detection time in low network congestion. The algorithm may lead to higher network convergence time under heavy congestion but improves the stability of the network.

We compared the performance of the tunable and standard OSPF protocols with CBR data traffic which closely models the real-time data traffic and with FTP data traffic. The tunable OSPF protocol has shown to provide a more stable network environment with uninterrupted data transfer and facilitated shorter download times. Furthermore, it is intuitively explained that the performance and stability of the emerging Wireless Mesh Networks (WMN), which is subject to radio channel anomalies, can be improved by employing the proposed tunable OSPF protocol.

References

1. G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an IP backbone," *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2002
2. J. Moy, “OSPF Version 2,” RFC 2328, April 1998
3. Cisco Systems, “OSPF design guide,” [online], http://www.cisco.com/warp/public/104/1.html
4. C. Alaettinoglu, V. Jacobson and H. Yu, “Towards millisecond IGP convergence,” [online], http://www.nanog.org/mtg-0010/ppt/cengiz.pdf , NANOG 20, October 2000
5. G. Choudhury, V. Sapozhnikov, G. Ash, A. Maunder and V. Manral, “Prioritized treatment of specific OSPF version 2 packets and congestion avoidance,” Internet Draft, draft-ietf-ospf-scalability-09.txt, December 2004
6. J. Pu, E. Manning and G. C. Shoja, “Routing reliability analysis of partially disjoint paths,” in *IEEE Conference on Communications, Computers, and Signal Processing*, 2001, Vol. 1, pp. 79-82, August 2001
7. P. Narvaez, K. Siu and H. Tzeng, “Local restoration algorithm for link-state routing protocols,” in *Eight International Conference on Computer Communications and Networks*, pp. 352-357, 1999
8. J. Wu, X. Lin, J. Cao and J. Weijia, “An extended fault-tolerant link-state routing protocol in the Internet,” in *Eighth Internal Conference on Parallel and Distributed Systems*, pp. 331-337, 2001
9. C. Villamizar, “Convergence and restoration techniques for ISP interior routing,” [online], http://www.nanog.org/mtg-0206/ppt/curtis.pdf
10. M. Goyal, K. Ramakrishnan, and W. Feng, “Achieving faster failure detection in OSPF networks,” *Proceedings of ICC 2003*, Vol. 1, pp. 296-300, May 2003
11. A. Siddiqi, “Improving Network Convergence Time and Network Stability of an OSPF-Routed IP Network,” Master’s Dissertation, Ottawa-Carleton Institute for Electrical and Computer Engineering, Carleton University, Ottawa 2004
12. J. Banks, J. Carson, B. Nelson and D. Nicol, *Discrete-event system simulation.* Prentice Hall, pp. 456-467, 2000
13. J. Jun and M. Sichitiu, “The nominal capacity of wireless mesh networks,” *IEEE Wireless Communications*, Vol. 10, pp. 8-14, October 2003
14. A. Acharya and A. Misra, “High-performance architectures for IP-based multihop 802.11 networks,” *IEEE Wireless Communications*, Vol. 10, pp. 22-28, October 2003