Proceedings of the 2003 IEEE
International Conference on Robotics & Automation
Taipei, Taiwan, September 14-19, 2003

Connectivity-Through-Time Protocols for Dynamic Wireless Networks to Support Mobile Robot Teams

Nageswara S. V. Rao
Center for Engineering Science
Advanced Research
Oak Ridge National Laboratory
Oak Ridge, TN 37831, USA.

Qishi Wu, S. Sitharama Iyengar
Department of Computer Science
Louisiana State University
Baton Rouge, LA 70803, USA

Arul Manickam
Department of Electrical and
Computer Engineering,
Carnegie Mellon University,
Pittsburgh, PA 15213, USA

***Abstract*— Mobile robot teams are increasingly deployed in various applications involving remote operations in unstructured environments that do not support wireless network infrastructures. We propose a class of protocols based on the connectivity-through-time concepts that exploit the robot movements to extend the traditional notions of network connectivity. These protocols enable the formation of ad hoc networks of mobile robots without the infrastructure of access points by utilizing the robots as routers. These protocols are implemented as a collection of daemons that track connectivity changes, compute single and multiple hop connectivity, route the packets via robots with suitable buffering, and adapt the transport parameters to the connection characteristics. The implementation employs UDP with window-based flow control that is tuned to the nature of connections. We present experimental performance results based on our implementation on robot teams to illustrate the salient features of this approach.**

I. INTRODUCTION

Mobile robot teams are being increasingly employed in remote and unstructured environments for operations such as terrain mapping and surveillance [1]. Such robot teams offer powerful capabilities. For example, a robot team can be deployed (perhaps air-dropped) to build a radiation map of an urban area suspected of nuclear or chemical contamination before human operators are allowed into the area. In another application, the robot team can cooperatively search for a target (such as a human in need of medical attention) in an area unsafe for humans. Typically, in these applications there is a need for the robots to effectively communicate to coordinate their activities as well as to combine the gathered information. The networking needs for this class of applications are quite specific and are not adequately addressed by the existing wireless ad hoc networking technologies, which are often the off-shoots of Internet-based approaches.

To describe the operational space of our networks, we present an example that is prototypical to the above class of applications. Consider that a team of mobile robots is deployed in a remote building to cooperatively scan the floors of the terrain for radiation levels by combining the sensor information. The robots are equipped with off-the-shelf wireless cards and support conventional TCP/IP stack. The building consists of steel reinforcements, which could affect the radio connectivity in an unpredictable manner. The radiation levels in the building are unknown and hence humans are not allowed into the building until it is completely scanned and found to be safe.

There is a wide spectrum of applications in which the wireless networks are deployed, ranging from campus access to robot teams to sensor networks [9]. The networking technologies, including hardware components and software modules, could be quite specific to the application. In campus networks, an infrastructure is deployed so that the node movements are covered by the access points [12]. In the sensor networks area, various types of scenarios call for different type of wireless networks [3]. In ad hoc wireless networks the challenge is to form and operate a network without the infrastructure. In dynamic networks the additional challenge is to cope with the changes in network connectivity. Several network protocols have been developed for various sensor network scenarios (see [2] and references therein). While robot teams can be considered a special case of sensors networks, their specific considerations require a closer inspection of the connectivity and transport performance issues. In particular, the latter has received very little attention since a vast majority of works focus on the connectivity issues alone.

The specific class of wireless ad hoc networks needed for the operation of mobile robot teams in the above scenario lead to the following considerations.

- *Small and mobile robot teams:* We consider teams of no more than ten mobile robots which cooperatively perform a task. Its primary focus is to execute a cooperative mission, and the movements of the robots are not tasked exclusively for communication purposes, i.e., network connectivity plays a secondary role to primary mission of the robots. Thus the methods designed for dense swarm type teams or stationary networks typical of sensor networks are not adequate here [2].
- *No infrastructure:* The robots operate over a wireless network in areas that are typical indoor or urban environments. In the above scenario, it is not feasible to

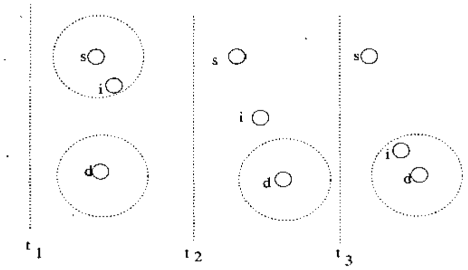

Fig. 1. Snapshots of a network of three nodes at times t_1, t_2 and t_3.

