Journal of Network and Systems Management, Vol. 9, No. 2, 2001

On the Design of Link-State Routing Protocol for Connection-Oriented Networks

M. Sivabalan[1] and H. T. Mouftah[1,2]

Link-state routing protocols are being increasingly used in modern communications networks. A salient feature of this class of routing protocols is that network connectivity and state information of all links are available to nodes for making routing decision. Two main components of a link-state routing protocol are an update mechanism and a routing algorithm. These components must be properly designed for efficient routing. Various alternatives are possible for each of these components leading to different scenarios for routing protocol. In this paper, we quantitatively examine the impact of these alternatives on network performance using call-by-call simulations. Our design objective is to reduce call blocking ratio without significantly increasing routing overhead. We also present a new signaling scheme that can be used in conjunction with link-state protocols. We show that, if properly designed, this scheme can enhance the network performance.

KEY WORDS: Routing: link-state protocol: signaling: simulations.

1. INTRODUCTION

In connection-oriented networks, transfer of information between two users is accomplished by choosing a path and reserving resources (e.g., link bandwidth) along that path. A logical association between communicating users is referred to as a "call." A network uses its routing protocol for choosing paths. Routing protocols should be designed in such a way that the chosen paths meet objectives of both network operators and users. The user's objective can be translated into quality-of-service [1] supported by the path. The network operators, on the other hand, are concerned mainly with the revenue generated by the network, which implies efficient usage of network resources. Link-state protocol is a class of routing protocols in which a node knows the network connectivity and the

[1] Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada K7L 3N6.

[2] To whom correspondence should be addressed. E-mail: *mouftah@eleceng.ee.queensu.ca*

1064-7570/01/0600-0223$19.50/0

state information of all the links in the network. The state information helps a node to determine the degree of desirability of including a link in paths. A node can directly gather the state information of its outgoing links. It also broadcasts (advertises) the state information of its outgoing links to all other nodes in the network through Link-State Update (LSU) messages. Furthermore, a node is responsible for determining when to trigger an LSU. Since complete information is available to every node in the network, link-state protocols permit source routing where a source node can autonomously choose a path to any other destination node. A major concern with link-state protocol is the processing overhead incurred due to dissemination of LSU messages. A hierarchical routing approach can be used to reduce such overhead, but at the cost of poor routing accuracy [2, 3]. OSPF (Open Shortest Path First) [4] and PNNI (Private Network Node Interface) [5] are two well-known examples of link-state protocols.

In this paper, we examine the influence of the components of a link-state protocol on the network performance using call-level simulations. We assume connection-oriented networks with distributed source routing paradigm, where a source node is responsible for computing the paths without involving a specialized central processor or any other nodes in the network. The network performance is evaluated mainly with respect to call blocking ratio, which is the ratio of rejected calls to offered calls. From an operator's perspective, the lower the call blocking ratio the higher the revenue. A tolerable blocking ratio is also important to attract users, and can therefore be considered as a call-level quality-of-service measure. We also present an intelligent signaling scheme that can be used in conjunction with link-state protocols. With this scheme, nodes gather routing information from not only LSU messages but also signaling messages during call establishment. This scheme is simple to implement and can enhance the network performance.

The rest of the paper is organized as follows: Section 2 describes the components of a link-state protocol. In Section 3 we describe the specific details of the network and the call establishment/termination models used for the simulations. We then present and discuss the simulation results in Section 4. In Section 5, we explain the proposed signaling scheme and demonstrate its impact on the network performance. Finally, Section 6 concludes the paper.

2. COMPONENTS OF LINK-STATE PROTOCOL

As shown in Fig. 1, a link-state protocol consists of a *routing algorithm* and an *update mechanism*, along with a routing table and a Routing Information Base (RIB). All these components reside in every node that is capable of making routing decision (path selection). A node computes the paths using the routing algorithm. The information required to compute paths is stored in RIB. Furthermore, a node constructs and maintains its RIB using its own knowledge

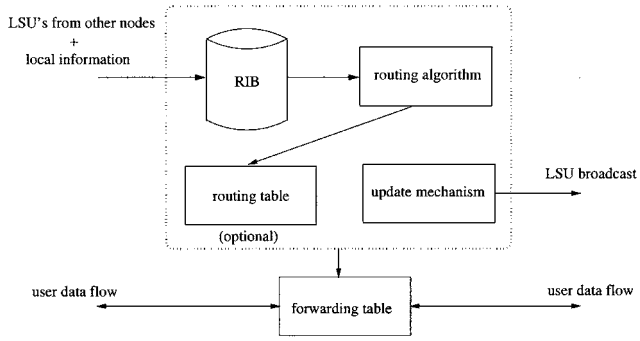

Fig. 1. Components of a link-state protocol.

about its outgoing links as well as the information from LSU messages received from the rest of the nodes. At any given time, a node can directly assess the state of any of its outgoing links. Thus, the content of RIB corresponding to the outgoing links is always accurate. But for the state information of a nonincident link, a node has to rely on LSU messages. The accuracy of the contents of RIB corresponding to the nonincident links depends on LSU advertisement rate; the higher the rate the better is the accuracy. A node advertises the state of its outgoing links using an update mechanism. [Note: An efficient link-state protocol could also take nodal state, e.g., CPU and buffer utilization, into consideration. However, we ignore the nodal effects in this study.] As we explain later, depending on the implementation of the routing algorithm, computed paths can be optionally stored in a routing table. In a connection-oriented network, the routing decision is made only at the call set-up time. Once a path is computed, the user data flow through the same path throughout the duration of the call. The input-output mapping for the calls passing through the node is stored in a forwarding table.

2.1. Update Mechanism

The simplest and most commonly used way of disseminating LSU messages is flooding [6]. Using sequence numbers, a node is prevented from unnecessarily transmitting duplicate messages. Another LSU mechanism is based on spanning-tree approach in which a node maintains a spanning-tree connecting all nodes in the network with itself as the root [6]. This scheme reduces the number of LSU messages generated per update at the cost of increased complexity due to setting up and maintaining the spanning-trees. We have also used flooding for our simulations.

