Open Shortest Path First (OSPF) Routing Protocol Simulation*

Deepinder Sidhu, Tayang Fu, Shukri Abdallah and Raj Nair†
Maryland Center for Telecommunications Research &
Department of Computer Science
University of Maryland — BC
Baltimore, MD 21228
and
Institute for Advanced Computer Studies
University of Maryland — CP
College Park, MD 20742

Rob Coltun
Consultant

Abstract

Open Shortest Path First (OSPF) is a dynamic, hierarchical routing protocol designed to support routing in TCP/IP networks. A simulation of the OSPF Election Protocol shows three results: (1) The Designated Router(DR) can be elected in constant time. (2) If a router has a limited number of input buffers, a competition for buffers between the Election and the Flooding Protocols increases the election time and causes an oscillatory behavior. At each router, the Router-ID of the DR continuously changes causing instability. (3) In the worst case, when the DR and the BDR fail at the same time, the DR-agreement-time is bounded above by twice the HelloInterval. A simulation of the OSPF Flooding Protocol, using 20, 50 and 80 router point-to-point networks, shows three results: (1) For the 50 router network, as link speed exceeds 4000 Kbps, the probability of overflowing the input buffers increases causing retransmissions. The increase in bootup-convergence-time from retransmissions is bounded by two and three times the RxmtInterval for link speeds of 4000 to 6000 Kbps and above 50 Mbps respectively. The increase in the bootup-convergence-time is due to large number of unacknowledged flooding packets received within RxmtInterval. (2) For 20 and 50 router networks, the input buffer size has little impact on the bootup-convergence-time. For the 80 router network, a small change in the input buffer size drastically changes the bootup-convergence-time. (3) Reducing the value of the RxmtInterval lowers the bootup-convergence-time at high link speeds.

*This research was supported in part by the Department of Defense at the University of Maryland Baltimore County. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Defense or the U.S. Government.

† Present address: Netrix Corporation, 13595 Dulles Technology Dr., Herndon, VA 22071

SIGCOMM'93 - Ithaca, N.Y., USA /9/93

1 Introduction

Open Shortest Path First '(OSPF) is a *dynamic, hierarchical* routing protocol designed to support routing in TCP/IP networks [1]. The OSPF routing protocol is a collection of interrelated algorithms: the Hello, Election, Flooding and Shortest-Path-First (SPF). The Hello, Election, and Flooding Protocols distribute and synchronize routing information within an autonomous system. The Shortest-Path-First algorithm computes the shortest-path tree.

In this paper, we present a simulation study of the Election and Flooding Protocols of OSPF. Section 2 presents simulation results of the Election and the Flooding Protocols. Section 3 contains summary and conclusions.

2 OSPF Simulation

In this section, we present the results of the discrete event simulation of the Election and Flooding Protocols.

2.1 Election Protocol

The Election Protocol elects a *Designated Router* (DR) and a *Backup Designated Router* (BDR) to distribute and synchronize topology information among routers on a broadcast network. Within the network, the DR reduces the number of messages needed to broadcast topology information and hides topology information from other routers within the autonomous system.

A router is *eligible* to participate in the Election Protocol if its Router-Priority is positive. A router nominates a DR and a BDR using the DR and BDR fields of the hello packet. Every HelloInterval, each router X transmits a hello packet containing among other information its Router-Id, its Router-Priority and a list of Router-Ids from whom X has received a hello packet. Router X discovers router Y when X receives for the first time a hello packet from router Y. Router X detects the absence of router Y when X does not receive a hello packet from router Y for a period of RouterDeadInterval. Router X considers router Y as a *bidirectional neighbor* when X sees its Router-Id in the list of Router-Ids in the hello packet sent by router Y.

A router is said to *declare* itself a DR(BDR) if it elects itself DR(BDR) and inserts its Router-Id in the DR(BDR) field of the hello packet. A null value in these two fields indicates the absence of DR and the BDR. We refer to the router that wins the election as the "winning-DR".

Initial Election Time

Let t_1 be the time at which the first router is booted and t_2 be the time at which the winning-DR elects itself. The objective of this experiment is to determine the DR-election-time, $t_2 - t_1$, on a broadcast network.

The network sizes vary from 10 to 100 routers all of which have unlimited amount of input and output buffers. The Wait Timer, RouterDeadInterval and the Hello Timer of each router are set to 40, 40 and 10 seconds respectively. We assume zero propagation and processing delays. We also assume that the Election Protocol runs in zero seconds. Initially, all routers are eligible routers in the DOWN state. If router R_x was booted at time t_x and the next router R_y was booted at time t_y such that $t_y \geq t_x$, then $t_y - t_x$ is the inter-boot-time, $\triangle t$. The first router is booted at time $\triangle t$ seconds, and the remaining routers are booted in increasing order of Router-Id. The experiment is repeated for $\triangle t$ of 7, 10, 22, 30 and 40 seconds.

