WTRP—Wireless Token Ring Protocol

Mustafa Ergen, *Student Member, IEEE*, Duke Lee, Raja Sengupta, *Member, IEEE*, and Pravin Varaiya, *Fellow, IEEE*

***Abstract*—The wireless token ring protocol (WTRP) is a novel medium access control (MAC) protocol for wireless local area networks (WLANs). In contrast with IEEE 802.11 networks, WTRP guarantees quality of service (QoS) in terms of bounded latency and reserved bandwidth, which are critical in many real-time applications. Compared to 802.11, WTRP improves efficiency by reducing the number of retransmissions due to collisions, and it is more fair as all stations use the channel for the same amount of time. Stations take turns transmitting and give up the right to transmit after a specified amount of time. WTRP is a distributed protocol that supports many topologies, as not all stations need to be connected to each other or to a central station. WTRP is robust against single node failures, and recovers gracefully from multiple simultaneous faults. WTRP is suitable for interaccess point coordination in ITS DSRC, safety-critical vehicle-to-vehicle communications, and home networking, and provides extensions to other networks and Mobile IP.**

***Index Terms*—Home networking, IEEE802.4, IEEE802.11, intelligent transportation systems, medium access control, mission critical systems, quality of service (QoS), vehicle-to-vehicle communication, wireless token ring protocol (WTRP).**

I. Introduction

WIRELESS token ring protocol (WTRP) is a medium access control (MAC) protocol that provides delay and bandwidth [quality of service (QoS)] guarantees for applications on wireless *ad hoc* networks. In *ad hoc* networks, participating stations may join or leave the network at any time, creating a dynamic topology. Because of hidden terminals and a partially connected network topology, contention among stations using the 802.11 MAC protocol can cause severe throughput degradation if the load becomes high, which also results in large medium access times.

QoS efforts have often focused on network layer queuing and routing techniques [7], [8]. With an unreliable wireless medium, QoS must also be addressed at the data link layer. The IEEE 802.11 [9] in point coordination function (PCF) mode, HiperLAN [20], and Bluetooth [21] achieve bounded latency by having a central station poll the slave stations. Academic research has also focused on this centralized approach in which the network is managed from a central station [10], [11].

WTRP follows a distributed approach. Its advantages are robustness against single node failure and support for flexible topologies, in which nodes can be partially connected and are not connected to a master. Current wireless distributed MAC protocols such as the IEEE 802.11 [distributed coordination function (DCF) mode] and the ETSI HiperLAN do not provide QoS guarantees that some applications require. Moreover, the medium is not shared fairly among stations and access time can be arbitrarily long [14], [15].

As in the IEEE 802.4 [6] standards, WTRP is designed to recover from multiple simultaneous failures. One major challenge that WTRP overcomes is that of partial connectivity. To overcome this challenge, WTRP places management, special tokens, and additional fields in the tokens, and adds new timers. When a node joins a ring, WTRP requires the joining node to be connected to its prospective predecessor and successor. The joining node obtains this information by looking up its connectivity table. When a node leaves a ring, its predecessor in the ring finds the next available node to close the ring by looking up its connectivity table.

Partial connectivity also affects the multiple token resolution protocol (deleting all multiple tokens but one). In a partially connected network, simply dropping the token whenever a station hears another transmission is not sufficient. To delete tokens that a station is unable to hear, WTRP uses a unique priority assignment scheme for tokens. Stations only accept a token that has greater priority than the token the station last accepted. WTRP also has algorithms for keeping each ring address unique, to enable the operation of multiple nearby rings.

WTRP has applications where there is a need for the rapid establishment of a communication infrastructure with QoS guarantees [1]. WTRP was deployed in the Automated Highway System program [12] and the Berkeley Aerobot Project [17]. These applications impose stringent bandwidth, latency, and speed of failure recovery requirements on the medium access protocol. For example, the automated highway project involves of a platoon of up to 20 vehicles, and requires that data (approximately 100 bytes per vehicle for speed, acceleration, and coordination maneuvers) be transmitted periodically. WTRP meets the application requirements in terms of bounded delay, fair share of bandwidth to all stations in the network, and fast failure recovery.

The outline of the paper is as follows. The overall architecture of WTRP is described in Section II. Initial versions of WTRP are presented in [3]–[5]. The current version, partly presented in [2], changes the packet format to convey more information and perform robust and quick ring creation. This version is presented in Section III. The algorithm is now simpler and more efficient than in [4] as we show in Section IV. There are three modes of WTRP: kernel, user-space, and simulator. The main modules of these modes are described in Section V. A revised proof of the stability for the new WTRP is in Section VI. A simple analysis

Manuscript received October 18, 2002; revised August 2, 2004. This work was supported in part by ONR and the California Department of Transportation.

M. Ergen, D. Lee, and P. Varaiya are with the Department of Electrical Engineering and Computer Science, University of California, Berkeley, CA 94720 USA (e-mail: ergen@eecs.berkeley.edu; duke@eecs.berkeley.edu; varaiya@eecs.berkeley.edu).

R. Sengupta is with the Department of Civil Engineering, University of California, Berkeley, CA 94720 USA (e-mail: sengupta@ce.berkeley.edu).

Digital Object Identifier 10.1109/TVT.2004.836928

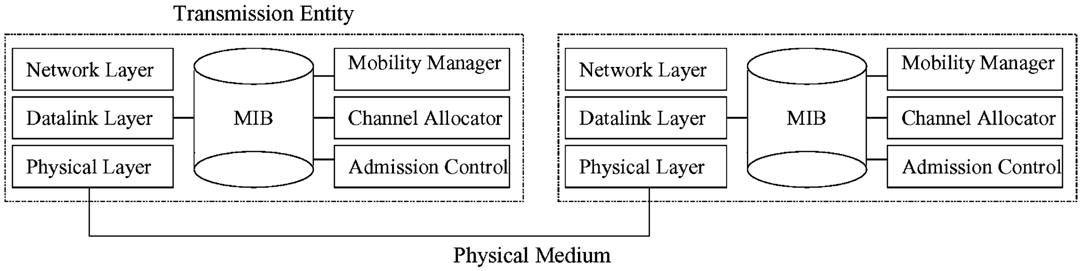


Fig. 1. System architecture.

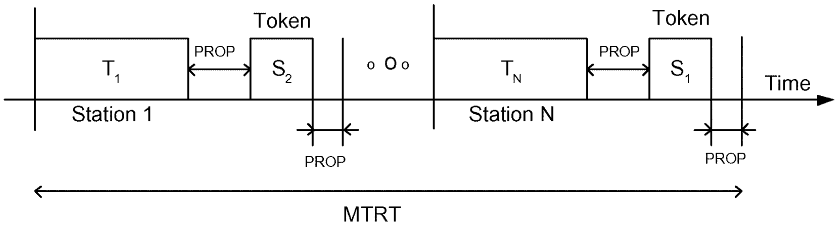


Fig. 2. Timing diagram for WTRP. PROP is signal propagation time.

of the saturation throughput performance is in Section VII. Results from an extensive simulation comparing the performance of a WTRP ring and the corresponding IEEE 802.11 protocol are given in Section VIII. Possible extensions and some conclusions are summarized in Section IX.

II. Overall System Architecture

Fig. 1 places WTRP within the overall system architecture. In addition to the communication stack including the data link layer where WTRP is located, we need the Mobility Manager, Channel Allocator, Management Information Base (MIB), and Admission Control Manager. We assume that there are multiple channels, and that different rings are on different channels, assigned by the channel allocator (Section II-B).

A. MAC

The core of WTRP is located in the MAC module. The main function of MAC is to control the timing of transmissions by different nodes to increase the chance of successful transmission.

In WTRP, the MAC layer performs ring management and timing of the transmissions. Ring management involves:

1) ensuring that each ring has a unique ring address;
2) ensuring that one and only one token exists in a ring;
3) ensuring that the rings are proper;
4) managing joining and leaving operations.

B. Channel Allocator

If there are a number of nearby token rings, their efficiency can be increased by achieving spatial reuse through sensible channel allocation.[1]

Finding the channel allocation that maximizes network capacity is difficult in any large deployment of mobile nodes. We propose a distributed approach. In our implementation, the channel allocator is local to each station, and accesses network topology information through the MIB. Each node decides which channel to join using information in the token, which includes the number of nodes (NoN) in the ring. If NoN reaches the maximum value, this is an indication for the nodes outside the ring to shift to the next channel and search for another ring.

C. Mobility Manager

The Mobility Manager decides when a station should join or leave the ring. The Mobility Manager thus solves a mobile hand-off problem. When a mobile node is drifting away from its current ring and into the vicinity of another, at some threshold the Mobility Manager decides to move to the next ring. The level of connection of a node to a ring is found in its connectivity table, described in Section III.

D. Admission Control

The Admission Control Manager limits the number of stations that can transmit on the medium. This ensures bounded latency and reserved bandwidth for stations that already have permission to transmit on the medium. There is an Admission Control Manager in each ring. The Admission Control Manager may move with the token but does not have to move every time the token moves. The Admission Control Manager periodically solicits other stations to join if "resources" on the ring are available.[2] Only stations that require fewer resources than those available in the ring may join.

E. Policer

The policer monitors the traffic generated by the application. It throttles the application when more traffic than reserved is produced. In WTRP, because the token holding timer polices

[1] Spatial reuse is central to wireless cellular telephony: the same channel (or set of channels) can be reused in regions that are separated by a sufficient distance, measured in terms of the signal-to-interference ratio.

[2] The "resource" of the token ring is defined as follows. MAX_MTRT [maximum token rotation time (MTRT)] is the maximum latency that each station in the ring can tolerate. RESV_MTRT is the sum of token holding times (THT) of each station. MAX_NoN is the maximum NoN allowed in the ring. The Admission Control Manager has to ensure the inequalities: $\mathrm{RESV\_MTRT} < \mathrm{MAX\_MTRT}$ and $\mathrm{NoN} < \mathrm{MAX\_NoN}$. A detailed description is in [1].

FC	RA	DA	SA	NoN	GenSeq	Seq	
1	6	6	6	2	4	4	bytes

Fig. 3. Token frame.

Connectivity Table of D

My Table		Other Table
Seq=1	Addr=E	Addr=H
Seq=2	Unknown	Addr=G
Seq=3	Unknown	
Seq=4	Addr=B	
Seq=5	Addr=C	

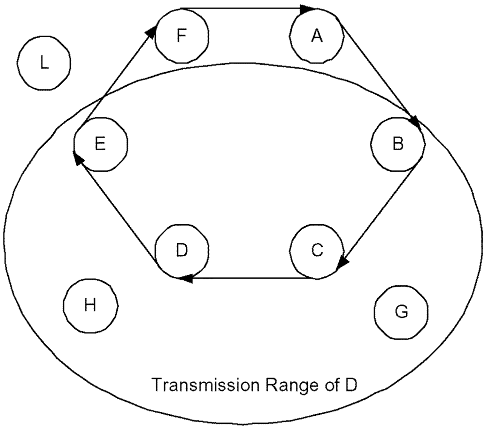


Fig. 4. Connectivity table.

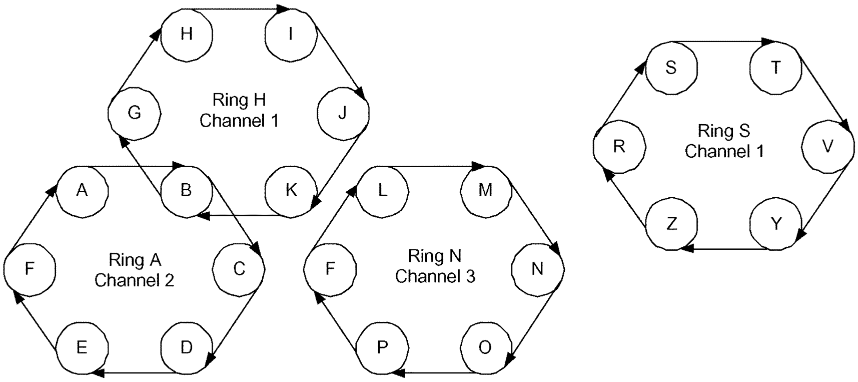


Fig. 5. Multiple rings.

the traffic generated by a station, no special policer module is necessary.

