Multiple Polling Slots for RICH RTR Packets
CMPE 255 Class Project

Marc Mosko

June 6, 2000

Abstract

The RICH wireless protocol [4] use receiver initiated polling of known sources on a common wireless channel. Since RICH does not use carrier sense, the common polling channel functions like a slotted ALOHA system. We study two techniques to boost polling success rate. First, we introduce a new scheme of listing multiple receivers in a single Request to Receive (RTR) packet. Second, we examine through simulation the effects of Dual Poll (DP) mode in the RICH protocols. Our analysis also presents two Discrete Time Markov models of the original RICH protocols.

We find that multiple scheduling slots does not have a significant affect on the number of concurrent conversations. Multiple slots may have beneficial effects on the mean waiting time of the transmit queue when packets are short. One of our Markov models has a good approximation of simulation for short packet lengths.

1 Introduction

The RICH protocols [4] seek to exploit session concurrency in spread spectrum communication by pairing transmitters and receivers on orthogonal frequency patters. In this study, we assume that each transmitter/receiver pair may join on a private frequency or hop pattern for the duration of their data exchange. We have not addressed the important issue of frequency reuse limits or how to coordinate orthogonal frequency usage.

In the RICH protocols, time is divided in to slots. The duration of each slot is sufficient to transmit and receive a single RTR. Data packets are assumed to last an integral number of slots. The slot time is equal to the dwell time on any one frequency. Nodes coordinate their transmissions with their hop patterns.

All nodes know a common hop sequence. This common channel is the signalling channel. Receivers use it to poll transmitters. At certain intervals, receivers poll their neighbors to see if the neighbors have data to send. This receiver initiated polling avoids data packet collisions. Since RTRs are much smaller than data packets, the collision window and backoff times are proportionally smaller.

After a receiver transmits an RTR, it hops to a frequency specified in the RTR packet next slot. If it hears data from the transmitter, it knows the RTR was successful. If there is no data, it concludes that either the RTR collided. The polling node backs off a time drawn from an exponential distribution with parameter β. If one uses NACKs, then the polled node could transmit a one slot NACK to prevent the polling node from backing off.

RICH also has a mode called Dual Poll (RICH-DP). In this mode, a polling receiver may also indicate that it has data to send. The polled node may reverse the sense of the conversation and allow the polling node to transmit data.

This scheme is essentially a slotted ALOHA [2] shared channel for RTR packets. It will have a maximum throughput of 0.38 when the RTR rate is $1/N$, where N is the number of nodes. We let λ be the RTR arrival rate and γ be the probability that a polled node has a packet for a polling node. We can put a loose bound the RTR goodput as $S_{rtr} = \gamma \cdot \lambda e^{\lambda N}$. This is the probability that a node successfully transmits an RTR on the shared slotted ALOHA channel and that the receiver has a packet for the transmitter. It ignores the possibility that the polled node may be busy in data transfer with another node. As nodes become back logged, γ quickly becomes 1.

Our innovation to the RICH protocols is to extend the RTR packet to include an ordered list of possible transmitters. After transmitting an RTR, the polling node goes to the private hop frequency. If it hears data next slot, it continues receiving. If there was no data, on the next slot it will transmit an gratuitous RTR for the 2nd node listed in the original RTR. If the second node had data and was available, it would hear the second RTR and begin transmitting on the following slot. The polling node repeats this for all nodes listed in the original RTR until there are node more nodes or one responds with data.

What the following sections will show is that using two to three RTR slots can increase RICH-SP and RICH-DP performance by a small amount, depending on the configuration. Listing more than three nodes in a single RTR packet becomes counter productive. Polling nodes spend too much time trying to acquire a node from their RTR packet. Under high load conditions, it is most likely that the RTR packet collided and the polling node is just wasting time trying to hear a transmitter. In general, under our simulation conditions of Poisson arrivals, the improvements of multiple RTR slots fall within statistical uncertainty and therefore do not exhibit significant system improvement.

Multiple scheduling slots can make a statistically significant difference in the mean waiting time of the transmit queue for shorter packet lengths when the system is backlogged. When the system is not backlogged or when the system uses long packets, there is negligible difference between one and multiple scheduling slots.

2 K-ary RTR Model

Fig. 1 shows the general state machine for our implementation of the RICH protocols. A node may be in Idle, Receive or Transmit mode. Each of Receive and Transmit are further refined in Figs. 2 and 3 respectively. Backoff mode is part of the Receive state. The state diagrams presented in this section are general to RICH-SP, RICH-DP, K-ary RICH, and K-ary RICH with NACK, or any combination thereof.

