RTRD: Real-Time and Reliable Data Delivery in Ad Hoc Networks

Kai Han* Guanhong Pei* Hyeonjoong Cho* Binoy Ravindran* E. D.Jensen‡

*ECE Dept., Virginia Tech
Blacksburg, VA 24061, USA
{khan05,somehi,hjcho,binoy}@vt.edu

‡The MITRE Corporation
Bedford, MA 01730, USA
jensen@mitre.org

***Abstract*—In this paper, we present a reliable real-time data delivery (communication) mechanism for ad-hoc networks, called RTRD. The mechanism makes use of a proactive wireless routing protocol (DSDV) for path finding and maintenance, and timely delivers data through a priori bandwidth reservation. In addition, to be robust to network failures, or to deliver large data chunks, it simultaneously delivers data in multiple paths. The simulation results conducted by NS-2 validate RTRD's effectiveness.**

I. Introduction

Over the last years, ad-hoc networks are widely used in many smart applications, including battlefields and earthquake response systems. While these applications remain diverse, one common point they all share is the need of an efficient and reliable real-time communication mechanism. However, the potential contention in MAC protocols (e.g., IEEE 802.11 and 802.15.4), the node mobility nature of the ad-hoc networks, and the interference between the transmitting nodes, all make it difficult to achieve good quality real-time communication (data delivery).

We consider an ad-hoc network with a set of mobile nodes $N = \{n_1, n_2, n_3, \ldots\}$. A basic wireless routing protocol is available for packet transmission between nodes. Medium Access Control (MAC) protocol is IEEE 802.11. Node clocks are synchronized. Nodes may join, leave, move or fail at any time, resulting in unexpected packet losses.

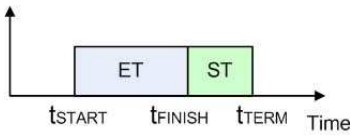


Fig. 1. Real-time Task

Real-time data delivery is often triggered by the completion of a real-time task. A real-time task executes on an individual node, with a given termination time t_{Term}. As shown in Figure 1, a task starts at time t_{Start}. After a given execution time ET, it finishes at time t_{Finish} and has some data which requires to be delivered to another node before t_{Term}. The time for real-time data delivery is the task's slack time ST, $ST = t_{Term} - t_{Finish}$. Before task execution starts, ET and the size of delivered data can be estimated through application profiling techniques [1]–[3].

In this paper, we present a mechanism called "Real-Time and Reliable Data Delivery" (or RTRD), for real-time data delivery in ad-hoc networks. While a task is executing, RTRD reserves enough bandwidth between the source and destination nodes. After the task completes, RTRD executes real-time data delivery within the required time period ST. In addition, to deal with possible message losses, or to deliver extremely large data chunks, RTRD simultaneously transmits data in multiple paths.

The rest of the paper is organized into three sections. In Section II, we describe the RTRD mechanism by first overviewing RTRD and its underlying routing protocol DSDV. After that, we illustrate each RTRD component, including packet priority management, bandwidth reservation, packet blocking control, and multi-path data delivery. In Section III, we present and analyze simulation results, which validate RTRD's effectiveness with comparison to other algorithms. We conclude the paper and present our future work in Section IV.

II. The RTRD Mechanism

Based on prediction on the possible size of real-time data, RTRD executes bandwidth reservation (or BR) before the real data delivery (or DD) begins. In this way, when a real-time task completes and data is ready for delivery, it can immediately transmit data with desired sending rate. RRTD's basic strategy is shown in Figure 2.

RTRD uses an existing wireless routing protocol (DSDV in this paper) to provide immediate data delivery path. It also needs to reserve enough bandwidth before data is delivered, and achieves bandwidth reservation by controlling its and its neighbors' packet sending and receiving rates. In

addition, it delivers data through multiple paths, in order to robustly transmit data, or separately transmit large data chunks.

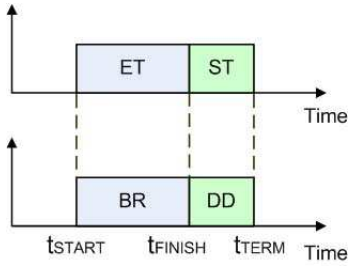


Fig. 2. RTRD Strategy

A. Underlying Routing Protocol

RTRD makes use of routing protocols to discover and maintain data delivery paths. Because deliveries are real-time, it expects path-finding time to be as short as possible.