Figure 1a shows the result for $\triangle t = 7$ seconds, and Fig. 1b shows the result for $\triangle t = 30$ and 40 seconds. In Fig. 1a, the DR-election-time increases linearly with the number of routers. In Fig. 1b, the DR-election-time is constant. To explain the linear increase of the DR-election-time in Fig. 1a, we trace the sequence of events executed at routers R_1 and R_2 attached to a broadcast network. Then, we generalize this explanation to a network of n routers.

At time 7 seconds, when router R_1 is booted up, it broadcasts a hello packet containing its Router-Id and enters the WAIT state for a period of 40 seconds. Similarly, when router R_2 is booted at time 14 seconds, it broadcasts a hello packet and enters the WAIT state. Router R_1 upon receiving the hello packet from R_2 at time 14 seconds estab-

lishes one-way communication with R_2. At time 17 seconds, the second HelloInterval, router R_1 broadcasts a hello packet with the Router-Ids of R_1 and R_2. Upon receiving this hello packet, R_2 establishes bidirectional communication with router R_1. At time 24 seconds, when R_2 broadcasts a hello packet, both routers establish bidirectional communication becoming candidates for election. A router exits the WAIT state if its Wait Timer expires or a Backup_Seen event is triggered.

A Backup_Seen event is triggered at any router R_x if R_x receives a hello packet from another router R_y such that (1) R_y declares itself to be the BDR, or (2) R_y declares itself to be the DR and declares that it has not elected a BDR. Since R_1 and R_2 are in the WAIT state, they cannot declare themselves as DR or BDR.

At time 47 seconds, the Wait Timer at R_1 expires, and R_1 elects R_2 as the DR because R_2 has a higher Router-Id. R_1 broadcasts the result of the election in its hello packet. A Backup_Seen event at R_2 is not triggered because R_1 is not declaring itself to be DR or BDR. At time 54 seconds, when the Wait Timer at R_2 expires, R_2 elects itself as the DR and R_1 as the BDR. At the same time, the Hello Timer at R_2 expires and R_2 broadcasts the results of the election. When R_1 receives the hello packet, a Neighbor_Change event is triggered causing R_1 to run the election. R_1 selects R_2 as the DR and itself as the BDR. At time 57 seconds, R_1 declares itself as a BDR to the whole network by broadcasting a hello packet. Thus, the DR-election-time for a network of two routers is $54 - 7 = 47$ seconds.

To generalize the explanation, consider a network with n routers with Router-Ids of $R_1, R_2, \ldots, R_n$. The boot time of these routers are $7, 14, 21, \ldots, 7 * n$ seconds respectively. The Wait Timer at each router expires at $47, 54, 61, \ldots, 7*n+40$ seconds respectively. Each router remains in the WAIT state for the whole period of 40 seconds because the Backup_Seen event cannot be triggered by any router. Before the Wait Timer of router i $(1 \leq i \leq n-1)$ expires, router $i+1$ is booted, and both routers establish bidirectional communication. When the Wait Timer expires at router i, it elects the router with the highest Router-Id with whom it established bidirectional communication.