F. MIB

The MIB holds all the information that each management module needs to manage the MAC module. Most of this information is collected by the MAC module and stored there. However, some of the information may need to be communicated. This is gathered and refreshed by the SNMP agent. Details on this are still being investigated.

III. Description

Transmission proceeds in one direction along the ring. Each station has a unique successor and predecessor. We use Fig. 2 to analyze the protocol. Suppose N stations are in the ring. Let T_n be the time during which station S_n transmits when it gets the *token*, and before it releases the token. So T_n can range from 0 to THT. S_n first sends its data during T_n and if enough time is left, the station invites nodes outside the ring to join. Fig. 3 shows the token frame.

Frame Control (FC) identifies the type of packet, namely $\{$*token, solicit-successor token, set-predecessor token, claim-token, set-successor token, token-deleted token, data*$\}$. *Source address* (SA) is the station where the packet originates. *Destination address* (DA) determines the destination station. *Ring address* (RA) is the ring to which the token belongs. *Sequence number* (Seq) is initialized to zero and incremented by every station when it passes the token. *Generation sequence number* (GenSeq) is initialized to zero and incremented at every rotation of the token by the creator of the token. *Number of nodes* (NoN) in the token frame is calculated by taking the difference of sequence numbers in one rotation.

Ring owner is the station with the same MAC address as the ring address. A station can claim to be the ring owner by changing the ring address of the token that is being passed around. In Fig. 5, the ring address of each of four rings is the address of one of its stations. The uniqueness of the MAC address allows the stations to distinguish between messages coming from different rings.

To ensure that the ring owner is present in the ring, when the ring owner leaves the ring, the successor of the owner claims the ring address and becomes the ring owner. The protocol deals with the case in which the ring owner leaves the ring without notifying the rest of the stations in the ring, as follows: The ring owner updates the generation sequence number of the token every time it receives a valid token. If a station receives a token

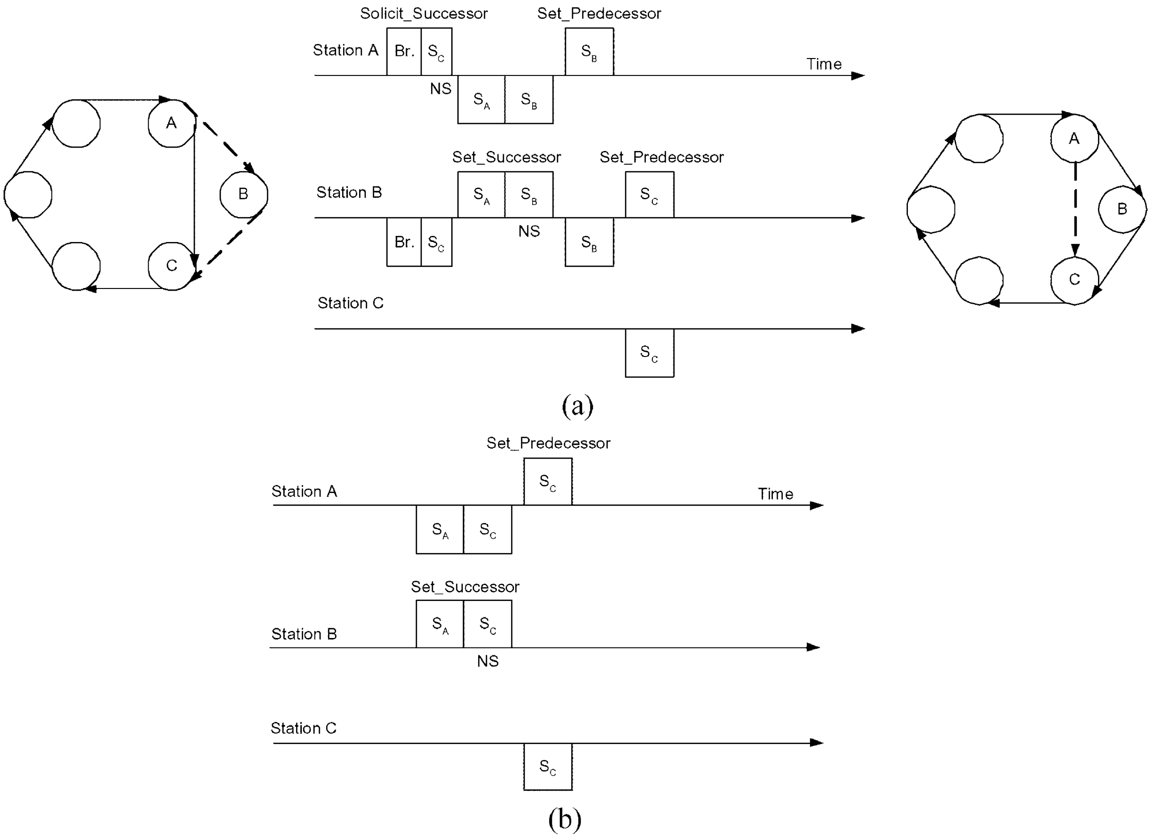


Fig. 6. Management procedures. (a) Joining. (b) Exiting.

without its generation sequence number updated, it assumes that the ring owner is unreachable and elects itself to be the ring owner.

Multiple ring management is left open and cited in Section IX and [1]. It is possible for a station to belong to more than one ring or to listen to more than one ring.

Successful token transmission relies on implicit acknowledgment. An implicit acknowledgment is any packet heard after token transmission that has the same ring address as the station. Another acceptable implicit acknowledgment is any transmission from a successive node regardless of the ring address in the transmission. A successive node is a station that was in the ring during the last token rotation. In other words, successive stations are those present in the local connectivity table.

Each station resets its timer (*idle_timer*) whenever it receives an implicit acknowledgment. If the token is lost in the ring, no implicit acknowledgment will be heard in the ring, and the *idle_timer* will expire. When the *idle_timer* expires, the station generates a new token, thereby becoming the owner of the ring.

Multiple token resolution (to delete all tokens but one) is based on priority. The generation sequence number and the ring address define the priority of a token. A token with a higher generation sequence number has higher priority. When the generation sequence numbers of tokens are the same, ring addresses of each token are used to break the tie. The priority of a station is the priority of the token that the station accepted or generated. When a station receives a token with a lower priority than itself, it deletes the token and notifies its predecessor without accepting the token. With this scheme, it can be shown that the protocol deletes all multiple tokens in a single token rotation provided no more tokens are generated [1], [3].

Connectivity manager resident on each node tracks transmissions from its own ring and those from other nearby rings. By monitoring the sequence number of the transmitted tokens, the connectivity manager builds an ordered local list of stations in its own ring and an unordered global list of stations outside its ring (see Fig. 4).

Ring recovery mechanism is invoked when the monitoring node decides that its successor is unreachable. In this case, the station tries to recover from the failure by forming the ring again. WTRP tries to reform the ring by excluding as few stations as possible. Using the connectivity manager, the monitoring station is able to quickly find the next connected node in the transmission order. The monitoring station then sends the *set_predecessor* token to the next connected node to close the ring.

Joining a ring is dynamic and handled one at a time, provided that the token rotation time (sum of THT per node, plus overhead such as token transmission times) does not grow unacceptably large upon addition of a new node. As illustrated in Fig. 6(a), suppose station B wants to join the ring. Let us also say that the admission control manager on station A broadcasts (Br.) to other nodes to join the ring by sending out a *solicit_successor* that includes the address of the successor (C) of A. The admission control manager waits for the duration of the response window for interested nodes to respond. The response window is the window of opportunity for a new node to join the ring. The response window is divided into slots of the du-

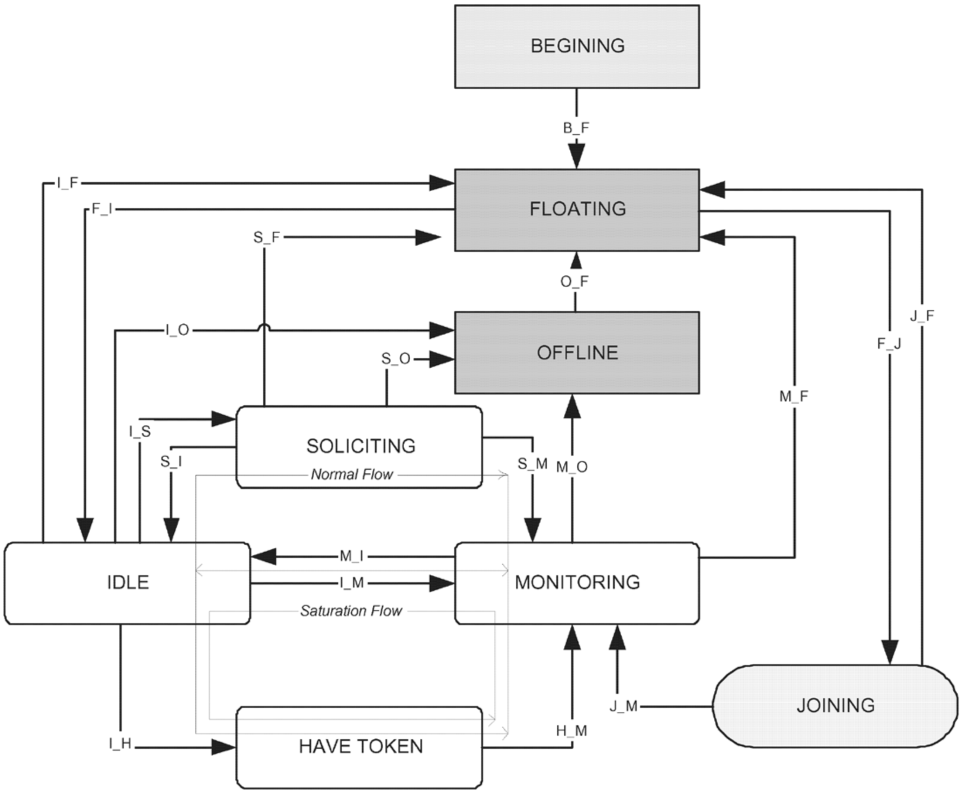


Fig. 7. Main flow.

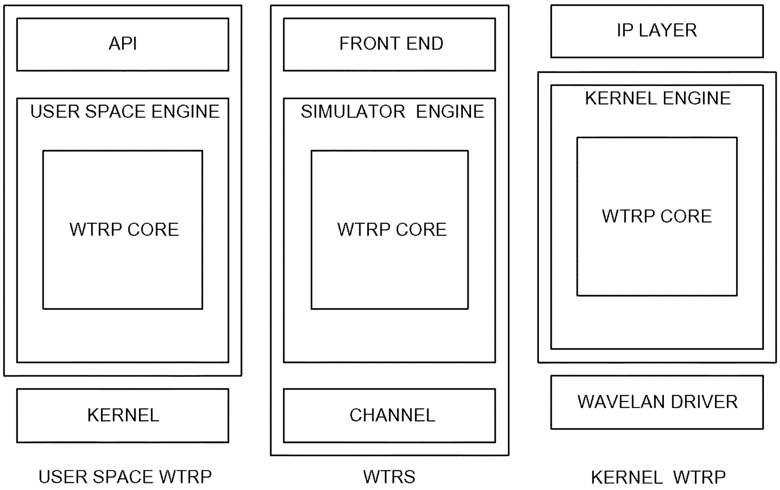


Fig. 8. Implementation overview.

ration of the *set_successor* transmission time. When a node like B that wants to join the ring, hears a *solicit_successor* token, it picks a random slot and transmits a *set_successor* token. When the response window passes, the host node A can decide among the slot winners. Suppose that B wins the contention. Then the host node passes the *set_predecessor* token to B, and B sends the *set_predecessor* to node C, the successor of the host node A. The joining process concludes.