Some well-known routing protocols, such as Adhoc On-demand Distance Vector (or AODV) and Dynamic Source Routing (or DSR), are reactive in that they form a path on-demand when a transmitting node requests one. Reactive protocols are not considered by RTRD, because path-finding time might be very long, especially when several intermediate nodes exist in the path — i.e., long path-finding time might cause data delivery violate its time constraint.

RTRD therefore adopts a proactive routing protocol called "Destination-Sequenced Distance Vector" (or DSDV) [4]. DSDV achieves proactiveness by letting nodes periodically discover and maintain paths. With DSDV, when a packet needs to be delivered, the path is already known and can be immediately used. Table II-A shows one DSDV node's (N1) simplified forwarding table.

TABLE I
N1's Simplified Forwarding Table

Destination	Next Hop	Hops	Sequence Number
N1	N1	0	710
N2	N3	2	392
N3	N3	1	676
N4	N3	2	128
N5	N3	3	350

Each node maintains a forwarding table for all reachable destinations. The table contains the next hop, number of hops and sequence number for each destination. A sequence number shows the freshness of a path, and is used to help nodes distinguish stale paths from the new ones and thus avoid formation of path loops.

Nodes broadcast routing updates periodically or at the time the network topology changes. Each path is tagged with a Time-to-Live (TTL) to indicate its freshness. Before a real-time data delivery starts, RTRD will reduce the TTL of the path between the source and destination nodes, in order to enhance path maintenance frequency. In this way, it remains a reliable path when delivery begins.

B. Packet Priority Management

RTRD orders packet delivery sequence according to their priorities. The packet with the highest priority will be first delivered. Priorities in RTRD are illustrated in Table II-B.

TABLE II
Priorities in RTRD

Priority Level	Priority	Packet Type
P1	11	Real-time
P2	21	RTRD Control (Reservation)
P2	22	RTRD Control (Multi-path)
P2	23	DSDV Routing
P3	31	Non-real-time

Here, RTRD control message (reservation) is sent by sender nodes (the source or intermediate nodes) to tell receiver nodes (the destination or intermediate nodes) the bandwidth to be reserved. RTRD control message (multi-path) is used by the source node to find multiple paths to the destination node. The number of paths is application-specific. Note that this number is an upper bound, because there might be less than the required number of paths between the source and destination nodes.

C. Delivery Bandwidth Computation

In order to know how much bandwidth is available for a node to use, we must take into consideration all transmissions that directly affect its opportunities to transmit.

IEEE 802.11 provides a CSMA/CA-based mechanism to allow nodes access wireless medium. To avoid the "hidden terminal" problem, before data transmission, the source node sends "Request to Send" (or RTS), and the destination node replies "Clear to Send" (or CTS). Every other node receiving RTS/CTS should remain in silence during the transmission period. With RTS/CTS, a node is not allowed to transmit whenever [5]:

1) It is receiving data;
2) One of its neighbors is receiving data (due to the reception of a CTS);
3) One of its neighbors is transmitting data to a node that is neither another neighbor nor the node itself (due to the reception of a RTS).

The available bandwidth for a node i to transmit (AB_i) is calculated as follows:

$$AB_i = EB_i - \left(b_i + \sum_{j \in \aleph_i} b_j + \sum_{j \in \aleph_i, k \neq \aleph_i^+} b_{jk} \right) \quad (1)$$

where b_i/b_j is the receiving bandwidth used by node i/j, b_{jk} is the traffic from node j to k, and $\aleph_i/\aleph_i^+$ is the set of neighbors of node i excluding/including itself. In real systems, poor link quality and the interference between nodes makes only a portion of the total bandwidth is usable. Therefore, here we use the total effective bandwidth EB_i for available bandwidth computation.

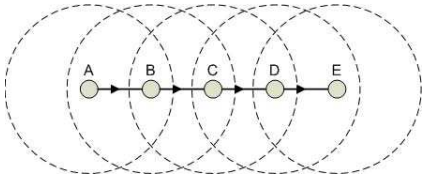


Fig. 3. Data Delivery in a 4-Hop Path

TABLE III
Data Delivery in a 4-hop path

	A	B	C	D	E
Hop1	S	R	CTS	-	-
Hop2	RTS	S	R	CTS	-
Hop3	-	RTS	S	R	CTS
Hop4	-	-	RTS	S	R

Denote r the required bandwidth for real-time data delivery. In delivery processes, nodes not only need to reserve r bandwidth, but also need to consider the extra bandwidth which they use to remain in silence due to the reception of RTS/CTS. Figure 3 and Table II-C show bandwidth consumption of nodes in data delivery path.