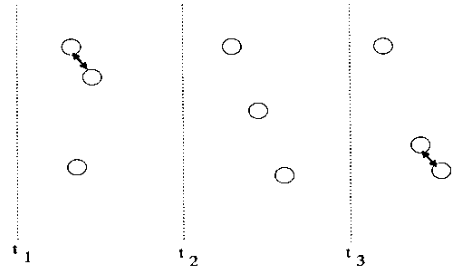

Fig. 2. $G(t)$ varies in time.

manually setup a network of access points before the robots are put into operation. The radio connectivity is highly dynamic and unpredictable due to the unstructured nature of the terrain and robot movements. As a result the connectivity is sparse since the robots are few and could be widely dispersed.

- *No special hardware:* We consider that the robots are equipped with IEEE 802.11 wireless cards, and no special communication hardware is available. The operating system is capable of executing socket-based codes.

These considerations are motivated by the current state of the art in commercially available mobile robots and networking technologies that can be deployed under short time frames.

In existing commercial mobile robot networks it is common to employ Internet wireless network technologies, typically IEEE 802.11 wireless cards and default TCP/IP stack. Typically the wireless cards are configured in the default infrastructure mode wherein the robots communicate exclusively through the access points. Indeed, a backbone of the access points is needed to connect various robots, which is not feasible in our case. The 802.11 cards can be operated in ad hoc mode in which case the robots that are within the radio range can communicate with each another but their connectivity is restricted to pairs that are within the radio range. While the multi-hop connectivity can be achieved in recent Linux kernels using IP forwarding, such method is effective only when source and destination robots are connected (via single or multiple hops) for the entire duration of message transmission. In particular, such approach does not exploit the connectivity gains due to robot movements.

Apart from the technological considerations, these robot networks also involve certain conceptual issues that are not main stream in current networking technologies, which are mostly (not all) Internet-based. Typically, the connectivity changes are treated as aberrations and are handled as exceptions. On the other hand, in the above scenarios the connectivity changes are integral parts of the operation. More importantly, if suitable protocols are employed, the connectivity changes can actually improve the network throughput as analytically shown in [4]. We show that the connectivity-through-time concept provides a way to conceptualize such phenomenon and to design protocols to exploit it. This concept was originally proposed in [10] and implemented in a working protocol for MS Windows operating system in [11] to support group meeting applications. The protocol proposed in this paper is based on a similar basic concept. But several components are redesigned for better performance tailored to the robot teams, and additional transport controls are added to achieve better throughputs.

The proposed implementation called CTIME of this protocol consists of three main parts. First, the path computation part updates the connectivity information from periodic messages from nearby nodes. Second, the routing part sends the packets to various nodes depending on the connectivity to the destination. Third, the transport part handles the various sending rates and duplicate packets. Conceptually this method is a combination of reactive and proactive approaches [9] in that it uses the former for routing along currently reachable destinations and the latter otherwise. While flooding is the basic mechanism for communication in highly dynamic networks, there are a number of specific parameters that need to be properly adapted to the network conditions to ensure good throughput. The implementation in [11] uses TCP for transport between direct neighbors. Such method is not suited here since the robot movements cause packet losses which are interpreted as congestions losses by TCP, thereby severely reducing the throughput. We adopt a window-based UDP mechanism for transport control in this paper. We tune the throughput rates of the sources based on the connection type such as direct, multiple hop or through-time using the experimental data.

The organization of this paper is as follows. The concept of connectivity-through-time is described in Section II. The CTIME implementation of the protocol based on this concept is described in Section III. Experimental results based on the implementation including settings for various parameters are described in IV.

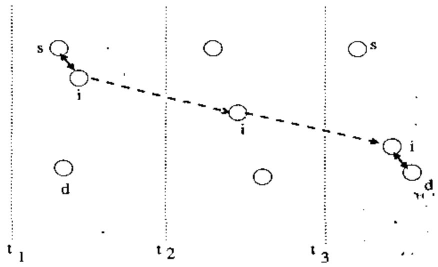

Fig. 3. Network of three nodes.