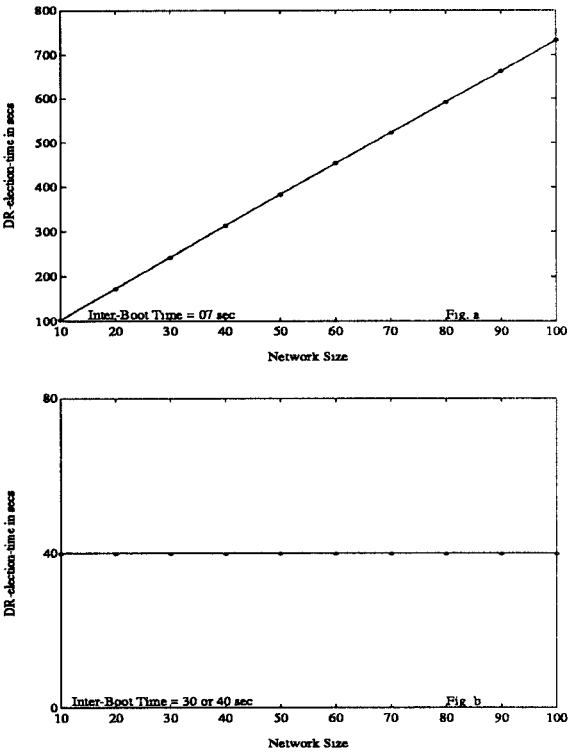

Figure 1: Election Time for Broadcast Networks

Finally, router R_n is booted at time $7 * n$ seconds. All routers establish bidirectional communication with R_n before its Wait Timer expires. As a result, all routers elect R_n as the DR. At time, $7*n+40$ seconds, the Wait Timer of R_n expires, and R_n elects itself as the DR and R_{n-1} as the BDR. Thus, the $DR - election - time = 7 * n + 40 - 7$, where $\triangle t = 7$. The election time increases linearly because a Backup_Seen event cannot be triggered at any router. The same explanation holds for $\triangle t = 10$ and 22 seconds.

Each router excluding R_n and R_{n-1} runs the Election Protocol two additional times. R_n and R_{n-1} run the Election Protocol one additional time. First, R_n broadcasts a hello packet declaring itself as the DR which causes a Neighbor_Change event at all other routers. Second, R_{n-1} broadcasts a hello packet declaring itself as the BDR which causes a Neighbor_Change event at all other routers.

Figure 1b shows a constant DR-election-time. In this scenario, $\triangle t = 30$ and router R_1 is booted at time 30 seconds. Router R_2 is booted at time 60

seconds and enters the WAIT state. R_1 establishes bidirectional communication with R_2 at time 80 seconds. When the Wait Timer of R_1 expires at time 70 seconds, R_1 elects itself as the DR. Since R_1 has no bidirectional neighbors, it does not elect a BDR. Therefore, the DR-election-time is 40 seconds. At time 70 seconds, R_1 broadcasts the results of the election in a hello packet which triggers a Backup_Seen event at R_2. R_2 exits the WAIT state and elects R_1 as the DR. For each of the remaining routers, a Backup_Seen event is triggered upon receiving a hello packet from the BDR. These routers exit their WAIT state and accept the existing DR and BDR.

Figure 1b also shows a constant DR-election-time under a different scenario, where $\triangle t = 40$ seconds. The Wait timer of R_1 expires at time 80 seconds when router R_2 is booted. Since R_1 and R_2 have not yet established bidirectional communication and the Wait Timer of R_1 has expired, R_1 elects itself as the DR. R_1 does not elect a BDR since it is has not established bidirectional communication with any other router. On receiving the hello packet from R_1 which establishes bidirectional communication, a Backup_Seen event is triggered at R_2 forcing R_2 to exit its WAIT state and elect R_1 as the DR. A Backup_Seen event is triggered at the remaining routers causing them to elect R_1 and R_2 as the DR and BDR respectively.

The order of handling the events impacts the performance of the Election Protocol. The DeadRouterInterval_expiration, Wait_Timer_expiration, Backup_Seen and Neighbor_Change events determine the content of a hello packet. An efficient OSPF implementation handles these events before it handles the Hello_Timer_expiration event; otherwise, the most recent information(results of election, new bidirectional neighbors) will be broadcast after one HelloInterval resulting in a degradation in performance.

To achieve the constant DR-election-time as in Fig. 1b, (1) choose two routers to be the intended DR and BDR, and assign them positive RouterPriority and the highest two Router-Ids. (2) boot the intended DR first and wait for a period of at least Wait Timer before booting the intended BDR, (3) boot the intended BDR and wait for a period of at least Wait Timer, and (4) boot the remaining routers. This result can be generalized by assigning different positive priorities to the routers.

Election Time and Topological Change

Assume that all routers are booted in increasing order of Router-Id and eventually reach a FULL state with the DR and the BDR. let t_{d1} (t_{b1}) be the time at which the first router detects the absence of the DR(BDR), and $t_{d2}(t_{b2})$ be the time at which the last router enters the EXCHANGE-START state with the newly elected DR(BDR). The objective of this experiment is to determine the DR(BDR)-agreement-time, $t_{d2} - t_{d1}$ ($t_{b2} - t_{b1}$), for a broadcast network after a topological change.

The time t_{d1} is measured after RouterDeadInterval seconds have elapsed, where RouterDeadInterval is the minimum period before a router detects the absence of another router. At time t_{db}, the DR and the BDR are brought down, and the DR-agreement-time and BDR-agreement-time are measured. This experiment is run with $\triangle t$ equal to 7, 10, 22, 30 and 40 seconds. Figures 2a-2c show the results of this experiment.