Algorithm 1: Computing Required Bandwidth

```
if i = source/destination then
    if destination/source in neighbors then
        r ≤ AB_i;
    else
        r ≤ AB_i/2;
else if i ∈ ℵ_{source/destination} then
    r ≤ AB_i/3;
else
    r ≤ AB_i/4;
```

where S and R are data sender and receiver, respectively. Note that S may be the source node (A) or intermediate nodes (B, C, D), and R may be the destination node (E) or intermediate nodes as well. Similar to [5], we show how to compute reserved bandwidth for node i in Algorithm 1.

D. RTRD Bandwidth Reservation

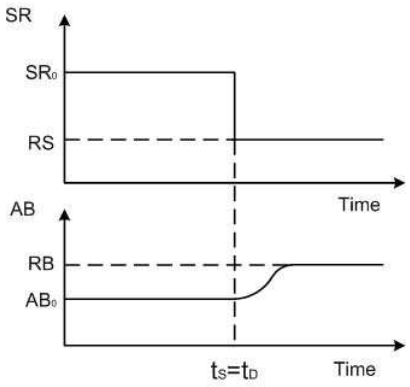
In Equation 1, we observe that a node's available bandwidth is affected by its receiving bandwidth (b_i), neighbor nodes' receiving bandwidth (b_j) and sending bandwidth (b_{jk}). In order to reserve enough bandwidth, a node (the source, destination or intermediate) should collaborate with its neighbors.

Except current available bandwidth AB_i, the extra required bandwidth for real-time data delivery, er_i, is given by:

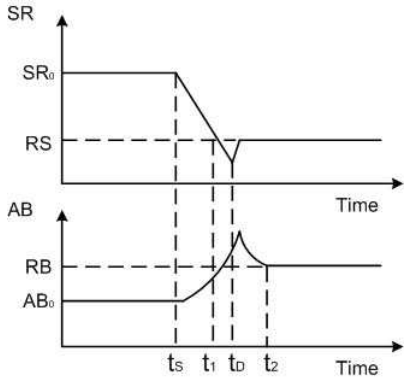
$$er_i = r - AB_i \quad (2)$$

If $er_i \leq 0$, there is no need to reserve extra bandwidth. But the node should exchange messages with its neighbors, telling them control their bandwidth usage. For instance, node j can only use $|er_{ij}|$ more bandwidth for its own transmission ($|er_{ij}| \leq |er_i|$).

If $er_i > 0$, the RTRD mechanism needs to reserve er_i more bandwidth for the coming delivery. For both neighbors and the node itself, if a data-receiving process does not begin, they can postpone this process by delaying to send CTS messages. In this way, they can reduce b_i or b_j in order to obtain er_i more bandwidth. This is as well done through collaboration between the node and its neighbors.



(a) Sharply Reducing Sending Rate



(b) Gradually Reducing Sending Rate

Fig. 4. Reducing Sending Rate on Neighbor Nodes

The RTRD mechanism also adjusts each neighbor's sending rate b_{jk} for bandwidth reservation, because the receiving process on nodes may have started, or nodes are receiving real-time data, or reducing b_i and b_j does not obtain enough bandwidth. Node i sends a meta-message to its neighbor j, telling it start to reduce $|er_{ij}|$ sending rate from a given time t_S ($|er_{ij}| \leq |er_i|$).

Figure 4 shows two ways to reduce the sending rate. In Figure 4(a), sending rate SR is sharply reduced from SR_0 to the required value RS at t_D, which is the starting time of the real-time data delivery. However, there is a time delay for available bandwidth AB to increase from current AB_0 to the required bandwidth RB. Because real-time delivery begins at t_D, the delay time causes the delivery violate its time constraint.

RTRD adopts the sending rate control shown in Figure 4(b), where the sending rate reducing begins earlier than t_D. At time $t_1 < t_D$, it reaches the required RS, and continues to decrease till delivery begins. After that, the sending rate increases to RS. Although the available bandwidth increases with a time delay, at t_D, the total available bandwidth exceeds the required value RB, and comes back to RB at time t_2. In this way, RTRD satisfies each real-time data delivery's time constraint.

