1995

Quo Vadis - Adaptive Heuristics for Routing in Large Communication Networks

Armin Mikler
Iowa State University

Johnny S. Wong
Iowa State University, wong@iastate.edu

Vasant Honavar
Iowa State University

Follow this and additional works at: http://lib.dr.iastate.edu/cs_techreports

Part of the OS and Networks Commons, and the Systems Architecture Commons

Recommended Citation
http://lib.dr.iastate.edu/cs_techreports/106

This Article is brought to you for free and open access by the Computer Science at Digital Repository @ Iowa State University. It has been accepted for inclusion in Computer Science Technical Reports by an authorized administrator of Digital Repository @ Iowa State University. For more information, please contact digirep@iastate.edu.
Quo Vadis - Adaptive Heuristics for 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.
Message routing and congestion control are typical traffic management tasks. These functions are generally thought of being hosted by the layer 3 of the Open Systems Interconnection (OSI/ISO) 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.

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,
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.

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, namely, distance vector routing and link state routing [Perlman, 1992].

In both strategies, network nodes rely on knowledge about every node or link in 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. At the same time, the imprecision or uncertainty associated with network state information grows with the size of the network as 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.

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].

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.

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 $\tau$ 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

Any intelligent traffic management mechanism capable of performing in a large communication environment must include an effective knowledge representation (KR) mechanism as well as an efficient knowledge acquisition (KA) engine, that minimizes the overhead that is associated with acquiring and maintaining network state information. In addition, adaptive decision making methods are needed which 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. In practice, all routing decisions in a large communication network are based on imprecise, uncertain knowledge of the current network state. This 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.

2. 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 [Snyder, 1989].

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. It follows that the likelihood of finding alternative paths of comparable length is a non-decreasing function of the distance to the destination.

4. The utilization $\rho$ of network nodes is generally determined by the ratio $\lambda/\mu$ where $\lambda$ represents the arrival rate to that node and $\mu$ 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. 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 $\rho_i$. The expected delay $D_i$ is given by

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

$D_i$ grows exponentially as $\rho_i$ increases.

### 3.1 The Design of Quo Vadis

The current design of Quo Vadis [Mikler, Honavar, & Wong, 1992; Mikler, Wong, & Honavar, 1993, 1994] 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

The KR 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, 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 a node $n_i$ 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 have no knowledge of the network connectivity which is implicitly available in routing tables, it needs an alternative scheme for addressing nodes and for computing their positions relative to each other. This is accomplished by assigning each network node a unique coordinate which reflects its location in the euclidean plane. Thus, each node $n_i$ is addressed by its respective coordinates $(x_i, y_i)$.

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 $V^d_i(t - \tau)$ (where $\tau$ 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 $g^d_{i,k}$ – 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 $V^d_i$ 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 $H_i = \{n_k \mid n_k$ 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:

$$G^S_{i,k} = \begin{cases} 1 + \eta \times \left( \frac{y_i - y_k}{D_{i,k}} \right) & \text{if } y_i \geq y_k \\ 0 & \text{otherwise} \end{cases}$$

where $\eta$ is to be chosen such that the directional gain appropriately amplifies load information from nodes in direction $d$. The directional gains $G^N_{i,k}$, $G^E_{i,k}$, and $G^W_{i,k}$ for the north, east, and west component of $V^d_i(t)$ are given by similar formulae. The normalization factor $G^d_i$ for direction $d$ for gains $G^d_{i,k}$ computed at node $n_i$ is given by:

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

The corresponding normalized directional gains are given by:

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

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

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

where $\tau$ 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 $\alpha$ 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, such as utilization, delay measures, or cell loss probability. In our model, network nodes are modeled as M/M/1 queues and $\rho_k(t)$ corresponds to the node utilization.
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 $\alpha$. 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 $\tau$). 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 performance of Quo Vadis would depend on how well it reflects the actual state of the network.

Suitable mechanisms that adapt parameters such as $\alpha$ and $\tau$ in response to variations in network dynamics and/or changes in performance demands are of interest. It is possible for $\alpha$ (and all other parameters) assume 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.

