To Update Network State or Not?

B. Fu, F. A. Kuipers, P. Van Mieghem

Faculty of Electrical Engineering, Mathematics and Computer Science,
Delft University of Technology,
P.O. Box 5031, 2600 GA, Delft, The Netherlands.
B.Fu,F.A.Kuipers,P.F.A.VanMieghem@tudelft.nl

Abstract— **A Link-State Update Policy (LSUP) has the task to distribute information regarding the network resources, and is therefore considered to be an integral part of future Quality of Service (QoS) routing protocols. The argument is that in order to guarantee QoS to applications, one must know the available resources. Unfortunately, the high dynamics in available resources complicates the development of an LSUP, which on its turn will result in high deployment costs for Internet Service Providers. In this paper we will re-examine whether the gain in network performance, which is expected under the deployment of LSUPs, will outweigh its investment and complexity costs. To capture the complete range of possible LSUPs, we take a pragmatic approach and confine ourselves to examining two extreme strategies: *routing with exact resource information* and *routing with no resource information*.**

Our study comprises of an analytical exercise and extensive simulations on various network topologies under a large range of network loads. Our objective is to determine where static information provides acceptable network performance and where dynamic LSUPs are indispensable.

I. Introduction

Most real-time applications require certain Quality of Service (QoS) guarantees (e.g., on capacity, loss, and delay) to achieve good performance. The study of QoS routing has mainly focused on algorithms that can find paths based on multiple constraints (cf. [1] for a survey). Those algorithms assume that the knowledge on the state of the network is provided by a QoS protocol. Unfortunately, the development of a QoS protocol has received much less attention, most probably due to the complexity of handling network dynamics.

In particular for QoS routing, a QoS routing protocol is expected to determine, update and distribute the set of dynamically changing link weights. The QoS routing protocol possesses a functionality to give each router a consistent view of the network and its resources by using a link-state update policy (LSUP). The resource information varies rapidly compared to the infrequent changes in network topology (such as the breakdown of a link or router). Based on the resource information, a QoS algorithm will compute a suitable path to accommodate the QoS requirements of a particular flow.

The complexity of QoS routing in comparison to "simple" best-effort routing may result in high costs. Internet Service Providers (ISPs) may not be willing to change their current non-QoS-aware systems if the performance gain of QoS-aware networking is not substantial enough compared to the additional cost. Beside the monetary cost involved in implementing QoS technology, cost also relates to computation and protocol overhead. The computation overhead is bounded by the worst-case complexity of the routing algorithm, while the protocol overhead is caused by the flooding and updating of resource information.

Most of the proposed link-state update policies (LSUPs) consider a trade-off between the protocol overhead and the accuracy of the resource information (see [2]). To reduce the protocol overhead, the number of updates needs to be limited, and consequently, not all resource changes are advertised. This leads to the problem of QoS routing with stale resource information.

Stale resource information may affect routing in the following ways:

- A feasible path cannot be found by the algorithm although one exists.
- A path is found by the routing algorithm but rejected during the set-up process, because not all links along the path can provide the required capacity.
- A non-optimal path is found while there exist other paths, which are more suitable.

It is difficult to indicate how much cost is acceptable in order to gain a certain improvement in network performance, as this may differ per ISP. In this paper, the profits of using link-state routing systems are estimated indirectly via the comparison of two extreme strategies: (A) routing with exact resource information, and (B) routing without resource information, i.e., without link-state updates (LSUs). Our work is motivated by the fact that there are no in-depth studies that quantify the performance gain that can be attained with accurate LSUPs.

The rest of this paper is constructed as follows. In Section II, we discuss related work. In Section III, we motivate our study and explain the two extreme strategies in which we are interested. Examples of performance evaluation are given in Section IV. Section V presents the simulation models. In Section VI, we show the performance in selected situations and discuss the simulation results. Finally, we present our conclusions in Section VII.

II. Related work

In this section we briefly discuss the work related to LSUPs and stale resource information.

A. LSUPs

For intradomain QoS routing, many LSUPs have been proposed.

Periodic LSUP: The periodic LSUP distributes the resource information through the network periodically. It is, for instance, employed by OSPF to update the topology information. The periodic LSUP is easy to implement and the update rate is fixed once the period is set. Hence, the periodic LSUP is not coupled to any traffic dynamics.