Each node has two queues. The RTRQ is a FIFO queue for transmitting RTR packets. The XMITQ is FIFO per destination for transmitting data packets. At the beginning of each slot, each node scans the XMITQ. If it finds a destination for which there is no packet in the RTRQ, it will append an RTR packet to the RTRQ with probability ϕ. We experimented with different values of ϕ and found they made little overall difference. All simulation runs used $\phi = 1$.

If there are any items in the RTRQ, a node will transition in to Receive mode. The node will create an RTR packet with up to k scheduling slots (sslots). It will pop the first k requests from the RTRQ and place them in the RTR packet, which it then sends. The

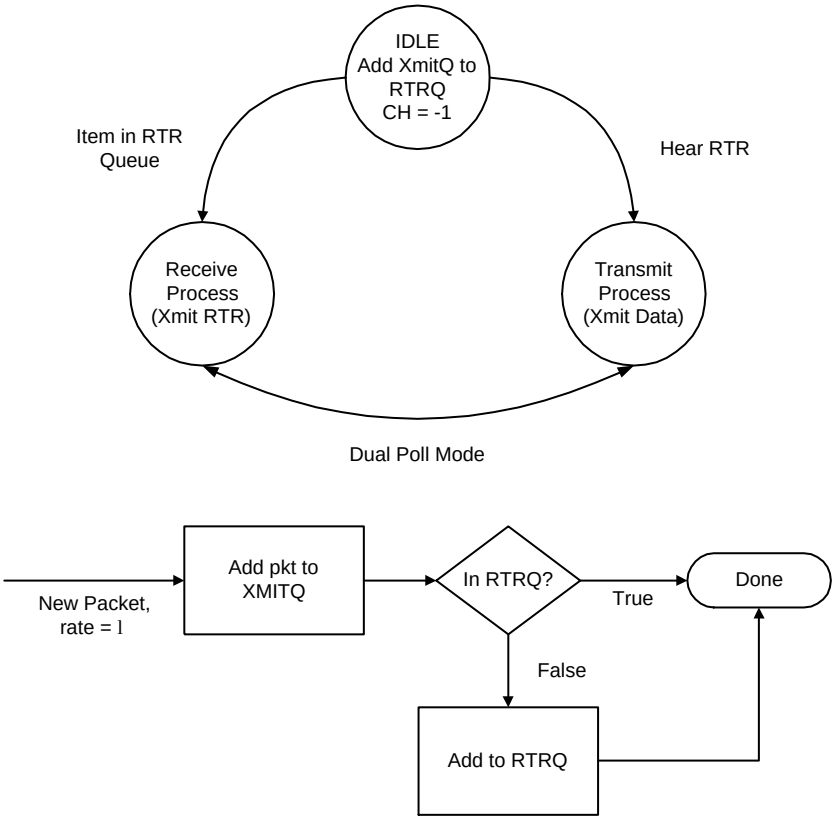

Figure 1: General State Model and New Packet Process

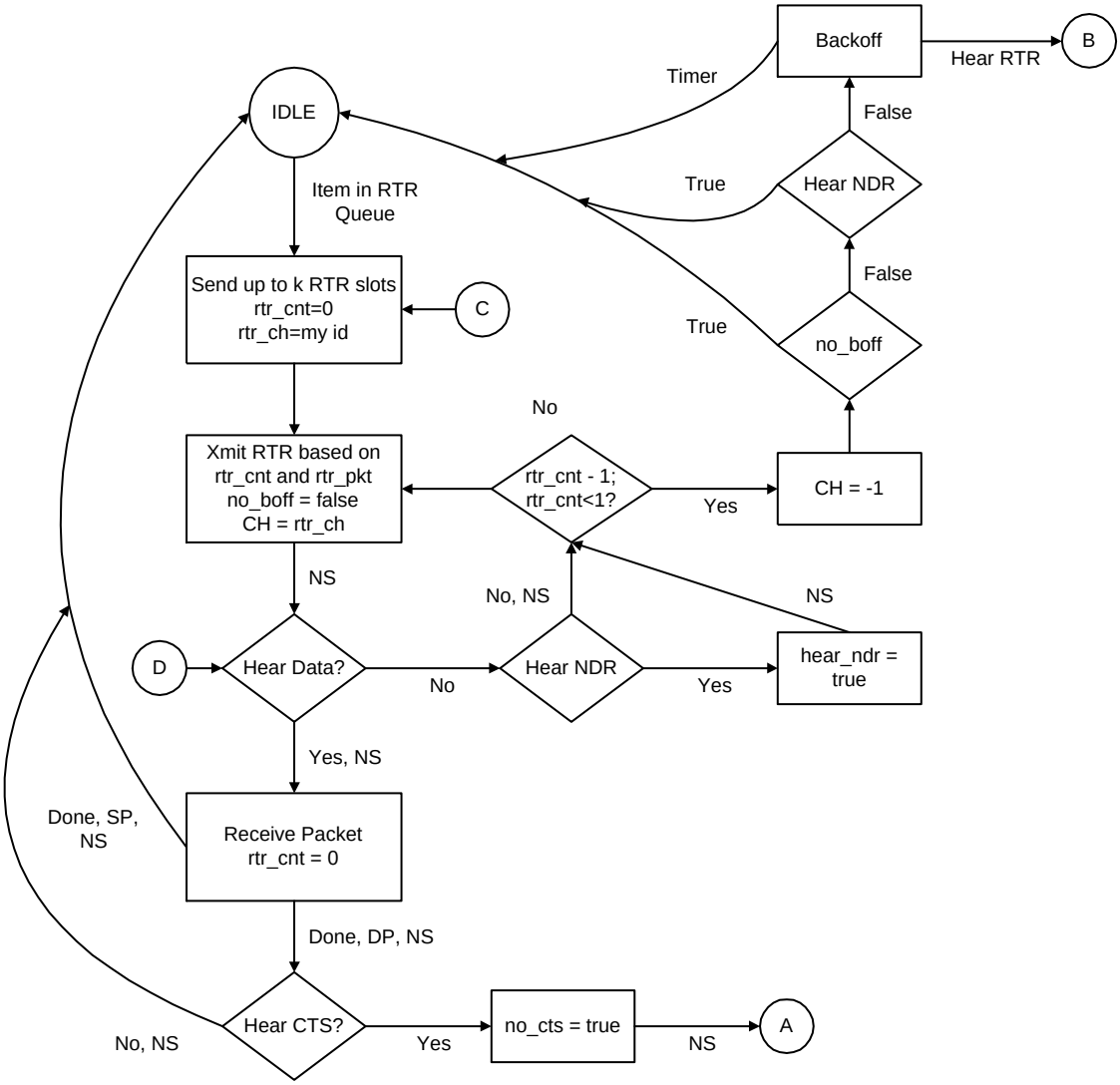

Figure 2: Receiver Process

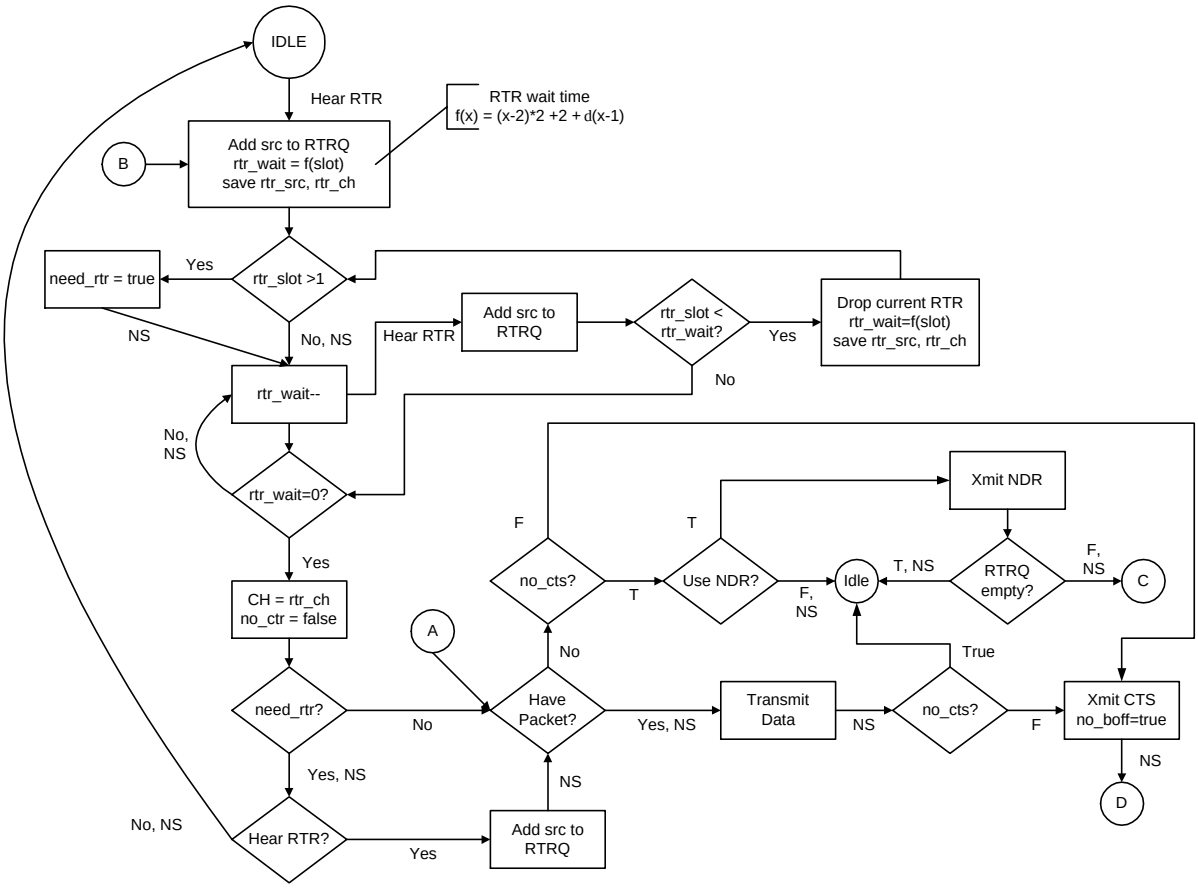

Figure 3: Transmitter Process

variable $rtr\_cnt = k$.

The Receiving node then enters a loop. If it hears data on the next slot, it begins receiving the packet. If it hears a NACK or no data, it loops for up to $rtr\_cnt - 1$ iterations. For each scheduling slot beyond the first, the Receiver will transmit an RTR on the private channel. This informs the waiting nodes that they may transmit. Since waiting nodes may be hidden from one another, the Receiver must continue to schedule the multiple transmitters.