II. CONNECTIVITY-THROUGH-TIME CONCEPT

Let the graph $G(t) = (V, E(t))$ represent the connectivity of the network at time t such that node $v \in V$ represents a robot and edge $(u,v) \in E(t)$ represents that the nodes u and v are in direct wireless communication. At time t, a path from node s to d in $G(t)$ represents a multi-hop network connection since a message can be routed along nodes of the path. If this path persists for a time interval $[T_1, T_2]$ a message with the end-to-end delay of $T_2 - T_1$ can be successfully delivered from s to d. On the other hand, if there is no path from s to d in $G(t)$ for any t, it does not necessarily mean that a message cannot be delivered. To illustrate this consider Figures 1-3 which show a team of three mobile robots. Initially, the robots s and i are directly connected. Then i moves away from s and all three robots are disconnected. Finally, i moves within the range of d and hence is connected to it. A message sent from s at time t_1 can be sent to i initially where it can be buffered and then delivered to d at time t_3. Essentially, the movements of i are utilized to deliver the message to d.

To discuss the performance of a protocol that achieves such delivery we need to identify a reasonable performance criterion. Since the topology is dynamic, it is too weak to expect that a datagram be delivered from s to d only if they are connected at some time (as is done in TORA [8], for example). On the other hand, it is unreasonable to expect messages to wait indefinitely long in the network; if d is not reachable from s at all, flooding the messages could lead to inordinate amounts of datagrams being generated, thereby causing the denial of service between the nodes that are connected. To address the issue, the concept of *connectivity-through-time* was proposed in [10].

Let the topology changes occur at unique times, denoted in increasing order by $t_1, t_2, \ldots, t_k$ for $t_i \in [0, T]$. Note that $G(t)$ remains constant for all $t \in [t_i, t_{i+1})$ and is given by $G(t_i)$. We define that s and d are 0-*connected-through-time* for interval $[T_L, T_H]$ if they are connected in $G(t)$ for some $t \in [T_L, T_H]$. Consider $[T_L, T_H] \subset (t_{i-1}, t_{i+1})$ containing t_i. We define that s and d are 1-*connected-through-time* for interval $[T_L, T_H]$ if

(a) they are 0-connected through-time for $[T_L, T_H]$, or
(b) there exists a node v such that: (i) s and v are connected in $G(t_i)$, and (ii) v and d are connected in $G(t_{i+1})$.

The *time-path* in $[T_L, T_H]$ is represented by the composition of path from s to v in $G(t_i)$, followed by *time-edge* $(v; t_i, v; t_{i+1})$, and followed by path from v to s in $G(t_{i+1})$. The time interval $T_H - T_L$ is called the *hold-time* of the path. This definition is recursively applied to an interval containing more than one t_i's as follows. We define that s and d are *k-connected-through-time* for interval $[T_L, T_H]$ containing $t_1, t_2, \ldots t_k$ if they are

(a) 1-connected-through-time for $[T_L, t_1)$, and
(b) $(k-1)$-connected-through-time for $[t_1, T_H]$.

Then s and d are *connected-through-time* for interval $[T_L, T_H]$ if they are k-connected-through-time. We consider that each node v is connected to itself at all times through time-edges denoted by $(v; t_i, v; t_{i+1})$.

Intuitively speaking, if s and d are connected-through-time in $[0, T]$, a datagram from s can be delivered to d by transmitting along graph paths and buffering along the time-edges for a time period given by the hold-time. Under the conditions of infinite buffer sizes and bandwidths, the throughput of the network is related to the connectivity-through-time in that a message can be delivered from node s to d *if and only* if they are connected through time. There are two practical considerations in implementing the above approach. First, the nodes have finite buffers and packets cannot be indefinitely stored. Second, the transmission time is non-zero and could be significant for newly made connections. As a result, not all messages in the buffers may be delivered during the time a connection is available. We parameterize the packets delivery along the connectivity-through-time with two parameters, *time-to-live* and *minimum-connection time*. The first parameter specifies the time during which the current message is useful. For example, the location information of a moving robot is obsolete after certain time. So we delete the messages from the buffers after the expiry of their time-to-live values. Then packets with appropriate time-to-live value can be delivered along a time-path with sufficient minimum-connection time.