Trigger-based LSUPs: Due to the shortcomings of the periodic LSUP, the trigger-based LSUPs were proposed to better catch the dynamics in link state [2][3][4]. In the following we assume that the available capacity is our link-state metric, but the same principles apply to other metrics. Trigger-based LSUPs can be further classified into class-based and threshold-based LSUPs. Class-based LSUPs trigger updates if the available capacity crosses the pre-defined boundaries; while threshold-based LSUPs consider the relative difference between the available capacity known by the network and the actual available capacity known only by the immediate node.

To reduce the protocol overhead caused by LSU traffic, the update rate should be limited. Some complementary strategies can be added to limit the number of updates. The hold-down timer [2][5] limits the update rates by defining the smallest period between two consecutive updates on the same link. The moving average strategy [4] considers the average available capacity (over a certain window size) to trigger updates. The average value is used to follow the trend of a link-state metric and to avoid updates triggered by a short-lived metric change.

Combined LSUPs: Based on the LSUPs mentioned above, some works suggest policies combining two or more of them, e.g. [6].

B. *The impact of stale resource information*

Many works study the impact of stale resource information on QoS routing performance. The effects of stale information introduced by the LSUPs mentioned in Section II-A have been evaluated in [4][5][6][7], where the LSUP parameters are tuned to achieve different information granularity and thus different network performance. Ma and Steenkiste [8] have compared static routing (routing without LSUs) and other routing algorithms under periodic LSUP. Shaikh *et al.* [2] give an overview of how the combination of LSUP, routing algorithm, traffic pattern and network topology influences the network performance.

As inaccuracy in resource information is inevitable, routing algorithms were proposed that can tolerate imprecise resource information. Some works [9][10] suggest using intelligent routing algorithms with local resource information.

III. Two extreme strategies

A full-fledged QoS architecture requires, apart from topology updates, also resource availability updates. The first kind of updates are slowly varying in time, while the second type – the traffic related – updates are changing much faster. This article only focusses on the second kind of updates. The key question boils down to: "Is it worth to implement complicated update strategies for the second kind of changes in a network?" That question is undoubtedly not new, but a clear answer is still lacking. A quite important motivation to *not* update is that the Internet, where dynamic strategies of the second kind are absent, has featured a reasonably good overall functioning.

To simplify the setting, we confine to two extreme strategies A and B. Strategy A is the optimal update case that takes *all* information of the past and present into account to allocate a flow in the network at each flow request instant t. "Optimal" here refers to any property we would like to maximize or minimize in a network, such as, for example, throughput, rejection rate or blocking probability, or revenue. Each flow is allocated along a path for its entire life-time; we do not consider re-allocations of previous flows. Since the strategy A assumes the knowledge of any past or present information, any network property can be computed optimally in our "Gedanke" experiment. We thus ignore the practical issues that may prevent the strategy A to achieve optimality, such as e.g. the computation time, flooding time and the memory needed. In practice, strategy A cannot be implemented, because the flooding of information always takes a non-zero time such that not all routing topology databases can be guaranteed to be synchronized. Strategy B is completely static and does not update at all. Each flow is just allocated along a fixed, initially computed shortest-hop path from source to destination.

In most real networks, we cannot predict the future. At best, we may possess estimates of the likelihood of a future traffic demand. Hence, strategy A is only optimal in the time interval $[0, t]$, but not necessarily in $[t, T]$, where $T > t$, because future demands are not taken into account.

Our analysis thus belongs to the field of dynamic decision theory, where the key question is "Is my myopic decision rule optimal?". In most cases, update rules are designed to improve the performance in the heavy traffic regime. For, if the traffic is low, the resources are usually plentiful and no need for update rules is perceived. Contrary to common intuition, we show that, in the heavy traffic regime, there are network scenarios where the static strategy B outperforms the "optimal" strategy A.

IV. Performance in time

In routing with exact resource information (strategy A), the network elements need to update every change, and paths are computed based on this exact information. In routing without LSUs (strategy B), only information about the topology is used. Paths are not computed for each flow, but instead, a static routing table is used. If one or more links on the path cannot accommodate the flow, the flow is rejected.