If the receiver does not hear a resonse from any polled node, it enters an exponential backoff period. While in backoff, no new packets are generated. The length of the backoff period is selected from a negative exponential distribution with parameter β.

A polled node obeys the following rules. When it receives an RTR addressed to it, the

node transitions to Transmit mode. If it was listed in scheduling slot 1, it should begin transmission on the next slot. It will either send data or an NACK, if NACKs are enabled. If the polled node was listed a scheduling slot greater than 1, it will stay on the common channel until it's transmission slot. When the transmission slot occurs ($rtr\_wait = 0$), the Transmitting node will switch to the private channel and listen for an RTR. If it does not hear the RTR, the receiver is most likely in communication with another node. The transmitter will switch to idle mode on the next slot. If the transmitter hears an RTR on the private channel addressed to it, it will begin transmitting data on the next slot.

While waiting for it's second RTR, another Receiver may preempt the pending RTR with an RTR scheduled sooner than the remaining waiting time. If this happens, the pending RTR is dropped and the new RTR becomes the current RTR. An improvement would be to queue multiple RTRs and service them in order, similar to how a Receiver sequences through Transmitters until it finds one.

If using Dual-Poll (DP) mode, a Transmitter may reverse the data flow direction by transmitting a CTS. In this case, the Transmitting node must set the variable $no\_boff = true$. When the node switches from Transmit mode to Receive mode, no_boff will prevent it from going in to exponential backoff if it does not hear a data packet. If NACKs are enabled, this variable is not absolutely necessary.

We also experimented with using NACK packets. If a Transmitter does not have a packet for a receiver, it may transmit either a CTS, if using DP, or a NACK. This allows the Receiver to know that it's RTR was received and did not collide. A receiver that hears a NACK will not go in to Backoff mode. Out simulations did not use NACK packets since they were not specified in the RICH description. We did compare using NACKs in a few configurations and found they made little difference to overall system performance.

