# Reversing the Collision-Avoidance Handshake in Wireless Networks

J.J. Garcia-Luna-Aceves and Asimakis Tzamaloukas *
Computer Engineering Department
Baskin School of Engineering
University of California, Santa Cruz, California 95064
{jj, jamal}@cse.ucsc.edu

**Abstract**

Many medium-access control (MAC) protocols for wireless networks proposed or implemented to date are based on collision-avoidance handshakes between sender and receiver. In the vast majority of these protocols, including the IEEE 802.11 standard, the handshake is sender initiated, in that the sender asks the receiver for permission to transmit using a short control packet, and transmits only after the receiver sends a short clear-to-send notification. We analyze the effect of reversing the collision-avoidance handshake, making it receiver initiated and compare the performance of a number of these receiver-initiated protocols with the performance of protocols based on sender-initiated collision avoidance. The receiver-initiated protocols we present make use of carrier sensing, and are therefore applicable to either baseband or slow frequency-hopping radios in which an entire packet can be sent within the same frequency hop (which is the case of FHSS commercial radios that support IEEE 802.11). It is shown that the best-performing MAC protocol based on receiver-initiated or sender-initiated collision avoidance is one in which a node with data to send transmits a dual-purpose small control packet inviting a given neighbor to transmit and asking the same neighbor for permission to transmit.

## 1 Introduction

There is a large body of work on the design of MAC (medium access control) protocols for wireless networks with hidden terminals. Kleinrock and Tobagi [7] identified the hidden-terminal problem of carrier sensing, which makes carrier-sense multiple access (CSMA) perform as poorly as the pure ALOHA protocol when the senders of packets cannot hear one another and the vulnerability period of packets becomes twice a packet length. The BTMA (busy tone multiple access) protocol was a first attempt to solve the hidden-terminal problem by introducing a separate busy tone channel [12]. The same authors proposed SRMA (split-channel reservation multiple access) [13], which attempts to avoid collisions by introducing a control-signal handshake between the sender and the receiver. A station that needs to transmit data to a receiver first sends a request-to-send (RTS) packet to the receiver, who responds with a clear-to-send (CTS) if it receives the RTS correctly. A sender transmits a data packet only after receiving a CTS successfully. ALOHA or CSMA can be used by the senders to transmit RTSs.

Several variations of this scheme have been developed since SRMA was first proposed, including MACA [6], MACAW [1], IEEE 802.11 [5], and FAMA [3]. These examples of MAC protocols, and most protocols based on collision-avoidance handshakes to date are sender-initiated, in that the node wanting to send a data packet first transmits a short RTS asking permission from the receiver. In contrast, in the MACA by invitation (MACA-BI) protocol [11], the receiver polls one of its neighbors asking if it has a data packet to send. A receiver-initiated collision avoidance strategy is attractive because it can, at least in principle, reduce the number of control packets needed to avoid collisions. However, as we show in this paper, MACA-BI cannot ensure that data packets never collide with other packets in networks with hidden terminals.

In this paper, we present MAC protocols with receiver-initiated collision avoidance that do provide *correct collision avoidance*, i.e., prevent data packets addressed to a given receiver from colliding with any other packets at the receiver. We analyze the effect of reversing the collision-avoidance handshake used to eliminate the hidden-terminal problem of carrier sensing. Our study of receiver-initiated collision avoidance focuses on single-channel networks with asynchronous transmissions, but many of our results extrapolate to networks with multiple channels.

The key contributions of this paper are recasting collision avoidance dialogues as a technique that can be controlled by senders, receivers, or both; showing that receiver-initiated collision avoidance can be even more efficient than sender-initiated collision avoidance; and presenting a method for proving that a receiver-initiated collision avoidance strategy works correctly.