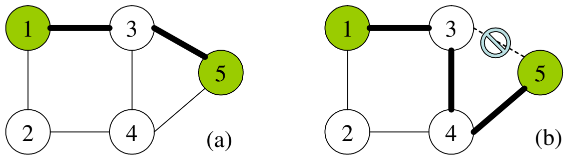

Fig. 1. (a) the routing without LSUs; (b)the routing with exact resources information

Consider Figure 1, where a flow is requested from source node 1 to destination node 5. Routing without LSUs uses the shortest path $1 \to 3 \to 5$ (Figure 1 (a)). We assume that link $3 \to 5$ does not have enough capacity to accommodate the new flow. Since the routing protocol without LSUs (strategy B) always uses the path $1 \to 3 \to 5$, the new flow is blocked. Strategy A, on the other hand, can find another path for the new flow (Figure 1 (b)). However, the new path in this example has a larger hopcount than the shortest path, which might be detrimental for future flows.

The effect on future flows is more pronounced under high network loads. Indeed, if an available path with h hops is found for a flow with capacity requirement c_r, the total network capacity consumed by this flow is $c_r h$. Ergo, the more hops a path has, the more network capacity will be consumed. The static strategy B chooses for each source-destination pair the path with minimum hopcount. In the first instance where the static routing in strategy B blocks a flow, strategy A has the potential to find an alternative, likely with longer hopcount. Thus, in the first few instances where strategy B blocks flows, strategy A clearly can outperform strategy B, but strategy A consumes in those cases, more than the minimum amount of capacity. If the traffic demand remains high, strategy A will find itself suffocated by those expensive flows accepted in the past, which may result in a higher blocking from then on compared to strategy B. In Figure 1 (b), future flows from 3 to 4 and from 4 to 5, can be affected by the deviation from the shortest path. Strategy A greedily allocates each subsequent flow, until no network capacity is available anymore. Strategy B allocates always a minimum amount of capacity, but spreads that allocation over time. Over a long time interval $[0, T]$, optimal flow scheduling assumes the knowledge of all future demands. Since the general problem of network flow scheduling in $[0, T]$ is a combinatorial optimization problem that is NP-complete, there do not exist simple greedy algorithms that are optimal. Hence, although strategy A seems optimal, because it uses all possible available information up to time t, the impact of the unknown future demands may result in an inferior performance compared to the static strategy B.

In the following example we will show that indeed static routing can outperform dynamic routing with exact information. Consider the complete graph K_N with N nodes and $L = \binom{N}{2}$ links. We consider the time-frame $[0, T]$ and assume that each allocated flow consumes the entire link capacity for a duration $d \geq T$. Strategy A perfectly routes paths with hopcount restricted to 2. Strategy B only allocates paths on the direct links. We denote the steady state link availability, as $p_A(i)$ and $p_B(i)$, which are a function of the number of flow requests i. The blocking probability for strategy B as function of i equals $P_B(i) = 1 - p_B(i)$, which is the probability that the direct link is already allocated. The corresponding blocking probability for strategy A is the probability that the direct link *and* all possible two hop paths between that source-destination pair are occupied. That probability is computed in [11] as

$$P_A(i) = (1 - p_A(i)) \left(1 - p_A^2(i)\right)^{N-2}$$

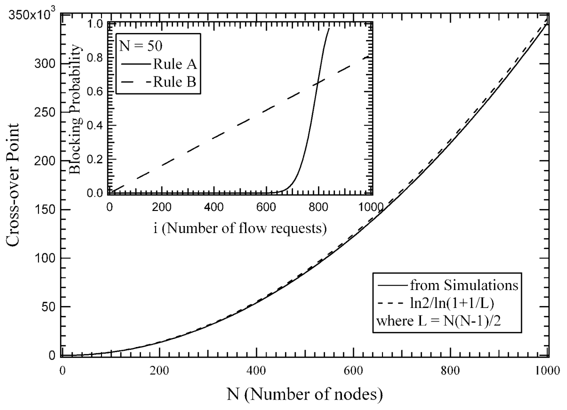

Fig. 2. The cross-over point as a function of N. The blocking probability for rules A and B as function of the number of flow requests i is shown in the insert.