III. CTIME-PROTOCOL

The overall idea of this protocol is to track the connectivity and route the packets by suitably buffering them if there is no path to the destination. Each network node acts as a router in delivering the messages. The source nodes decompose the messages as UDP datagrams and then send them over the network. The received datagrams are reassembled at the destination. This protocol is specified by two variables *time-to-live* and *minimum-connection time* both determined empirically. Each packet is given the

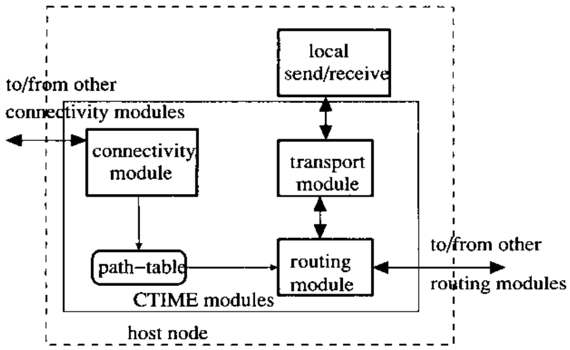

Fig. 4. Network of three nodes.

same time-to-live value. The minimum-connection time is assumed to be sufficient to clear the buffered packets.

The CTIME protocol is implemented using three components implemented as daemons as shown in Figure 4.

(a) *Connectivity Computation:* The direct and multiple-hop connectivity at each node is continually updated in response to the I-am-here datagrams broadcast periodically by each node.
(b) *Message Routing:* The messages are decomposed into UDP datagrams and routed by suitably buffering as needed.
(c) *Message Transport:* Packet losses, throughput rates and duplicates are handled by using a window-based mechanism akin to TCP but adapted to the current environment.

A. Connectivity Computation

Each node v maintains a list of *direct neighbors* $DN(v)$ that it is in direct contact with and list of *multiple hop neighbors* $MN(v)$ that it is connected via other nodes. Each node periodically broadcasts an *I-am-here* UDP broadcast packet, which is heard by all nodes within the direct range. This packet includes the list of all direct neighbors as well as multiple hop neighbors. Using these messages from the neighbors, each node computes its direct and multi-hop neighbors using the distributed transitive closure algorithm. This information is periodically updated as the I-am-here messages from other nodes are received. The counting to infinity problem is avoided by not sending the connectivity information to its original source as is usually done.

B. Routing

The message at a source node is decomposed into fixed sized datagrams, which are routed along the nodes. The datagrams are written to the local routing daemon. The routing module also receives packets from other nodes to be routed or buffered. The packets targeted to the local node are simply sent to the send/receive module. For other datagrams if a route to destination exits, then it is sent along the known path by writing it to the next node on the path to destination. That is, if the destination is directly connected, it is sent to it, and if not, it is written to the next node on the path to destination. If no path to the destination exists, then it is buffered locally if not already buffered and then is broadcast to all its immediate neighbors. When new connections are made, this module examines the list of buffered packets and routes them as above. Also, the buffered packets are periodically examined and those that outlived their time-to-live values are simply deleted.

C. Transport Method

The transport module at the source generates UDP datagrams from the message and keeps track of packets that have not been acknowledged. It maintains a buffer of unacknowledged packets and sends them to the router module at the appropriate rates as described below. It also resends the unacknowledged packets after a time out period.

A rate control mechanism is used for sending the packets to the appropriate nodes. We propose a UDP-based method using a simple window-based flow control strategy. Each node maintains a window-size w and window-time T_w to compute its throughput in terms of the number of packets sent during T_w. The preference of this method over TCP for transport is dictated by the following considerations:

(a) *High Physical-Layer Losses:* The usual TCP interprets the physical layer losses as congestion signals and reduces its throughput. In the current scenario, however, the opposite is needed: the throughput must be increased to account for packet loss.
(b) *Graceful disconnects:* Due to high rate of disconnects, TCP based method will wait for connection time out.
(c) *Application-Level Tuning:* Most TCP parameters are not available to be tuned to suit the application without the kernel modification. For example, it is not easy to select the congestion window size based on the current connection parameters.

At each source, we specify the throughput rate depending on the connectivity to destination. We collected throughput rates at the source and destination which showed a unimodal behavior (akin to those used in TCP models [7], [6]). In Figure 5, we plot the receiver throughput and packet loss as functions of sending rate while both robots are stationary. It clearly shows that maximum throughput is achieved in the vicinity of sending rate of 0.8Mbps. Below this value the losses are low but so is the throughput at the destination and above this value the losses increase rapidly thereby reducing the useful