In our models, the RTR distribution is a composite of the Poisson new packet arrivals and the exponential backoff. Depending on queuing, a node's RTR ready rate will vary. Under light loads with little queuing, the RTR ready rate is dominated by the Poisson arrivals. Under heavy load conditions when the node is perpetually backlogged, it is dominated by

the exponential backoff. Around the knee of the graph, the rate will oscillate between the Poisson arrivals and the exponential backoff. Since this will lead to nodes having unequal arrivals rates, the original estimate of $S_{rtr} \approx \lambda e^{\lambda N}$ is too low [1].

3 Markov Models

We developed two discrete time Markov models for the RICH protocols. Our initial inspiration was a paper by Souza and Silvester [3]. Like Souza and Silvester, our performance metric is the number of concurrent transmissions. We note that our model yields different results since ours accounts for nodes in backoff mode. In Model 2, nodes that exit backoff mode immediately become ready with an RTR.

We use three state variables: the number of transmitting pairs n, the number of nodes with an RTR packet m, and the number of nodes in backoff b. The probability p_{nmb} is the probability of being in a given state. We present the single step transition equations for movement from state (n, m, b) to (n', m', b'). We denote the transition probability as $p_{nmb,n'm'b'}$. It is the probability that a system in state (n, m, b) at the end of slot t will be in state (n', m', b') at the beginning of slot $t + 1$.

Our models are for RICH-SP. They do not yet include DP mode or multiple RTR transmitter slots.

Model 1 obeys the following rules. RTRs arrive at idle nodes with rate λ in a Poisson process where only 0 or 1 RTR may arrive per slot per node. Data packets have geometric length with parameter α. A pair of communicating nodes, when they finish, will go to the Idle mode. After 1 slot, they may then become ready. A polled node has a packet for the polling node with probability γ. If $m = 1$, then the RTR may succeed or fail. If $m > 1$, all RTRs fail and move to backoff mode. Nodes in idle or backoff mode may receive an RTR. Once nodes complete their backoff period, they become idle.

We had difficulty modeling the negative exponential distribution for backoff duration. A negative exponential distribution returns a variate for the number of slots to remain in backoff mode. The probability that b nodes transition to b' nodes in a slot is $\binom{b}{b'}\beta^{b-b'}e^{b}$, where

$1/\beta$ is the negative exponential parameter. This distribution is correct over a renewal cycle, but does not normalize per slot. Instead, we used a geometric backoff length with parameter β. The two distributions have the same mean, but their variances are substantially different.

Model 2 is the same as Model 1, except nodes that finish their backoff period go directly to RTR ready mode.

We divide our state space in to eight parts. Each part has acceptance rules on the state (n, m, b). Each rule will specify a valid range for (n', m', b') and the transition probabilities.

A node in backoff mode will leave backoff mode with probability βe^{β} per slot. A pair of nodes in data exchange will leave that mode with probability $(1 - \alpha)$ per slot. A node will have exactly 1 RTR arrival with probability λe^{λ} per slot. It will have 0 RTR arrivals with probability $1 - \lambda e^{\lambda}$.

$\delta(0) = 1$ or 0 otherwise.

Define the function $B(i, j, p) = \binom{i}{j} p^j (1-p)^{i-j}$ to be the binomial distribution. Let $\eta = \frac{2n}{N-1} + \frac{N-2n-1}{N-1}(1-\gamma)$. η is the probability of picking a busy node (a node in a conversation) or an available node (in idle or backoff mode) but that node does not have a data packet for the receiver.

The function $P(i, j, \lambda)$ is a Poisson process for j arrivals out of i nodes with rate λ per node. It may have only 0 or 1 arrivals per node per slot. The probability of 1 arrival for a node is $\lambda e^{-\lambda}$. For j arrivals out of i nodes, is the Poisson probability for j arrivals times the probability that $i - j$ nodes have 0 arrivals. The probability of 0 arrivals out of i nodes is 1 minus the sum that at least one node had an arrival.

$$P(i, j, \lambda) = \begin{cases} 1 & i = 0 \\ 1 - \sum_{k=1}^{i} P(i, k, \lambda) & i > 0, j = 0 \\ \binom{i}{j} \lambda^j e^{-i\lambda} \cdot P(i - j, 0, \lambda) & i > 0, j > 0 \end{cases} \quad (1)$$

Using these definitions, we have the following relations per slot. n pairs of nodes in a conversation become n' pairs with probability $B(n, n', \alpha)$. ι idle nodes become m' ready