LSU messages carry link-state information. The state of a link can be char-

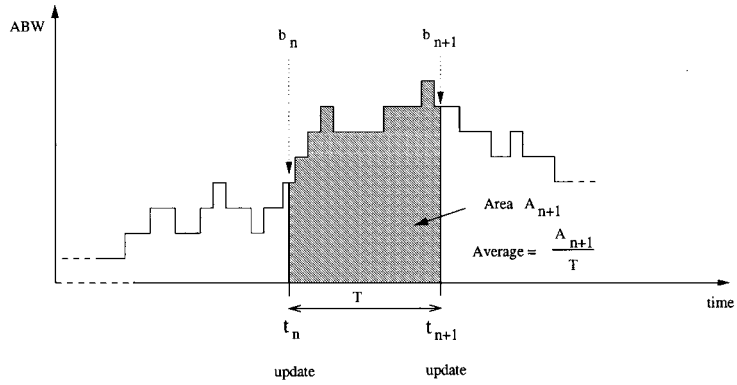

Fig. 2. A Sample profile of ABW on a link.

acterized by metrics and attributes. A metric is quantitative, such as link bandwidth, Available Band-Width (ABW), and delay. Attributes of a link are qualitative, and are considered individually to determine the desirability of including the link in a path. They include link type (e.g., satellite, terrestrial) and ABW. Note that ABW can be used as a metric to select a path consisting of lightly loaded links as well as an attribute to avoid the links which cannot meet the requirements of an incoming call.

In this paper we assume that the link bandwidth is the only resource consumed by the calls. The state of a link is characterized by ABW on the link. This metric changes as connections are added to or torn down from the link. If an LSU is triggered following every single connection, the amount of LSU messages generated will be potentially high resulting in increased utilization of nodal processing power. On the other hand, if LSU messages are advertised at a very low rate' the resulting paths may be poor leading to inefficient use of network resources. Thus, the update mechanism must be properly designed in order to achieve efficient routing with minimal overhead.

An issue we like to explore is whether instantaneous or some kind of statistical average of ABW should be advertised. Let t_n and t_{n+1} be two consecutive update instants, and b_n and b_{n+1} be the instantaneous ABW at these instants, where $n = 0, 1, \ldots$ (see Fig. 2). We denote the advertised value of the metric at time t_{n+1} by a_{n+1}. The following different representations of a_{n+1} are examined:

- Instantaneous: $a_{n+1} = b_{n+1}$
- Pure average: $a_{n+1} = (A_{n+1}/T)$ (see Fig. 2)
- Moving average: $a_{n+1} = \alpha a_n + (1 - \alpha)b_{n+1}$ where $0 < \alpha < 1$.

LSUs can be triggered in an event-driven or periodic fashion. In the former, a node sends out an LSU message whenever it detects an occurrence of a

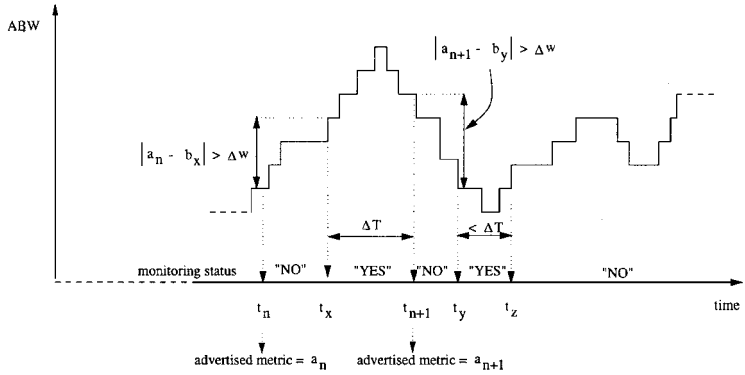

Fig. 3. Detecting significant events.

significant event in any one of its outgoing links. Proper definition of significant events is an important design issue. A change in the network topology, such as a link failure, is definitely a significant event. Moreover, with traffic-sensitive routing, the selection of path between a given pair of nodes depends on the traffic load on the links. Thus, timely update of changes in the link load is important for finding efficient paths. In the periodic approach, a node triggers the LSUs periodically after every T_{up} seconds regardless of the activities on the links. In practice, a combination of both approaches is used in the modern networks in order to provide efficient as well as robust routing [5]. We examine both periodic and event-driven LSUs. For the event-driven LSU, a change in the ABW is considered significant if the present ABW differs from the previously advertised value by some predefined margin.

Figure 3 illustrates how a node detects a significant change on an incident link. A significant event happens at time $t > t_n$ if $|a_n - b_t| > \Delta w$, where Δw is the predefined margin, b_t is the actual ABW at time t, and a_n is the last advertised value at time t_n (it can be either periodic or event-driven). Even though significant, a change may be short-lived and advertising such a change may not be useful to the routing algorithm, but can instead waste the nodal processing power. We suppress such trivial advertisements as follows: a node triggers an LSU message only if a significant change persists for a duration of at least ΔT seconds. When the node detects a significant change, it starts a timer and begins to monitor the link. If the change remains valid throughout ΔT seconds, the node triggers an LSU, resets the timer, and stops monitoring the link (e.g., between t_x and t_{n+1}). While monitoring the link, a node resets the timer if another change on the link makes the previous change obsolete (e.g., between t_y and t_z). Moreover, when a link is lightly loaded, advertising a change in the link may not be beneficial to the routing algorithm. We can take advantage of this fact to reduce

the number of LSU messages generated in the network. An *update threshold* U_{up} is set on the link utilization to decide whether or not to trigger LSU advertisement following a significant change. An event-driven update is triggered only if the utilization exceeds U_{up}. However, U_{up} does not affect the periodic-updates.