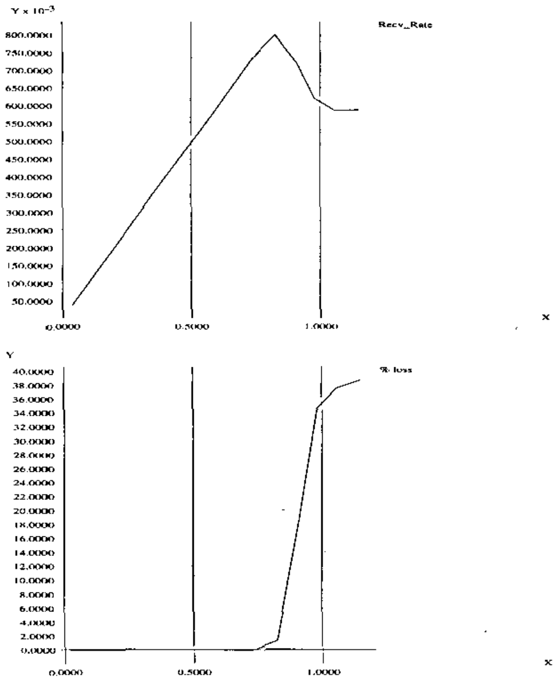

Fig. 5. Stationary nodes: Destination throughput (Mbps) and percentage loss versus source sending rate (along X-axis in Mbps) for direct connections in top and bottom plots, respectively.

throughput at the destination. It is interesting to note that the highest throughput is achieved under small but non-zero loss rate. When robots are in motion, there are somewhat higher losses and lower throughput. We choose the appropriate sending rate for direct connections based on whether the robots are moving or not.

When transmitting directly between two robots, we achieved comparable throughput with TCP byte-stream connection. However, there is a significant reduction in the overall throughput when packets are routed via other robots, because (a) same physical channel is used for two connections at the intermediate node thereby reducing the available raw bandwidth to at most half the peak; and (b) the overheads of routing introduce delays thereby further reducing the bandwidth. Note that TCP is not capable of sending packets using other nodes as routers. Generally TCP only needs to take care of the transport at the two ends, while the intermediate nodes in CTIME protocol have much more complicated transport controls to ensure reliable delivery.

In CTIME, we apply the direct sending rate (a) in transmissions to the destination when directly reachable , or (b) in broadcasting if the destination is not reachable. If the destination is reachable via multiple hops, the lower

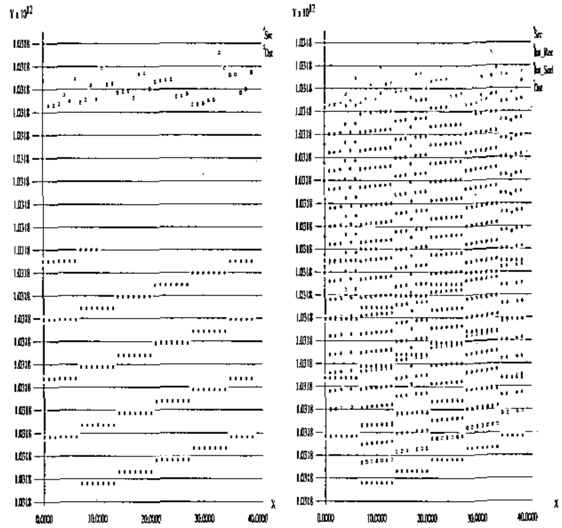

Fig. 6. Messages are delivered through time-connectivity. Packet number (X-axis) versus send/receive time.

sending rate is employed as per the observations shown in Figure ??.

Packet acknowledgments are sent from the destination toward the source to clear out the packets that have been received by the destination. The router modules examines the buffer upon receiving an acknowledgment packet and deletes the corresponding packet if it exists in the buffer.

IV. EXPERIMENTAL RESULTS

The protocol is implemented in C++ under Linux OS using socket-level programming. The testing is carried out on a team of four Mini ATRV mobile robots equipped with 802.11 wireless cards. We describe experimental results for a complicated scenario to illustrate the salient features of CTIME (more scenarios and details of implementation can be found at at www.cesar.ornl.gov/~nrao). Here messages are delivered between source and destination which are never connected to each other even via multiple hops. As shown in Figure 6, this scenario has three stages. First, the connection exists only between the source and intermediate node. This connection breaks in the second stage where the intermediate node is the only active node performing broadcasts. In the third stage, the connection between the intermediate node and destination comes up. In the left plot of Figure 6, the datagrams are sent multiple times from the source since the destination is not reachable at any time. In the right plot of Figure 6, the datagrams are shown to be received and retransmitted by the the intermediate node. The corresponding average throughput is shown in Figure 7, where the left plot shows