Let $t_{dh}(t_{bh})$ be the last time at which the DR(BDR) broadcast a hello packet before going down ($t_{dh}, t_{bh} \leq t_{db}$). All routers should detect the absence of the DR exactly at $t_{dh} + RouterDeadInterval$ and the absence of the BDR exactly at $t_{bh} + RouterDeadInterval$ seconds. However, a router R_x checks if the RouterDeadInterval has expired only when R_x's Hello Timer expires. The Hello Timer expiration depends on the boot time of each router. Depending on its boot time, a router belongs to one of ten groups, G_i, where $i = (Router - Id * \triangle t) \bmod HelloInterval$. All routers in a group detect the RouterDeadInterval expiration of another router at the same time. Let the DR and BDR belong to the groups G_d and G_b respectively.

To calculate the DR(BDR)-agreement times, we partition the groups into three sets, S_1, S_2 and S_3 where S_1 is the set of groups which detect the absence of the DR and the BDR at the same time, S_2 is the set of groups which detect the absence of the DR before they detect the absence of the BDR and S_3 is the set of groups which detect the absence of the BDR before they detect the absence of the

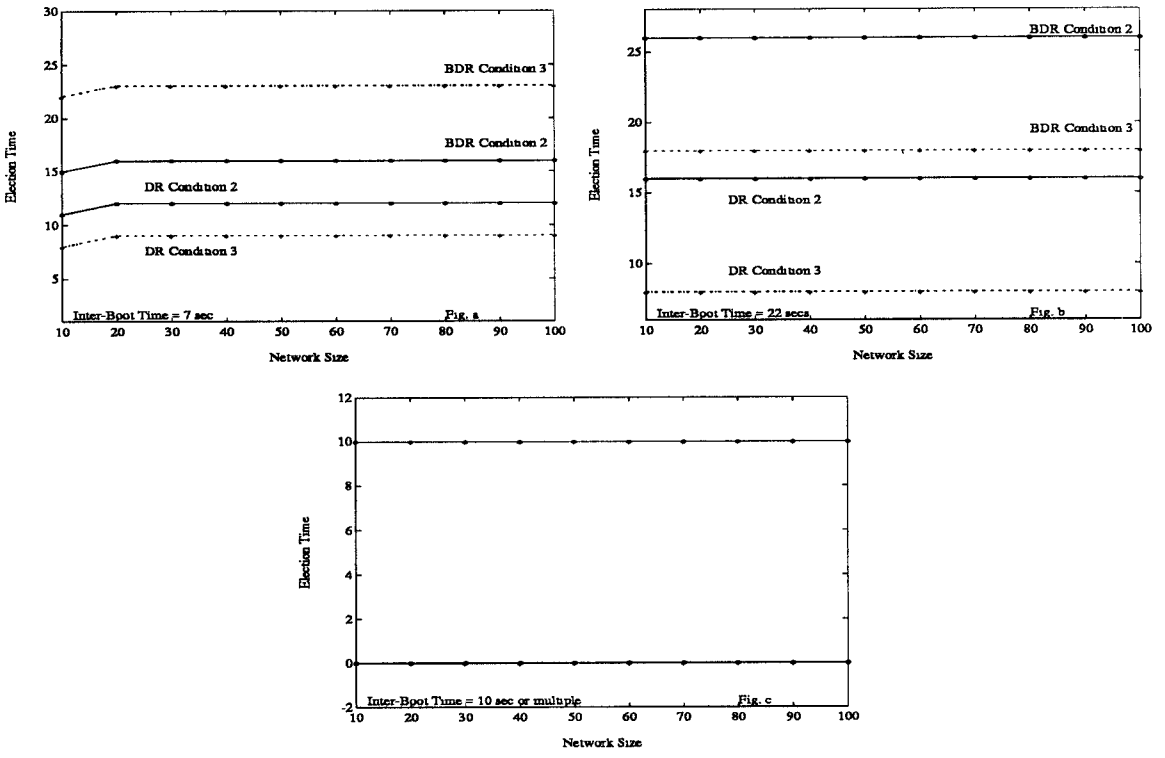

Figure 2: Election Time after Topological Change

DR.

To determine in which set a group G_k belongs, consider a router R_j in the DR-OTHER state which belongs to group G_k. Let t_{jh} be the first time the Hello Timer expires at router R_j after the DR and BDR are brought down($t_{jh} \geq t_{db}$). Let $x = t_{jh} - t_{dh}$ and $y = t_{jh} - t_{bh}$. If x and y are both less than the HelloInterval, or both are greater or equal to the HelloInterval, then $G_k \in S_1$; otherwise, if x is greater or equal to the HelloInterval, then $G_k \in S_2$; otherwise, $G_k \in S_3$.

