Quo Vadis - A Framework for Intelligent Routing in Large Communication Networks

Armin R. Mikler, Johnny S.K. Wong, Vasant Honavar
Department of Computer Science
Iowa State University
Ames, Iowa 50011, USA
e-mail: mikler|wong|honavar@cs.iastate.edu

Abstract

This paper presents Quo Vadis, an evolving framework for intelligent traffic management in very large communication networks. Quo Vadis is designed to exploit topological properties of large networks as well as their spatio-temporal dynamics to optimize multiple performance criteria through cooperation among nodes in the network. It employs a distributed representation of network state information using local load measurements supplemented by a less precise global summary. Routing decisions in Quo Vadis are based on parameterized heuristics designed to optimize various performance metrics in an anticipatory or pro-active as well as compensatory or reactive mode and to minimize the overhead associated with traffic management. The results of simulation experiments within a grid network clearly demonstrate the ability of Quo Vadis to avoid congestion and minimize message delay under a variety of network load conditions.

1 Introduction

Recent advances in computers and communications, along with the ever-increasing need for rapid and reliable information transfer over very long distances has led to unprecedented expansion of such communication infrastructures over the past several years. Such networks contain hundreds if not thousands of interconnected nodes [Snyder, 1989]. Traffic management mechanisms must be able to support a cost-effective, responsive, flexible, robust, customer-oriented high speed communication environment while minimizing the overhead associated with management functions. Conventional traffic management mechanisms for routing and congestion control algorithms entail tremendous resource overhead in storage and update of network state information. This will almost certainly result in increased cost and reduced performance with growth in the size of the networks. Thus, a careful reevaluation of conventional traffic management schemes with respect to their efficiency and effectiveness within large, constantly expanding communication environments is needed.

Message routing and congestion control are typical traffic management tasks. These functions are generally thought of being hosted by the layers 2-4 of the Open Systems Interconnection (OSI) protocol stack. The primary objective of routing mechanisms is to propagate messages across the network towards their destinations while simultaneously trying to optimize one or more performance criteria such as path length or message delay.

The primary objective of congestion control is to prevent uncontrolled influx of messages into a set of network nodes. Without congestion control, network nodes may experience over-utilization which in turn may lead to increased loss of messages due to limited availability of buffers [Gerla and Kleinrock, 1980; Jain, 1990]. As a consequence, the quality of service offered by the network will decrease. Furthermore, the loss of messages generally requires their retransmission which in turn reduces the overall network utilization (throughput). Even if network nodes have infinite buffer space available, thereby eliminating message loss, congestion tends to increase the overall delay encountered by messages.

Routing and congestion control are strongly interrelated as routing decisions determine the area through which a message is sent while moving towards its destination. Consequently, routing algorithms must be carefully designed to adapt rapidly to load changes in the network. In addition, routing techniques must minimize the associated resource overhead and should scale well without compromising performance as networks continue to grow in size. Resource overhead to be minimized can be divided into:

- bandwidth requirements;
- storage requirements; and

- computational complexity.

Additional desirable properties of routing and congestion control mechanisms for such communication environments include the ability to:

- route messages anticipating the consequences of routing decisions on the network dynamics (e.g., to pro-actively avoid congestion if possible),
- smoothly trade-off of some subsets of performance measures against others, and
- gracefully adapt without manual intervention to (predictable as well as unpredictable) changes in network dynamics without compromising performance.

This paper describes Quo Vadis, a framework for intelligent traffic management in very large, high-speed communication networks. Quo Vadis draws upon insights from hitherto disparate areas: communication networks, artificial intelligence, machine learning, and optimization in order to strike a balance among various performance criteria. The primary objective of Quo Vadis is to achieve reasonable network performance while minimizing the overhead associated with network traffic management.

2 Routing in Large Networks

Conventional approaches to routing [Cegrell, 1975; McQuillan, 1980; Schwartz and Stern, 1980] rely on the timely availability of large amounts of accurate network state information (for example, in the form of distance and routing tables) at each of the switching nodes so that they can make routing decisions designed to optimize (to the extent possible) the desired measures of overall network performance such as delay and throughput [Tanenbaum, 1988; Bertsekas and Gallager, 1992; Wong and Mikler, 1993]. In practice, frequent transmission of such network state information consumes valuable resources such as memory and bandwidth which could otherwise be used for message traffic. Most attempts to reduce the overhead involved in the update of network state information at each switching node lead to a degradation in the accuracy of the information available. As communication networks grow larger, the overhead associated with conventional routing mechanisms becomes prohibitive.

2.1 Basic Routing Algorithms

Most conventional routing protocols, such as the *routing information protocol* (RIP) and *open shortest path first* (OSPF) have their origin in either one of two basic strategies [Perlman, 1992], which are

1. distance vector routing, and

2. link state routing

Distance vector routing is often referred to as the *old* `ARPANET` routing algorithm. It is essentially a distributed version of the Bellman-Ford shortest path algorithm [Bertsekas and Gallager, 1992]. Distance vectors are generally stored in distance tables at each network node. Distance tables thus contain distance estimates to every destination in the network via each neighbor node n_k. The distance vector to a particular destination node is computed by adding the *distances* between nodes along the paths to the destination. A routing table that contains all possible destination nodes is constructed by selecting from the distance tables those routing vectors with minimum distance estimates. Upon receiving a message that is to be routed towards its destination, a network node initiates a table look-up resulting in a node to which the message is to be sent next.

Link state routing is based on the assembly of complete topological information. It is frequently referred to as the *new* `ARPANET` routing algorithm as it has replaced the earlier distance vector approach on the `ARPANET`. Each node measures the distance from itself to all its neighbor nodes and propagates a *link state packet* (LSP) to all other nodes in the network. This process is generally referred to as *flooding.* After a node receives a LSP from every node in the network it can construct a spanning tree that is rooted at the node itself. The construction of the spanning tree is based on Dijkstra's shortest path first (SPF) algorithm [McQuillan, 1980]. Network nodes must be able to assess the validity of each LSP received to avoid outdated information from corrupting the spanning tree. This is accomplished by employing costly timer, sequence number and aging schemes [Perlman, 1992].

Both link state and distance vector routing rely upon *complete* network state information. That is, each node needs to compile global knowledge of the entire network. While in distance vector routing this knowledge is represented by the set of all distance tables, link state routing relies on information about the state of every link in the network. Clearly, the amount of network state information used by both these routing strategies increases with the size of the network.

The imprecision or uncertainty associated with network state information grows also with the size of the network. This is a direct consequence of the temporal dynamics of the network which causes the network state to change even as the state information is being computed and propagated. The amount of storage required to maintain network state information at each switching node also grows with the size of the network. So does the network bandwidth required to maintain this information up-to-date.