2.2. Routing Algorithm

Path selection with link-state protocol is typically formulated as a shortest path problem. Each link is assigned a cost, and the cost of a path is the summation of the costs of all the links on the path. A path with the lowest cost is chosen for routing a call. The link cost is a function of link metric and represents the degree to which a certain link is preferable for a path. An efficient routing algorithm should yield paths which have a better chance of accepting incoming calls and, at the same time, should reduce the network resources consumed by the paths as much as possible. This is important to improve the success of not only the incoming calls but also the future calls. We try to achieve this objective using the following link cost function:

$$\text{link cost} = 1 + \frac{k}{\text{normalized ABW}} \tag{2.1}$$

where k is a constant which determines how quickly the routing algorithm reacts to the traffic load on the link; the higher the value of k the quicker the reaction. The normalized ABW is computed as the ratio of ABW to the total bandwidth. When $k = 0$, this function leads to minimum-hop paths. When the links have plenty of ABW, the routing algorithm picks the minimum-hop paths. But, as ABW is reduced, the minimum-hop paths get saturated and the routing algorithm attempts to choose paths that are more likely to accept calls. With appropriate selection k, the routing algorithm gracefully becomes load-sensitive as the network load increases. However, as we will show later, if the link cost is too sensitive to the link load the routing protocol can indeed degrade the network performance. The constant 1 in this cost function provides damping to prevent the routing algorithm from being too sensitive to the changes in the link load. Basically, the idea of using the link cost of this form is to enable the routing algorithm to choose paths which are more likely to accept calls while distributing the traffic to utilize the network resources efficiently.

We examine two approaches to computing paths: on-demand and pre-computed. With on-demand approach, a source node computes a path only upon the arrival of a new call using the latest link-state information in its RIB. The source node prunes its own view of the topology map to eliminate the links which are likely to reject the call (using ABW as the attribute). It then runs the Dijkstra's

shortest-path algorithm [6,7] to get a path with the lowest cost. With pre-computed approaches, path computation is triggered by certain events. We consider two approaches to triggering path computation. In the first one, called pre-computed(1), a new path between a source-destination pair is computed only when the previous path between that pair rejects a call. This approach is similar in spirit to the Dynamic Alternate Routing (DAR) protocol [8,9] used in the British Telecom networks. In the second approach, called pre-computed(2), a source node computes new paths to all destinations whenever it receives an LSU message [10]. In addition, a source node computes a new path whenever it finds that an earlier path can no longer support new calls due to insufficient bandwidth on its outgoing link. This is to take advantage of the accurate state information of the outgoing links available to the nodes. The pre-computed paths are stored in a routing table.

3. SIMULATION MODEL

We consider a 16-node mesh connected network shown in Fig. 4. Each link has a capacity of 50 units and a propagation delay of 5 ms. The processing delay for both signaling and LSU messages at each node is 1 ms. The network supports only a single class of service. Calls are bi-directional and point-to-point. Each call requires a unit bandwidth for both forward and reverse directions. Call holding times are exponentially distributed with a mean of 1 minute. Furthermore, calls are assumed to arrive at each node according to Poisson process, and all nodes have identical arrival rates. For any given call, the source node uniformly picks any other node as the destination. During simulations, the network load is altered by varying the mean call arrival rate.

To support a call, the network should allocate the required bandwidth in both forward and reverse links. The source node needs to allocate bandwidth only on the forward link, the destination node needs to reserve bandwidth only

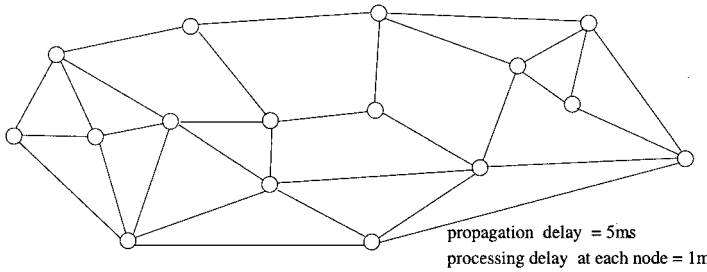

Fig. 4. The 16-node network used for simulations.

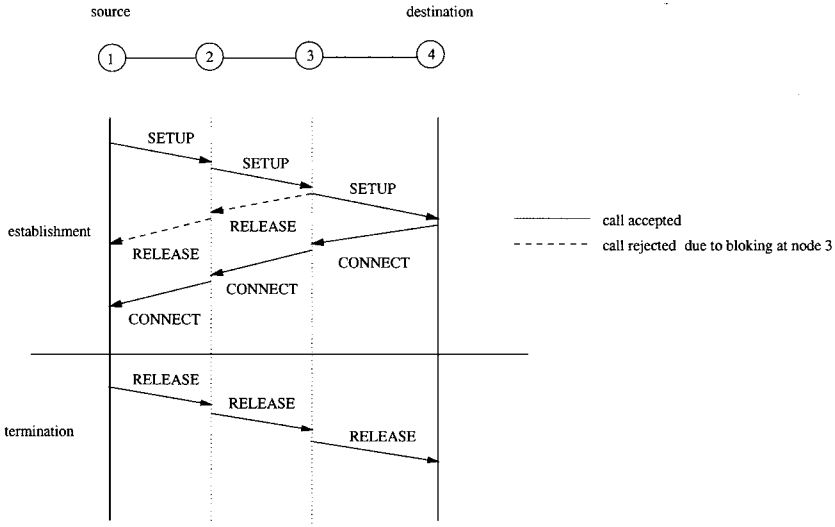

Fig. 5. Call establishment/termination model.

on the reverse link, and the intermediate nodes must allocate bandwidth on both forward and reverse links. The call establishment/termination model is depicted in Fig. 5. Upon arrival of a call, the source node finds a path to the destination. It then initiates the connection establishment by sending out a SETUP message along the chosen path. Since source routing is used, the SETUP message contains the complete description of the path (i.e., a list of nodes to be visited by the SETUP message). As the SETUP message proceeds, each node on the chosen path checks the required outgoing (forward and/or backward) link to see if it has enough bandwidth to support the call. If it does, bandwidth is reserved for that call and the SETUP message is forwarded to the next node enroute. Otherwise it sends a RELEASE message back towards the source node. As the RELEASE message travels, the bandwidth reserved on the up-links is released. A rejected call is assumed to be lost. If the SETUP message successfully reaches the destination node and if the destination node accepts the call, a CONNECT message is sent back to the source node. When the CONNECT message reaches the source node, the call is assumed to begin its data transmission. Note that we do not simulate the actual data transmission.