We use a fully-connected network topology to discern the relative performance advantages of different protocols. We opted to focus on fully-connected networks in our analysis because of two reasons: (a) it allows us to use a short analysis that can be applied to several protocols; and (b) our focus on protocols that provide correct collision avoidance means that the relative performance differences in a fully-connected network are very much the same when networks with hidden terminals are considered. In particular, results presented for FAMA protocols [2, 3] indicate that, in a network with hidden terminals, the performance of a MAC protocol with correct collision avoidance is almost identical to the performance of the same protocol in a fully-connected network if the vulnerability period of a control packet is made proportional to the length of the entire packet. This is intuitive, if a MAC protocol prevents data packets from colliding with other packets in any type of topology, hidden terminals can degrade the protocol's performance from that obtained in a fully-connected network only to the extent that control packets used to prevent data collisions are subject to

*This work was supported by the Defense Advanced Research Projects Agency (DARPA) under Grant No. F30602-97-2-0338.


Mobicom '99 Seattle Washington USA

additional interference caused by the fact that nodes cannot sense the transmissions of control packets by hidden sources.

The receiver-initiated protocols we introduce in this paper require that nodes accomplish carrier sensing. This can be done with baseband radios and today's commercial slow frequency hopping radios, in which complete packets are sent in the same frequency hop. The receiver-initiated protocols we present, as well as the sender-initiated protocols introduced in the past based on carrier sensing and a single channel (e.g., FAMA [3]), do not really apply to DSSS (direct sequence spread spectrum) radios, because radios capture none or one of multiple overlapping transmissions depending on the proximity and transmission power of the sources. Fortunately, there are many commercial radios, specially at the 2.4GHz band, which can make use of our collision-avoidance approach.

Section 2 introduces fundamental aspects of receiver-initiated collision-avoidance handshake, and Section 3 presents a number of MAC protocols based on receiver-initiated collision-avoidance. Section 4 proves that, in the absence of fading, all these protocols solve the hidden-terminal problem, i.e., they eliminate collisions of data packets. Section 5 analyzes the throughput of these protocols in fully-connected networks. Our analysis shows that receiver initiated multiple access with dual-use polling (RIMA-DP) is the most efficient approach among all the sender- and receiver-initiated MAC protocols proposed to date for single-channel networks with asynchronous transmissions.

## 2 Receiver-Initiated Collision Avoidance

Critical design issues in receiver-initiated MAC protocols over a single channel are: (a) whether or not to use carrier sensing, (b) how to persist transmitting packets, (c) how to resolve collisions, and (d) deciding how a receiver should poll its neighbors for data packets.

Carrier sensing has been shown to increase the throughput of sender-initiated collision avoidance tremendously [2]; furthermore, carrier sensing has also been shown to be necessary to avoid collisions of data packets in sender-initiated collision avoidance over single-channel networks in which transmissions occur in an asynchronous way, i.e., without time slotting [3].

We describe all receiver-initiated schemes assuming carrier sensing and asynchronous transmissions. To simplify the analysis of the protocols, we also assume non-persistent carrier sensing, which has been shown to provide better throughput characteristics than persistent disciplines for CSMA and CSMA/CD [8] at high loads. Furthermore, our treatment of receiver-initiated collision avoidance assumes simple back-off strategies; however, the benefits of using sophisticated back-off strategies or collision resolution algorithms has been analyzed for a number of sender-initiated MAC protocols [1, 4], and it should be clear that the same schemes could be adopted in any of the receiver-initiated approaches we address in this paper.

In sender-initiated collision avoidance, a node sends a request-to-send packet (RTS) whenever it has data to send and, in protocols using carrier sensing, the channel is free. However, deciding how to send polling packets in receiver-initiated protocols is not as immediate as sending transmission requests in sender-initiated protocols; furthermore, as we show in this paper, the polling discipline chosen determines to a large extent the performance of the protocol. A polling rate that is too small renders low throughput and long average delays, because each sender with a packet to send is slowed down by the polling rate of the receiver. Conversely, a polling rate that is too high also renders poor performance, because the polling packets are more likely to collide with each other and no source gets polled.

The polling discipline used in a receiver-initiated MAC protocol can be characterized by three different factors:

- Whether or not the polling rate is independent of the data rate at polling nodes,
- Whether the poll is sent to a particular neighbor or to all neighbors,
- Whether the polling packet asks for permission to transmit as well.

In terms of the relationship between the polling rate and the data rate, we can categorize polling disciplines in two major classes: independent polling and data-driven polling.