Thus, we determine t_{d1} and t_{b1} as follows. $t_{d1} = t_{dh} + RouterDeadInterval + d$ if $G_d \neq \phi$; otherwise, $t_{d1} = t_{dh} + RouterDeadInterval + f$ if $G_d = \phi$ and G_f is the first non-empty group that follows d in the ring of integers modulo 10. Similarly, $t_{b1} = t_{bh} + RouterDeadInterval + b$ if $G_b \neq \phi$; otherwise, $t_{b1} = t_{bh} + RouterDeadInterval + f$ if $G_b = \phi$ and G_f is the first non-empty group that follows b in the ring of integers modulo 10.

To determine t_{d2} and t_{b2}, we relate the times t_{dh}, t_{bh} and t_{db} according to one of three conditions: (1) $\triangle t$ is a multiple of the HelloInterval, or (2) $t_{dh} < t_{bh} < t_{db}$, or (3) $t_{bh} \leq t_{dh} \leq t_{db}$ and $\triangle t$ is not a multiple of the HelloInterval. Each condition determines a sequence of events to elect a new DR and a new BDR.

If condition 1 holds, all routers belong to group $G_0 \in S_1$ and $t_{d1} = t_{b1} = t_{dh} + RouterDeadInterval$. A Neighbor_Change event simultaneously causes all routers to run the Election Protocol, elect a DR and enter the EXCHANGE-START state with the new DR. In this case $t_{d2} = t_{d1}$ and the DR-agreement-time is zero as shown in Fig. 2c. After one HelloInterval, the new DR broadcasts a hello packet which causes all routers to run the Election Protocol, elect a new BDR and simultaneously enter the EXCHANGE-START state with the new BDR. Hence, $t_{b2} = t_{b1} + HelloInterval$ and the BDR-agreement-time is 10 seconds as shown in Fig. 2c.

If condition 2 holds, all groups belong to the sets S_1 and S_2, and the set S_3 is empty. Each router in S_1 elects a new DR and enters the EXCHANGE-START state with the new DR. Each router in S_2 promotes the present BDR to become the new DR

and detects the absence of the BDR at the next expiration of its HelloTimer. Let T_1 be the first time at which the new DR broadcasts a hello packet, and T_2 be the last time at which any group in S_2 detects the absence of its promoted BDR. T_1 and T_2 must occur within one HelloInterval from t_{d1}. All routers are guaranteed to have entered the EXCHANGE-START state with the new DR at time $\min(T_1, T_2)$. Then, $t_{d2} = \min(T_1, T_2)$. At time T_1 all routers run the Election Protocol and elect a new BDR. Therefore, $t_{b2} = T_1$. Figures 2a-2b show the DR(BDR)-agreement times for condition 2 when $\triangle t = 7$ and 22 seconds.

If condition 3 holds, then all the groups belong to the sets S_1 and S_3, and the set S_2 is empty. Routers in the set S_1 elect a new DR and enter the EXCHANGE-START state with the new DR. Each router in S_3 elects a new BDR and enters the EXCHANGE-START state with the new BDR. After one HelloInterval, routers in S_3 detects the absence of the DR, promote the new BDR to the new DR and continue the database synchronization process with the promoted DR. If the newly elected DR is in S_3, it declares itself as a new BDR, promotes itself and elects a new BDR. Let T_4 be the time at which the newly elected DR declared itself as a new BDR. After one more HelloInterval, it declares itself a new DR. The first declaration does not change the identities of the new DR and new BDR. On receiving the second declaration, all router agree on the new DR and new BDR. On the other hand, if the newly elected DR is in S_1, it elects a new BDR and declares itself new DR one HelloInterval after t_{d1}. Therefore, we expect that the DR-agreement time to be less than one HelloInterval as shown in Figs. 2a-2b. Let T_3 be the last time at which any group in S_3 detects the absence of the BDR. Therefore, all routers are guaranteed to have entered the EXCHANGE-START state with the new DR at time $\min(T_1, T_3)$. Then, $t_{d2} = \min(T_1, T_3)$. All routers know about the newly elected BDR at $t_{b2} = \max(T_1, T_4 + HelloInterval) + HelloInterval$.

In the worst case, when the DR and the BDR fail at the same time, the DR-agreement-time is bounded above by twice the HelloInterval.

In an OSPF implementation, a router may checks the expiration of RouterDeadInterval for a neighboring router when its Hello Timer expires. If the granularity of checking the RouterDeadInterval is finer than the HelloInterval, it is possible to obtain one group of routers which detect the absence of the DR and BDR at the same time as in Fig. 2c.