E. Packet Blocking Control

In Figure 4(b), if SR_0 is very large and the sending rate does not decrease very fast, it is possible that available bandwidth is less than the required value at delivery starting time t_D. RTRD provides a packet blocking control mechanism to deal with such conditions, as shown in Figure 5.

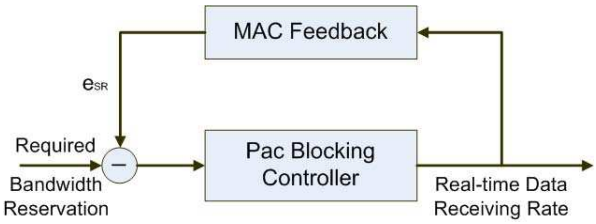


Fig. 5. Packet Blocking Control

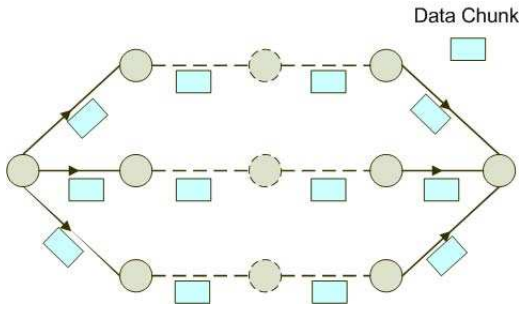
When starting to receive real-time data, a receiver node (an intermediate node or the destination node) computes the error e_{SR} between the real-time data receiving rate RR and the required bandwidth reservation RB, by extracting necessary information from the MAC layer [6]:

$$e_{SR} = \frac{RR - RB}{RB} \tag{3}$$

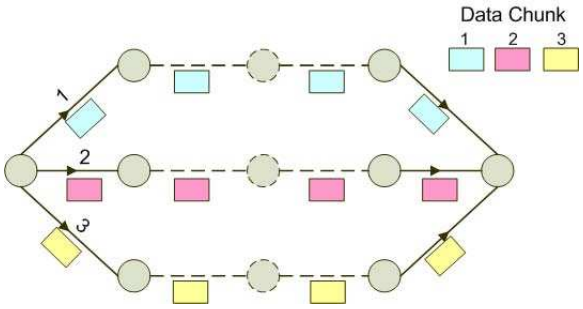
If $e_{SR} < 0$, the receiver randomly drops e_{SR} percentage P2- and P3- level non-real-time packets(Table II-B), in order to speed up real-time packet delivery.

F. Multi-path Data Delivery

Message losses and node failures are frequent in some ad-hoc networks. To achieve reliable real-time data delivery, RTRD adopts multi-path delivery mechanism (the number of paths is application-specific), as shown in Figure 6(a).



(a) Multi-path for Reliable Delivery



(b) Multi-path for Large Data Delivery

Fig. 6. Multi-path Real-time Data Delivery

If the required bandwidth exceeds a node i's total effective bandwidth EB_i, multi-path delivery can help to distribute the delivery work load to different paths, as shown in Figure 6(b).

III. Simulation Studies

We conducted a set of simulations of the RTRD mechanism using NS-2, and its performance is compared with several other mechanisms, e.g. SRTC [7], which is available in the literature that also provides real-time services in ad-hoc networks.

A. Simulation Environment

We consider a single broadcast region with an available link capacity of 2 Mb/sec under the IEEE 802.11 protocol with an effective data rate of approximately 1.43 Mb/sec. Each node generates variable-rate traffic (randomly uniform distribution) according to the exponential on-off traffic. Real-time data chunk sizes are randomly generated

subject to uniform distribution with the minimum value of 50 Bytes.

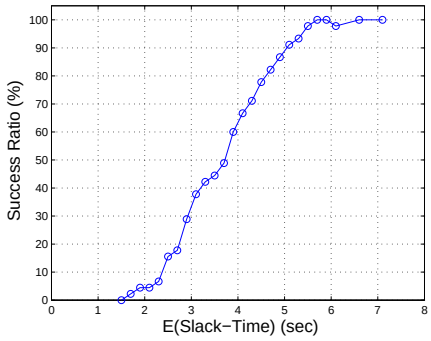
Non-real-time data transmission with a random uniformly distributed rate with the minimum value of 100 Bytes and the maximum of 400 Bytes. Within the transmission time, each delivery task's sending rate remains constant. Each real-time task's execution time (ET) and slack time (ST) are also in a randomly uniform distribution with the minimum value of 1000 ms (real-time tasks are stated in Section I). Inter-packet time between two packets is exponential distributed.