The expected number of links for strategy B after $i + 1$ requests equals $Lp_B(i + 1) = Lp_B(i) - p_B(i)$. For strategy A we have $Lp_A(i + 1) = Lp_A(i) - p_A(i) - 2(1 - p_A(i)) \left(1 - (1 - p_A^2(i))^{N-2}\right) \approx Lp_A(i) - 2 + p_A(i)$. Solving these equations gives [12]:

$$p_A(i) \approx 2 - \left(1 + \frac{1}{L}\right)^i \text{ and } p_B(i) = 1 - \frac{i}{L}$$

For small i, $p_A \approx p_B$, which results in $P_A(i) < P_B(i)$. Indeed less flows are blocked by strategy A, because more alternative paths exist. However, by using two-hop paths, resources are consumed faster (i.e., $p_A(i)$ decreases faster with i than $p_B(i)$), which increases the chance of blocking for future flows. Hence, after the allocation of a large number of flows, the strategy B is expected to outperform strategy A. The cross-over point where $P_A(i) = P_B(i)$ is approximately $i_c \approx \frac{\ln 2}{\ln(1+1/L)}$ as illustrated in Figure 2.

V. Simulation scenarios

In this section, we describe our simulation scenarios. We have used a flow-level simulator[1]. The network and traffic parameters are introduced as well as the routing algorithm and signaling model. At the end, we explain how we process the data from our simulation results.

A. Network model

We mainly concentrate on the class of random graphs $G_p(N)$ with N the number of nodes, and p the probability that there exists a connection between a pair of nodes. However, we also simulate in a square lattice topology and an MCI topology, which are given in Figure 3. Each connection is symmetric, with the two directions treated as two links separately, and all the links have unit capacity.

As the capacity guarantee is the most likely constraint for a customer requiring QoS, we only consider the capacity

[1]DESINE (DElft SImulator of NEtworks), developed at Delft University of Technology.

requirement as our QoS metric when carrying out the simulations. Other interesting QoS metrics like delay and packet loss are often highly correlated to the available capacity (e.g., a lower available capacity is likely to result in higher delays and packet loss). Hence, we believe that the trends observed for available capacity will match to a certain extend with trends in delay and packet loss.

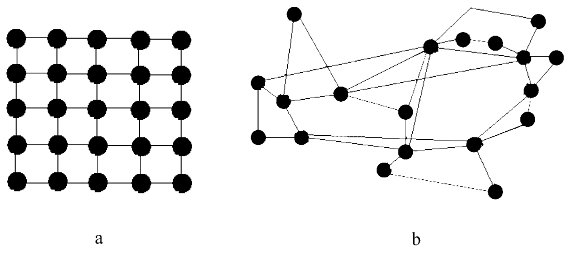

Fig. 3. a: the square lattice topology. b: the MCI topology

B. Traffic model

The arrival process of the incoming flows is modeled as a Poisson process with rate λ flows per unit time. The pairs of source and destination nodes are uniformly selected among the set of nodes. The service time of flows, i.e. the flow duration, is described by a random variable d. We denote by C_r ($0 \leq C_r \leq 1$) the capacity requirement of each flow, which is a certain percentage of the unit link capacity.

Following Shaikh *et al.* [2], the network load is defined as: $\rho_N = \lambda E[d]E[C_r]E[h]/L$, where $E[d]$ is the mean flow duration, $E[C_r]$ is the mean capacity requirement, $E[h]$ is the mean hopcount of the shortest paths between all pairs of source and destination nodes, and L is the number of links in the network.

C. Routing algorithm

For each new flow, the source node uses his own view on the resource information to compute the path based on the flow's QoS requirements.

We use the widest-shortest path (WSP) algorithm with pruning. As explained in [8], this algorithm gives high priority to limiting the hopcount, thus limiting the resource consumption. It is said to perform better than the shortest-widest path algorithm, which gives high priority to balancing the network load. The basic steps of the WSP algorithm are given below:

1) Based on the resource information of the source node, all the links $l_{i,j}$ (the link from node i to node j) for which the known available capacity $C_{l_{i,j}} < C_r$ are pruned.
2) Compute the minimum hop paths in the pruned graph.
3) If there is only one path found with the smallest hopcount, this path will be used to carry the flow; else if more than one shortest paths exist, we select the path P with the maximum available capacity ($C_P = \min_{l_{k,m} \in P}(C_{l_{k,m}})$).

The available capacity $C_{l_{i,j}}$ of link $l_{i,j}$ known by the source node, is of great importance in the process of selecting the path. Inaccuracy of $C_{l_{i,j}}$ may result in finding a non-optimal path. A flow fails during the routing process if no paths can be found by the routing algorithm.