With independent polling, a node polls its neighbors at a rate that is independent of the data rate of the node or the perceived data transmission rate of its neighbors. In contrast, with data-driven polling, a node attempts to poll its neighbors at a rate that is a function of the data rate with which it receives data to be sent, as well as the rate with which the node hears its neighbors send control and data packets. The specification of the MACA-BI protocol by Talucci et al. [11] assumes this type of polling. Throughout the rest of the paper, we assume data-driven polling, because it is very difficult in a real network to determine a good independent polling rate by the receivers, and because data-driven polling is far simpler to analyze.

In practice, to account for data rate differences at nodes and to eliminate the possibility of a data-driven polling discipline never allowing a node to receive data, a protocol based on data-driven polling should send a poll based on its local data to be sent or after a polling timeout elapses without the node having any packet to send to any neighbor.

The intended audience of a polling packet can be a single neighbor, a subset of neighbors, or all the neighbors of a node. A large audience for a poll packet introduces the possibility of contention of the responses to the poll, and either the collisions of responses need to be resolved, or a schedule must be provided to the poll audience instructing the neighbors when to respond to a poll.

The intent of a polling packet can be simply to ask one or more neighbors if they have data to send to the polling node, or it can both ask for data and permission to transmit in the absence of data from the polled neighbors. Intuitively, the latter approach should have better channel utilization, because data will be sent after every successful handshake, and more data per successful handshake are sent as traffic load increases even if the polled node does not have data for the polling node. We also note that a polling packet asking for data from a neighbor could allow the polled node to send data to *any* destination, not just to the polling node; however, this strategy would not work efficiently in multihop networks, because there is no guarantee that the recipient of a data packet who did not ask for it will receive the transmission in the clear.

It is clear that polls that specify transmission schedules can address the three key functions of a polling discipline that we have just discussed. In this paper, however, we concentrate on single-node polling and broadcast polling only. Receiver-initiated protocols based on schedules is an area of future research.

## 3 Receiver-Initiated Protocols

This section introduces new MAC protocols based on receiver initiated collision avoidance and relate them to the taxonomy of polling disciplines presented in Section 2. To our knowledge, these protocols are the first based on receiver-initiated collision avoidance that eliminate the collisions of data packets with any other control or data packets in the presence of hidden terminals.

For simplicity, we describe the new MAC protocols without the use of acknowledgments (ACKs); in practice, ACKs will be used. However, it should be clear that, because the protocols support correct collision avoidance, an acknowledgment to each data packet

can be sent collision-free by the receiver immediately after it processes the data packet. The only caveat is that the time that a node must back off to let data flow without collisions must include the time needed for the sender to receive the acknowledgment in the clear.

### 3.1 Protocols with Simple Polling

#### 3.1.1 MACA-BI

The original MACA-BI [11] protocol uses a ready-to-receive packet (RTR) to invite a node to send a data packet. A node is allowed to send a data packet only if it has previously received an RTR, whereas a node that receives an RTR that is destined to a different node has to back off long enough for a packet to be sent in the clear.

According to the description of MACA-BI, a polled node can send a data packet intended to the polling node or any other neighbor. In a fully-connected network, whether the data packet is sent to the polling node or not is not important, because all the nodes must back off after receiving an RTR in the clear. However, this is not the case in a network with hidden terminals.

By means of two simple examples, we can show that MACA-BI does not prevent data packets sent to a given receiver from colliding with other data packets sent concurrently in the neighborhood of the receiver. The first example illustrates the fact that, in order to avoid the transmission of data packets that the intended receiver cannot hear because of other colliding data packets, a polled node should send data packets only to the polling node. The second example illustrates the possibility that collisions of data packets at a receiver may occur because the receiver sent an RTR at approximately the same time when data meant for another receiver starts arriving.

In Fig. 1, nodes $a$ and $d$ send RTRs to nodes $b$ and $e$ at time $t_0$, respectively. This prompts the polled nodes to send data packets at time $t_1$; the problem in this example occurs when at least one of the polled nodes sends a data packet addressed to $c$, which cannot hear either packet.