Leaving the ring can be done with or without notification. Suppose station B wants to leave the ring as in Fig. 6(b). B first waits for the right to transmit. Upon receipt of the right to transmit, B sends the *set_successor* packet to its predecessor A with the MAC address of its successor, C. If A can hear C, A tries to connect with C by sending a *set_predecessor* token. If A cannot hear C, A will find the next connected node, in the transmission order, and send it the *set_predecessor* token. If B fails, then station A recognizes the failure when it does not get the implicit acknowledgment and tries to close the ring.

Interference is reduced by including NoN into the token packet. When a station detects a ring, it examines the NoN field. If NoN is set to MAX_NoN, the station changes its channel and searches for another ring. Otherwise, the station either waits for a *solicit_successor* token to become a ring member or changes its channel to search for another ring. As a result, a new station never interferes with the ring.

IV. Finite State Machine

We now describe details of the finite state machine shown in Fig. 7. The states are $\{$*Beginning, Floating, Offline, Joining,*

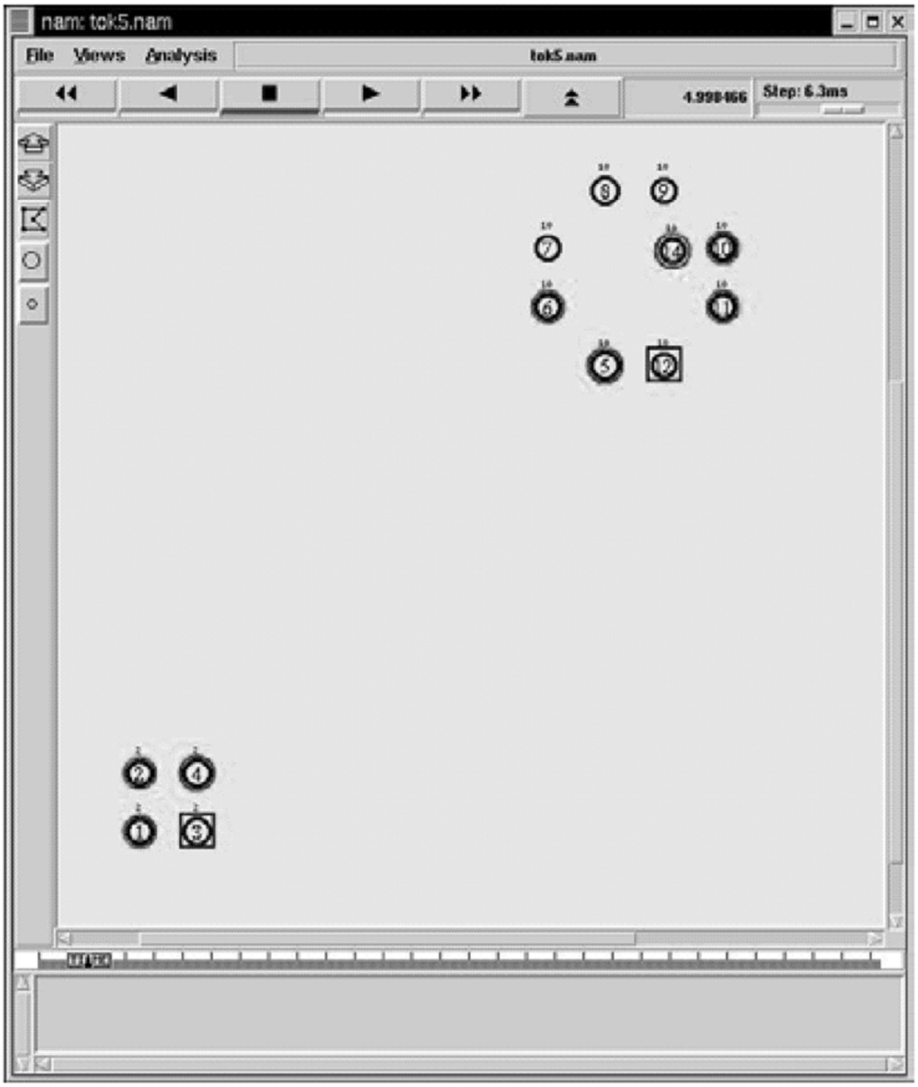


Fig. 9. Snapshot of visualization of the simulation.

Soliciting, Idle, Monitoring, Have Token}. To cope with "mobility," FSM has *joining* and *soliciting* states in which inviting and joining processes are handled. "Interference avoidance" to other rings is done by introducing *floating* and *offline* states, wherein a station suspends transmission in these states and waits to join a ring. "Collision avoidance" in the same ring is eliminated by the *idle* state, wherein a station suspends transmission until it gets the *token*. "Equal bandwidth share" is controlled by *have token* state wherein the station transmits packets as long as this is allowed. *Monitoring* state is for "guaranteed transmission," wherein a station checks its transmission and retransmits in case of a failure. We describe the states further in the remainder of the section. A full explanation can be found in [1].

STATES

A. Beginning State

Beginning state is a virtual state representing the start of the protocol. There is only one transition and the station directly goes to the *floating* state.

B. Floating State

Floating state is the state in which a station resets its parameters and waits to join a ring. When a station passes to the *floating* state, it resets its station parameters, cleans up its packet queues, and initializes the *claim_token_timer*.

A station passes to the floating state at the beginning and when there is a failure in the ring. When the station is *self_ring*[3] and in the *idle* state, it goes to *floating* state when it detects another ring. If the station does not get the token in the *idle* state or can not join a ring, *idle_timer* expires and it goes to *floating* state. If the station detects a ring in the *floating* state, it waits to be invited by the ring and suspends its transmission in order not to interfere with the ring transmission. If the station does not detect a ring, *claim_token_timer* expires and it goes to the *idle* state and creates a *self_ring*. If the station gets a *solicit_successor* token, and if it wants to join a ring, the station sends a *set_successor* token and goes to *joining* state.

[3] "Self-ring" is defined for a station when its successor and predecessor are itself.

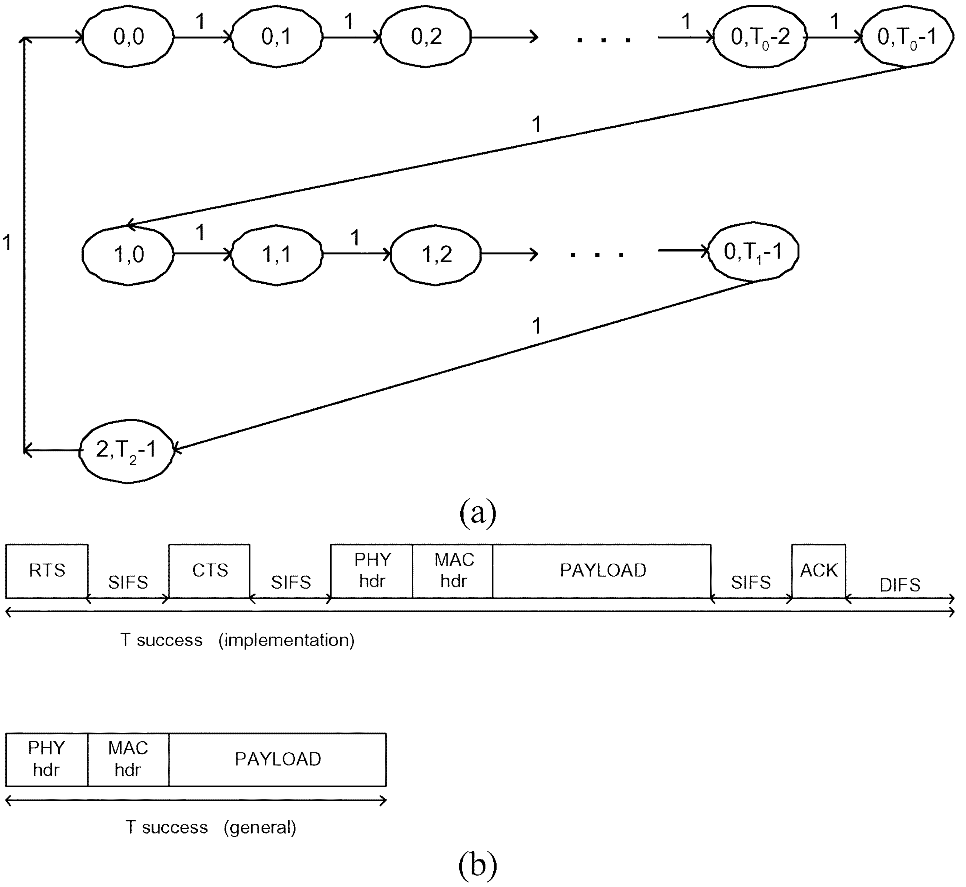


Fig. 10. Markov model and data frames. (a) Markov chain model. (b) T_s for implementation and general.

C. Offline State

When a station goes to the *offline* state, it reinitializes the station, clears the packet queues, and adds *offline_timer* to the scheduler. Since *offline_timer* is twice the *max_token_rotation_time*, the wait period in the *offline* state gives sufficient time to the former ring members to realize that the station is out of the ring. This prevents the station from joining a ring before the ring is closed. A station goes to the *offline* state if it does not belong to a *self_ring*, or it detects another ring, or it joins a ring but fails to pass the token to its successor. In the *offline* state, a station waits and does nothing until the *offline_timer* expires. From the *offline* state, a station only goes to the *floating* state.

D. Joining State

When a station goes to *joining* state, it initializes the *contention_timer*. A station goes to *joining* state only from *floating* state. When a station receives *solicit_successor* and decides to join the ring, the station sends *set_successor* and goes to the *joining* state. If the station receives *set_predecessor*, this means that joining is successful. Thereupon, the station sends *set_predecessor* and passes to the *monitoring* state. If the station does not get *set_predecessor*, *contention_timer* expires which means a failure in joining and the station goes back to the *floating* state.

TABLE I
FHSS SYSTEM PARAMETERS AND ADDITIONAL PARAMETERS USED TO OBTAIN NUMERICAL RESULTS

packet payload	8184bits
MAC header	272 bits
PHY header	128 bits
ACK	112 bits + PHY header
RTS	160 bits + PHY header
CTS	112 bits + PHY header
Channel Bit Rate	1 Mbit/s
Propagation Delay (δ)	1 μs
WTRP Slot Time	488 μs
802.11 Slot Time	50 μs
SIFS	28 μs
DIFS	128 μs
H	17
M	$N * (H + 1)$

E. Soliciting State

When a station goes to the *soliciting* state, the station initializes *solicit_wait_timer* and sets the *num_node* attribute of the station to $\mathrm{MAX\_NoN} + 1$ to suspend transmission of other stations if it is not a *self_ring*. When the station is in the *idle* state and receives the *token*, it checks its queues. If they are empty, it decides to send *solicit-successor*. If the decision is positive, then it sends *solicit_successor* token and goes to the *soliciting* state. If the station is *self_ring*, it adds *solicit_successor_timer*. When the timer expires, it sends *solicit_successor* token and goes to the soliciting state.

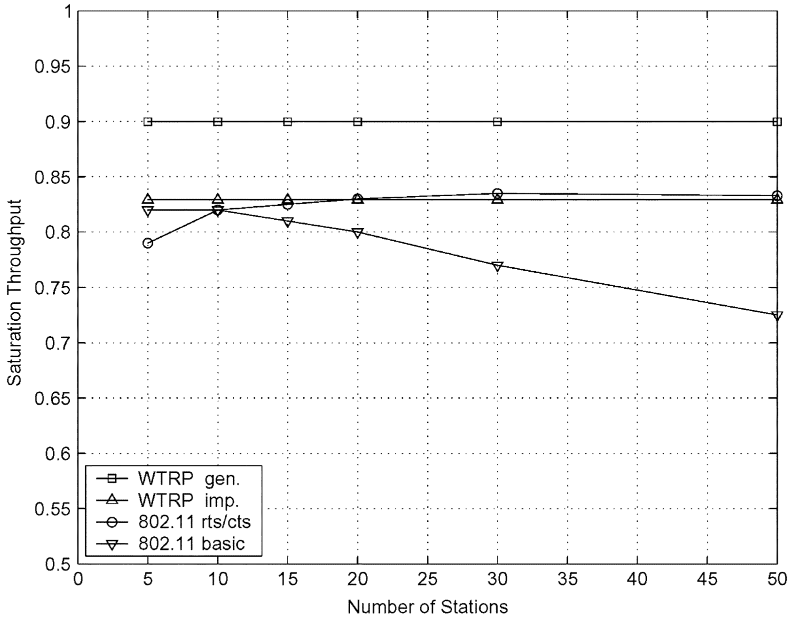


Fig. 11. Saturation throughput.