D. Signaling model

Once a path is found by the routing algorithm for a new flow, from the source node on, the amount of resources needed will be reserved on each link along the path. If all the links along this path can accommodate the new flow, i.e., $C_{l_{i,j}} \geq C_r, \forall\ l_{i,j} \in P$, the path is set up. The resources required by this flow will be reserved during the service time until this flow terminates. As the information of the source node may be inaccurate, the path computed by the routing algorithm might be unfeasible. In this case, the resources that have already been reserved need to be released, and the flow is rejected during the setup process. We call this a setup failure.

Upon a failure during either the routing process or the set up process, we drop the flow without rerouting, because our aim is to see how stale information affects the network performance under the same path selection process.

E. Statistical model

The square lattice topology and the MCI topology are fixed. For each class of random graphs, we generate 1000 different connected graphs.

For each combination of graph, network load and extreme LSUP, 10 realizations are simulated. In each simulation, a set of 200,000 flows are offered to the network in a Poissonian way. The first 100,000 flows are used as warm-up flows whose routing results will not be taken into account when collecting the data. In this way, we model a steady-state behavior. Once the 10 iterations are finished, an average will be taken for all the data.

VI. Simulation results

We first present our simulation results after which we collectively discuss them. The blocking ratio (blocking ratio $BR_x = \frac{Number\ of\ failed\ flows\ with\ x}{Total\ number\ of\ flows}$) , is the comparison metric.

Two types of traffic were tested: one with the flow duration d set to 100 time units, and the capacity requirement of each flow C_r set to 5%, a constant percentage of the unit link capacity; the other with the flow duration d following a Weibull distribution with mean 100 time units to capture the long-tailed nature of connection durations, and C_r set be to a uniform distribution in the range [2.5%, 7.5%] of the unit link capacity. Smaller capacity requirements may be more realistic, but we want to know how badly a certain routing strategy behaves. Thus, larger capacity requirements, which are more sensitive to routing decisions, are chosen to better capture the performance of routing strategies.

For each graph, we simulate with different load ρ_N ($\rho_N < 1$), and the flows' arrival rate λ is calculated as $\lambda = L\rho_N/E[d]E[C_r]E[h]$.

MCI topology: Figure 4 shows the network performance of the two extreme strategies in the MCI topology, as a function of the network load under two traffic patterns. For both traffic patterns, the blocking ratios for both extreme strategies increase quickly as the network load increases, and the biggest difference happens somewhere near $\rho_N = 0.5$. As expected, routing without LSUs performs worse than with exact resource information, but the difference is small under low and very high network loads.

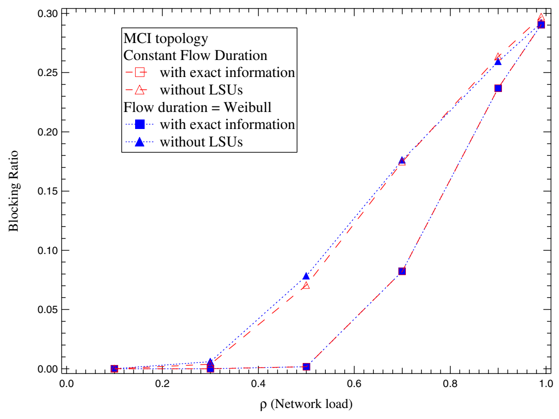

Fig. 4. The blocking ratio on MCI under different types of flow duration, different capacity requirements, and different network loads.

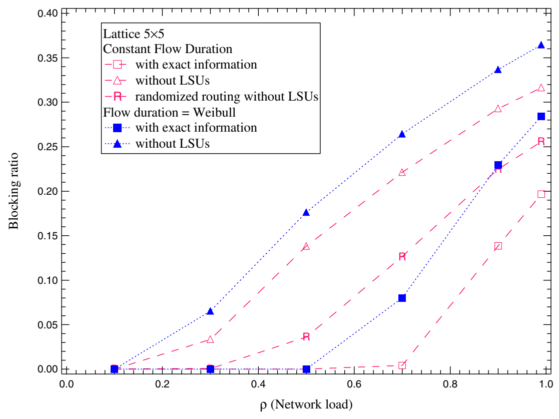

Fig. 5. The blocking ratio on lattice under different types of flow duration, different capacity requirements, and different network loads.