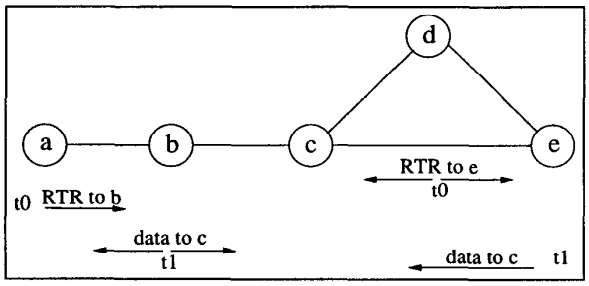


Figure 1: Data packets colliding in MACA-BI when packet is not sent to polling node

In the example shown in Fig. 2, node $a$ sends an RTR to $b$ at time $t_0$. This RTR makes node $b$ start sending data to node $a$ at time $t_1$ which in order to provide good throughput must be larger than $\gamma$ seconds, where $\gamma$ is the length of an RTR. At time $t_2$ node $c$ starts sending an RTR to node $d$. Because of carrier sensing, $t_2$ must be within $\tau$ seconds (maximum propagation delay) of $t_1$. In this example, after receiving node $c$'s RTR, node $d$ replies with data that must start arriving at node $c$ at time $t_3$. Because the maximum propagation delay is $\tau$, it must be true that $t_3 \leq t_2+\gamma+2\tau \leq t_1+\gamma+3\tau$. Hence, if data packets last longer than $\gamma+3\tau$ seconds, the data packets from $b$ and $d$ collide at node $c$. In practice, data packets must be much longer than RTRs to provide good throughput, and it thus follows that MACA-BI cannot prevent all data packets from experiencing collisions.

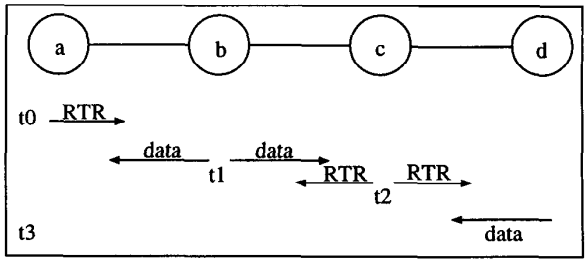


Figure 2: Data packets colliding in MACA-BI due to RTR not being heard

#### 3.1.2 RIMA-SP

The above problems in MACA-BI went unnoticed in the specification by Talucci et al. [11]. To make the RTR-data handshake in MACA-BI collision free, the following two minor modifications are required:

- The polled node should transmit data packets only if they are addressed to the polling node.
- A new control signal is also required, which we call No-Transmission-Request (NTR), and an additional collision-avoidance waiting period of $\xi$ seconds is required at a polled node prior to answering an RTR. During that period, if any channel activity is heard, the receiver (polling node) that originated an RTR sends an NTR telling the polled node not to send any data. Otherwise, if nothing happens during the waiting period, the polled sender transmits its data, if it has any to send to the polling node.

We call the protocol resulting from modifying MACA-BI with the above two rules RIMA-SP (receiver initiated multiple access with simple polling). Fig. 3 illustrates the operation of RIMA-SP. The complete proof that RIMA-SP provides correct collision avoidance when $\xi = \tau$ is given in Section 4.

In RIMA-SP, every node initializes itself in the START state, in which the node waits twice the maximum channel propagation delay, plus the hardware transmit-to-receive transition time ($\epsilon$), before sending anything over the channel. This enables the node to find out if there are any ongoing transmissions. After a node is properly initialized, it transitions to the PASSIVE state. In all the states, before transmitting anything to the channel, a node must listen to the channel for a period of time that is sufficient for the node to start receiving packets in transit.

If a node $x$ is in the PASSIVE state and senses carrier, it transitions to the REMOTE state to defer to ongoing transmissions. A node in REMOTE state must allow enough time for a complete successful handshake to take place, before attempting to transition from remote state.