If a station receives *set_successor* in the *soliciting* state, this means that there is a station responding to its *solicit_successor*, and so the station sends *set_predecessor* and goes to the *monitoring* state. If there is no response to the invitation, *solicit_wait_timer* expires and the station goes to the *idle* state if it is *self_ring*, or *monitoring* state if it is not *self_ring*.

F. Idle State

When the station goes to *idle* state, it adds *idle_timer* and *inring_timer* if it is not a *self_ring*; otherwise it adds only *solicit_successor_timer*. When the station passes the *token* to its *successor*, it goes to the *monitoring* state to listen for *implicit_ack* that is a transmission from its successor to the successor of its successor. After hearing the *implicit_ack*, the station goes back to the *idle* state. When the station gets the *token* with higher priority (lower priority tokens are deleted by sending *token_deleted* token) and packet queues are nonempty, the station goes to *have_token* state. If the packet queues are empty, it decides whether or not to send *solicit_successor*. If the decision is positive, it goes to *soliciting* state period. Otherwise, the station passes the *token* and goes to *monitoring* state. If the station receives *set_successor* token that means a station is leaving, the station sends *set_predecessor* and goes to the *monitoring* state. If the station wants to leave, it sends *set_successor* and goes to the *offline* state.

G. Monitoring State

When the station goes to the *monitoring* state, it resets *num_token_pass_try* and adds *token_pass_timer*. *Monitoring* state is to monitor the medium to make sure that the transmission is successful. *Implicit_ack* is used to decide if a transmission is successful. If there is a retransmission of the same token, the station sends *token_deleted* token. If the transmission is successful, the station goes to the *idle* state.

H. Have Token State

When the station goes to the *have_token* state, it initializes *token_holding_timer*. A station passes to *have_token* state only from the *idle* state if it has a packet to send. When a packet is transmitted, transmission handler is called. Transmission handler checks for the packet queues and transmits if those are nonempty. When the queues are empty or the *token_holding_timer* is expired, the station passes the *token* to its successor and goes to the *monitoring* state.

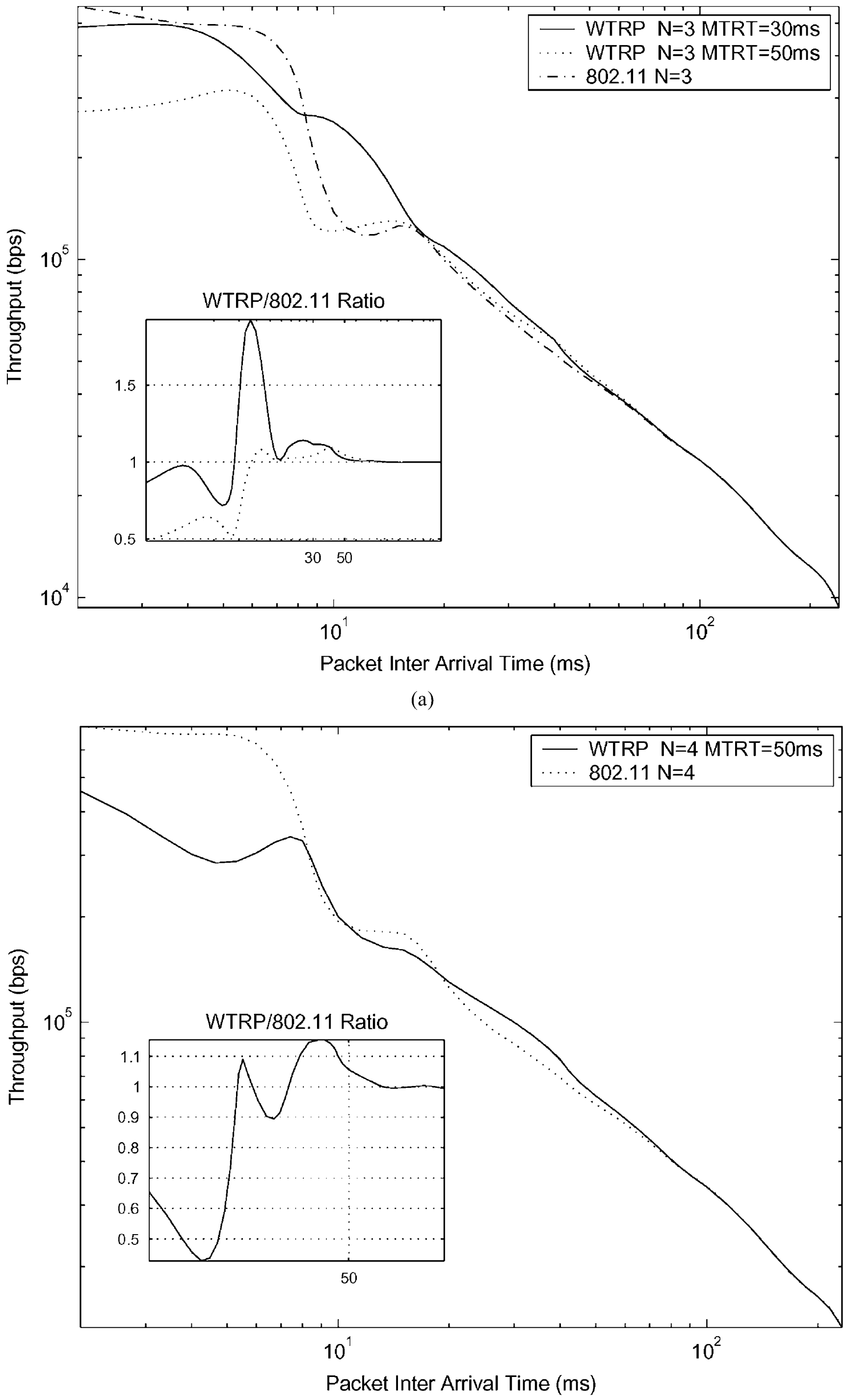
Operating Types

I. Normal Operating Flow

The station makes certain state changes when it is operating in *normal operating flow*. This is the flow when there is no joining process, which means either the ring is full or there is no station outside of the ring. When the station gets the *token* in the *idle* state, it goes to *have token, soliciting*, and *monitoring* state, when there is a packet to send, when it decides to send *solicit_successor* or when the packet queues are empty and *solicit_successor* decision is negative, respectively. The station goes back to the *idle* state from the *monitoring* state when it gets the *implicit_ack.*

J. Saturation Operating Flow

The station operates in saturation condition when there are always packets to send. In this case, the station passes to the *have_token* state from the *idle* state and keeps sending packets until the *token_holding_timer* expires. The station passes to the *monitoring* state after passing the *token* to its successor, and goes back to the *idle* state after receiving *implicit_ack*, as seen in Fig. 7. The difference between *normal* and *saturation* operating flow is that the latter does not send *solicit_successor* and state transitions always follow in order the *idle*, *have token* and *monitoring*, whereas the former does not follow an ordered state transition.



(a)

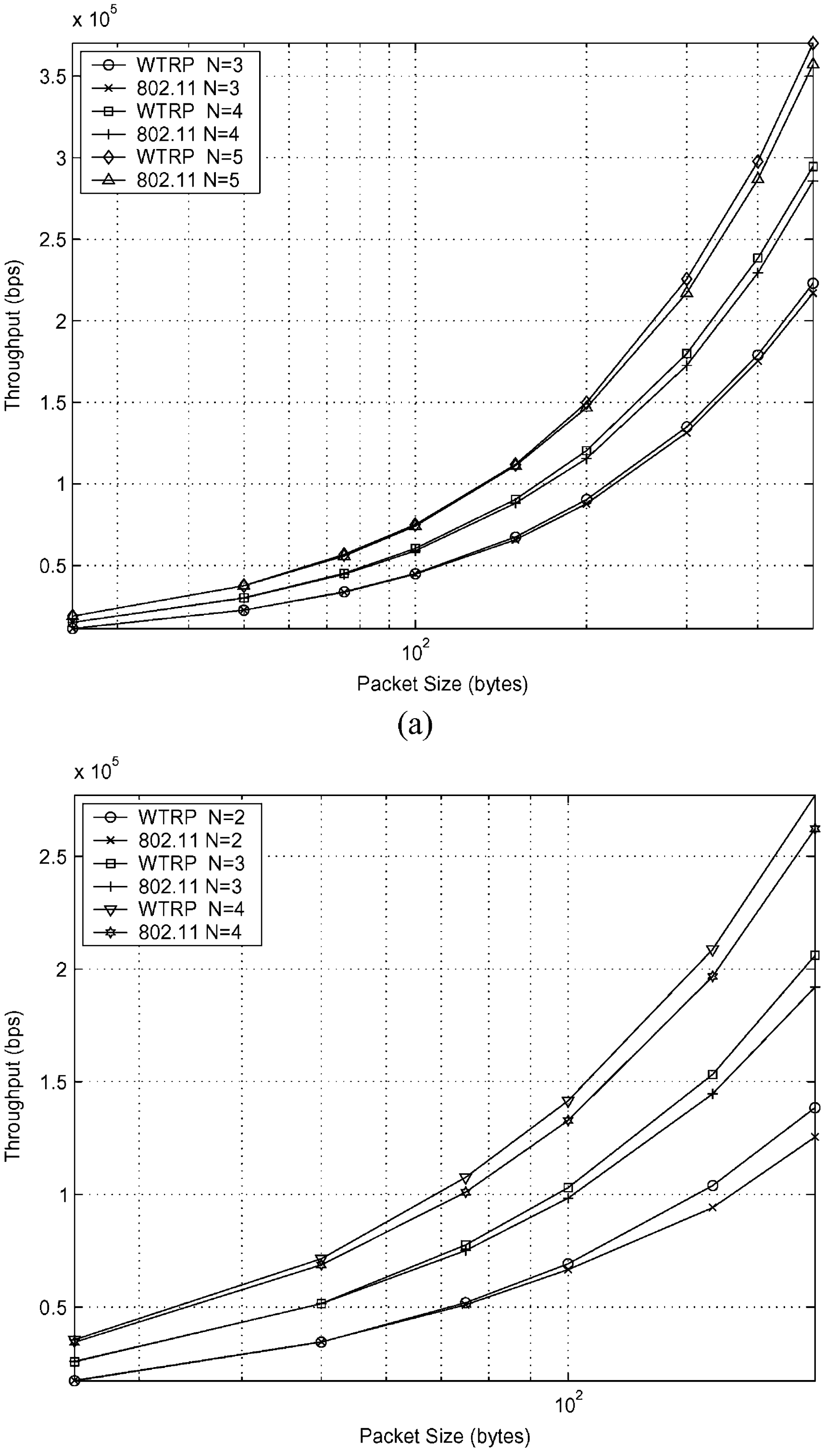
(b)

Fig. 12. Variable packet generating rate with different size of senders. (a) Three senders. (b) Four senders.

V. Implementation Overview

WRTP is deployed in three modes: {*Simulator, User-Space, and Kernel Implementations*}. All implementations share a unified codebase. The core of the protocol is the same for all three implementations, each of which introduces certain interfaces to the core as seen in Fig. 8. This unified codebase is made possible by the implementation of Linux kernel libraries for packet manipulations and event scheduler in user space. Detailed information about the implementation is in [1] and the open source code is in [23].