4. SIMULATION RESULTS

The main goal of this study is to examine the impact of the link-state protocol components on the network performance. The behavior of these components

are characterized by the parameters explained in section 2; α, k, Δw, ΔT, T_{up}, and U_{up}. When the influence of particular parameter is observed, the values of all other parameters remain fixed. For steady-state observations, each point in the following graphs is obtained as an average of five independent simulation runs. Each simulation run corresponds to 8 hours of network operation time with initial 10% discarded to account for transient effect.

4.1. Link Metric Representation

First, we like to understand the effect of link metric representation on the network performance. We carried out simulations with each of the three representations given in Section 2.1. We set $\alpha = 0.5$ and $k = 0.01$. In this experiment, only periodic update was used, and we examined the network with $T_{up} = 10$ and $T_{up} = 90$ seconds. We note that the accuracy of RIB is better with a shorter update period, and therefore we can expect a higher call blocking ratio when $T_{up} = 90$ seconds. Figure 6 shows the variation of call blocking ratio with the load when $T_{up} = 90$ seconds. We find that instantaneous value is a poor indicator of the link metric at both update periods. Pure average does not have any advantage over instantaneous value when $T_{up} = 10$ seconds, however it leads to a significant reduction in call blocking ratio when $T_{up} = 90$ seconds. But, with both update periods, moving average leads to the lowest call blocking ratio.

A node gathers the state information about the non-incident links from LSU messages. Therefore it is crucial that the advertised metric should serve as a good predictor of ABW on such a link between two consecutive LSUs. Figure

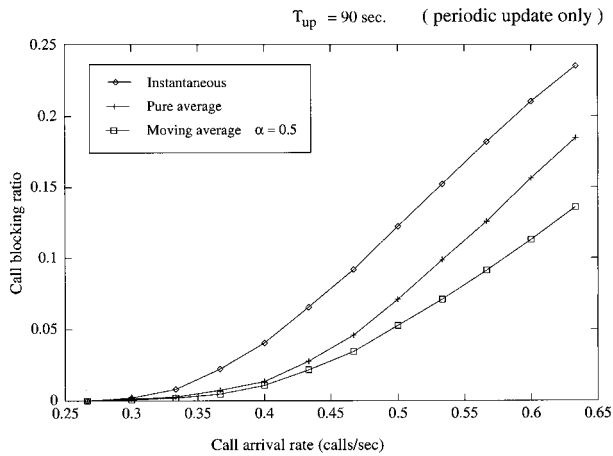

Fig. 6. Impact of link metric representation on the network performance.

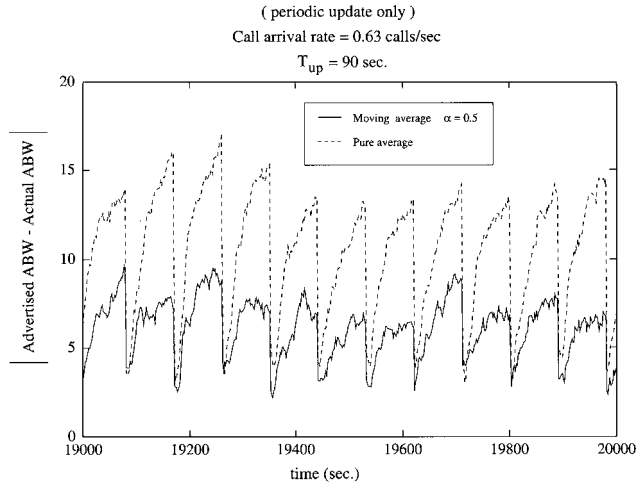

Fig. 7. The variation of absolute error in the advertised metric with time.

7 compares pure and moving average metric representations with respect to the absolute difference in the advertised and actual metric values (averaged over all links). This observation is made at an arrival rate of 0.63 calls/sec. Ideally, the lower the difference the higher the probability that the call is accepted on a link. In Fig. 7, the valleys lie in the vicinity of LSU advertisements and the difference increases soon after the advertisements. The difference is more pronounced when $T_{up} = 90$ sec. The moving average representation yields lower error than pure average with both update periods. We also observed when $T_{up} = 10$ sec., the error with the instantaneous representation (results are not shown here) is identical to that under the pure average representation. But when $T_{up} = 90$ sec., the instantaneous representation leads to the largest error. This result supports our earlier observation that, among all the representations we considered, moving average is the best in terms of blocking performance.

The mismatch between the advertised and actual metric is a good indicator of routing "oscillation." This phenomena can be best explained through an example: let us assume that a link advertises a very large metric. Following this advertisement, all the nodes receiving LSUs will find the link attractive and start using it more often. As a result, the link quickly becomes saturated and therefore will start rejecting the calls. The distant nodes are not aware of this situation and hence keep including the link in the path computations until the link triggers another LSU. The situation is reversed at the next advertisement in which the link reports a very low metric. Now, the link seems unattractive and hence all the nodes will try to exclude the link as much as possible during the

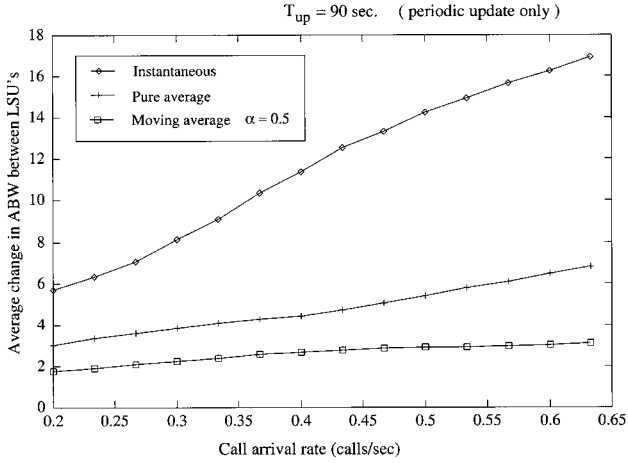

Fig. 8. The variation of average change in the advertised metric between consecutvie LSUs.

path computations. In the extreme case, the link would be completely unused. The outcome of the oscillation is that the link switches between saturated and unsaturated states. The oscillation makes the network inefficient.