Any node in PASSIVE state that detects noise in the channel must transition to the BACKOFF state. If node $x$ is in PASSIVE state and obtains an outgoing packet to send to neighbor $z$, it transitions to the RTR state. In the RTR state, node $x$ uses non-persistent carrier sensing to transmit an RTR. If node $x$ detects carrier when it attempts to send the RTR, it transitions to the BACKOFF state, which makes the node back off immediately for a sufficient amount of time to allow a complete handshake between a sender-receiver pair to occur; otherwise, $x$ sends its RTR.

If node $z$ receives the RTR correctly and has data for $x$, it waits for $\xi$ seconds. If during the waiting period there is no activity in the channel, node $z$ transitions to the XMIT state, where it transmits a data packet to $x$ (Fig. 3(a)); otherwise, node $z$ assumes that

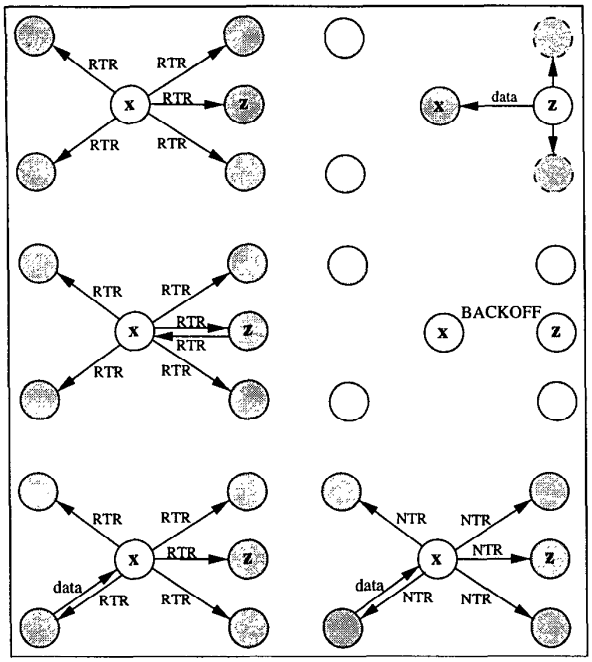


Figure 3: RIMA-SP illustrated

there was a collision and transitions to the BACKOFF state to allow floor acquisition by some other node. After sending its RTR, node $x$ senses the channel. If it detects carrier immediately after sending its RTR, node $x$ assumes that a collision or a successful data transfer to a hidden node is taking place. Accordingly, it sends a No transmission Request (NTR) to $z$ to stop $z$ from sending data that would only collide at x (Fig. 3(b)).

When multiple RTRs are transmitted within a one-way propagation delay a collision takes place and the nodes involved have to transition to the BACKOFF state and try again at a later time chosen at random, as shown in Fig. 3(b).

Node $x$ determines that its RTR was not received correctly by $z$ after a time period equal to the maximum round-trip delay to its neighbors plus turn-around times and processing delays at the nodes, plus the waiting period $\xi$. After sending its RTR, node $x$ listens to the channel for any ongoing transmission. Because of non zero propagation delays, if node $x$ detects carrier immediately after transmitting its RTR, it can conclude that it corresponds to a node other than $z$, which would take a longer time to respond due to its need to delay its data to $x$ to account for turn-around times. [1]

The lengths of RTRs and NTRs are the same. The same argument used in [2] to show that the length of an RTS must be longer than the maximum propagation delay between two neighbors to ensure correct collision avoidance can be used to show that RTRs and NTRs must last longer than a maximum propagation delay. In ad-hoc networks in ISM bands, propagation delays are much smaller compared with any packet that needs to be transmitted.

To reduce the probability that the same nodes compete repeatedly for the same receiver at the time of the next RTR, the RTR specifies a back-off-period unit for contention. The nodes that must enter the BACKOFF state compute a random time that is a multiple of the back-off-period unit advertised in the RTR. The simplest case consists of computing a random number of back-off-period units using a uniformly distributed random variable from 1 to $d$, where $d$ is the maximum number of neighbors for a receiver. The simplest back-off-period unit is the time it takes to send a small data packet successfully.

[1] Our analysis assumes 0 turn-around times and 0 processing delays for simplicity.