Interaction of Election and Flooding Protocols

The objective of this experiment is to determine if the Flooding Protocol affects the DR-election-time and DR(BDR)-agreement-time. The experimental settings are identical to the two previous experiments except that $\triangle t = 7$ and the size of the input-control-packet queues of all interfaces is set to 10 packets. Each interface has one input-control-packet queue which contains both hello and flooding packets. If the queue is full, incoming packets are dropped.

We conducted three different runs of the same experiment. The results of the three runs are different from each other and different from the results in Figs. 1a and 2a. This behavior results from the competition between the flooding packets and the hello packets for input buffer. The flooding packets prevented the hello packets from arriving at the routers every HelloInterval thus increasing the election and agreement times. As the size of the queues decreases, the election and agreement times increase. However, when we introduce separate input queues for the hello and flooding packets keeping the total size of both queues to 10, we obtain the same results as in Figs. 1a and 2a. We processed the hello packets before we processed the flooding packets. We strongly recommend that an OSPF implementation should have a separate control queue for hello packets and should process the hello packets at a higher priority than the other control packets.

If a router, R, has a limited amount of input buffer space, we observe an oscillatory behavior in the identity of the DR at R. If R does not receive a hello packet from the DR within a RouterDeadInterval seconds, R assumes that the DR is down. The DR may not be down except that its hello packets are being dropped due to lack of buffer space. R runs the election and elects a new DR and starts a new synchronization process

with the new DR. Upon receiving a hello packet from the old DR, R assumes that the old DR is up again(Neighbor_Change event) and runs the election. R elects the old DR and starts a new database synchronization process.

In networks with a strict performance requirements, for example convergence to occur within twenty seconds, it is crucial to implement a separate queue for hello packets.

2.2 Flooding Protocol

The Flooding Protocol is a reliable information exchange mechanism which ensures that all routers within an area have identical topology information for that area. Every pair of neighboring routers exchange topology summaries to learn about the most recent topology changes within the autonomous system. A router obtains the new information by synchronizing its topology database with a neighboring router using the Flooding Protocol.

In this section, we describe three experiments that measure the bootup-convergence-time and the convergence-time for point-to-point networks. The *bootup-convergence-time* is the interval between the time all routers and links in a network are initially brought up until the routing-convergence-state is reached. The routing-convergence-state is the state in which all routers reach the FULL state and have empty retransmission and request lists. Let t be the time at which a topological change occurs in a network that has reached a routing-convergence-state. The *convergence-time* is the time interval from t until the next routing-convergence-state is reached.

We use three topologies: $\langle 20, 4, 6\rangle$, $\langle 50, 6, 4\rangle$ and $\langle 80, 6, 5\rangle$. In the notation $\langle N, d, e\rangle$, N is the number of routers, d is the network diameter and e is the maximum router degree. To minimize topology-induced bias, we generate a topology with random interconnections. To exercise the Flooding Protocol, high values are chosen for d and e. All links have the same speed chosen from a range of 56 Kbps through 2 Gbps.

Link Speed and Convergence Time

The objective of this experiment is to determine the impact of link speed on the convergence-time and the bootup-convergence-time. All routers have unlimited amount of input and output buffers. The Wait Timer, RouterDeadInterval and the Hello Timer of each router are set to 40, 40 and 10 seconds respectively.

Initially, all routers are in the DOWN state and are booted simultaneously. After the network reaches routing-convergence-state, the bootup-convergence-time is measured, and a topological change is introduced at time t_0 by bringing down a link. The routers are allowed to respond to this topological change and reach the routing-convergence-state. The last action of the Flooding Protocol is the deletion of a link state request list or the receipt of a database description packet. Let the time of the last action be t_1. The convergence-time is measured as the time period $t_1 - t_0$.

Figures 3a-3c show the bootup-convergence-time and convergence-time over a range of link speeds for the 50 router network. In Figs. 3a-3b, for link speeds less than 4000 Kbps, the bootup-convergence-time for the RxmtInterval of 5 and 10 seconds are 20 and 30 seconds respectively. Since all routers are booted at the same time, they establish bidirectional communication in 10 seconds. If a link state advertisement is not acknowledged within RxmtInterval, a retransmission occurs. Consequently, we get the bootup-convergence-time to be the sum of one HelloInterval and twice the RxmtInterval. In Fig. 3b, for link speeds from 4000 to 6000 Kbps, the increase in the bootup-convergence-time is bounded by twice the RxmtInterval. For example, the bootup-convergence-time for the link speed of 6000 Kbps and RxmtInterval of 10 seconds is 50 seconds, giving an increase of 20 (50-30) seconds. In Fig. 3c, for link speeds above 50 Mbps, the increase in the bootup-convergence-time is bounded by three times RxmtInterval. As the link speed increases, the probability of input-buffer-overflow increases causing retransmissions because large numbers of flooding packets are received within an RxmtInterval. When a router, R_x, receives a flooding packet from a router, R_y, router R_x checks if this packet