2.2 Approaches to reducing overhead

The immense cost associated with the maintenance and frequent update of network state information prompted the exploration of a number of strategies designed to minimize the resource (e.g., storage and bandwidth) requirements of traffic management in large communication networks. Most of these strategies involve structuring of the network at the logical level, the physical level, or both. Some examples of structuring at the logical level include hierarchical routing [Kleinrock and Kamoun, 1977; Perlman, 1985] and landmark routing [Tsuchiya, 1988]. Large networks are organized into a hierarchy of logical units. Switching nodes maintain complete state information only for the nodes within their own logical unit supplemented by a summary of the network state information for other logical units. Collections of subnetworks connected via a backbone offer an example of structuring of the network at the physical level. While both hierarchical routing and landmark routing do reduce the amount of network state information stored at and transmitted between nodes, they suffer from a number of drawbacks. For instance, it has been shown that the manner in which reduction in network state information is realized in hierarchical and landmark routing results in an increased average path length between source and destination nodes. The existence of an optimal structuring of the network so as to limit the size of routing tables has been shown in [Kleinrock and Kamoun, 1977] and [Tsuchiya, 1988]. However, frequent restructuring of hierarchies and landmarks so as to maintain an optimal structure is required in order to provide for acceptable performance in an expanding communication environment. This clearly represents another drawback associated with such techniques.

Hierarchical routing and landmark routing are approaches to reduce the size of routing and distance tables in the underlying distance vector routing algorithm. No such approach is currently available for link state routing as routing tables are computed using a minimum spanning tree that can only be constructed from complete topological information. Instead, approaches such as SPF routing with emergency exits (SPF-EE) [Wang and Crowcroft, 1990] are designed to reduce the frequency of link state updates and thus the frequency of recalculating the spanning tree by reducing the degree of oscillation commonly experienced by link state routing.

In a network with n nodes and k-connectivity, the space required to store network state information is $O(k \times n)$ for both, distance vector and link state routing. While there are $k \times n$ links to be considered in the construction of a spanning tree, distance vector routing must construct k distance tables each with n entries. If the network is structured into a r-level hierarchy this requirement can be somewhat reduced [Perlman, 1992].

However, the space requirement of a routing strategy is not the only issue to be considered. Maintaining up-to-date knowledge about the network state requires frequent propagation of distance and delay estimates. Thus, all of the above routing mechanisms consume bandwidth proportional to their storage requirement. The precision of information that is ultimately used to construct routing tables clearly depends on the dynamics of the network as well as the update frequency. Even if the time interval τ between updates is small, a finite amount of time is needed to propagate network state information (or its impact) to every node. Consequently, network state information collected by network nodes almost never represents the state of the network at a time t when a routing decision is made. Some degree of uncertainty is therefore inevitable.

3 Quo Vadis

In our view, any intelligent traffic management mechanism must include:

- An effective knowledge representation (KR) mechanism capable of providing sufficiently precise information about the state of the network;
- An efficient knowledge acquisition (KA) technique, that minimizes the overhead that is associated with acquiring network state information.
- Adaptive decision making methods that are designed to optimize the network performance.

The approach adopted by Quo Vadis for traffic management (and routing in particular) in large communication networks is motivated by the following observations:

1. The quality of routing decisions (as measured by suitable metrics such as average delay, average path length, etc.), is a function of the imprecision or uncertainty associated with the network state information upon which such decisions are based (assuming a decision function that makes optimal use of the available information). The imprecision or uncertainty of network state information is a function of (among other things) network dynamics, frequency of state updates, network delay for control messages, etc. In practice, all routing decisions in a large communication network are based on imprecise, uncertain knowledge of the current network state.

2. As noted in Snyder's proposal for the so-called *traveller architecture* [Snyder, 1989], the significance attached to the state (e.g., load) of a node to routing decisions made by another node in the network should be an inverse function of the distance between the two nodes. It follows that switching nodes in large communication networks should be able to make routing decisions based on the network state in their local neighborhood with little overall degradation in the quality of routes. The intuition behind this observation becomes clear if one considers a traveller faced with the task of choosing a route from a current location to a final destination. Such decisions are usually based on the conditions (e.g., traffic density) in the immediate vicinity of his current location. At each step, he is likely to pick a general direction that takes him closer to his destination via a neighboring location that appears to be the best (as measured by the traffic density). A precise knowledge

of the current traffic conditions at locations that are sufficiently far from the traveller's current location is of little use to him because the conditions there almost certainly would have changed by the time he gets close to them.

3. The number of routes of comparable length between a source node N with coordinates (X_N, Y_N) and a destination node D with coordinates (X_D, Y_D) is a non-decreasing function of the distance between the two nodes. For example, in a regular square grid network, it can be easily shown that the number of possible shortest routes $\mathcal{P}$ between nodes N and D is
$$\mathcal{P} = \binom{\mathcal{D}_x + \mathcal{D}_y}{\mathcal{D}_x} \tag{1}$$
where
$$\mathcal{D}_x = \mid X_N - X_D \mid \quad \text{and} \quad \mathcal{D}_y = \mid Y_N - Y_D \mid$$
It follows that the likelihood of finding alternative paths of comparable length is a non-decreasing function of the distance to the destination. Quo Vadis exploits this fact through its use of a carefully designed knowledge representation mechanism that maintains at all time, at each node, a locally computed *view* of the network state. This view includes precise information about the state of the node (e.g., its load) supplemented by a less precise (spatially and temporally averaged) summary of the state of the entire network as viewed from that node. Thus, each node needs to communicate its state and its view only to a small set of nodes in its immediate neighborhood. Routing decisions made by each node in Quo Vadis are based on the network state as captured by the views of the nodes in its immediate vicinity, and the destination of the message to be routed.

4. The utilization ρ of network nodes is generally determined by the ratio λ/μ where λ represents the arrival rate to that node and μ designates the rate at which the node can service messages. Hence, high utilization may occur due to a reduced service rate (possibly caused by node failures), or an increased arrival of messages. An increase of a node's arrival rate can have essentially two causes:

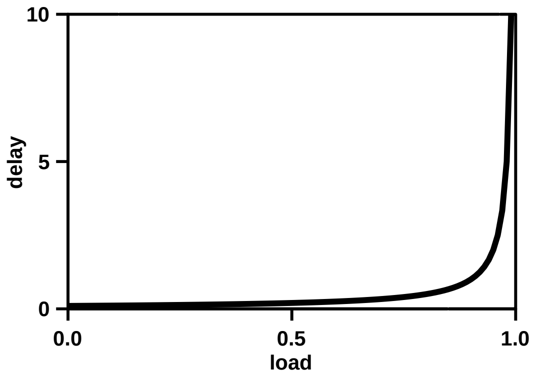

Figure 1: Delay vs. Utilization

(a) Many network nodes inject messages into the network destined to the same node (or network area).

(b) Routing decisions in neighbor nodes select the same node for a large number of messages. That is, a node is selected as *best neighbor* as determined by the routing metric used.