### 3.2 Protocols with Dual-Use Polling

The collision avoidance strategy described for RIMA-SP can be improved by increasing the probability that data will follow a successful RTR, without violating the rule that data packets should be transmitted only if they are addressed to the polling nodes. A simple way to achieve this with data-driven polling is to make an RTR entry both a request for data from the polled node, and a transmission request for the polling node to send data. The RIMA-DP (receiver-initiated multiple access with dual-purpose polling) protocol does exactly this. Fig. 4 illustrates the modified collision avoidance handshake to permit the polling node to either receive or send data without collisions.

As Fig. 4(a) illustrates, a key benefit of the dual-use polling in RIMA-DP is that both polling and polled nodes can send data in a round of collision avoidance. This is possible because the RTR makes all the neighbors of the polling node back-off, and the data from the polled node make all its neighbors back-off, which can then be used by the polling node to send its data.

RIMA-DP gives transmission priority to the polling nodes. When a node $z$ is polled by node $x$ and has data for node $x$, $z$ waits $\xi$ seconds before sending a data packet. In contrast, if the polled node does not have data for $x$, it immediately sends a CTS (Clear-To-Send packet) to $x$. This permits a polling node $x$ exposed to a neighbor sending data to hear part of that neighbor's data packet after sending its RTR; in such a case, node $x$ can send an NTR to the polled node to cancel its RTR. Section 4 shows that this prevents collisions of data packets, provided that $z$ waits for $\xi > \gamma + 7\tau$ seconds before sending any data after being polled and the length of a CTS is $2\tau$ seconds longer than the length of an RTS. As in RIMA-SP, the lengths of RTRs and RTSs are the same.

As in RIMA-SP, every node starts in the START state and transitions to to the PASSIVE state when it is initialized. If a node $x$ is in the PASSIVE state and senses carrier, it transitions to the REMOTE state to defer to ongoing transmissions. A node in REMOTE state must allow enough time for a complete successful handshake to take place, before attempting to transition from remote state.

Any node in PASSIVE state that detects noise in the channel must transition to the BACKOFF state where it must allow sufficient time for complete successful handshakes to occur. If node $x$ is in PASSIVE state and obtains an outgoing packet to send to neighbor $z$, it transitions to the RTR state. In the RTR state, node $x$ behaves as in RIMA-SP.

If node $z$ receives the RTR correctly and has data for $x$, it waits for $\xi$ seconds before sending a data packet to $x$. If during the waiting period there is no activity in the channel, node $z$ transitions to the XMIT state, where it transmits a data packet to $x$. Otherwise, $z$ assumes a collision or data transfer to a hidden node and goes to the BACKOFF state. If $z$ has no data for $x$, it sends a CTS to $x$ immediately.

If node $x$ detects carrier immediately after sending an RTR, it defers its transmission attempt and sends an NTR to the node it polled. The CTS length, which is $\tau$ seconds longer than an RTR, forces polling nodes that send RTRs at about the same time when a polled node sends a CTS to detect carrier from the CTS and stop their attempt to send or receive data. Any node other than $x$ receiving the CTS for $x$ transitions to the BACKOFF state. When node $x$ receives the CTS from $z$, it transitions to the XMIT state and transmits a data packet to $z$.