### 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:

$$U_k = -\left(\beta \times P_k + (1 - \beta) \times L_k\right); \quad 0 \leq \beta \leq 1$$

(6)
Figure 1: Possible position of \( n_i, n_k, n_d, \) and \( n_p \) in a network

where \( \beta \) 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:

\[
L_k = \gamma \times \rho_k(t) + (1 - \gamma) \times v_k(t); \quad 0 \leq \gamma \leq 1 \tag{7}
\]

where \( v_k(t) \) 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 \( n_k, v_k(t) \) is composed of two components, namely an east-west component \( C_{EW} \) and a north-south component \( C_{NS} \). 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 \( \theta \) 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 ??).

The components of \( v_k(t) \) are:

\[
C_{NS} = \begin{cases} 
\sin \theta \times V^N_k & \text{if } \sin \theta \geq 0 \\
\sin \theta \times V^S_k & \text{if } \sin \theta < 0 
\end{cases}
\]

\[
C_{EW} = \begin{cases} 
\cos \theta \times V^E_k & \text{if } \cos \theta \geq 0 \\
\cos \theta \times V^W_k & \text{if } \cos \theta < 0 
\end{cases}
\]

The projection \( v_k(t) \) is then computed as:

\[
v_k(t) = \sqrt{C^2_{NS} + C^2_{EW}} \tag{8}
\]

Thus, if \( n_d \) is to the north of \( n_k \), then \( V^N_k(t) \) (as one would expect logically) should contribute the most to \( L_k \). \( V^E_k(t) \) or \( V^W_k(t) \) contribute to a lesser extent, depending on the relative location of \( n_d \). \( V^S_k(t) \), 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 $\gamma$ 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:

$$P_k = \frac{D_{k,i}}{D_i} \times \rho_i(t)$$

where $D_{i,j}$ is the euclidean distance between $n_i$ and $n_j$. 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).

Other formulations that share the spirit of the examples shown above for the calculation of load and path liabilities are certainly possible. 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 Simulation of Quo Vadis

A prototype implementation of Quo Vadis was used to conduct a number of experiments to explore the effects of the various parameters used in Quo Vadis. These experiments 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.

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 $\text{M/M/1}$ 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. 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.

In order to study the effects of parameters $\alpha$, $\beta$, and $\gamma$, for each of the experiments an $m \times n$ grid network was simulated for $T$ seconds (real time).

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.

## 4.1 The Effects of $\alpha$

The parameter $\alpha$ determines how the composite load landscape of the network is reflected by the nodes’ individual views. Therefore, both the distance over which a specific load condition can have impact on routing decisions as well as the degree of this impact are governed by $\alpha$.

As all parameters in Quo Vadis are tightly coupled a demonstration of the effects of $\alpha$ with respect to the view computation required the isolation of the knowledge representation from the overall routing mechanism. For this simulation experiment, a $10 \times 10$ grid network was set in a particular state corresponding to a pre-determined load distribution. The underlying motivation of this approach is to statically model various load conditions and to determine their impact on the view $V_i(t)$ as acquired by node $n_i$. In order to eliminate the effects of routing decisions on the load distribution in the network, nodes generated only self-traffic at a constant rate. Thus, messages did not have to be routed among network nodes but could be delivered to the node itself at a node’s service rate. As a consequence, the values for parameters $\beta$ and $\gamma$ were rendered irrelevant for this experiment. The network together with its corresponding load distribution is shown in Figure ??.

Adverse load conditions were simulated by increasing the message generation rate at a single node (or a small number of nodes). Since no messages were sent across the network, the only information communicated among network nodes was view and load information. The views into each of the four directions (East, West, North, and South) as acquired after $T$ seconds of simulation by each individual node were then analyzed. This experiment was
repeated for different values of $\alpha$ (0.1 through 1.0 in steps of 0.1). Figures ?? and ?? show the East-Views, $V^E_i$, as acquired at every node $n_i$ in the network after $T$ seconds of simulation for different values of $\alpha$. It should be noted that figures ?? and ?? do not display view values for nodes $\{9, 19, 29, 39, 49, 59, 69, 79, 89, 99\}$, as the East-Views in these nodes are undefined.