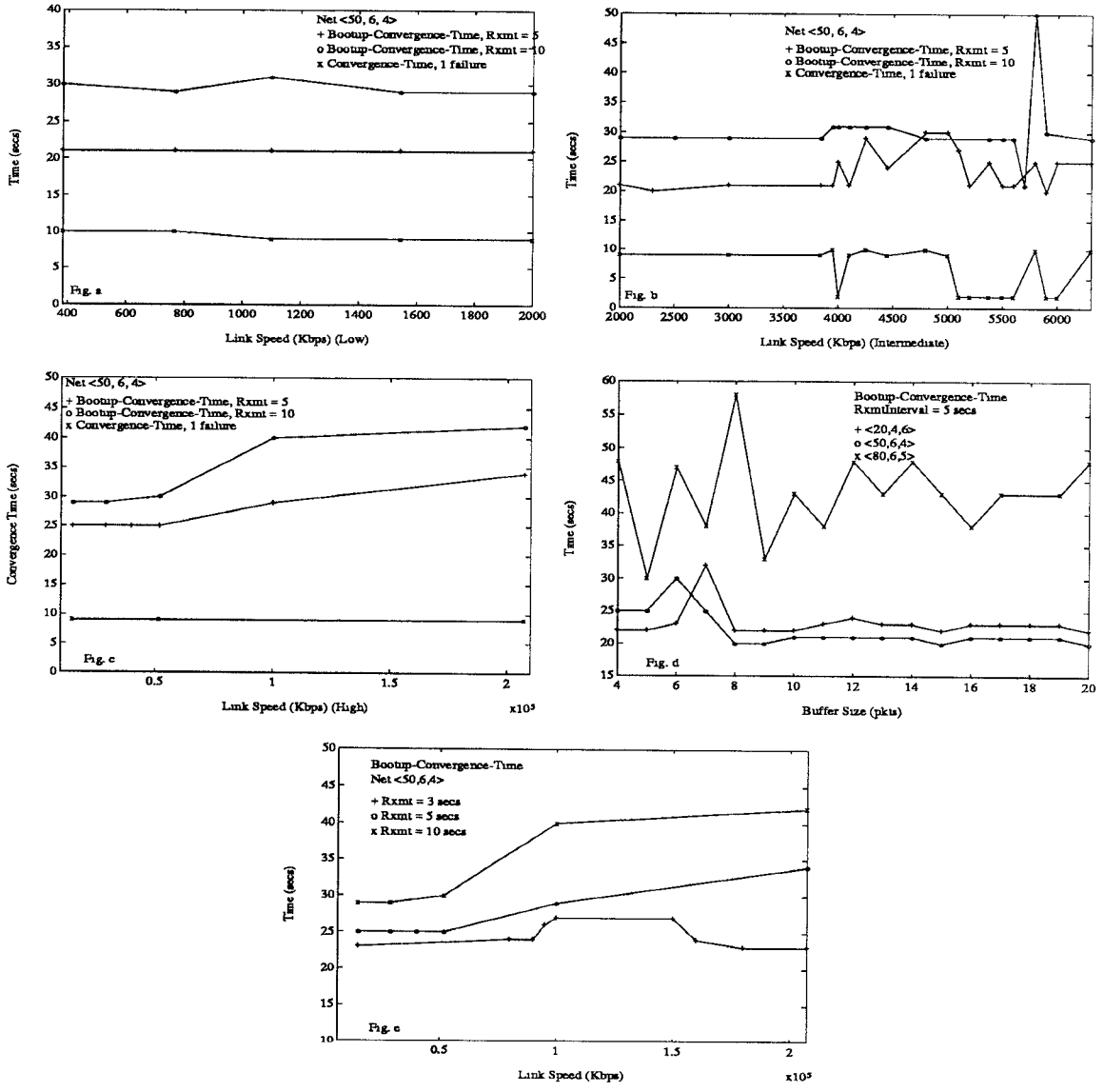

Figure 3: Impact of Link Speed, Buffer Size and RxmtInterval on Flooding Protocol

acknowledges a packet on R_x's retransmission list. Therefore, as the size of the retransmission list at R_x increases, the time to acknowledge a packet increases.