Assuming network nodes to be modeled as M/M/1 queues [Jain, 1991; Robertazzi, 1990], the message delay in each node i, among other things, depends on its utilization ρ_i. The expected delay D_i is given by

$$D_i = \frac{1/\mu_i}{1 - \rho_i} \tag{2}$$

D_i grows exponentially as ρ_i increases (see Figure 1). Clearly there exists a tradeoff between utilization and message delay, both of which are important performance measures.

In a uniformly utilized network, the best performance along a particular route can be obtained when the number of intermediate nodes is kept minimal. However, this is not necessarily true for non-uniform utilization as we will show below. This relationship between utilization and delay can be exploited in the design of routing algorithms.

Quo Vadis attempts to do precisely this through its use of parameterized heuristic knowledge representation, knowledge acquisition, and decision functions.

3.1 A Prototype Design of Quo Vadis

The current design of Quo Vadis [Mikler, Honavar, & Wong, 1992; Mikler, Wong, & Honavar, 1993] consists of two closely coupled modules:

- The knowledge representation module which is primarily responsible for the maintenance and update of network state information as viewed from each node.
- The decision module which implements routing and control algorithms.

Both these modules instantiate a family of parameterized heuristics that follow from the design philosophy of Quo Vadis. Future extensions to this design might include additional modules for adaptation of parameters to particular network dynamics and for learning appropriate classes of routing and congestion control strategies. A detailed description of the design and operation of knowledge representation and routing decision modules in Quo Vadis follows.

3.1.1 Knowledge Representation in Quo Vadis

As noted earlier, the knowledge representation mechanism in Quo Vadis is designed to maintain at all time, at each node, a locally computed view that includes precise information about the node itself as well as nodes in the immediate vicinity, supplemented by a spatially and temporally averaged summary of the state of the network as viewed from that node. This section explains exactly what constitutes such a view and how it is computed by each node based entirely on the information communicated to it by a small set of nodes in its immediate neighborhood.

Since the network nodes in Quo Vadis contain no representation of of directional information of the sort that is implicitly available through routing tables for guiding messages through the network, an alternative scheme is needed for addressing nodes and for computing their positions relative to each other. This is accomplished by superimposing a 2-dimensional grid on

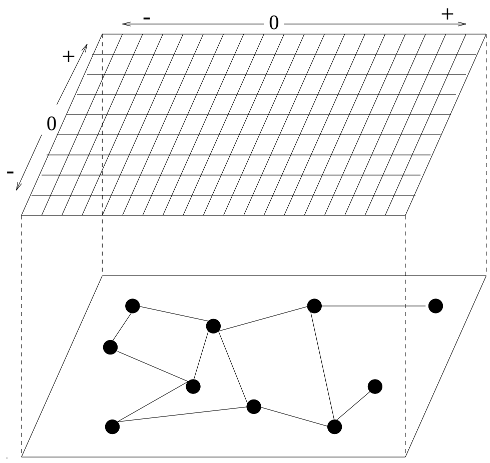

Figure 2: A superimposed coordinate system

the plane containing the network and identifying each node by its coordinates relative to the grid (see Figure 2). Thus, each node n_i can be addressed by its respective coordinates (x_i, y_i). Note that this does not restrict the allowable network topology to a regular grid. The coordinate system can be superimposed on any network topology. Furthermore, multiple nodes belonging to a subnetwork can share the same grid coordinate with routing within a subnetwork handled using any of the conventional routing strategies.

In some types of networks, particularly Metropolitan Area Networks (MAN), with Manhattan Street or HR^4 topologies, the location of network nodes and corresponding links coincide with the coordinates of a superimposed coordinate system [Wong and Kang 1990]. These networks can use routing algorithms that do not require explicit topological information. In more general topologies, the coordinates of a particular destination node must be retrieved as part of address/name resolution via some type of name-server. In addition, more complex network topologies may require nodes to maintain additional topological information.

While knowledge about the connectivity of nodes is implicit in networks with regular grid topology, more complex network topologies may require nodes to maintain information about the reachability of network nodes. Note that this knowledge is implicit in conventional routing tables (and link state representations). The resource overhead associated with providing information about the set of nodes that can be reached from each individual network node clearly depends on the network topology as well as factors such as frequency of changes in network topology (e.g., due to node and link failures or node and link additions as a result of network expansion or restructuring). For instance, if the network has a high degree of path redundancy, i.e., there exist multiple disjoint routes between any two nodes in the network, most of the network nodes will be reachable. Hence, a representation which only involves nodes that cannot be reached (i.e., the exception) will entail the least resource overhead. In the worst case, complete topological information must be maintained at each node. However, it is generally assumed that the topology of a network remains relatively static (with the exception of node/link failure and network growth). Therefore, reachability information can be assembled at startup time and only rarely updated (in response to changes in network topology). Knowledge representation mechanisms for this purpose and their implications on constraints on network topologies are currently under investigation.

The basic framework is based on the fact that each node n_i maintains a view $V_i(t)$ of the network from its vantage point at time t. This view can be decomposed into four components, one for each of the four directions - north, south, east, and west. Thus we have: $V_i(t) = [V_i^N(t), V_i^S(t), V_i^E(t), V_i^W(t)]$. Each component V_i^d : $(d \in \{N, S, E, W\})$ of the view $V_i(t)$ is computed using the corresponding view components $V_k^d(t-\tau)$ (where τ is the interval between view updates) together with local measurements $\rho_k(t)$ (see below) communicated by each of its neighbors n_k (suitably weighted by a normalized directional gain $g_{i,k}^d$ – see below). This ensures that the contribution of the information provided by the node n_k to the views computed at the node n_i is inversely proportional to the euclidean distance $D_{i,k}$ between the nodes n_i and n_k. Also note that the contribution of the node n_k to the view component V_i^d is directly proportional to its relative orientation as viewed from n_i with respect to the direction $d \in \{N, S, E, W\}$. This gain is normalized over the set of all neighbor nodes $H_i = \{n_k \mid n_k \text{ is a neighbor of } n_i\}$. (Note that this definition of directional gain is only one of the alternatives with qualitatively similar properties. Also, different definitions of neighborhood are possible).