From Equation ?? it is apparent that for $\alpha = 1.0$, a node $n_i$ computes its east-view $V^E_i$ solely as the weighted average of local load values $\rho_j$ obtained from neighbor nodes $n_j \in H_i$. The views, $V^E_j$, computed in neighbors $n_j$ do not contribute to $V^E_i$. Depending on the value of $\eta$ in Equation ??, $V^E_i$ is a more or less precise image of $\rho_j$ computed in $n_j$ if $n_j$ is east of $n_i$ and $n_j \in H_i$. For smaller values of $\alpha$ (i.e., $\alpha = 0.6$), Equation ?? takes the view $V^E_j$ of neighbors into account thus computing $V^E_i$ as an average of view and load measures of nodes in an extended neighborhood. That is, network nodes $n_k \notin H_i$ affect the magnitude of $V^E_i$. These effects are clearly displayed in Figure ??.

As $\alpha \to 0$ a load condition in a single node $n_k$ affects the view in a much larger set of nodes. However, the magnitude of impact on the view $V^E_i$ is significantly reduced. Figure ?? shows the change of magnitude as a function of distance from $n_k$.

How views $V^d_i$, can be used to optimize performance in an anticipatory fashion is further highlighted in the study of effects of parameter $\gamma$.

### 4.2 The Effects of $\beta$

For the study of the effects of $\beta$ on the selection of routes, Quo Vadis was simulated in a 1024-node grid network for 300 seconds. Each of the $N = m \times n$ network nodes created messages at the same rate, i.e., $0.3 \text{msgs/s}$. The destination nodes for messages are chosen at random at message creation. Every node in the network has equal probability of being selected as
Figure 3: $V_i^E$ for $\alpha = 1.0$ and $\alpha = 0.6$ respectively

Figure 4: $V_i^E$ for $\alpha = 0.3$ and $\alpha = 0.1$ respectively
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.

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 $\beta$ in Equation ???. To isolate the effect of $\beta$ on the performance of Quo Vadis, other parameters - namely, $\alpha$ and $\gamma$ - were maintained constant at $\alpha = \gamma = 0.5$.

### 4.2.1 Shortest Path vs. Quo Vadis Routing

From Equation ?? 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 $n_d$ depends on their relative hop-distance. 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 ???. 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 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
Table 1: Mean Hop Count ($\bar{h}$) and Mean Message Delay ($\bar{d}$) for different values of $\beta$. ($n > 85700$ messages)

<table>
<thead>
<tr>
<th>$\beta$</th>
<th>$\bar{h}$</th>
<th>$\bar{d}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3</td>
<td>23.07</td>
<td>2.43</td>
</tr>
<tr>
<td>0.4</td>
<td>22.76</td>
<td>2.41</td>
</tr>
<tr>
<td>0.5</td>
<td>22.44</td>
<td>2.36</td>
</tr>
<tr>
<td>0.6</td>
<td>22.15</td>
<td>2.34</td>
</tr>
<tr>
<td>0.7</td>
<td>21.89</td>
<td>2.33</td>
</tr>
<tr>
<td>0.8</td>
<td>21.58</td>
<td>2.35</td>
</tr>
<tr>
<td>0.9</td>
<td>21.33</td>
<td>2.51</td>
</tr>
<tr>
<td>1.0</td>
<td>21.29</td>
<td>2.79</td>
</tr>
</tbody>
</table>

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 reduces the impact of this effect given an appropriate setting of $\beta$. While a shortest path routing algorithm makes a random decision among neighbors with equal path utility (Equation ??), Quo Vadis takes network load into account and biases the selection towards neighbors with better utility (Equation ??). 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 $\beta$ are summarized in the Table ??.

Figure ?? shows the corresponding graphs for the $\bar{d}$ and $\bar{h}$. Figure ?? indicates the existence of an optimal value for $\beta$, $\beta^*$ 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 $L_L$. 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 $\beta$ is shown in Figure ??.

Clearly, a load sensitive setting of $\beta$ 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 $\beta$), 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 $\mu = 10$ msg/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

$$\rho'_{1-3} = (0.3, 0.8, 0.4).$$
While the total load along $P_1$ and $P_2$ are the same, Equation ?? 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 $\beta$. 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 through 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 ???. The effects of adaptive measures taken by Quo Vadis are shown in Figures ???. Shortest path routing (i.e., $\beta = 1.0$) does not attempt to reduce the influx of traffic into the affected area in order to normalize the load conditions at the hotspot. Quo Vadis, however, balances load conditions in the network in a relatively short 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 $\alpha$, $\beta$, and $\gamma$ is currently being investigated.