Rule	Source State	Destination State	Purpose
1	(0,0,b)	(0,m',b')	Nodes become ready
2	(0,1,b)	(0,m',b')	Ready node fails
3	(0,1,b)	(1,m',b')	Ready node succeeds
3a	(0,1,b)	(1,m',b')	Ready node succeeds from idle
3b	(0,1,b)	(1,m',b')	Ready node succeeds from backoff
4	(0,m,b)	(0,m',b')	Multiple RTRs collide
5	(n,0,b)	(n',m',b')	Nodes become ready
6	(n,1,b)	(n',m',b')	Ready node fails
7	(n,1,b)	(n',m',b')	Ready node succeeds
7a	(n,1,b)	(n',m',b')	Ready node succeeds from idle
7b	(n,1,b)	(n',m',b')	Ready node succeeds from backoff
8	(n,m,b)	(n',m',b')	Multiple RTRs collide

Table 1: Transition Rules

nodes with an RTR with probability $P(\iota, m', \lambda)$. b nodes in backoff mode become b' nodes in backoff with probability $B(b, b', \beta)$.

Each model has 8 rules. These rules model the transitions shown in Table 1. Rules 2 and 3 (or 2 and 3a, 3b) always execute together. Rules 6 and 7 (or 6 and 7a, 7b) always execute together. Model 1 uses rule 7. Model 2 uses rules 3a,3b and 7a, 7b. As one can see, rules 1-4 and 5-8 are essentially the same, except for the difference in state n. This overlap reflects our process of deriving the rules. We started with the simpler cases with $n = 0$ then derived the general rules. Not having time to reduce the rule sets, we have left them as possibly redundant.

We solved the Markov model through iterative evaluation of the single step matrix until the eigenvalue varied by less than 0.01% from the previous iteration. Since our transition matrix is generally dense, the solution converges in under 100 iterations.

4 Simulation

The simulations modeled RICH-SP and RICH-DP with and without multiple scheduling slots per RTR. The simulations implemented the state machines presented in Section 2. We

Rule	Acceptance	Output States	State Equation
		$\iota = N - 2n - m - b$	
1	$n = 0$ $0 \le b \le N$ $m = 0$	$n' = 0$ $0 \le b' \le b$ $0 \le m' \le \iota$	$P(\iota, m', \lambda) \cdot B(b, b' - m, \beta)$
2	$n = 0$ $0 \le b \le N - 1$ $m = 1$	$n' = 0$ $1 \le b' \le b + 1$ $0 \le m' \le \iota$	$\eta \cdot P(\iota, m', \lambda) \cdot$ $B(b, b' - 1, \beta)$
3	$n = 0$ $0 \le b \le N - 1$ $m = 1$	$k = \delta(N - 1 - b)$ $n' = 1$ $0 \le b' \le b - k$ $0 \le m' \le \iota - 1 + k$	$[\gamma \frac{\iota}{N-1}] \cdot P(\iota - 1 + k, m', \lambda) \cdot$ $B(b, b', \beta) + [\gamma \frac{b}{N-1}] \cdot B(b - 1, b', \beta) \cdot$ $P(\iota - 1 + k, m', \lambda)$
4	$n = 0$ $0 \le b \le N - m$ $2 \le m \le N$	$n' = 0$ $m \le b' \le b + m$ $0 \le m' \le \iota$	$P(\iota, m', \lambda) \cdot B(b, b' - m, \beta)$
5	$1 \le n \le \lfloor \frac{N}{2} \rfloor$ $0 \le b \le N - 2n$ $m = 0$	$0 \le n' \le n$ $0 \le b' \le b$ $0 \le m' \le \iota$	$P(\iota, m', \lambda) \cdot B(b, b', \beta) \cdot$ $B(n, n', \alpha)$
6	$1 \le n \le \lfloor \frac{N-1}{2} \rfloor$ $0 \le b \le N - 2n - 1$ $m = 1$	$0 \le n' \le n$ $m \le b' \le b + m$ $0 \le m' \le \iota$	$P(\iota, m', \lambda) \cdot B(b, b' - 1, \beta) \cdot$ $B(n, n', \alpha) \cdot$ $(N - 2n \ge 2)?\eta : 1.0$
7	$1 \le n \le \lfloor \frac{N}{2} \rfloor - 1$ $0 \le b \le N - 2n - 1$ $m = 1$	$1 \le n' \le n + 1$ $0 \le b' \le b$ $0 \le m' \le \iota$	$(m' < \iota)?[\gamma \frac{\iota}{N-1}] \cdot B(b, b', \beta) \cdot$ $P(\iota - 1, m', \lambda) \cdot B(n, n' - 1, \alpha) : 0$ $+(b < b')?[\gamma \frac{b}{N-1}] \cdot B(b - 1, b', \beta) \cdot$ $P(\iota, m', \lambda) \cdot B(n, n' - 1, \alpha) : 0$
8	$1 \le n \le \lfloor \frac{N}{2} \rfloor - 1$ $0 \le b \le N - 2n - m$ $2 \le m \le N - 2n$	$0 \le n' \le n$ $m \le b' \le b + m$ $0 \le m' \le \iota$	$P(\iota, m', \lambda) \cdot B(b, b' - m, \beta) \cdot$ $B(n, n', \alpha)$