Ad hoc networks can scale to large configurations. Design and development costs for such systems are reduced by efficient techniques for evaluating design alternatives and predicting their impact on overall system performance. Simulation is the most common performance evaluation technique for such systems. The WTRP simulator reduces design and development



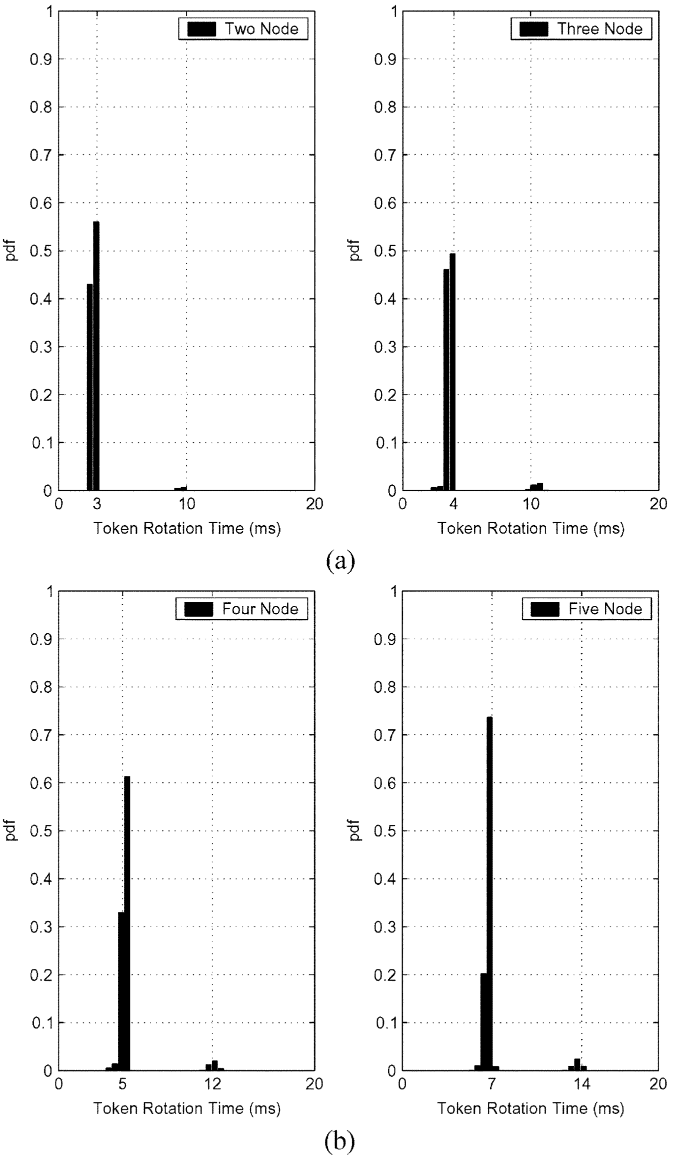
(a)

(b)

Fig. 13. Variable packet size with different MTRT. (a) $\mathrm{MTRT} = 30$ ms. (b) $\mathrm{MTRT} = 50$ ms.

cost by providing an easy path for migrating simulation models to operational software prototypes and support for visual and hierarchical model design. The WTRP simulator (see Fig. 8) allows debugging and simulating the WTRP core shared by all implementations. A snapshot of visualization of a scenario in the simulator is seen in Fig. 9.

The importance of the user-space implementation (Fig. 8) is its platform independence. In a managed network, applications can reduce the frequency of packet collisions by sending the UDP packet using the UDP interface. In this case, WTRP is used as a transmission scheduling method rather than for medium access control.



(a)

(b)

Fig. 14. Token rotation time distribution—four and five senders. (a) Two and three senders. (b) Four and five senders.

The Linux kernel module is developed for kernel version 2.2.19, and it is inserted between the IP and WaveLAN libraries as seen in Fig. 8 to work as WaveLAN driver.

VI. Proof of Stability

A. Introduction

In this section, we show that if after some time t there are no transmission losses and topological changes, and if stations do not voluntarily go into the *offline* state, then at some time $s > t$, the algorithm will lead to a stable state in which all stations cluster into rings. Moreover, $|t - s|$ can be bounded in terms of the various timer constants and the number of stations.

The following is a brief outline of the proof. More details are in [1] and [3]. We first prove that interference is reduced after the creation of the first ring, if only that ring is allowed to transmit and all equivalent tokens are deleted by the multiple token resolution protocol.

Next we show that the ring becomes stable in a finite time. Finally, we show that in any ring, the bijection representing the correct relationship between predecessor and successor increases monotonically to the maximum number of nodes in finite time. When all bijections converge, all rings, operating in different channels, are operating correctly.

B. Assumptions

1) No transmission error occurs after t.
2) Topology remains constant after t.
3) Stations do not voluntarily go out of a ring (into the *offline* state) after t.
4) If station x is in the transmission range of station y, y is in the transmission range of x.
5) If station x is in reception range of station y, y is in the reception range of x.

C. Constants

1) Maximum token rotation time (MTRT): In the proof, MTRT is the maximum time it takes for a token to visit a station twice.
2) IDLE_TIME: amount of time that a station waits before regenerating a token when the medium is quiescent; $\mathrm{IDLE\_TIME} \geq \mathrm{MTRT}$.
3) INRING_TIME: amount of time that a station waits without receiving any acceptable token before going into the *offline* state. $\mathrm{INRING\_TIME} \geq \mathrm{IDLE\_TIME} \geq \mathrm{MTRT}$; INRING_TIME < 2IDLE_TIME.
4) CLAIM_TOKEN_TIME: amount of time that a station waits when the station is in the *floating* state.
5) MAX_NoN: maximum number of nodes allowed in a ring.

D. Definitions

1) $r_i(t)$: set of nodes in ring i.
2) $t_i(t)$: set of tokens in $r_i(t)$.
3) Two tokens are said to be equivalent if their ring addresses are the same.
4) $x \cdot \mathrm{NS}(t)$: MAC address of the station to which station x forward tokens.
5) $x \cdot \mathrm{PS}(t)$: MAC address of the station from which station x accepts tokens.
6) $|V(t)|$: number of stations at time t.

E. Network

1) For stations x and y, if $x \cdot \mathrm{NS} = y, y \cdot \mathrm{PS} = x$ and if y can receive and accepts *normal* tokens from x, we say that x has the bijection with y.
2) $S = \{\langle x \cdot y\rangle | x \text{ has the bijection with } y\}$
3) A set $r_i(t)$ of stations is a *ring* if for all $x \in r_i(t), x$ has the bijection with its successor $y \in r_i(t)$.

F. Proof

1) Out-Ring Interference Cancellation:

Lemma 1: When a station hears a transmission from another ring, it suspends its transmission.

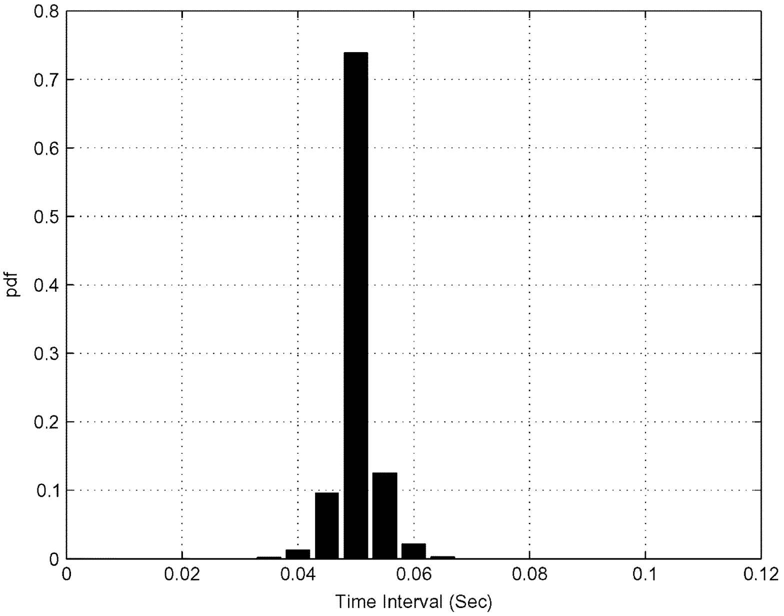


Fig. 15. A pdf of jitter.

Proof: If a station hears a transmission from another ring, it goes to *floating* or *offline* state when it is in a self-ring or normal ring, respectively. ■

2) In-Ring Interference Cancellation:

Lemma 2: Consider a token p at time $t = t_0$, and let the path taken by p be $\langle(x_0,t_0),(x_1,t_1),(x_2,t_2),\ldots,(x_m,t_m)\rangle$, in which t_n is the time that p visits station x_n, and $t_{i+1} > t_i$. If $x_i = x_j$, with $0 \leq i < j \leq m$ and $t_j - t_i < \mathrm{MTRT}$, then there is k, with $i \leq k \leq j$, such that x_k owns p.

Proof: Assume in contradiction that $x_i = x_j, t_j - t_i < \mathrm{MTRT}$, but there is no owner x_k of token p with $i \leq k \leq j$. Then the generation sequence numbers of the token when it arrives at x_i and when it arrives at x_j are the same, because only the owner of the token can modify the generation sequence number.

Also, x_j could not have been in the *offline* state at any time after t_i because, by the protocol, x_i cannot rejoin another ring after exiting a ring for one MTRT (more precisely, exiting of the *offline* state for one MTRT). Because a station cannot receive a token in the *offline* state, it could not have received p before time $t_i + \mathrm{MTRT}$. Thus, x_i could not have been in the *offline* state after time t_i. The priority of a station can only increase while it is not in the *offline* state and so token p could not have survived station x_j. ■

Lemma 3: If there are no multiple equivalent tokens at time t, there are no multiple equivalent tokens at time $s > t$.

Proof: Assume in contradiction that there are multiple equivalent tokens at $s > t$. By assumption, a station cannot generate multiple equivalent tokens from transmission errors. So the station must have generated a token while a token that it has previously generated is still in the ring. But $\mathrm{IDLE\_TIME} > \mathrm{MTRT}$, and a token dies if it does not visits its owner for MTRT. Thus the station could not have generated the equivalent token. ■

Lemma 4: No multiple equivalent tokens exist at time $t + 2\mathrm{MTRT}$.

Proof: All surviving equivalent tokens go through the token owner within one MTRT. After one MTRT, the owner remembers the highest priority token among them. Within the next MTRT, all or all but one equivalent token will be deleted, as the owner will not pass any token with priority lower than the highest priority token that it received.