![Figure 7: Effects of sudden load increase in node $n_i$ under Shortest Path Routing](image)

4.3 The Effects of $\gamma$

In Equation ???, $\gamma$ defines the significance of load measures $\rho_k$ versus $v_k$, the projections of a node’s view $V_k(t)$ with respect to a particular destination. The underlying motivation is to enable network nodes to make routing decisions in either reactive or anticipatory fashion. For $\gamma = 1$, only $\rho_k$ determines the load liability of $n_k$, thereby enabling $n_i$ to route messages so as to circumvent the neighbor node $n_k \in H_i$ with the highest utilization, thus reacting to adverse load conditions in the immediate neighborhood. On the other hand, small values of $\gamma$ (i.e., $\gamma \to 0$) node $n_i$ will base its evaluation of neighbors $n_k$ on a load summary as represented by $V_k(t)$ with respect to the relative location of the destination. Hence, adverse load conditions on the path towards the destination can be sensed by $n_i$ so as to adjust the routing decision.
As for the evaluation for $\alpha$, the isolation of the effects of $\gamma$ required the network to remain in a pre-determined state. The corresponding network and load graph are shown in Figure 8.

In addition, nodes 40 and 49 have been selected to serve as source and destination nodes for a single message which is traced on its journey through the network. The purpose of the trace is to identify all nodes that are visited by the message thus revealing the routing decisions made by intermediate nodes. This experiment is repeated for various values of $\gamma$. Since $\beta$ controls the significance of the load liability, it has been chosen so as to amplify the effects of $\gamma$, i.e., $\beta$ was maintained constant at 0.2. The value of $\alpha$ was set to 0.3, thus making the effects of adverse load condition visible at distant nodes.

The different routes traveled by a test message are presented in Table 2 for various values of $\gamma$. Clearly, the shortest path between source node 40 and destination node 49 is

<table>
<thead>
<tr>
<th>$\gamma$</th>
<th>Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>(40,30,31,32,33,34,35,36,37,38,48,49)</td>
</tr>
<tr>
<td>0.1</td>
<td>(40,41,31,32,33,34,35,36,37,38,39,49)</td>
</tr>
<tr>
<td>0.2</td>
<td>(40,41,31,32,33,34,35,36,37,38,39,49)</td>
</tr>
<tr>
<td>0.3</td>
<td>(40,41,42,32,33,34,35,36,37,38,39,49)</td>
</tr>
<tr>
<td>0.4</td>
<td>(40,41,42,43,33,34,35,36,37,38,39,49)</td>
</tr>
<tr>
<td>0.5</td>
<td>(40,41,42,43,33,34,35,36,37,38,39,49)</td>
</tr>
<tr>
<td>0.6</td>
<td>(40,41,42,43,44,45,46,47,55,56,57,58,59,49)</td>
</tr>
<tr>
<td>0.7</td>
<td>(40,41,42,43,44,45,46,47,57,58,59,49)</td>
</tr>
<tr>
<td>0.8</td>
<td>(40,41,42,43,44,45,46,47,57,58,59,49)</td>
</tr>
<tr>
<td>0.9</td>
<td>(40,41,42,43,44,45,46,47,57,58,59,49)</td>
</tr>
<tr>
<td>1.0</td>
<td>(40,41,42,43,44,45,46,47,57,58,59,49)</td>
</tr>
</tbody>
</table>

**Table 2:** Points of deflection for different values of $\gamma$
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 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.

Since Quo Vadis propagates only local measurements $\rho_j(t)$ and the view vector $V_j(t)$ 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 is 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 re-actively avoid congestion in the network while simultaneously minimizing message delay. More systematic parametric study of Quo Vadis in a dynamic environment with emphasis on parameters such as, $\alpha$, $\beta$, $\gamma$, $\eta$, and update interval $\tau$ (and the interrelationships among them as well as $\beta$) 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 ($\alpha$, $\beta$, $\gamma$, $\eta$, $\tau$) 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. 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