Assume that the x and y coordinates increase as one travels further east and north respectively. Let (x_i, y_i) and (x_k, y_k) be the coordinates of nodes n_i and n_k respectively, and the euclidean distance between n_i and n_k be $D_{i,k}$. The directional gain to the south at node n_i for node n_k is given by:

$$\mathcal{G}^S_{i,k} = \begin{cases} 1 + \frac{y_i - y_k}{D_{i,k}} & \text{if } y_i \geq y_k \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The directional gains $\mathcal{G}^N_{i,k}$, $\mathcal{G}^E_{i,k}$, and $\mathcal{G}^W_{i,k}$ for the north, east, and west component of $V_i(t)$ are given by similar formulae. The normalization factor $\mathcal{G}^d_i$ for direction d for gains $\mathcal{G}^d_{i,k}$ computed at node n_i is given by:

$$\mathcal{G}^d_i = \sum_{n_k \in H_i} \mathcal{G}^d_{i,k} \tag{4}$$

The corresponding normalized directional gains are given by:

$$g^d_{i,k} = \frac{\mathcal{G}^d_{i,k}}{\mathcal{G}^d_i} \tag{5}$$

Now the view component $V^d_i(t)$ at node n_i at time t is given by:

$$V^d_i(t) = \sum_{n_k \in H_i} g^d_{i,k}(\alpha * \rho_k(t) + (1-\alpha) * V^d_k(t-\tau)); \quad 0 < \alpha \leq 1 \tag{6}$$

where τ is the time elapsed since the previous view update at the node n_i. (It is possible to make the update frequency a function of the local network dynamics. Such an approach is currently under study and will be discussed in a forthcoming paper). The parameter α determines the degree to which the effects of an event (i.e., load change) can impact routing decisions at other network nodes.

The local measurement $\rho_k(t)$ of node n_k has a number of natural interpretations. For example, if each network node is modeled as an M/M/1 queue, $\rho_k(t)$ may correspond to the utilization or the ratio of the arrival rate $\lambda_k(t)$ to the service rate $\mu_k(t)$ at time t. Note that there is nothing in the design of Quo Vadis that forces it to use an M/M/1 queue to model each node. A variety of more sophisticated queueing models can be used if necessary. The relative importance attached to the local measurements as opposed to the (spatially and temporally averaged) global view of the network as seen from a node is governed by the parameter α. It is a candidate for adaptation to cope with changes in network dynamics. So is the frequency of update of views maintained by nodes in the network (controlled by τ). Note that each node n_i computes its own view V_i only to disseminate it among its neighbors so as to enable them to update their knowledge of the network state. This knowledge is maintained at each node n_i in a knowledge base $S_i(t) = \{(\rho_k(t), V_k(t)) \mid n_k \in H_i\}$. As explained below, the routing decisions at each node n_i are based on its current knowledge base $S_i(t)$. The quality of routing decisions in Quo Vadis would depend on how well the knowledge base reflects the actual state of the network. In addition to choosing an appropriate representation of the network state, an efficient update mechanism must be deployed.

In order to maintain a precise view of the network, each node must communicate current network state information with each of its neighbors. Such updates can be time driven, event driven, or a combination of both. In the time driven approach, every τ time units a node propagates its network state information to every neighbor. This results in a periodic refreshing of the knowledge base, whether or not the state of the network has changed. This would require the setting of a suitable value of the parameter τ as a function of the dynamics of the network. The event driven approach forces a node to propagate its current view of the network state whenever a change of a certain magnitude TH has occurred during an interval of τ time units. Event driven strategies can reduce the overhead as an update is only initiated in response to significant changes in network state (e.g., those caused by changes in network load). A disadvantage of the event driven approach is that multiple changes each of which individually may be too small to trigger an update, may, due to their cumulative effect, lead to large degree of imprecision in network state representation. A hybrid approach to updating network state information has been used in the *new ARPANET* [McQuillan et al 1980]. This strategy guarantees the update of neighbor nodes every T seconds, however, an update is initiated earlier if the change in network state exceeds TH. The threshold TH is reduced in every measurement interval τ until $TH = 0$, thereby forcing a node to initiate the propagation of network state information.

Any of the above strategies may be deployed any in Quo Vadis. Of particular interest are mechanisms that adapt parameters such as α and τ in response to variations in network dynamics and/or changes in performance demands. It is only in the interest of simplicity of notation that such parameters have been treated as though they were constants in the equations above. Thus it is possible to let them take on different values at different nodes in the network and change their values as a function of spatio-temporal variations in traffic patterns and performance requirements. It is also worth emphasizing that the particular equations for view computation given above represent only one of many possibilities given the overall design philosophy of Quo Vadis.

The size of the knowledge base $S_i(t)$ at node n_i depends solely on the number of neighbors in its neighborhood H_i and is independent of the size of the network. Thus if M is the total number of nodes in the network and h the average connectivity (i.e., the average cardinality of H_i), then the storage required at each node in Quo Vadis is $O(h)$. This constitutes a significant reduction in storage and processing overhead (especially in very large networks where $M \gg h$) over conventional routing mechanisms (e.g., those that use global routing tables) which require $O(M)$ storage at each node.

3.1.2 Routing and Control in Quo Vadis

As pointed out earlier, each node n_i in Quo Vadis, when it receives (or generates) a message that needs to be sent to a different destination, it makes a routing decision based on the destination of the message and its current knowledge base S_i. This section describes in detail the routing mechanism used in a prototype implementation of Quo Vadis. Consider a message that is on its way from a source n_s to a destination n_d through a node n_i. Now n_i is faced with the task of routing the message along a path that would take it to its destination so as to optimize some desired performance criteria (e.g., average path length, average delay, or other suitable routing metrics). The node n_i does this by selecting one of the nodes in its neighborhood H_i that appears to best serve this objective. Choosing the best neighbor is based on the use of an evaluation function (in much the same spirit as the heuristic evaluation functions used in state space search in artificial intelligence problems [Pearl, 1984]). The node n_i computes the utility U_k of each node $n_k \in H_i$ and chooses the one that has the largest utility (it is assumed that during this computation, the view and load values do not change). In the prototype implementation of Quo Vadis, U_k is a function of two separate components:

1. the load liability L_k which estimates the load likely to be encountered by the message on its way to its destination n_d if it were to be routed through n_k; and

2. the path liability P_k that assigns a value to each neighbor n_k so that neighbors that are closer to the destination of the message being routed reflect lower values of P_k.

The overall utility U_k of the node n_k is given by:

$$U_k = -(\beta * P_k + (1-\beta) * L_k); \quad 0 \leq \beta \leq 1 \tag{7}$$

where β determines the emphasis placed on finding the shortest path to the destination relative to the desire of avoiding heavily loaded paths. Given this general framework for computing the utility of nodes, several different choices exist for the exact form of the expressions used to compute L_k and P_k. The particular forms used in the prototype implementation of Quo Vadis are explained below.

The load liability of node n_k is given by:

$$L_k = \gamma * \rho_k(t) + (1-\gamma) * \upsilon_k(t); \quad 0 \leq \gamma \leq 1 \tag{8}$$

where $\upsilon_k(t)$ is the sum of the projections of the appropriate components of the view V_k of the neighbor node n_k onto the vector connecting n_k to the destination node n_d.