A link-state protocol should not be too responsive to the link load to cause oscillation. The oscillation increases the call blocking ratio especially when T_{up} is very long. In this case, the routing algorithm keeps selecting the poor paths for a long time, even though better paths are available in the network. The relationship between routing oscillation and the network load is illustrated in Fig. 8. We measured the degree of oscillation by the average of absolute change in the advertised metric values (averaged over all links) between consecutive LSUs. First we noted that, with larger T_{up} the network experiences a higher degree of oscillation. Also, for a given T_{up}, the oscillation increases with the load. Secondly, the pure average representation helps dump the oscillation, and the moving average representation provides the highest damping. So, moving average of link metric is easy to compute and leads to better performance. We used this representation for the rest of the experiments.

We now examine the effect of α on the network performance. A larger α gives more weight to the previously advertised values and hence reduces the adaptation of the routing protocol. A smaller α, on the other hand, makes the routing protocol too responsive to the network load and hence increases the oscillation. In the previous experiment, we intuitively set $\alpha = 0.5$ to have a compromise between adaptation and oscillation. However, here we investigated the variation of call blocking ratio with load under different values of α. It has been observed that at a low T_{up} (10 sec.), among the tested values ($\alpha = 0.1$ to $\alpha = 0.9$)

$\alpha = 0.5$ gives the lowest call blocking ratio and $\alpha = 0.9$ yields the highest call blocking ratio. When $T_{up} = 90$ seconds the best performance is achieved when $\alpha = 0.7$ and the worst performance is obtained when $\alpha = 0.1$. These observations also show that a dynamic routing protocol should not be too adaptive if it has to operate with inaccurate state information (as with 90 sec. update). If LSUs are triggered at a reasonably high rate, $\alpha = 0.5$ can be a better choice.

4.2. Triggering LSUs

So far we have considered only periodic updates. As mentioned before, for robust performance both periodic and event-driven updates are used in practice. Event-driven updates provide timely updates of changes in the link-state, and are crucial to efficient routing. Periodic update adds robustness to the update mechanism. The update mechanism considered in this paper is characterized by Δw, Δt, T_{up}, and U_{up} (see Section 2.1). We now examine the effect of these parameters. An efficient update mechanism should provide the necessary information for the routing algorithm while keeping the overhead minimal. We measure the processing overhead of the update protocol by the average number of LSU messages processed per second at each node, assuming that one update message is generated per link.

We investigated the influence of the update protocol parameters Δw and Δt on the network performance when $T_{up} = 90$ seconds. We found that, when $\Delta t = 1$ second, call blocking ratio reduces as Δw gets smaller. It was also evident that as Δw gets smaller, more LSU messages are generated and hence LSU processing overhead increases. Also we noted that, when Δw is small (say 4%), LSU processing overhead increases with the network load and then asymptotes. The value of Δw should be chosen such that the performance gain should justify the overhead. In our example, increasing Δw from 4% to 10% increases the call blocking probability by only 0.5% but is associated with approximately a three-fold decrease is the LSU processing overhead at the heavy load. Also we noted that the combined update with $\Delta w = 10\%$ is identical to the pure periodic update with $T_{up} = 10$ sec. in terms of blocking ratio, but the former reduces the LSU processing overhead by approximately four times. As intuitively expected, we found that an increase in Δt is associated with a decrease in the overhead. As long as Δt is not very large, the blocking performance is insensitive to Δt. In our case, we do not see any performance degradation even when $\Delta t = 5$ sec. But when $\Delta t = 10$ sec., we found that call blocking ratio begins to show a sign of increase at heavy load.

These experiments indicate that, with the combined update mode, the event-driven update (with the proper selection of parameters) takes the major role in providing routing information. Therefore it is not necessary to have a smaller periodic update interval. Our results have shown that, with $\Delta w = 10\%$ and $\Delta t =$

1 second, the processing overhead can be significantly reduced by using longer update interval. However, a very large T_{up} may compromise the reliability of the update mechanism. Also, we found taht the blocking performance is insensitive to the update interval.

As we mentioned before the LSU processing overhead can be further reduced by advertising a significant change in the link metric only if the link utilization is above the update threshold U_{up}. This is evident from the results obtained on the impact of U_{up} on the network performance. As U_{up} is increased from 20% to 70%, the LSU processing overhead at each node is reduced by approximately one message/sec under heavy load. But a very large U_{th} will certainly make the routing protocol less adaptive. For instance, in our example the network performance degrades when $U_{th} = 80\%$. However, we found that U_{up} does not significantly affect the blocking performance.

4.3. Link Cost

In all these experiments we set $k = 0.01$. We now examine the influence of k on the routing performance. When $k = 0$, the bandwidth dependent term in the expression for the link cost given by Eq. (2.1) vanishes and the routing algorithm provides minimum-hop paths. Conventionally, minimum-hop path between a source-destination pair remains the same regardless of the link load. In our experiments, the paths are computed based on the pruned topology, excluding the links with insufficient ABW. So, they may change occasionally. When $k \neq 0$, the link cost increases as ABW decreases, and a path with larger number of underutilized links is preferred over one with fewer but heavily utilized links. The constant k determines how quickly the routing protocol retreats from minimum-hop routing. Note that $k = \infty$ is an extreme in which the damping constant in Eq. (2.1) has no effect.