Square lattice topology: For the lattice, the network performance under two traffic patterns is shown in Figure 5. Again, under low network loads, routing without LSUs has a similar performance as routing with exact resource information. However, its performance decreases quickly as the network load increases. The reason is that, in a topology like lattice, between each pair of nodes, there may exist multiple shortest paths, and using only one out of this set could result in earlier blocking. Randomized routing [13] was introduced to balance the load. For each pair of source and destination nodes, a set of shortest paths are computed, once a flow comes, the source selects one path randomly from this set. We applied the randomized routing to the traffic pattern with constant flow duration. Figure 5 shows that randomized routing improves the network performance considerably for the no LSUs case, but it still gives higher blocking than routing with exact resource information. The routing with exact resource information starts giving increased blocking ratios at a heavier network load.

Random graphs: We have performed simulations in the class of random graphs $G_p(N)$, for different N and different p. The critical link density p_c in the random graphs equals $p_c \approx \frac{\ln N}{N}$. Choosing $p < p_c$ results, with high probability, in disconnected graphs. We consider only $G_p(N)$ with $p > p_c$.

Figure 6 shows the network behavior in $G_{0.1}(50)$ under different network loads for the two traffic patterns.

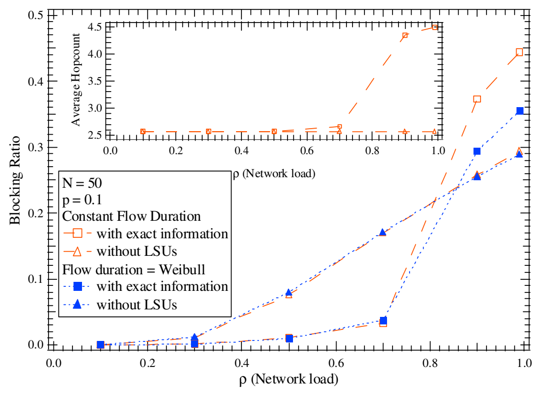

Fig. 6. The blocking ratios on the class of random graphs under different types of flow duration, different capacity requirements, and different network loads with $N = 50$ and $p = 0.1$. The average hopcount for the constant flow duration is put as insert.

Figure 7 shows the network behavior in the class of random graphs $G_p(N)$ with different N, and different p under network load $\rho = 0.99$ for the type of traffic with constant flow duration.

As can be seen from the Figures, when the network load is low, the difference between strategy A and strategy B is small, since it is unlikely that the shortest paths will get congested. The difference increases with ρ_N, provided the network load is not very high. For $\rho_N \to 1$, shortest paths become rapidly congested and, under strategy A, longer-hop paths are used. Allocating long-hop paths may block future requests from being allocated, which explains the steep increase in blocking probability for the class of random graph. This explanation is substantiated by the insert in Figure 6, which gives the expected hopcount between all pairs of nodes as a function of network load.

In the case of strategy B, no matter how many possible paths there exist for a certain pair of source and destination nodes, only the one with the smallest hopcount is selected without considering the resource information. Thus, the average hopcount does not change as the network load changes.

In the case of strategy A, under small loads, the expected hopcount equals that of the shortest paths (used with strategy B). However, when ρ_N reaches a certain value, the average hopcount $E[h']$ for routing with strategy A increases quickly.

The actual load $\rho^{'}$ which each link experiences, can be given as $\rho^{'} = \lambda E[d]E[C_r]E[h^{'}]/L$. $\rho^{'}$ goes to 1 before ρ_N does. This is the reason why the curves for routing with strategy A cross those for routing with strategy B in Figure 6.

Figure 7 shows that under the traffic model with the constant flow duration, BR_A increases as N increases. As the network grows, the routing with strategy A gives worse performance with more computation and communication overhead. This is neither economical nor scalable.

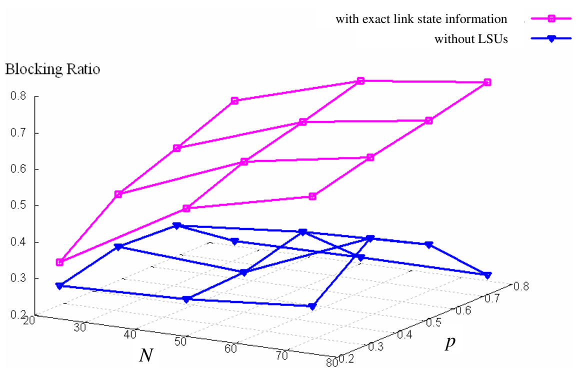

Fig. 7. The blocking ratios in the class of random graphs $G_p(N)$ with different N, and different p under network load $\rho = 0.99$, with the flow arrival process following the Poisson process; flow duration d set to 100 time units; capacity requirement C_r set to be 5% of the unit link capacity.