Depending on n_d's location relative to n_k, $\upsilon_k(t)$ is composed of two components, namely an east-west component $\mathcal{C}_{EW}$ and a north-south component $\mathcal{C}_{NS}$. Let (x_k, y_k) and (x_d, y_d) be the coordinates of node n_k and the destination node n_d respectively. Let θ be the angle formed by n_d, n_k, n_p, where n_p is a virtual point in the grid with coordinates (x_d, y_k) (see Figure 3).

The components of $\upsilon_k(t)$ are:

$$\mathcal{C}_{NS} = \begin{cases} \mid \sin\theta * V_k^N \mid & \text{if } \sin\theta \geq 0 \\ \mid \sin\theta * V_k^S \mid & \text{if } \sin\theta < 0 \end{cases}$$

$$\mathcal{C}_{EW} = \begin{cases} \mid \cos\theta * V_k^E \mid & \text{if } \cos\theta \geq 0 \\ \mid \cos\theta * V_k^W \mid & \text{if } \cos\theta < 0 \end{cases}$$

The projection $\upsilon_k(t)$ is then computed as:

$$\upsilon_k(t) = \sqrt{\mathcal{C}_{NS}^2 + \mathcal{C}_{EW}^2} \tag{9}$$

Thus, if n_d is to the north of n_k, then $V_k^N(t)$ (as one would expect logically) should contribute the most to L_k. $V_k^E(t)$ or $V_k^W(t)$ contribute to a lesser

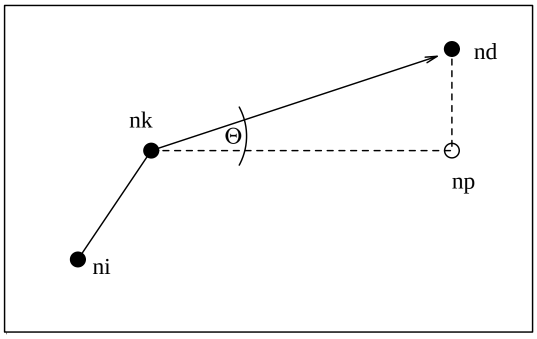

Figure 3: Possible position of n_i, n_k, n_d, and n_p in a network

extent, depending on the relative location of n_d. $V_k^S(t)$, in this particular case, does not make any contribution to L_k at all, as the south view of n_k is of little consequence to a message destined to go north through n_k. The tunable parameter γ determines the relative emphasis placed on the load (as measured by $\rho_k(t)$) versus the appropriate projections of $V_k(t)$ (as reflected by $v_k(t)$).

The path liability of a node n_k with respect to a message passing through n_i on its way to a destination n_d is given by:

$$P_k = \frac{D_{k,d}}{D_{i,d}} * \rho_i(t) \tag{10}$$

Clearly, choice of a neighbor node that has the smallest P_k biases Quo Vadis to route messages along paths that cover the largest fraction of the remaining distance to the destination (provided other things being equal).

It is possible to use a variety of other formulations that share the spirit of the examples shown above for the calculation of load and path liabilities. It is also possible to incorporate additional terms suggested by other performance criteria into the calculation of U_k. Routing decisions are based on parameterized heuristics so as to permit a range of tradeoffs through adaptation of tunable parameters to accommodate different (perhaps even conflicting) performance criteria under a range of different network dynamics.

4 A Prototype Simulation of Quo Vadis

A prototype implementation of Quo Vadis was used to conduct a number of experiments to explore the behavior of parameterized knowledge representation and heuristic routing mechanisms. The experiments described in this paper were conducted in simple regular $m \times n$ grid networks. We anticipate that more general network topologies might present several additional specific issues that may need to be addressed by Quo Vadis. However, our primary objective in this paper was to study and understand the behavior of Quo Vadis within a relatively simple setting through a set of carefully designed experiments.

4.1 Implementation of Quo Vadis

Quo Vadis has been implemented within an object-oriented discrete event-driven simulation environment [Mikler, Honavar & Wong, 1992; Mikler, Wong & Honavar, 1995]. Each network node is represented as a single M/M/1 queue with infinite buffer space, guaranteeing that every message in the network will ultimately be delivered to its respective destination node. Upon arriving at a particular node, the message is added to the queue, awaiting service by the routing mechanism. The queuing discipline is strictly First-In-First-Out (FIFO), so that a message is stalled until all messages that arrived earlier at this node are serviced. Service consists of two possible actions:

1. If the routing mechanism determines that the message has reached its destination, it is passed on to the higher protocol layers. Within the simulation, this entails the removal of the message from the network and the recording of its contribution to statistics for delay and hop-count.

2. If it is determined that the message needs to be passed on to other network nodes to further propagate towards its destination, the routing mechanism employs the heuristic decision mechanism (described above) to select a *best* next node.

The update of routing information is assumed to take place via a separate channel hence bypassing the FIFO queuing used for messages. Effectively, this could have been implemented through priority queuing, giving state change information the highest priority.

A message in the network is represented by general protocol information such as *creation time, source node, destination node, hop-count* and *message* `ID` together with a field that represents the simulated message size, i.e., the number of data bytes in the message. Message sizes are exponentially distributed with a mean that is specified at simulation startup. Provided that all nodes service messages at a constant rate, this results in an exponentially distributed service rate over all messages. Additional protocol information may have to be associated with each message in order to enable nodes to adapt decision parameters and to perform well in various network topologies. This is currently being investigated and will be described in a forthcoming paper.

4.2 Experimental Results

In order to study the performance of Quo Vadis, an $m \times n$ grid network was simulated for T seconds (real time). Each of the $N = m * n$ network nodes created messages at the same rate, i.e., 0.3 $msgs/s$. The destination nodes for messages are chosen at random at message creation. Every node in the network has equal probability of being selected as destination node for a particular message. Self-traffic however does not occur. It is further assumed that links have sufficient bandwidth so that transmission delays are negligible. Message delays are thus assumed to be caused solely by queueing delays encountered in network nodes.

To avoid biasing the results by the transient behavior of the networks at the beginning and the end of the simulation, statistics were recorded for only those messages that reached their destination during the time interval $(0.1T, 0.9T)$. Clearly, T must be chosen such that a sufficiently large number of messages can be recorded, thus yielding a good approximation of the various means computed. The results shown here were obtained by simulating 300 seconds of network operation using a 1024-node grid.

The experiments described below were designed to investigate the behavior of Quo Vadis especially in the context of its design objectives of gracefully minimizing delay, pro-actively avoiding congestion by distributing the load across the network, and reactively responding to unexpected localized increases in load, etc.

4.2.1 Shortest Path vs. Quo Vadis Routing

The following simulation results clearly demonstrate the success of Quo Vadis in selecting routes so as to reactively as well as pro-actively avoid highly utilized network areas. This behavior is governed primarily by the setting of the parameter β in Equation 7. To isolate the effect of β on the performance of Quo Vadis, other parameters — namely, α and γ — were maintained constant at $\alpha = \gamma = 0.5$. (Their impact on the overall performance of Quo Vadis is one of the topics of ongoing research).