We employ random topologies with the node mobility uniformly distributed ranging from 0.1m/s to 3 m/s. If there is no possible path between source and destination, the packet is deleted and counted as an unsuccessful delivery.

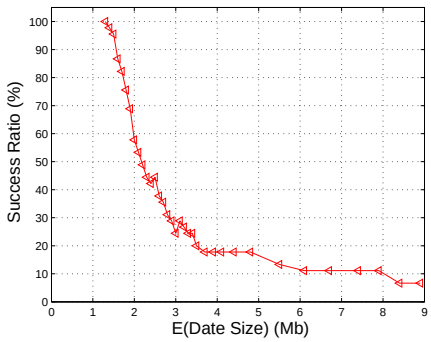
Other parameters, e.g., 802.11 physical layer parameters, were set to default values as recommended in [8] and in NS-2.

B. Simulation Analysis

1) Correctness and Effectiveness: Given the scenario that there are at least 2 intermediate nodes between the source and destination nodes, we measure the RTRD performance with respect to the success ratio (or $SucR$).



(a) SucR Regarding Slack Time



(b) SucR Regarding Data Chunk Size

Fig. 7. SucR under Various Slack Time/Data Chunk Size

In Figure 7(a), we fix the data chunk size, and randomly generate slack time according to different expectation (E(Slack-Time)). We fix the slack time and change (E(Data Size)) in Figure 7(b). We observe that $SucR$ approaches to 100% as E(slack-time) increases in Figure 7(a), or as the E(data size) decreases in Figure 7(b). Therefore, RTRD achieves perfect $SucR$ when given enough slack time, or when the data chunk size is moderate. Only when the slack time is very tight, or the required bandwidth makes the link overloaded, will RTRD's performance degrade.

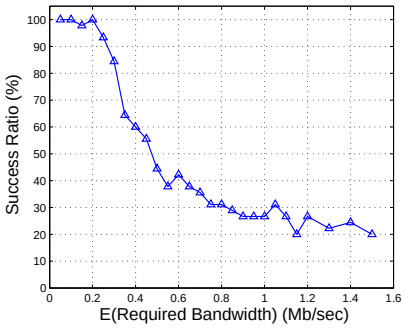


Fig. 8. SucR Regarding Required Bandwidth

Figure 8 shows the success ratio $SucR$ with two intermediate nodes in the path. The upper bound of the required bandwidth should be 1/3 of the total effective bandwidth the network could afford, and the upper bound here is 0.48 Mb/sec. Given an expectation of the required bandwidth less than 0.25 Mb/sec, we can achieve 100% $SucR$. When the expectation of the required bandwidth (E(Required Bandwidth)) goes beyond 0.25 Mb/sec, the $SucR$ decreases. This is because the maximum possible required bandwidth begins exceeding 0.48 Mb/sec as $E(RequiredBandwidth) \geq 0.25Mb/sec$.

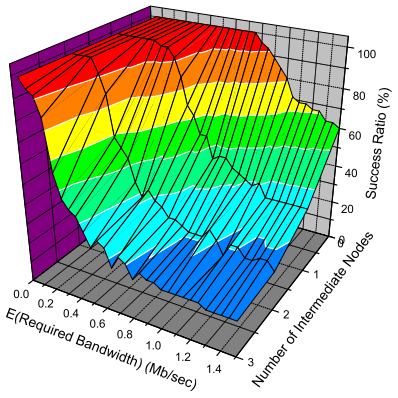


Fig. 9. RTRD SucR Performance

Figure 9 shows the success ratio $SucR$ under different number of intermediate nodes and E(Required Bandwidth). We observe that RTRD can fully explore the total network effective bandwidth for real-time data delivery.