If the token owner leaves the ring at any time, all tokens will be deleted since the owner cannot return to the ring in less than one MTRT. ■

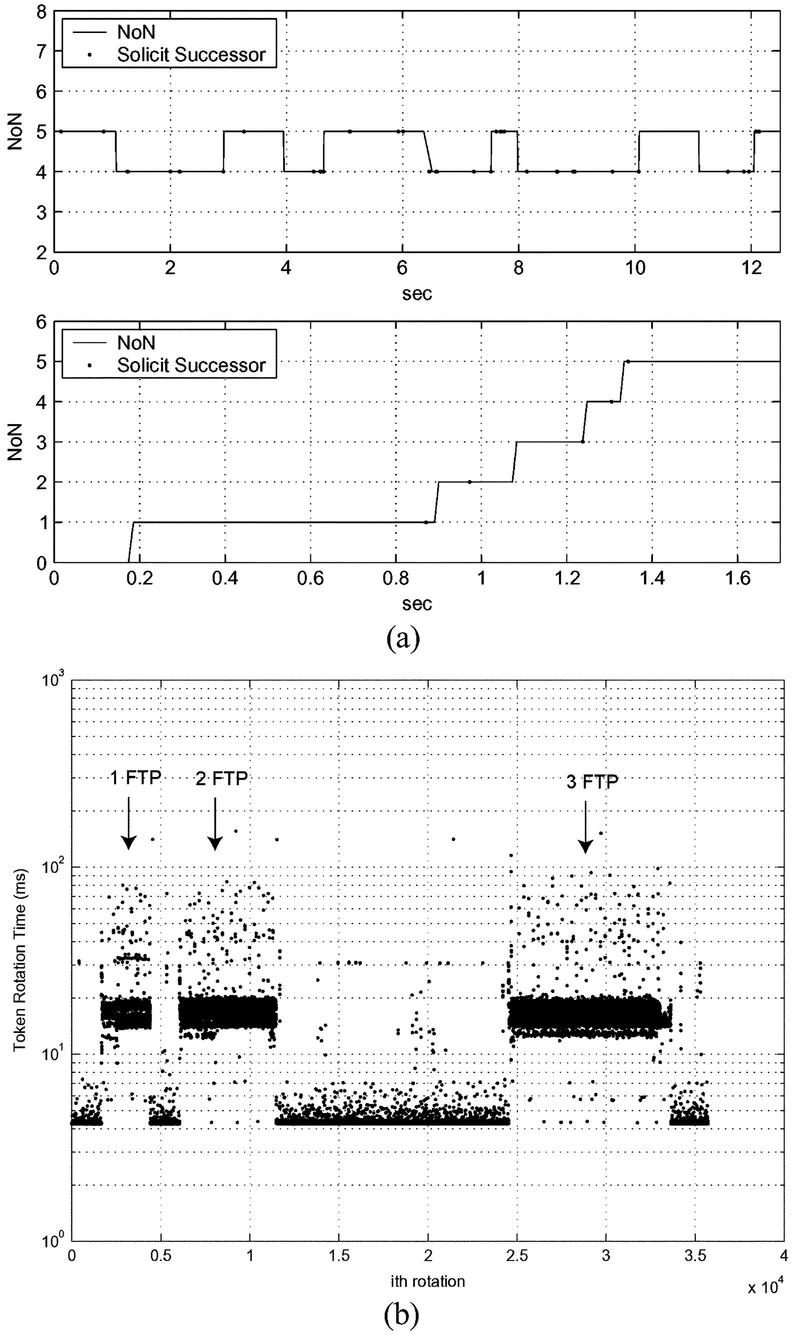
Lemma 5: There exists time $s < t + 2\mathrm{MTRT}$ such that there are no multiple equivalent tokens at time $u > s$.

Proof: By Lemma 4 no multiple equivalent tokens exist at time $t + 2\mathrm{MTRT}$, i.e., all multiple equivalent tokens are removed at some time $s < t + 2\mathrm{MTRT}$. By Lemma 3 there are no multiple equivalent tokens at time $u > s$. ■

Lemma 6: At some time s, with $t < s < t + \mathrm{IDLE\_TIME} + 3\mathrm{MTRT}$, there is one and only one token in any ring.

Proof: There could be multiple tokens in a ring at time s. By Lemma 5 only one or none of these tokens will survive by time $t + 2\mathrm{MTRT}$. Station y in a ring will only accept a token if it has higher priority than the last token that it accepted. So after one rotation, the priority of all existing tokens must be increasing in terms of its order of visits to station y. All of these tokens must visit station y within another MTRT, and will be deleted except for one token.

Even if all tokens get deleted at time $u > s$, at least one token will exist by time $u + \mathrm{IDLE\_TIME}$. From the bijection, we know that if x has the bijection with y, y must accept tokens from x. This means that the station that holds the token has higher priority than its successor. The only station that is an exception to this rule is the owner of the token, because the owner increments the generation sequence number by one when it passes the token. The only generation sequence number assignment that will satisfy these constraints is the following. The generation sequence number of the stations from the successor of the owner to the station with the token has



(a)

(b)

Fig. 16. Behavior with load and node failures. (a) Robustness and responsiveness. (b) Token rotation time during FTP—3 nodes.

the same generation sequence number as the token. The generation sequence number from the successor of the station with the token to the owner is one less than that of the token. This means that when one or more stations regenerate tokens during $[u, u+\text{IDLE\_TIME}]$, the generation sequence number of these tokens will be higher than that of any stations at time s. Because these tokens will be passed around as a *normal* token, only the highest priority token will survive within one MTRT, and lower priority tokens will be deleted. ■

3) Stability:

Lemma 7: A ring will not break after time $t + 2\text{MTRT} + \text{INRING\_TIME}$, and the number of stations in the ring will not decrease.

Proof: A station updates its NS pointer when it is unable to pass the token to its successor, leaves the ring, or receives *set_successor*. Because each station in a ring has a bijection with its successor, it is always able to pass the token to its successor. Thus, no station will be kicked out of the ring. Also, by assumption, a station does not voluntarily leave the ring. When a station receives the *set_successor*, the NS pointer of the station will change. However, the ring will still not break since all contending stations must have a connection with the successor of the soliciting station, by assumption.

A station updates its PS pointer when it accepts a token from a station different from its predecessor. A *set_predecessor* token from another station may be accepted if the station has the same ring address, causing the ring to break. However, this situation cannot possibly arise. A station in the ring cannot receive a token from a station in the same ring that is not its predecessor because each station has a bijection with its successor and cannot fail to pass a token to its successor. Moreover, by assumption, a station is not allowed to leave the ring voluntarily and induce its predecessor to generate *set_predecessor*.

We now show that after $t + \mathrm{MTRT} + \mathrm{INRING\_TIME}$, a station outside a ring cannot possibly send a *set_predecessor* token to a station inside the ring with the same ring address. Suppose that station y, with ring address B, receives a token from station w, outside the ring. Let x be the predecessor of y. From time $t + \mathrm{MTRT}$ onwards, there will be no multiple equivalent tokens by Lemma 5. A station cannot possibly remember a token that did not exist at time $t+\mathrm{MTRT}$, and at time $t + \mathrm{MTRT} + \mathrm{INRING\_TIME}$, because a station must have accepted a token during $[t+\mathrm{MTRT}, t+\mathrm{MTRT}+\mathrm{NRING\_TIME}]$ or have formed a *self-ring*.

Thus, by $t + \mathrm{MTRT} + \mathrm{INRING\_TIME}$, if a station remembers anything about the token with a particular ring address, B, it is remembering the same token. If a station receives a token from a station outside the ring, then the token must have made a loop from station y to w and back to y. This means that there was a break in the ring to which x and y belong, because when a token travels it makes bijection between the sender and the acceptor of the token. For station x and y to be in the same ring at time $t + \mathrm{MTRT} + \mathrm{INRING\_TIME}$, a new token must have been regenerated by a station in the loop after the token with ring address B has passed them by. But this is not possible because y must have received the token with the ring address B within MTRT since the last time it saw it, and all stations in the loop have seen the token after station y accepted it. ■

4) Ring Enlargement:

Lemma 8: If station x has the bijection with its successor y, then the INRING_TIMER of x goes off before y.

Proof: The fact that x has the bijection with y shows that the last token y accepted was from x. Then x must have reset its INRING_TIMER before y, so the INRING_TIMER of y cannot go off before x. ■

Lemma 9: When a station goes out of ring (into the *offline* state), $|S|$, the number of the bijections, remains positive.

Proof: Suppose the predecessor of y is x, and the successor of y is z. When y goes into the *offline* state, it waits for an invitation without changing its channel and changes when it detects a token $p \cdot \mathrm{non} = \mathrm{MAX\_NoN}$.

We distinguish the two cases in which a station can be kicked out. The first case is when the INRING_TIMER expires. In this case, from the Lemma 4, x could not have the bijection with y, because if it did, the INRING_TIMER of x would have gone off before y. In this case, whether or not y had bijection with z, $|S|$ will not decrease, because in the worst case $|S|$ stays the same if y had the bijection with z. The second case is when y is kicked out because it is not successful in finding a successor. Again, whether or not x had the bijection with y, $|S|$ will remain positive, because in the worst case we lose the bijection from x to y, but x creates a bijection with another node or a self-bijection. ■

Lemma 10: $|S|$ monotonically increases and converges to $|V|$ in a finite time.

Proof: By Lemma 9, $|S|$ is nondecreasing. By Lemma 7 a ring will neither break nor decrease in size after time $t + \mathrm{MTRT} + \mathrm{INRING\_TIME}$. In addition, a station will not solicit another station to join unless it sees two successful token rotations. If there is no topological change, a station will not join a node unless it is part of a ring. This means that the number of stations in a ring does not decrease.

If $|S| \neq |V|$, at some time $s > t + 2\mathrm{MTRT} + \mathrm{INRING\_TIME}$, there is a station y that is waiting to join at time s.

Station y waits to join until it hears a token indicating $p \cdot \mathrm{non} = \mathrm{MAX\_NoN}$, and it then changes the channel. In this case, we gained a full ring in one channel and start to create another ring in another channel. From Lemma 1, the stations operating in the other channel detect a ring in one CLAIM_TOKEN_TIME. If the station y accepts a token, allowing another station to form the bijection with y, then we gain in the size of $|S|$. Since within every INRING_TIME, the number of bijections increases, $|S|$ will converge to $|V|$ in finite time. When $|S|$ finally converges to $|V|$ at time $u > s$, using in-ring interference cancellation, each ring has exactly one token. ■

VII. Saturation Throughput Analysis

We calculate the saturation throughput assuming an ideal channel (no errors) and all terminals in a single ring, following the approach in [13] and [16] for saturation throughput analysis for IEEE 802.11.

A. Model

There are N stations in the ring. Each station has a packet immediately available for transmission after the completion of each successful transmission. Unlike in 802.11, consecutive packets do not wait for a random backoff time before transmitting. Instead, they stop transmitting packets at the end of the THT, and wait until they recapture the token.

Let $c(t)$ be the stochastic process representing the time counter for a given station. A discrete time scale is adopted: t and $t + 1$ correspond to the beginning of two consecutive slot times. This discrete time scale does not directly relate to the system time: As illustrated in Fig. 10(a), one jiffy is adopted as a slot time size j since it may include the token transmission.

Let I, M, H be the "Idle Time," "MTRT" and "THT," respectively, and let $T_0 = I = M - H - 1, T_1 = H, T_2 = 1$ be timer values. We assume that the token passing time equals one slot time. The state at slot time t is $s(t) = 0, 1,$ or 2 corresponding to *idle, have_token*, and *monitoring* state, respectively.

Each station is represented by the two-dimensional (2-D) discrete Markov chain $(s(t), c(t))$ depicted in Fig. 10(a).[4] This is a

[4] The Markov chain formulation is not needed for our calculation. We present it for comparison with the model for 802.11 in [13].