From Equation 7 it is apparent that choosing parameter $\beta = 1.0$ forces Quo Vadis to select routes so as to minimize the remaining distance to the destination node. This is equivalent to what is generally referred to as *shortest path routing*. In a grid topology, the number of shortest paths between a node n_i and the destination node n_d is given by Equation 1. As one might expect, not all nodes in the grid network experience the same amount of traffic. In fact, nodes in the center of the grid network have to route a larger number of messages on average as compared to nodes at the fringes of the grid. This is due to the fact that a larger number of shortest paths between randomly chosen source-destination pairs pass through nodes in the center of the grid. The corresponding load-graph is shown in Figure 4. It clearly displays an increased load in nodes closer to the center of the grid and less load in those nodes at the grid's edges.

As seen in Figure 1, the message delay in a network node increases exponentially with its load. It follows that nodes in the center of the grid contribute most to the overall message delay along path traversed by the message. Thus, load at these nodes impacts the total message delay to a much higher degree than nodes at the fringes of the grid. This effect is amplified as the average network load increases.

Quo Vadis delays the onset as well as impact of this effect given an appropriate setting of β. While a shortest path routing algorithm makes a random decision among neighbors with equal *path utility* (Equation 10), Quo Vadis takes network load into account and biases the selection towards neighbors with better utility (Equation 8). The price paid for the ability to circumvent a highly utilized network area is an increase in mean path length $\bar{h}$.

The means of path length and message delay for different values of β are summarized in the Table 1. Figures 5 and 6 show the corresponding graphs for the $\bar{d}$ and $\bar{h}$ respectively. Figure 5 indicates the existence of an optimal

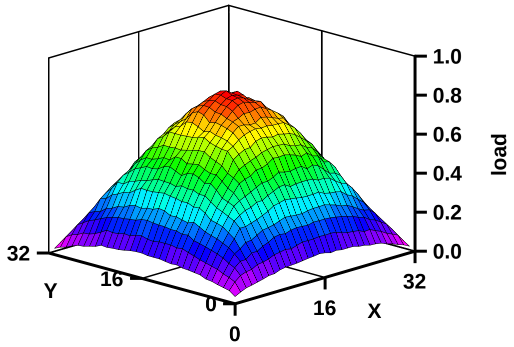

Figure 4: Load Distribution in a 1024 node grid network using Shortest Path Routing i.e., $\beta = 1.0$

value for β, β^* that minimizes the mean message delay. An increase in the mean delay is observed for $\beta < \beta^*$ as the routing decisions are dominated by the *load liability* L_k. For $\beta \ll \beta^*$ the performance can approach that of random routing. For $\beta > \beta^*$, Quo Vadis approaches shortest path routing thereby causing an increased mean message delay as discussed above.

The load distribution in the network using Quo Vadis routing with different values of β is shown in Figures 7, 8, and 9.

Clearly, a load sensitive setting of β results in a more balanced distribution of load, thus preventing a single network area from becoming overutilized. If load vigilance is high (i.e., small β), routing decisions may result in extended path length. However, this does not necessarily lead to an increase in total message delay along the path if the message is routed through a lightly loaded area. The exponential increase in delay with increasing load justifies such a tradeoff. The following example clarifies this point:

Let $\mu = 10\ msgs/s$ and consider two paths P_1 and P_2 with path lengths 5 and 3 respectively. Further assume the loads along P_1 to be

$$\rho_{1-5} = (0.3, 0.3, 0.2, 0.3, 0.4)$$

and loads along P_2 to be

$$\rho'_{1-3} = (0.3, 0.8, 0.4).$$

$\beta =$	$\bar{h}$	$\bar{d}$
0.3	23.07	2.43
0.4	22.76	2.41
0.5	22.44	2.36
0.6	22.15	2.34
0.7	21.89	2.33
0.8	21.58	2.35
0.9	21.33	2.51
1.0	21.29	2.79

Table 1: Mean Hop Count ($\bar{h}$) and Mean Message Delay ($\bar{d}$) for different values of β. ($n > 85700$ $messages$)