For BR_B we observe for each N, a bell-shaped curve, with a peak depending on the number of nodes N. For small p, the graph will approximate a tree-like structure and the shortest paths between all pairs of nodes are likely to span nearly all links. In that case the load is well balanced. Similarly for $p \rightarrow 1$, the shortest paths are likely to be the direct links and by using these links the network does not waste many resources, which would affect the allocation of other flows. In between these densities, the shortest paths may only span (i.e., use) a small portion of the network, causing an increased chance of blocking (which might be reduced via randomized routing as shown for the lattice networks).

We carried out similar simulations under the traffic model with Weibull distributed flow duration, and obtained similar results as under the traffic model with constant flow duration.

VII. Conclusion

In this paper, we have presented analytical and simulation results for two extreme link state update policies (LSUPs), namely *routing with exact resource information* and *routing with static information*. Different traffic and network scenarios have been evaluated. Our results show that the extremes react differently to different network models. In each network model, the difference in performance varies with the network load, and stays small under low network loads.

For the MCI topology, the performance difference remained fairly small.

For the square lattice topology, routing with exact resource information gives small blocking ratio even under heavy network load, while routing with static information degrades very fast as the network load increases. However, by introducing randomized routing without LSUs, the network performance could be improved substantially.

The class of random graphs $G_p(N)$ performs differently as p or N varies. When the amount of traffic running in the network is relatively small compared to the high capacity provided by the core network, routing without LSUs is expected to give satisfactory performance. However, under very high network loads, the intelligence of knowing exactly the resource information backfires, because it leads to longer-hop paths, which results in a more congested network, leading to high blocking ratios.

The most striking observation in our work is that under high network load the "no updates" routing performs close to or even better than the "exact" routing. If we would have counted the amount of traffic overhead induced by the LSUs, this observation would even be more pronounced. We believe that with the fast increase in access technologies (e.g., FttH), the ISP and core networks will have to operate under high load regimes. They even have an economic incentive to utilize their network as much as possible. Our results argue that in this case an investment in LSUPs will not be cost efficient and that a simple shortest-hop framework is likely to suffice.

Acknowledgment

The work is funded by the STW foundation as part of the Network Dynamics and QoS project (DTC.6421).

References

[1] F. Kuipers, T. Korkmaz, M. Krunz, and P. Van Mieghem, "An overview of constraint-based path selection algorithms for Qos routing," *IEEE Communication Magazine*, vol. 40, no. 12, pp. 50–55, December 2002.
[2] A. Shaikh, J. Rexford, and K. Shin, "Evaluating the impact of stale link state on Quality-of-service routing," *IEEE/ACM Transactions on Networking*, vol. vol. 9, April 2001.
[3] A. Ariza, E. Casilari, and F. Sandoval, "Strategies for updating link states in QoS routers," *Electronic Letters*, vol. vol. 36, no. No. 20, 2000.
[4] B. Lekovic and P. Van Mieghem, "Link state update policies for Quality of service routing," *Eighth IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT2001)*, 2001.
[5] G. Apostolopoulos, R. Guerin, and S. Kamat, "Quality of service based routing: A performance perspective," *ACM SIGCOMM*, 1998.
[6] M. Kabatepe and M. G. Hluchyj, "On the effectiveness of topology update mechanisms for ATM networks," *ICC'98*, 1998.
[7] Q. Ma and P. Steenkiste, "Qos routing for traffic with performance guarantees," *Proc. IFIP Int. Workshop Quality of Service*, 1997.
[8] ——, "On path selection for traffic with bandwidth guarantees," *ICNP'97*, 1997.
[9] S. Nelakuditi, Z. Zhang, R. P. Tsang, and D. H. Du, "Adaptive proportional routing: a localized QoS routing approach," *IEEE/ACM Transactions on Networking*, 2002.
[10] X. Yuan and A. Saifee, "Path selection methods for localized Quality of service routing," *IC3N'01*, October 2001.
[11] P. Van Mieghem, "Performance analysis of communications networks and systems," *Cambridge University Press*, 2006.
[12] T. Kleiberg and P. Van Mieghem, "A queueing approach to model network flow dynamics," *submitted to Performance Evaluation*, 2006.
[13] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving Qos routing performance under inaccurate link state information," *Proceedings of the 16th International Teletraffic Congress*, 1999.