Table 2: Method 1 Markov Model

Rule	Acceptance	Output States	State Equation
		$\iota = N - 2n - m - b$	$\kappa = m' - b + b' - m$
1	$n = 0$ $0 \le b \le N$ $m = 0$	$n' = 0$ $0 \le b' \le b$ $b - b' \le m' \le N - b'$	$P(\iota, \kappa, \lambda) \cdot B(b, b', \beta)$
2	$n = 0$ $0 \le b \le N - 1$ $m = 1$	$n' = 0$ $1 \le b' \le b + 1$ $b - b' + 1 \le m' \le N - b' - m + 1$	$\eta \cdot P(\iota, \kappa, \lambda)\cdot$ $B(b, b' - 1, \beta)$
3a	$n = 0$ $0 \le b \le N - 1$ $m = 1$	$n' = 1$ $0 \le b' \le b$ $b - b' \le m' \le N - 2 - b'$	$\frac{\gamma(N-b-1)}{N-1} \cdot B(b, b', \beta)\cdot$ $P(\iota, m' - b + b', \lambda)$
3b	$n = 0$ $0 \le b \le N - 1$ $m = 1$	$n' = 1$ $0 \le b' \le b - 1$ $b - b' - 1 \le m' \le N - 2 - b'$	$\frac{\gamma b}{N-1} \cdot B(b - 1, b', \beta)\cdot$ $P(\iota, m' - b + b' + 1, \lambda)$
4	$n = 0$ $0 \le b \le N - m$ $2 \le m \le N$	$n' = 0$ $m \le b' \le b + m$ $b - b' + m \le m' \le N - b'$	$P(\iota, \kappa, \lambda) \cdot B(b, b' - m, \beta)$
5	$1 \le n \le \lfloor \frac{N}{2} \rfloor$ $0 \le b \le N - 2n$ $m = 0$	$0 \le n' \le n$ $0 \le b' \le b$ $b - b' \le m' \le N - 2n - b'$	$P(\iota, \kappa, \lambda) \cdot B(b, b', \beta)\cdot$ $B(n, n', 1 - \alpha)$
6	$1 \le n \le \lfloor \frac{N-1}{2} \rfloor$ $0 \le b \le N - 2n - 1$ $m = 1$	$0 \le n' \le n$ $1 \le b' \le b + 1$ $b - b' + 1 \le m' \le N - 2n - b'$	$P(\iota, \kappa, \lambda) \cdot B(b, b', \beta)\cdot$ $B(n, n', 1 - \alpha)\cdot$ $[(N - 2n \ge 2)?\eta : 1]$
7a	$1 \le n \le \lfloor \frac{N}{2} \rfloor - 1$ $0 \le b \le N - 2n - 1$ $m = 1$	$1 \le n' \le n + 1$ $0 \le b' \le b$ $b - b' \le m' \le \iota - 1 + b - b'$	$\frac{\gamma\iota}{N-1} \cdot B(b, b', \beta)\cdot$ $P(\iota - 1, m' - b + b', \lambda)\cdot$ $B(n, n' - 1, 1 - \alpha)$
7b	$1 \le n \le \lfloor \frac{N}{2} \rfloor - 1$ $0 \le b \le N - 2n - 1$ $m = 1$	$1 \le n' \le n + 1$ $0 \le b' \le b - 1$ $b - b' - 1 \le m' \le \iota - 1 + b - b'$	$\frac{\gamma b}{N-1} \cdot B(b - 1, b', \beta)\cdot$ $P(\iota - 1, m' - b + b' + 1, \lambda)\cdot$ $B(n, n' - 1, 1 - \alpha)$
8	$1 \le n \le \lfloor \frac{N}{2} \rfloor - 1$ $0 \le b \le N - 2n - m$ $2 \le m \le N - 2n$	$0 \le n' \le n$ $m \le b' \le b + m$ $b - b' + m \le m' \le N - 2n - b'$	$P(\iota, \kappa, \lambda) \cdot B(b, b' - m, \beta)\cdot$ $B(n, n', 1 - \alpha)$