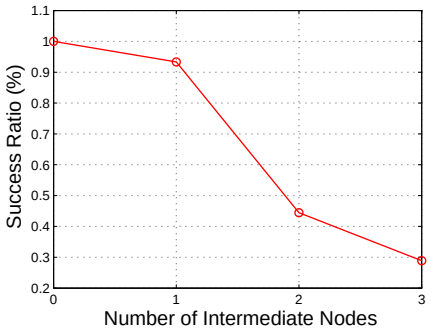


Fig. 10. SucR Regarding Number of Intermediate Nodes

Figure 10 shows how the number of intermediate nodes affects the success ratio $SucR$, when $E(RequiredBandwidth) = 0.6Mb/sec$. As stated in Algorithm 1, if there are more than 2 intermediate nodes in the path, the upper bound of required bandwidth is limited to 1/4 of the total effective data bandwidth (0.357 Mb/sec in our simulation), thus gives the lowest $SucR$.

2) Comparison Test: Figure 11 compares RTRD with other mechanisms. The experiment is conducted with respect to E(Required Bandwidth). The increment step of E(Required Bandwidth) is 0.05 MB/sec. We run 45 rounds of simulation for each value of E(Required Bandwidth), each of which lasts over 1200 seconds.

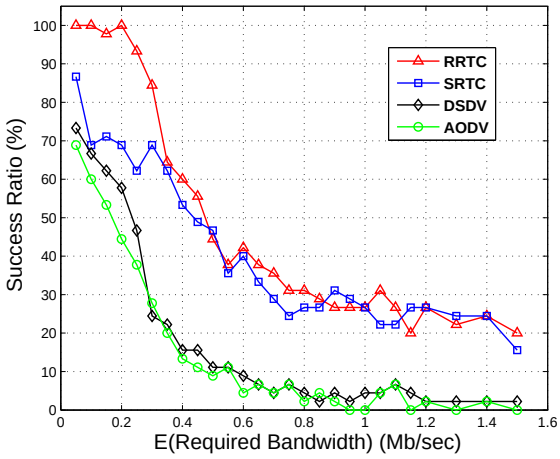


Fig. 11. Mechanism Comparison

The experiment can be intuitively divided into two phases. In the first, the maximum possible required bandwidth does not overload the link, which in the figure is that $E(RequiredBandwidth) \leq 0.2Mb/sec$. RTRD achieves near 100% $SucR$, and outperforms SRTC with at least 13.3% and at most 31.1%. The advantages come from RTRD's unique bandwidth reservation strategy illustrated in Section II-D and II-E. We consider both the sending and receiving rate of the neighbor nodes, and use packet blocking control strategy for ongoing real-time transmission. In comparison, DSDV and AODV exhibit poor $SucR$, which is at least 27% lower than RTRD due to their lack of real-time properties.

In the second phase ($E(RequiredBandwidth) > 0.2Mb/sec$), the maximum possible required bandwidth overloads the link, which seriously degrades all mechanisms' performance. RTRD still maintains the leading role, although the gap between SRTC and RTRD shrinks because of being under such severe traffic condition. Same as that in the first phase, DSDV and AODV fall far behind the former two.

IV. Conclusions and Future Work

We present RTRD to reliably deliver real-time data in ad-hoc networks. RTRD uses proactive routing protocol DSDV to obtain immediate path, and timely delivers data through a priori bandwidth reservation. In addition, it simultaneously delivers data to achieve fault tolerance or large data chunk delivery. The simulation results validate RTRD's effectiveness.

Our future work including the study of transmission fairness on neighbor nodes, and conducting RTRD experiments on IEEE 802.15.4-based wireless sensor networks.

References

[1] OMG, "Real-time corba 2.0: Dynamic scheduling specification," Tech. Rep., OMG, September 2001.

[2] B. Reistad and D. K. Gifford, "Static dependent costs for estimating program execution time," in *ACM Conference on Lisp and Functional Programming*, 1994.

[3] P. Giusto et al., "Reliable estimation of execution time of embedded software," in *DATE*, 2001.

[4] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *ACM SIGCOMM*, December 1999, pp. 234–244.

[5] R. Guimaraes and L. Cerda, "Bandwidth reservation on wireless networks with rts/cts signalling," in *12th European Wireless Conference (EW06)*, April 2006.

[6] T. He et al., "Speed: A stateless protocol for real-time communication in sensor networks," in *ICDCS03*, May 2003.

[7] B. D. Bui et al., "Soft real-time chains for multi-hop wireless ad-hoc networks," in *RTAS07*, 2007, pp. 69–80.

[8] ANSI/IEEE Std. 802.11 1999 Edition (R2003), "Part 11: Wireles lan medium access control (mac) and physical layer (phy) specifications," Tech. Rep., Institute of Electrical and Electronic Engineers, August 1999.