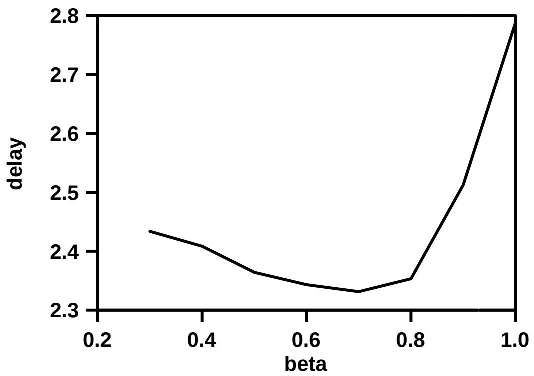

Figure 5: Effects of different values of β on $\bar{d}$

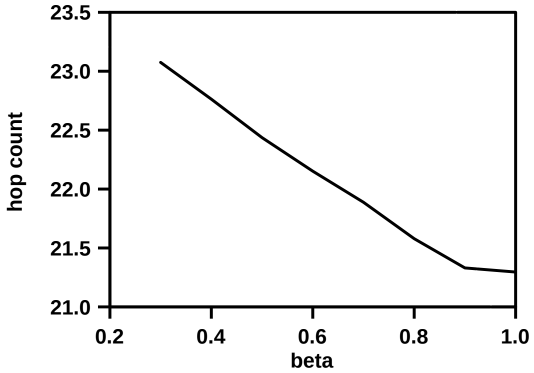

Figure 6: Effects of different values of β on $\bar{h}$

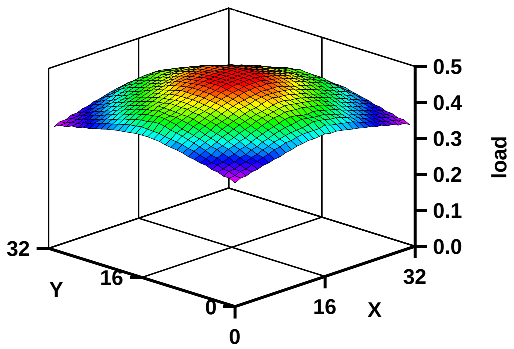

Figure 7: Load Distribution using Quo Vadis with $\beta = 0.4$

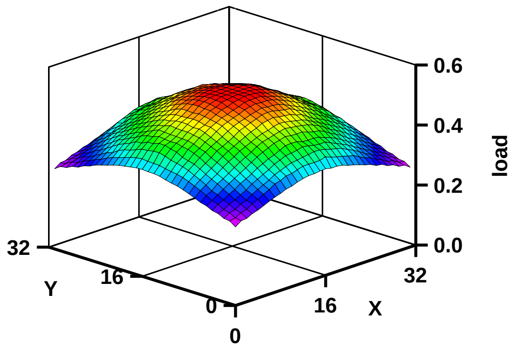

Figure 8: Load Distribution using Quo Vadis with $\beta = 0.6$

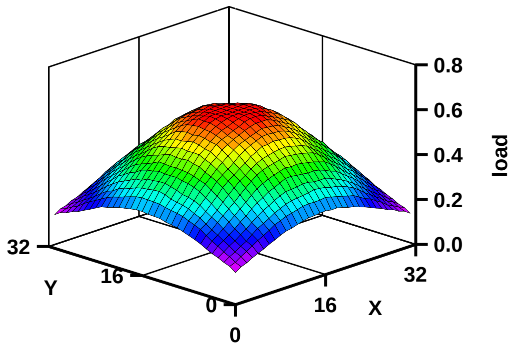

Figure 9: Load Distribution using Quo Vadis with $\beta = 0.8$

While the total load along P_1 and P_2 are the same, Equation 2 yields total *delays* of 0.720 s and 0.810 s along P_1 and P_2 respectively. Though longer, P_1 clearly is a better choice when delay is to be minimized.

If routing decisions result in path P_2, the message not only experiences a larger delay, but in addition would make things worse for messages that cannot avoid intersecting P_2 on their way to their destination.

4.2.2 Routing in the Presence of Hotspots

Hot spot refers to a single node or a small group of nodes in the network that experience a sudden increase in utilization. Such hotspots may be caused due (among other things) to:

- localized increases in arrival rate, or
- localized node or link failures.

One of the desirable properties of a routing mechanism is its ability to react to such load changes. A good routing algorithm should attempt to route messages around the hotspot, thereby reducing the message delay, perhaps at the expense of increasing the total length of the route.

The ability to adapt to such localized load changes quickly has been deliberately designed into Quo Vadis. Nodes in the neighborhood of a suddenly *over-utilized* node start to divert traffic as soon as the load increase is made known to them. High load in an affected node (as in highly loaded network areas) has a repulsive effect on traffic and routing decisions are automatically biased towards avoiding that node. Again, the extent of this bias is determined by β. Such dispersion of traffic is accomplished with minimal impact on nodes that are sufficiently distant from those that are affected by local increases in load.

While the increase in a node's load should clearly repel messages from being routed though it, a sudden load decrease should be utilized by nodes in the neighborhood in their effort to distribute network load uniformly.

Sudden load changes have been simulated by increasing and decreasing a node's service rate. The effects of such a change when shortest path routing is in place is shown in Figure 10. The effects of adaptive measures taken by Quo Vadis are shown in Figures 11 and 12. Shortest path routing (i.e., $\beta = 1.0$) does not attempt to reduce the influx of traffic into the affected

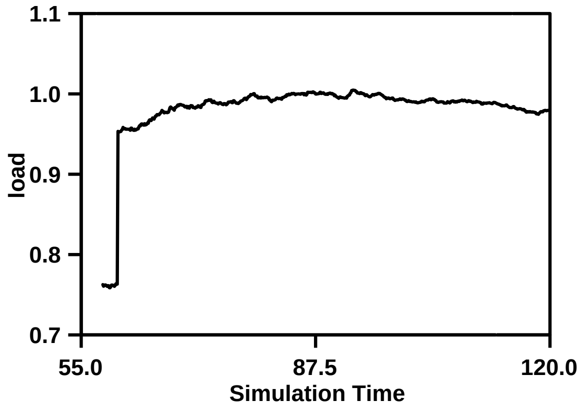

Figure 10: Effects of sudden load increase in node n_i under Shortest Path Routing

area in order to normalize the load conditions at the hotspot. Quo Vadis, however, balances load conditions in the network in a relatively short some time. This is accomplished by the dispersion of traffic which would otherwise have been routed through the hotspot area. The relationship between the time needed for the normalization of load conditions and parameters α, β, and γ is currently being investigated.

5 Discussion & Future Work

Quo Vadis attempts to reduce the resource requirement for storage, acquisition, and use of network state information while achieving the desired performance (as defined by the criteria such as average message delay).

The knowledge base in an n-node network with k-connectivity has size $O(k)$ since only information obtained from each of the k neighbors is stored. This information consists of local load measures as well as summarized view information in 4 directions from each neighbor. Thus, the storage requirements of Quo Vadis are essentially independent of the size of the network and hence scale very well with increasing network size. Since Quo Vadis makes do with propagation of only local measurements $\rho_j(t)$ and the view vector

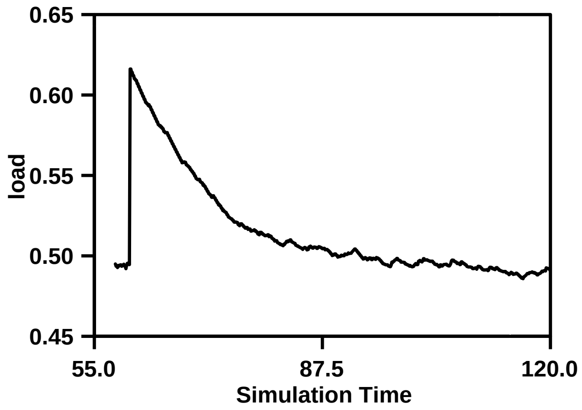

Figure 11: Effects of Quo Vadis on a sudden load increase in node n_i

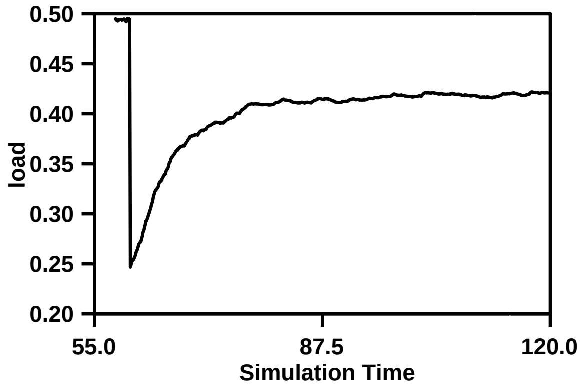

Figure 12: Effects of Quo Vadis on a sudden load decrease in node n_i

$V_j(t)$ between neighboring nodes n_j and n_i, the bandwidth requirement is small compared to conventional routing mechanisms. As explained in previous sections, Quo Vadis does not attempt to construct a precise picture of the network state as imprecision increases with distance and uncertainty of routing decisions in inevitable. Instead, it uses a coordinate system that provides for directional orientation together with a summary of network state information. This allows Quo Vadis to avoid the costly validity check of information as required by routing methods that use the link state protocol.

The experimental results presented in this paper clearly demonstrate that Quo Vadis is largely successful in meeting its primary design objectives, at least when it is used within the relatively simple regular grid network. Particularly noteworthy is the ability of Quo Vadis to pro-actively as well as reactively avoid congestion in the network while simultaneously minimizing message delay. More systematic parametric study of Quo Vadis with emphasis on parameters such as, α, γ, and update interval τ (and the interrelationships among them as well as β) is in progress.