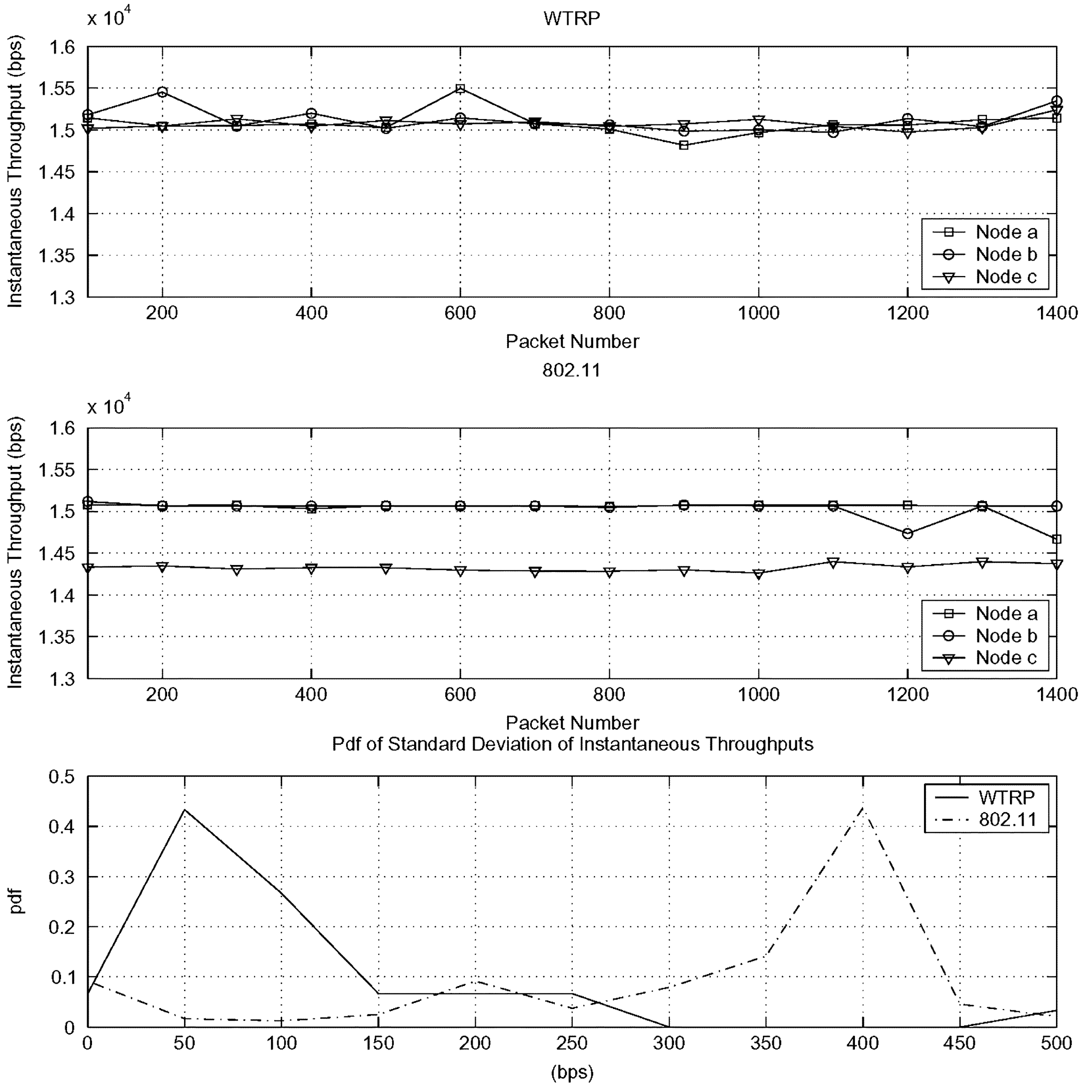


Fig. 17. Instantaneous throughput.

very simple chain in which the only nonzero one-step transition probabilities are

$$\begin{cases} P\{i,k|i,k-1\}=1, & 0<k<T_i \\ P\{i,0|i-1,k\}=1, & k=T_i \\ P\{0,0|2,0\}=1. \end{cases} \tag{1}$$

The first equation in (1) says that the time counter is incremented at the beginning of each slot time. The second and third equations of (1) indicate that the state changes when the corresponding timer expires. There are $M = N \times (H+1)$ states in all, and so the stationary probability distribution is simply

$$b_{i,k} = P\{(c(t), s(t)) = (i,k)\} = \frac{1}{M} \tag{2}$$

for all (i,k).

A station transmits when $s(t) = 1$ (*have_token*), so the probability β that it transmits data in a randomly chosen slot time is

$$\beta = \sum_{k=1}^{H} b_{1,k} = \frac{H}{M}. \tag{3}$$

We will assume that during the holding time of H slots, a station can transmit exactly K packets.

B. Throughput

In one MTRT of M slots, each station transmits K packets and monitors the station for one slot time. Thus the normalized throughput in bits/s is

$$S = \frac{E[P] \times K \times N}{\text{length of } M \text{ slots}} = \frac{E[P]KN}{jN + KNT_s} = \frac{E[P]K}{j + KT_s} \tag{4}$$

in which $E[P]$ is the payload (in bits) in one packet, j is the size (in bits) in one slot, and T_s is the packet size (in bits).

We calculate T_s for two possible implementations illustrated in Fig. 10(b). In the first case, WRTP is implemented on top of 802.11 using RTS/CTS, and incurs the correponding overhead, and so

$$\begin{aligned} T_s^{\text{imp}} = {} & \text{RTS} + \text{SIFS} + \delta + \text{CTS} + \delta + L + E[P] \\ & + \text{SIFS} + \delta + \text{ACK} + \text{DIFS} + \delta. \end{aligned} \tag{5}$$

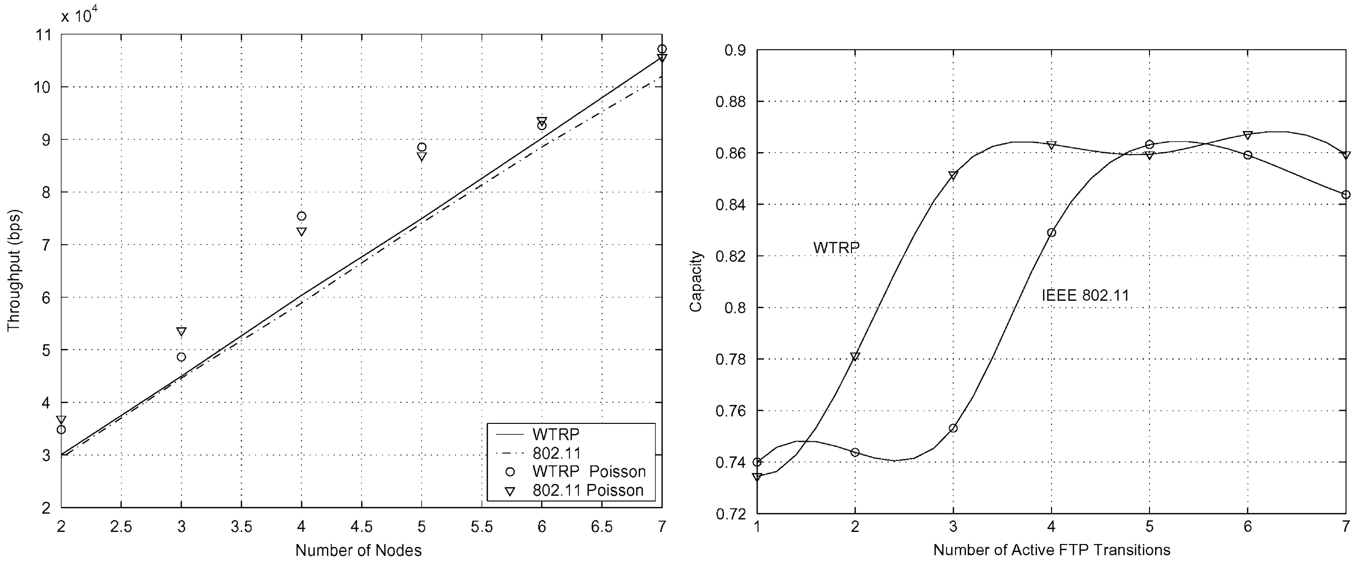


Fig. 18. Number of nodes versus throughput. (a) CBR performance. (b) FTP performance.

In the second case, WRTP are broadcast over 802.11 using the basic access mode, so RTS/CTS is disabled and

$$T_s = T_s^{\mathrm{gen}} = L + E[P] \tag{6}$$

in which L is the WRTP packet header length. In both cases, L is the sum of the PHY header and MAC header.

Table I lists the parameter values. The system values are those specified for the frequency hopping spread spectrum (FHSS) PHY layer [24]. The channel bit rate is assumed to be equal to 1 Mbits. This rate is lower than that of wireless card we used in the experiments, but allows comparison with results of 802.11 throughput [13].

Packet payload size is 8184 bits is considered, which takes one THT, so $K = 1$. Using these parameters to calculate $T_s^{\mathrm{imp}}, T_s^{\mathrm{gen}}$ and substiuting in (4) gives the WRTP saturation throughputs of

$$S^{\mathrm{imp}} \approx 0.83 \text{ Mbps}, \quad S^{\mathrm{gen}} \approx 0.9 \text{ Mbps}.$$

Fig. 11 also plots saturation throughputs for 802.11 using both basic access and RTS/CTS mechanisms, reproduced from [13].

As expected, WRTP throughput is superior. It is unaffected by the number of stations, because the MTRT, $M = N \times (H+1)$, is "tuned" to the number of stations. (This ideal condition is removed in the next section.) The 802.11 throughput first increases and later decreases with N, because of collisions, whose effect is more pronounced in the basic access mechanism.

VIII. Performance Analysis

The througput analysis is for idealized conditions of no channel errors, no topology changes and with MTRT adjusted to the number of stations N. In this section, we evaluate the effectiveness of a single ring WTRP under more realistic conditions. We consider "latency bound," the time that a station has to wait to transmit; "fairness" measured in terms of the amount of transmission bandwidth that stations get; "robustness" measured as time to recover from a ring collapse; "responsiveness" of the ring to changes in the medium; and "utilization" of the channel capacity. WTRP and 802.11 (with RTS/CTS) performance is compared for both constant bit rate (CBR) and FTP traffic. Note that WTRP is implemented on top of 802.11 in basic mode, so it incurs some extra overhead.

A. Optimum Operating Frequency

Fig. 12(a) and (b) compare performance for periodic traffic (CBR) with 100-byte packets as the packet interarrival time increases. The performance of WTRP is superior when the interarrival time is near MTRT. Under high load (interarrival time smaller than MTRT), WTRP performance fluctuates because of its fixed THT and packet queue size. In the scenario, THT only allows a node to transmit one or two packets. This fills the buffer and consequently, stops the upper layer with logical link control, causing delay and performance degradation. Peaks in the figures indicate that node transmits more packets than normal in a short period.

With packet interarrival close to MTRT, WTRP only sends one packet and one token, showing performance improvement compared to 802.11 which suffers from collision. Under low load, with packet interarrival larger than 75 ms, WTRP and 802.11 performance are indistinguishable because of low collision probability.

Fig. 13(a) and (b) compares the performance with constant packet generation rate (one MTRT), as the packet size is increased. WTRP throughput is proportional to load (number of stations). 802.11 shows a slight degradation as packet size increases (the gap between the WRTP and 802.11 curves grows). This is because large packets can be fragmented by the IEEE 802.11 protocol and the station in backoff stage samples and senses the channel. In between sampling times, station may detect an empty channel as the packet interarrival time is longer than the sensing time, even though the channel is not really empty.

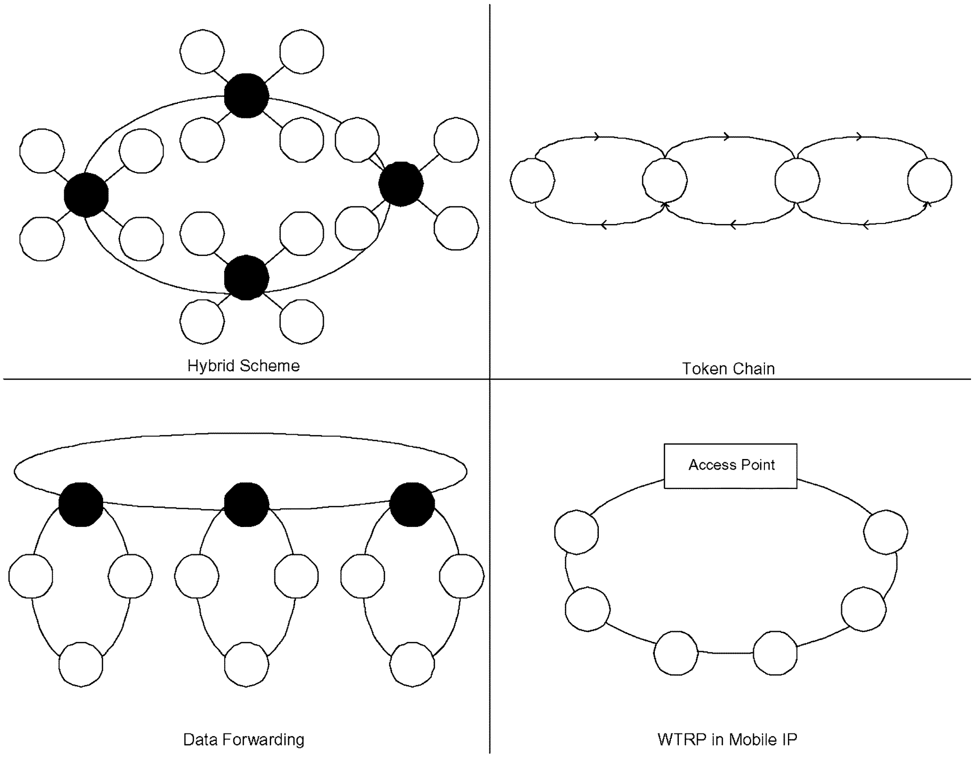


Fig. 19. System extensions.