In Figs. 3a-3c, RxmtInterval does not affect the convergence-time which is bounded by 10 seconds. A router in this OSPF implementation responds to a topological change when its Hello Timer expires. To explain the bounding value, let $t_2 \leq t_1$, be the time at which the Hello Timers at all routers expire. The interval $t_2 - t_1$ is less than or equal to the HelloInterval, 10 seconds.

Buffer Size and Convergence Time

The objective of this experiment is to determine the impact of input buffer size on the bootup-convergence-time. In this experiment, for networks of 20, 50 and 80 routers, the link speed is fixed at T1 (1.544 Mbps) and the input buffer size is varied from 4 through 20. The experimental procedure is as described above.

The results of this experiment are shown in Fig. 3d. For any of the networks, convergence does not occur if the buffer size is less than or equal to three. This result implies that the lower bound of the input buffer size for the operation of an OSPF network of 20 or more routers is greater than three. Consequently, a router requires an input buffer memory size of at least 4 times the maximum size of a flooding packet.

For the 20 and 50 router networks, the buffer size has little impact on bootup-convergence-time. The bootup-convergence-time increases at a buffer size of 6 in the 50 router network and at a buffer size of 7 in the 20 router network. This increase results from two retransmissions which occur after the loss of a packet and its acknowledgment. This retransmission occurs at a higher buffer size for the 20 router network than for the 50 router network because the 20 router network has a higher router degree and must send out more link-state advertisements per output buffer than the 50 router network. The 80 router network demonstrates very noisy behavior because the number of link-state advertisements that must be flooded is large. Clearly, a buffer size of 20 or more is needed for networks with more than 80 routers.

RxmtInterval and Convergence Time

The objective of this experiment is to demonstrate the effect of a low setting of the RxmtInterval. The experimental procedure is as described above. In this experiment, we used the 50 router network with link speeds from 25 to 200 Mbps. Output and input buffer size is unlimited.

The results of this experiment are shown in Fig. 3e. Reducing the value of the retransmission timer lowers the bootup-convergence-time for all link speeds. However, the reduction is larger for higher link speeds. For example, at the link speed of 100 Mbps, a speedup of 14 seconds is observed when the retransmission timer is reduced from 10 to 3. This result verifies the suggestion in [1] that the setting of the retransmission timer can be reduced for high speed networks.

Buffer management is vital to the performance of the Flooding Protocol; otherwise, there is potential for performance degradation due to high contention for memory. The Flooding Protocol in an OSPF implementation may use inherent rate-based control mechanisms such as: (1) limit the number of simultaneous synchronizations, or (2) reduce the value of the retransmission timer. It is also recommended that a linear search of the retransmission list be avoided.

3 Summary and Conclusions

Open Shortest Path First (OSPF) is a dynamic, hierarchical routing protocol to support the TCP/IP networks. In this paper, a simulation of the OSPF Election Protocol shows three results: (1) The Designated Router(DR) can be elected in constant time. (2) If a router has a limited number of input buffers, a competition for buffers between the Election and the Flooding Protocols increases the election time and causes an oscillatory behavior. At each router, the Router-ID of the DR continuously changes causing instability. To solve these problems, Hello packets must be queued in a separate control queue and processed at a higher priority. (3) In the worst case, when the DR and the BDR fail at the same time, the DR-agreement-time is bounded above by twice the HelloInterval.

A simulation of the OSPF Flooding Protocol

using 20, 50 and 80 router point-to-point networks shows three results: (1) For the 50 router network, as link speed exceeds 4000 Kbps, the probability of overflowing the input buffers increases causing retransmissions. The increase in bootup-convergence-time from retransmissions is bounded by two and three times the RxmtInterval for link speeds of 4000 to 6000 Kbps and above 50 Mbps respectively. The increase in the bootup-convergence-time is due to large number of unacknowledged flooding packets received within RxmtInterval. (2) For 20 and 50 router networks, the input buffer size has little impact on the bootup-convergence-time. For the 80 router network, a small change in the input buffer size drastically change the bootup-convergence-time. (3) Reducing the value of the RxmtInterval lowers the bootup-convergence-time at high link speeds.

References

[1] J. Moy. The open shortest path first (OSPF) specification. Technical Report RFC-1131, SRI Network Information Center, October 1989.

[2] Deepinder Sidhu, Tayang Fu, Shukri Abdallah, Raj Nair, and Rob Coltun. Open shortest path first simulation. under preparation.