Extensive research by other researchers on both link state and distance vector routing algorithms have uncovered many issues that need to be considered in the design of new routing mechanisms. Examples of such design issues are bandwidth and storage overhead, performance in the presence of failure [Merlin and Segall, 1979; Jaffe and Moss 1982; Wong and Kang 1990], message looping and bouncing. The current design of Quo Vadis aims at reducing resource overhead. Issues such as message looping, message bouncing, as well as mechanisms to deal with node and link failures are currently under study.

A long-term objective of this research is the design of completely autonomous self-managing, intelligent, low-overhead, robust and adaptive traffic management mechanisms for very large high speed communication networks of the future. Towards this end, mechanisms that dynamically adapt the tunable parameters (α, β, γ, τ) used by Quo Vadis at each node in response to changes in network dynamics are of interest. In particular, variations of techniques drawn from adaptive control [White & Sofge, 1992] and machine learning [Honavar, 1994], especially reinforcement learning [Keerthi & Ravindran, 1994] are currently under investigation. For examples of preliminary work by other investigators on this topic, the reader is referred to [Littman and Boyan 1993; Lehman et al. 1993].

In conclusion, it must be noted that Quo Vadis exemplifies a family of parameterized algorithms, different instances of which may be appropriate for optimization of different performance criteria. The basic mechanism can be applied in various network topologies after being supplemented by additional protocol elements as necessary. Different approaches to facilitate network nodes with information which would enhance the quality of routing decisions in networks with complex topologies are currently being investigated.

The results presented in this paper clearly indicate the advantage of viewing routing as a distributed, heuristic multi-criterion optimization task with adaptive properties so as to respond quickly to various forms of network dynamics.

Acknowledgement

The authors would like to thank Todd Campbell and Deepa Parthasarathy for their contribution to the design of object classes used in the implementation of the Quo Vadis simulation environment. Vasant Honavar would like to acknowledge the support of National Science Foundation (grant IRI-9409580) and the Iowa State University College of Liberal Arts and Sciences during this research.

References

Bertsekas, D. and Gallager, R. *Data Networks.* 2nd ed. Prentice-Hall, 1992.

Cegrell, T. A Routing Procedure for the TIDAS Message-Switching Network. *IEEE Transactions on Communications* Com-23, No.6 (1975): 575-585.

Gerla, M. and Kleinrock, L. Flow Control: A Comparative Survey. *IEEE Transactions on Communications* Com-28, No.4 (1980): 553-574.

Honavar, V. Toward Learning Systems That Use Multiple Strategies and Representations. In: *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration.* Honavar, V. and Uhr, L. (Ed.) San Diego: Academic Press (1994): 615-644.

Jaffe, J.M. and Moss, F.H. A Responsive Distributed Routing Algorithm for Computer Networks. *IEEE Transactions on Communications* Com-30, No.7 (1982): 1758-1762.

Jain, R. Congestion Control in Computer Networks: Issues and Trends. *IEEE Network Magazine* May 1990 (1990): 24-30.

Jain, R. *The Art of Computer Systems Performance Analysis.* Wiley & Sons, Inc., New York: 1991.

Keerthi, S.S. and Ravindran, B. A Tutorial Survey of Reinforcement Learning. (preprint) (1994).

Kleinrock, L. and Kamoun, F. Hierarchical Routing for Large Networks. *Computer Networks* 1 (1977): 155-174.

Lehmann, F., Seising, R., and Walther-Klaus, E. Simulation of Learning in Communication Networks. *Simulation Practice and Theory* 1, No.1 (1993): 41-48.

Littman, M. and Boyan, J. A Distributed Reinforcement Learning Scheme for Network Routing. *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications.* Alspector, J., Goodman, R. and Brown, T. X. (Ed.) (1993): 45-51.

McQuillan, J.M., Richter, I., and Rose, R.C. The New Routing Algorithm for the ARPANET. *IEEE Transactions on Communications* Com-28, No.5 (1980): 711-719.

Merlin, P.M. and Segall, A. A Failsafe Distributed Routing Protocol. *IEEE Transactions on Communications* Com-27, No.9 (1979): 1280-1287.

Mikler, A.R., Honavar, V.G., and Wong, J.S.K. Simulating a Traveller: A Heuristic Approach to Routing in Large Communication Networks. *Proceedings of the European Simulation Symposium, ESS '92* November 1992 (1992): 297-301.

Mikler, A.R., Wong, J.S.K., and Honavar, V.G. Quo Vadis – A Framework for Adaptive Routing in Very Large Communication Networks. *Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications;* Alspector, J., Goodman, R. and Brown, T. X. (Ed.) (1993):196-202.

Mikler, A.R., Wong, J.S.K., and Honavar, V.G. Modelling and Simulation of Traffic Management in Large Communication Networks: A Case Study Using Quo Vadis. In preparation (1995).

Pearl, J. *Heuristics: Intelligent Search Srategies for Computer Problem Solving.* Addison-Wesley, Reading, Massachusetts: 1984.

Perlman, R. Hierarchical Networks and the Subnetwork Partition Problem. *Computer Networks and ISDN Systems* 9 (1985): 297-303.

Perlman, R. *Interconnections, Bridges and Routers.* Addison-Wesley, Reading, Massachusetts: 1992.

Robertazzi, T.G. *Computer Networks and Systems: Queueing Theory and Performance Evaluation.* Springer Verlag, New York: 1990.

Schwartz, M. and Stern, T.E. Routing Techniques used in Computer Communication Networks. *IEEE Transactions on Communications* Com-28, No.4 (1980): 539-552.

Snyder, J.M. A Routing Architecture for Very Large Networks Undergoing Rapid Reconfigurations. *Proc ACM SIGCOMM 1989* (1989): 57-63.

Tanenbaum, A.S. *Computer Networks.* Prentice-Hall, 1988.

Tsuchiya, P.F. The Landmark Hierarchy: A new Hierarchy for Routing in Very Large Networks. *Proc ACM SIGCOMM 1988* (1988): 35-42.

Wang, Z. and Crowcroft, J. Shortest Path First with Emergency Exits. *SIGCOMM '90* (1990): 166-176.

White, D.A., and Sofge, D.A. (Ed.) *Handbook of Intelligent Control.* New York: Van Nostrand Rheinhold (1992).

Wong, J.S.K. and Kang, Y. Distributed and Fail-Safe Routing Algorithms in Toroidal-Based Metropolitan Area Networks. *Computer Networks and ISDN Systems* 18 (1990): 379-391.

Wong, J.S.K. and Mikler, A.R. Routing Algorithms for High-Speed Communications Networks. *Broadband Communications Systems*; (Conard, J.W. , Ed.) Auerbach Publications, 1993. 97-105.