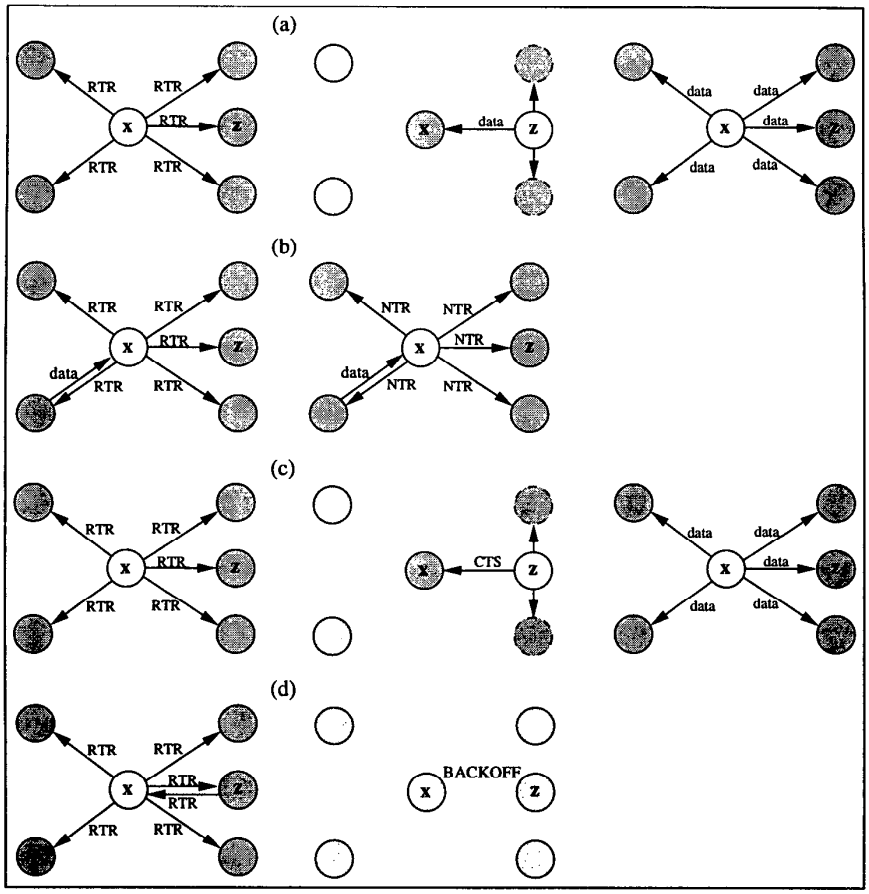


Figure 4: RIMA-DP illustrated

### 3.3 Protocols with Broadcast Polling

Contrary to the prior two approaches, an RTR can be sent to multiple neighbors. We describe a modification of RIMA-SP based on this variant.

A node broadcasts an RTR only when there is a local data packet (data-driven polling). Only after a node has received an invitation, it is allowed to send any data. Because a poll broadcast to all the neighbors of a node can cause multiple nodes to attempt sending data to the polling node, an additional control packet is needed to ensure that transmissions that collide last a short period and do not carry user data. Accordingly, a polled node sends a short RTS (Ready-To-Send packet) before sending data. Furthermore, after sending its RTS, the polled node must wait for $\xi$ seconds to allow the polling node to send an NTR when collisions of RTSs occur at the polling node. We call this protocol RIMA-BP (Broadcast Polling).

It can be shown that RIMA-BP provides correct collision avoidance if $\xi = 4\tau$. Fig. 5 illustrates the receiver-initiated handshake of RIMA-BP. As it is shown in the figure, the key difference with RIMA-SP is the use of an RTS prior to the transmission of a data packet.

## 4 Correct Collision Avoidance in RIMA protocols

Theorems 1 and 2 below show that RIMA-SP and RIMA-DP ensure that there are no collisions between data packets and any other transmissions. A similar proof to that of Theorem 1 can be used to show that RIMA-BP provides correct collision avoidance if $\xi = 4\tau$. The following assumptions are made to demonstrate correct collision avoidance in RIMA protocols [3]:

A0) A node transmits an RTR that does not collide with any other transmissions with a non-zero probability.

A1) The maximum end-to-end propagation time in the channel is $\tau < \infty$.

A2) A packet sent over the channel that does not collide with other transmissions is delivered error free with a non-zero probability.

A3) All nodes execute a RIMA protocol correctly.

A4) The transmission time of an RTR and a CTS is $\gamma$, the transmission time of a data packet is $\delta$, and the hardware transmit-to-receive transition time is zero; furthermore, $2\tau < \gamma \leq \delta < \infty$.

A5) There is no capture or fading in the channel.

DOCKET ALARM

# Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

## Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

## Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

## Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

## API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

**LAW FIRMS**
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

**FINANCIAL INSTITUTIONS**
Litigation and bankruptcy checks for companies and debtors.

**E-DISCOVERY AND LEGAL VENDORS**
Sync your system to PACER to automate legal marketing.