Our results showed two important observations. First, as long as $k \geq 0.001$ (this is the lowest nonzero value we used in the simulations and perhaps even smaller values are possible), the blocking performance is almost insensitive to the selection of k. Second, as k increases LSU processing overhead increases specially at heavy load. A large value of k makes the routing algorithm reacts fast to the changes in ABW, and with the combined update, this means increase in the rate of generation of LSU messages. So, k should be large enough to achieve better blocking performance, but not too large to unnecessarily increase the overhead.

4.4. When to Compute Paths

We have considered only on-demand routing algorithm thus far. We now compare the two pre-computed approaches, pre-computed(1) and pre-com-

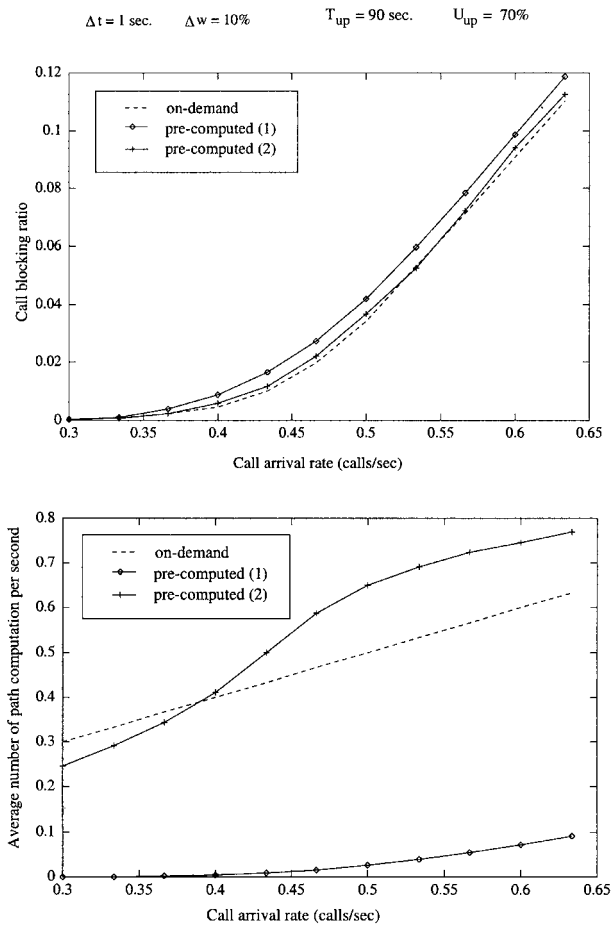

Fig. 9. Impact of path computation method on the network performance.

puted(2) (see Section 2.2) with the on-demand routing in terms of the computational overhead and call blocking ratio. The computational overhead is determined by the number of path computations carried out per second at each node. Figure 9 compares the path computation approaches in terms of call blocking ratio and the computational overhead. With the on-demand approach path is computed whenever there is a call request and hence the path computation rate is equal to the call arrival rate at each node. Since a path is computed at the time of call arrival, this approach yields paths which are more likely to accept the calls. With pre-computed(1) new paths are computed only when the previous path begins to block calls, and this approach leads to the highest call block-

ing ratio. In terms of blocking, pre-computed(2) and on-demand approaches are almost identical.

In terms of computational overhead, pre-computed(1) is less demanding than other two. Under light load, pre-computed(2) has less path computation rates than on-demand scheme. Since the LSU generation rate increases with the load, the path computation rate of pre-computed(2) also increases with the load. Above the arrival rate of 0.4 calls/sec., the path computation rate of pre-computed(2) exceeds that of on-demand. We like to note that with both on-demand and pre-compute(1) only one path is computed at a time, but with pre-compute(2) each node computes the paths to all the destinations, i.e., a shortest-path tree is computed with itself as the root. On a SUN SPARC2 workstation, it takes about 700 μs to compute a single path and about 900 μs to compute a path tree for our network model. Therefore, in terms of blocking performance and computational overhead on-demand approach outperforms the two pre-computed approaches.

5. AN INTELLIGENT SIGNALING SCHEME

During call establishment and termination, signaling messages traverse the links along the chosen paths. A source node chooses its paths based on the latest information available in its RIB (which cannot be always accurate). While traversing the links, the signaling messages get an opportunity to observe the accurate state of the links. Normally routing protocols do not utilize the signaling messages to gather the network state information. We argue that this is inefficient. Instead of completely relying on the LSU messages, the nodes can use the signaling messages as intermediate agents to learn the state information and use it for subsequent path selection. For example, using a congested link that has just blocked a call immediately for the next arriving call is not prudent. Because it may take a while for the congestion to relieve, and the new call has a better chance of being blocked by the link.

In this section we present an intelligent signaling scheme in which the source node remembers the heavily loaded links for a short time. During the call establishment, the source node learns the first link with the lowest ABW encountered by the SETUP message. This implementation requires two additional fields in the SETUP message; one for recording the lowest ABW and another for recording the identity (ID) of the corresponding link. These fields are *Min_abw* and *Link ID* in the example shown in Fig. 10. Note that, before fowarding the SETUP message, the source node sets the value of *Min_abw* to ∞. Upon receiving the SETUP message, a node compares the ABW of the required outgoing link on the path with the minimum ABW encountered by the SETUP message thus far, and if the former is less than the later the node updates the fields in the message with the link ID and its ABW. The comparison is carried out by the call admission entity in the node during the bandwidth allocation

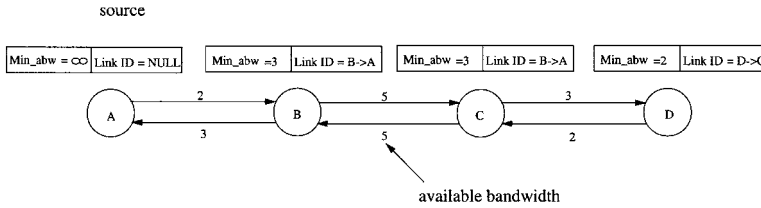

Fig. 10. An example of the operation of the new signaling scheme.

process, and therefore does not significantly increase the computational overhead at the nodes. If the call is successful, the SETUP message returns to the source node as a CONNECT message, and no bandwidth comparison is made along the return path. If the call fails the RELEASE message returns to the source node and the bandwidth comparison is not made along the return path. In this case, the blocking link is marked as the congested link.