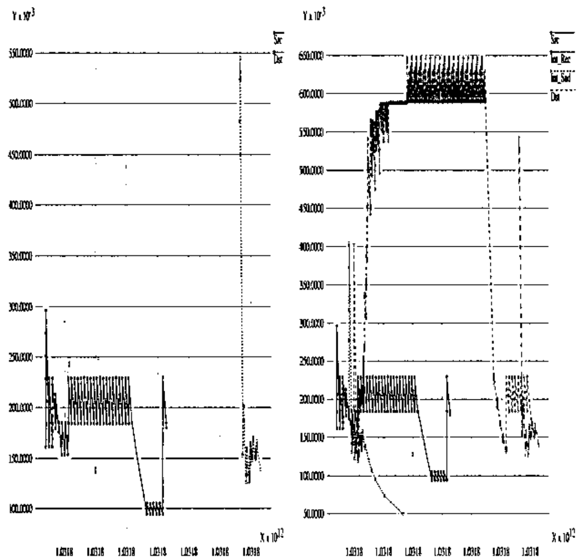

Fig. 7. Average throughput vs. time.

throughputs at source and destination only, and the right plot additionally shows the throughputs corresponding to reception and transmission at the intermediate node. Observe that throughput at the destination is almost the same as at intermediate node. The explanation for this observation is that the two hop connections never exist at the same time so that each of them has the exclusive bandwidth utilization at different times.

V. CONCLUSIONS

By utilizing the connectivity-through-time concepts we propose a class of protocols that exploit the robot movements to go beyond the traditional notions of connectivity. Our protocols enable the formation of ad hoc networks of mobile robots without the infrastructure of access points by utilizing the robots as routers. We implemented CTIME protocol based on these principles on RWI MiniATRV mobile robots using UDP with window-based flow control that is tuned to the nature of connections. There are several components of the proposed protocol and its implementation that require future work. On-line optimization of time-to-live parameters and smooth adaptation of the source sending rates would be of future interest. The presented protocol realizes a flat network in which the physical medium is shared by all nodes within the range. The intended application is for small teams of robots, typically less than 10. Such flat network with shared medium can severely limit the capacity of larger networks [5]. A hierarchical approach might be investigated for such applications. The open question here is to exploit the time-connectivity in such a hierarchical network. Another future direction is to make the protocol completely automatic so that various parameter values are adaptively learned.

VI. ACKNOWLEDGMENTS

This research is sponsored by the Defense Advanced Projects Research Agency under MIPR No. K153 and the Engineering Research Program of the Office of Science, U.S. Department of Energy, under Contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

VII. REFERENCES

[1] T. Balch and L. E. Parker, editors. *Robot Teams: From Diversity to Polymorphism*. AK Peter Pub., 2002.

[2] D. Braginsky and D. Estrin. Rumor routing algorithm for sensor networks. In *First Workshop on Sensor Networks and Applications*, 2002. to appear.

[3] D. Estrin, L. Girod, G. Pottie, and M. Srivastava. Intrumenting the world with wireless sensor networks. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, 2001.

[4] M. Grossglauser and D. Tse. Mobility increases the capacity of ad-hoc networks. *IEEE Transactions on Networking*, 2002. to be published.

[5] P. Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, 2000.

[6] F. P. Kelley. Mathematical modelling of the Internet. In *Proceedings of 4th International Congress of Industrial Applied Mathematics*, 1999.

[7] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, 2002.

[8] V. C. Park and M. S. Corson. A performance comparison of the temporally-ordered routing algorithm and ideal link-state routing. In *Proc. of INFOCOM'98*. 1998.

[9] C. Perkins, editor. *Ad Hoc Networking*. Addison-Welsey, 2001.

[10] S. Radhakrishnan, G. Racherla, N. Sekharan, N. S. V. Rao, and S. G.Batsell. DST- a routing protocol for ah-hoc networks using distributed spanning trees. In *Proc. of 1999 IEEE Wireless Communications and Networking Conference*. 1999.

[11] N. S. V. Rao, S. G. Batsell, and J. Jonassaint. Connectivity-through-time protocol for dynamic wireless networks. 2001. Department of Energy Invention Disclosure, ID 0893, S-96,637,2000.

[12] W. Stallings. *Wireless Communications and Networks*. Prentice-Hall, 2001.