B. Bound on Latency

Fig. 14(a) and (b) shows that the variance of the token rotation time is nearly zero, with THT of 1.5 ms, so each station can transmit periodically. The token rotation time increases by less than THT when a new node is added, while the variance remains close to zero. Thus an increase in network size does not cause instability.

The small variance in token rotation time is also seen in the distribution of packet interarrivals (or jitter). Fig. 15 shows this distribution at the listener when there are three senders. Each sender sends 100 bytes every 50 ms and MTRT is set to 50 ms. The variance is small despite the overhead of 802.11 in terms of introducing backoff interval.

The experimental results in Fig. 16(b) are reproduced from [3]–[5]. The same 7.5-MB file is transferred, with one, two, and three concurrent FTP sessions. Evidently, the token rotation time does not change with the increased number of simultaneous sessions. Obviously, the total time is proportional to the number of sessions.

C. Robustness and Responsiveness

Fig. 16(a) shows the response to topology changes. In the upper plot, there are five nodes, one of which turns on and off every second. The dots represent the *solicit_successor* token. WRTP handles leaving and joining in a robust way, as the number of stations in the ring never drops below four. The lower plot is a trace of ring formation when all five nodes turn on at the same time: The ring size increases monotically and reaches its maximum. Fig. 16(a) also illustrates that WTRP recovers quickly from single-node failures.

D. Fairness

IEEE 802.11 is not fair; it favors the station with the last successful transmission [14]. Fig. 17 shows the instantaneous throughput of three nodes. The top plot shows that with WTRP, all stations achieve the same throughput. The middle plot shows that with 802.11, the station that arrived late is disadvantaged. The bottom plot shows that the standard deviation of the throuput is much worse under 802.11.

E. Network Size

In 802.11, each additional node increases in collision probability. Fig. 18(a) compares the throughput of WTRP and 802.11 as we increase the number of nodes. Each node sends 100-byte packets every 50 ms. WTRP performs better than 802.11 and the improvement increases with network size. Note that as WTRP is implemented on top of 802.11, it incurs a performance penalty. The figure also compares performance with random (Poisson) packet generation with an average interarrival time of 50 ms.

Fig. 18(b) compares the throughput as the number of simultaneous FTP transfers is varied. For both 802.11 and WTRP networks, the number of nodes is equal to the number of simultaneous FTP transfers. For instance, in the case of three simultaneous transfers, the transfers are from 1 to 2, from 2 to 3, and from 3 to 1. For 802.11, throughput first increases, and then decreases, as collisions create congestion [16].

WTRP throughput is constant when the number of transfers is larger than three. The reason the throughput is lower one or two nodes is that they cannot saturate the network.

IX. Conclusion

WTRP is inspired by the IEEE 802.4 token bus protocol, which in turn was motivated by applications in factory automation, with requirements of bounded latency and robustness against multiple node failures [18], [22]. WTRP combines these features of bounded latency and robustness with the more versatile *ad hoc* wireless networks.

There are three WTRP implementations. The simulator implementation is a tool for WTRP design. The user-space implementation is a platform-independent implementation in the application layer. The kernel implementation is a Linux link layer module built on top of the IEEE802.11 in DCF mode. It incurs the overhead associated with IEEE 802.11.

We compared WTRP stability and saturation throughput with IEEE 802.11. Results show that WTRP recovers quickly from failures, has higher throughput because of lower collision probability, and allocates bandwidth equally among stations. The consistency of the token rotation time, regardless of the number of simultaneous transmissions, leads to predictable medium access latency. These features make WTRP attractive for real time applications.

The properties of WTRP can provide performance gains when it is combined with other architectures. We discuss four extensions illustrated in Fig. 19.

Hybrid Schemes hierarchically combine the centralized star topology and the distributed ring topology. *Token chain* creates a ring by bidirectional token passing. The token coming from the predecessor is passed to the successor and token coming from the successor is passed to the predecessor. The nodes at the edges have the same station as their predecessor and the successor. This kind of network structure is suitable if nodes have a linear arrangement.

Data forwarding can be done by clustering stations into multiple rings. Different channels are assigned to different rings to avoid interference. There are several possibilities for the routing scheme. In the depicted scheme the rings are connected by “gateway” nodes which, in addition to their own “local” ring, also belong to a “backbone” ring that the gateway nodes create. Routing is divided into intraring routing between nodes in the same ring and interring routing between gateway nodes.

In *Mobile IP design with WTRP*, resource information can be included in the token signal, which enables the mobile node to detect the attachment point quickly, without waiting for beacon signals.[5] The attachment point can also detect movement by checking its connectivity table to execute a smooth hand-off algorithm. Since *Network resource* information can be included in the token, a mobile node can search for a ring with the appropriate resources.

Acknowledgment

We are grateful to R. Datta, J. Ko, A. Puri, R. Attias, and S. Tripakis for their help in the development of WTRP.

References

[1] M. Ergen, “WTRP-wireless token ring protocol,” Master’s degree Thesis, Berkeley, CA, 2002.

[2] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya, “Wireless token ring protocol-performance comparison with IEEE 802.11,” in *8th IEEE Int. Symp. Comput. Commun.*, Antalya, Turkey, July 2003.

[3] D. Lee, “Wireless token ring protocol,” Master’s degree Thesis, Berkeley, CA, 2001.

[4] D. Lee, R. Attias, A. Puri, R. Sengupta, S. Tripakis, and P. Varaiya, “A wireless token ring protocol for ad hoc networks,” in *2002 IEEE Aerospace Conf. Proc.*, MT, Mar. 9–16, 2002.

[5] M. Ergen, D. Lee, R. Attias, A. Puri, R. Sengupta, S. Tripakis, and P. Varaiya, *Wireless Token Ring Protocol*: SCI Orlando, July 2002.

[6] International Standard ISO IEC8802-4:1990—, ANSI/IEEE Std. 802.4-1990.

[7] K. Nichols, S. Blake, F. Baker, and D. Black, RFC2474 Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers, Dec. 1998.

[8] S. Herzog, RFC2750 RSVP extensions for policy control, Jan. 2000.

[9] *Draft Int. Standard ISO IEC 8802-11—IEEE P802.11/D10*, Jan. 14, 1999.

[10] M. Hannikainen, J. Knuutila, A. Letonsaari, T. Hamalainen, J. Jokela, J. Ala-Laurila, and J. Saarinen, “TUTMAC: A medium access control protocol for a new multimedia wireless local area network,” in *9th IEEE Int. Symp. Personal, Indoor and Mobile Radio Commun.*, vol. 2.3. New York, 1998, pp. 592–596. vol. 1574.

[11] I. F. Akyildiz, J. McNair, C. L. Martorell, R. Puigjaner, and Y. Yesha, “Medium access control protocols for multimedia traffic in wireless networks,” in *IEEE Network*: IEEE, July–Aug. 1999, vol. 13, pp. 39–47.

[12] P. Varaiya, “Smart cars on smart roads: Problems of control,” *IEEE Trans. Automat. Contr.*, vol. 38, pp. 195–207, Feb. 1993.

[13] G. Bianchi, “Performance analysis of the IEEE 802.11 distributed coordination function,” in *IEEE J. Select. Areas Commun.*, vol. 18, Mar. 2000.

[14] Y. Wang, “Achieving fairness in IEEE 802.11 DFWMAC with variable packet size,” in *Global Telecommun. Conf.*, 2001.

[15] T. Pagtzis, P. Kirstein, and S. Hailes, *Operational and Fairness Issues with Connection-Less Traffic Over IEEE802.11b*: ICC, 2001.

[16] G. Bianchi, “IEEE 802.11-saturation throughput analysis,” *IEEE Commun. Lett.*, vol. 2, pp. 318–320, Dec. 1998.

[17] H. Shim, T. J. Koo, F. Hoffman, and S. Sastry, “A comprehensive study on control design of autonomous helicopter,” in *Proc. IEEE Conf. Decision Contr.*, FL, Dec. 1998.

[18] A. Tanenbaum, *Computer Networks*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[19] M. Ergen, S. Coleri, B. Dundar, A. Puri, and P. Varaiya, “Position leverage smooth handover algorithm for mobile IP,” in *IEEE ICN*, Atlanta, GA, Aug. 2002.

[20] *High Performance Radio Local Area Network (HIPERLAN), Type 1; Functional Specification*, Oct. 1996. Radio equipment and systems (RES).

[21] [Online]. Available: http://www.bluetooth.com

[22] W. McCoy, RFC1008: Implementation guide for the ISO transport protocol, 1987.

[23] [Online]. Available: http://wow.eecs.berkeley.edu/WTRP

[24] *IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Nov. 1997.

Mustafa Ergen (S’98) received the B.S. degree from Middle East Technical University (METU) and was the METU Valedictorian in 2000. He received the M.S. degree from the University of California, Berkeley, in 2002. He is currently working toward the Ph.D. degree at the University of California, Berkeley, all in electrical engineering.

He is the Coordinator of the Distributed Sensing Group. He has been conducting research in wireless communication networks with an emphasis on WLAN and OFDM systems and is the author of many works in the field, including the book (with A. R. S. Bahai and B. Saltzberg) *Multi-Carrier Digital Communications: Theory and Applications of OFDM* (New York: Springer, 2004).

Duke Lee is a graduate student with the University of California, Berkeley (UC Berkeley). He has been involved in the UC Berkeley PATH Project since 1997.

[5] Research shows that hand-off time in Mobile IP is dominated by the beacon period of the attachment points [19].

Raja Sengupta (S'93–M'02) received the Ph.D. degree from the Electrical Engineering and Computer Science Department, University of Michigan, Ann Arbor.

He is currently an Assistant Professor in the Systems Program of the Department of Civil and Environmental Engineering, University of California, Berkeley. He is Director of the PATH Wireless Laboratory and Deputy Director of the Center for Collaborative Control of Unmanned Vehicles. His current research interests are in Dedicated Short Range Communications (DSRC), networked estimation and control, vision based control, and collaborative behavior in robotic systems.

Dr. Sengupta is Associate Editor of the IEEE CONTROL SYSTEMS MAGAZINE and of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS. He was Program Chair of the IEEE Conference on Autonomous Intelligent Networked Systems 2003. He is Co-Chair of the first ACM MOBICOM Workshop on Vehicular Ad-hoc Networks 2004.

Pravin Varaiya (M'68–SM'78–F'80) is a Nortel Networks Distinguished Professor with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. From 1975 to 1992, he was also Professor of Economics at Berkeley. From 1994 to 1997, he was Director of the California PATH program, a multiuniversity research program dedicated to the solution of California's transportation problems. His current research is concerned with communication networks, transportation, and hybrid systems. He has taught at the Massachusetts Institute of Technology, Cambridge, and the Federal University, Rio de Janeiro. He has coauthored three books and more than 250 technical papers. The second edition of *High-Performance Communication Networks* (with J. Walrand) (San Mateo, CA: Morgan-Kaufmann, 2000), and *Structure and Interpretation of Signals and Systems* (with Edward Lee) (Reading, MA: Addison-Wesley, 2003).

Prof. Varaiya has held a Guggenheim Fellowship and a Miller Research Professorship. He received an Honorary Doctorate from L'Institut National Polytechnique de Toulouse, and the Field Medal of the IEEE Control Systems Society. He is a member of the National Academy of Engineering. He is on the editorial board of several journals, including *Discrete Event Dynamical Systems* and *Transportation Research—C*. He is a member of the Board of Directors of Teja and a Founder of Sensys Networks.