Upon receiving either a CONNECT or RELEASE message, the source node examines the message to see if *Min_adw* recorded in the message is below some predefined threshold C_{th}. If so, the source node removes the corresponding link from its topology map for a time t_{bo} called *backoff* time. The link is not considered for path computation during the backoff period. If, during the backoff period, the source node receives an LSU message corresponding to the excluded link, it includes the link in the topology map. This is because the LSU message contains more recent information, and the routing algorithm can use the latest information to decide whether or not to use the link. If there is no LSU generatewd from the link during the backoff time, the source node includes the link in the topology map only after the backoff time. By doing so, the source node use the information gathered during a call set-up process to make a better routing decision for subsequent calls. With on-demand routing, when a path is computed, ABW of a link is used as an attribute to decide whether or not the link can support a new call. The backoff scheme introduces another time-dependent attribute to the link. When a link is excluded from topology, it is marked with an earliest time (which is t_{bo} away from the time of exclusion) at which it will become eligible for path computation. The source node can consider the link only if the link is eligible and has an adequate ABW. We note that it may be possible to update the RIB with the value of C_{th} instead of backing off. With this approach, the routing information at the nodes would no longer be identical. We will examine the impact of such an approach in the future.

Two parameters characterizing this scheme, C_{th} and t_{bo} must be carefully chosen for effective operation of this scheme. With a larger C_{th}, the source node would tend to avoid many links at a time. Thus, the resulting paths tend to be longer. This may not matter at low load where the network resources are plenty

but can increase the call blocking ratio at high load. The selection of t_{bo} also has a similar effect. With large values of C_{th} and t_{bo}, the blocking performance improves at low load and degrades at high load. So, C_{th} and t_{bo} must be adjusted in accordance with the load. We implement such a control which dynamically modifies t_{bo} according to the call arrival rate at the source node. When calls are arriving at rate λ the corresponding t_{bo} is computed using the following rule:

$$t_{bo} = T_0 \left(\frac{\lambda_0}{\lambda} \right)^n \tag{5.1}$$

where T_0 is the backoff time used with reference load λ_0, and n is a constant which determines how t_{bo} varies with λ (above and below λ_0). The higher the value of n the faster is the reduction of t_{bo} as λ exceeds λ_0. Figure 11 illustrates the behavior of the signaling scheme using a fixed timer and a proportional timer specified by this equation with $T_0 = 5$ seconds and $\lambda_0 = 0.45$ calls/sec. By adjusting the timer according to the load at the source, we can reduce the call blocking ratio at low load through intelligent signaling without causing performance degradation at high load.

The new signaling scheme is used with a base link-state protocol whose parameters have been tuned to minimize the call blocking ratio. The maximum reduction in the call blocking ratio with the new scheme is only about 1%. However, the real power of the scheme is visible when the network operates with a lot of inaccurate information. To illustrate this power, we compare the performance of network with and without intelligent signaling when there is no LSU at all (see Fig. 12). In this case, a source node attempts to route a call on a minimum-hop path. Now the performance improvement due to the new scheme is quite significant.

We finally note that we have not considered alternate path routing, i.e., we allow only one attempt per call. If used in controlled manner, alternate path routing can certainly reduce call blocking ratio [11,12]. But price paid for the improvement is the increased call set-up delay. Using intelligent signaling, we try to increase the success probability of the call set-up attempt at the first trial without significantly increasing the network complexity. Alternate path routing can be combined with the intelligent signaling scheme to further enhance the network performance. We will explore this approach in the future.

6. CONCLUSIONS

We have presented a comprehensive simulation study of link-state routing protocol. The important components of the protocol were identified and their influence on the network performance was studied in a step-by-step manner.

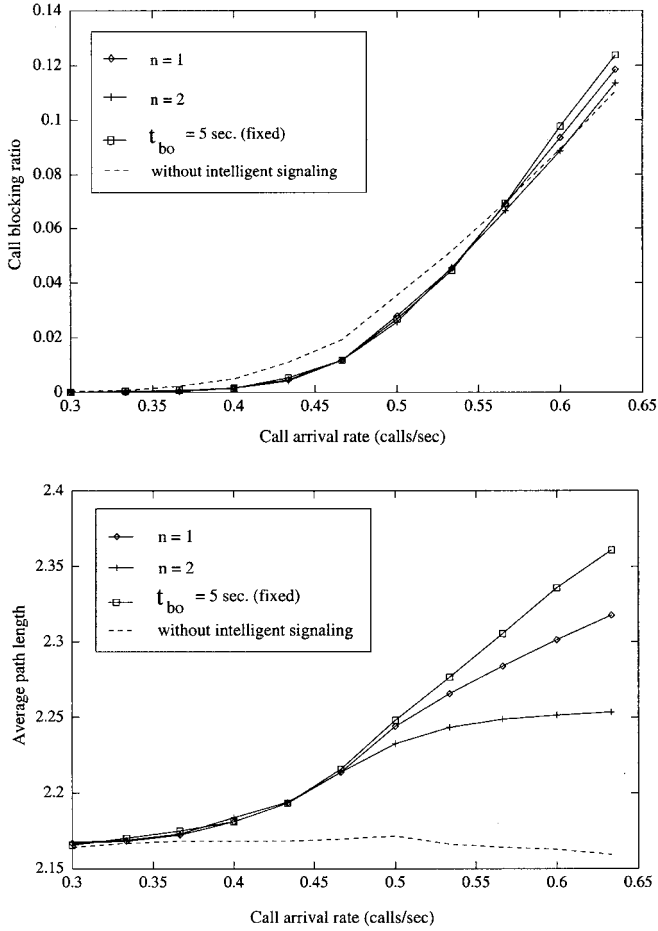

Fig. 11. The performance of the new signaling scheme with fixed and proportional timer.

The simulation experiments have shown that the moving average representation is a very effective way of advertising the available bandwidth on a link in terms of blocking performance. A link-state protocol that is highly responsive to the traffic on the links could degrade the network performance due to oscillation. The impact of oscillation is pronounced if the link-state information is highly inaccurate. Also, with a combined updates, a highly responsive routing protocol tends to generate more update traffic. We have also shown how to reduce the LSU processing overhead through filtering. Finally, an intelligent signaling scheme was proposed for improving the network performance. This scheme is

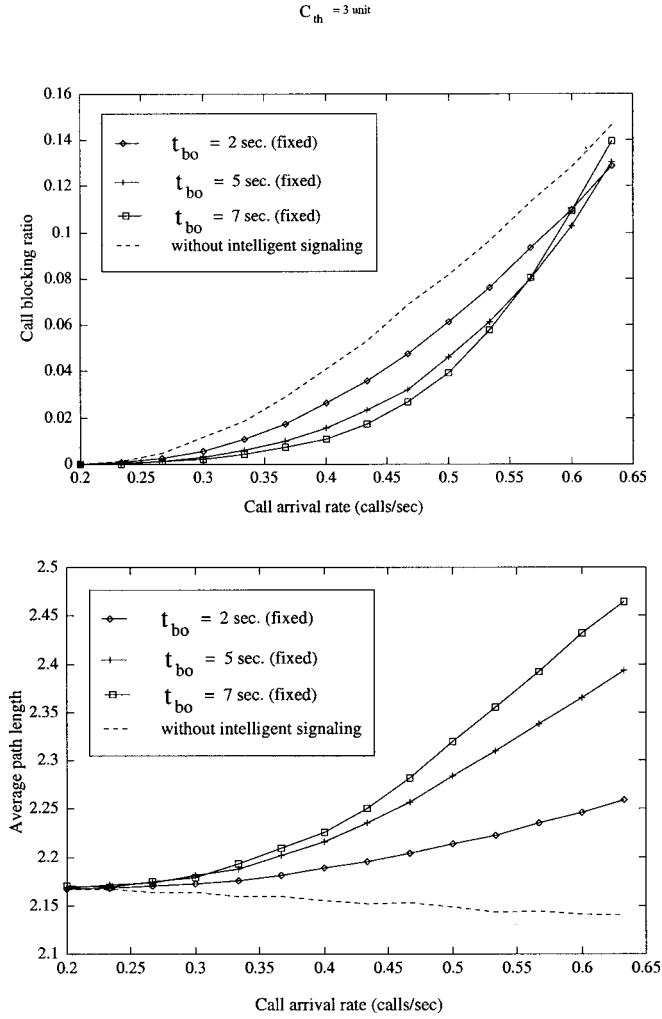

Fig. 12. The performance of the network with intelligent signaling when there is no LSU.

simple to implement and yields significant performance improvement specially when accurate link-state information is unavailable for path computation.

REFERENCES

1. W. C. Lee and M. G. Hluchyi, and P. A. Humblet, Routing subject to quality-of-service constraints in integrated communication networks, *IEEE Network Magazine*, pp. 46–55, 1995.

2. L. Kleinrock and F. Kamoun, Hierarchical routing for large networks, *Computer Networks*, Vol. 1, pp. 155–174, 1977.
3. W. C. Lee, Topology aggregation for hierarchical routing in ATM networks, *ACM SIGCOMM'94*, Vol. 25, No. 2, pp. 82–92, 1995.
4. W. Moy, The OSPF Specification Version 2 (RFC 1247), OSPF IGP Working Group of IETF, 1990.
5. The ATM Forum, Private Network Node Interface Specifications Version 1.0, PNNI Subworking Group, 1996.
6. D. Bertsekas and R. Gallager, *Data Networks*, Prentice Hall, New Jersey, 1992.
7. E. Dijkstra, A note on two problems in connection with graphs, *Numerical Math.*, Vol. 1, pp. 269–271, 1959.
8. R. J. Gibbens, F. P. Kelly, and P. B. Key, Dynamic alternative routing-modeling and behavior, *Proceedings of the 12th International Teletraffic Congress*, Torino, pp. 3.4A.3.1–3.4A.3.7, 1988.
9. J. Y. LeBoudec and T. Prezygienda, A route pre-computation algorithm for integrated services networks, *Journal of Network and Systems Management*, Vol. 3, No. 4, pp. 427–449, December 1995.
10. C. Parris and H. Zhang, Dynamic rerouting of guaranteed quality-of-service connections, *Journal of Network and Systems Management*, Vol. 4, No. 2, pp. 181–220, June 1996.
11. J. M. Akinpelu, The overload performance of engineered networks with nonhierarchical routing, *AT&T Bell Laboratories Technical Journal*, Vol. 63, No. 7, pp. 1261–1281, 1984.
12. M. Sivabalan and H. T. Mouftah, Alternate path routing over ATM networks, *Proceedings of OCRI/TRIO Seminar on Future Trends in Fast Packet Switching*, Kingston, Ontario, 1995.
13. M. de Prycker, *Asynchronous Transfer Mode: A Solution for Broadband ISDN*, Ellis Horwood Ltd., New York, 1991.

M. Sivabalan received a B.Sc. degree in Electrical and Electronic Engineering from University of Peradeniya, Sri Lanka, and an M.Sc. and a Ph.D. degree in the areas of Cryptography and high-speed network routing, respectively, from the Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada.

Hussein T. Mouftah has been a professor in the Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada since 1979. Prior to that he worked with the Data Systems Planning Department at Nortel Networks (then BNR), Ottawa, Canada, for three years. He is a Fellow of the IEEE and the receipient of the Engineering Medal for Research and Development of the Professional Engineers of Ontario (PEO).