Table 3: Method 2 Markov Model

Variable	Value Set
N	10, 20
α	0.80, 0.90, 0.98
β	0.20, 0.10, 0.06
use_dp	0, 1
λ	0.001, 0.002, 0.003, 0.004, 0.005, 0.010, 0.015 0.020, 0.025, 0.030, 0.035, 0.040, 0.045, 0.050

Table 4: Simulation Parameters

created a custom discrete time simulator in C++. A custom simulator, as opposed to NS2 or Parsec, allowed us to focus on the protocols and state machines. Now that we have a debugged state machine model, we could transplant our code to a more robust simulation environment.

We ran simulations on 36 different configurations over 14 packet arrival rates. This gives us 504 data points. N is the number of fully connected nodes. λ is the packet arrival rate (which is also the RTR arrival rate). New packets only arrive while in Idle mode. β is the exponential backoff parameter. α is the geometric packet length parameter. $sslot$ is the number of scheduling slots per RTR. dp is a flag to indicate whether or not to use Dual Poll mode.

The 504 different scenarios come from the values shown in Table 4. We also tried varying the use of NACKs and ϕ. These last two made very minor differences, so they were not included with the general simulation runs. Unless otherwise stated, we did not use NACKs and set $\phi = 1$.

We simulated each data point for 10000 slots. We repeated each simulation 10 times and averaged the results. Over all 504 data points, we calculated the 90% confidence interval. The largest interval was 17.7% of the mean for $N = 20, \alpha = 0.90, \beta = 10, \lambda = 0.050$. The average 90% interval was 6.6% of the mean. So, in general, we can say that the average of our ten samples lies within a reasonable range of the true population mean.

For each configuration of the 36 configurations, we created several graphs over the λ values. We graphed (1) The Hear RTR Rate, (2) Good RTR Rate, (3) Percent Time in

Backoff, (4) Average Transmit Queue Length, (5) Average Transmit Queue Waiting Time, and (6) Average Concurrent Conversations per Slot. In this paper, we only persent (4) and (6). Further, since much of the data was repetitive, we only include 8 of the 36 configurations.

The configurations for which we show graphs are, as (N, α, β): (10, .8, .2), (10, .8, .06), (10, .98, .2), (10, .98, .06), (20, .8, .2), (20, .8, .06), (20, .98, .2), (20, .98, .06). These are the extreme cases for 10 and 20 nodes. The have the shortest ($\alpha = 0.8$) and longest ($\alpha = 0.98$) packets and the shortest ($\beta = 0.2$) and longest ($\beta = 0.06$) backoff periods. It turns out that the data is not sensitive to β but is sensitive to α. Therefore, we will generally focus on the differences between α cases.

Fig. 4 shows the number of concurrent conversations for $N = 10$. Fig. 5 shows the same graphs for $N = 20$. Both figures include a data series labeled "Model 2" which represents our second Markov model. Fig. 6 shows the mean waiting time in the transmmit queue. We only show graphs for $N = 20$ nodes. The graphs for 10 nodes follow a similar pattern.

5 Conclusion

Under our simulation conditions, multiple RTR scheduling slots did not make a significant statistical difference to the number of concurrent conversations. Multiple scheduling slots can make a statistically significant difference in the mean waiting time of the transmit queue for shorter packet lengths when the system is backlogged. When the system is not backlogged or when the system uses long packets, there is negligible difference between one and multiple scheduling slots.

Our simulations and Markov models had four main modes: Communicating, Idle, Ready, and Backoff. Communicating pairs stayed in the Communicating mode for a geometric length. Nodes in Ready mode failed if there were more than one node in Ready mode in the same slot. If there were exactly one node ready, it may succeed or fail based on the polled node status. Nodes stayed in Backoff mode for a time drawn from a negative exponential distribution. Nodes not in Communicating, Ready, or Backoff mode were in Idle mode.

We tested with Poisson arrivals while only in Idle mode. Communication leghts were

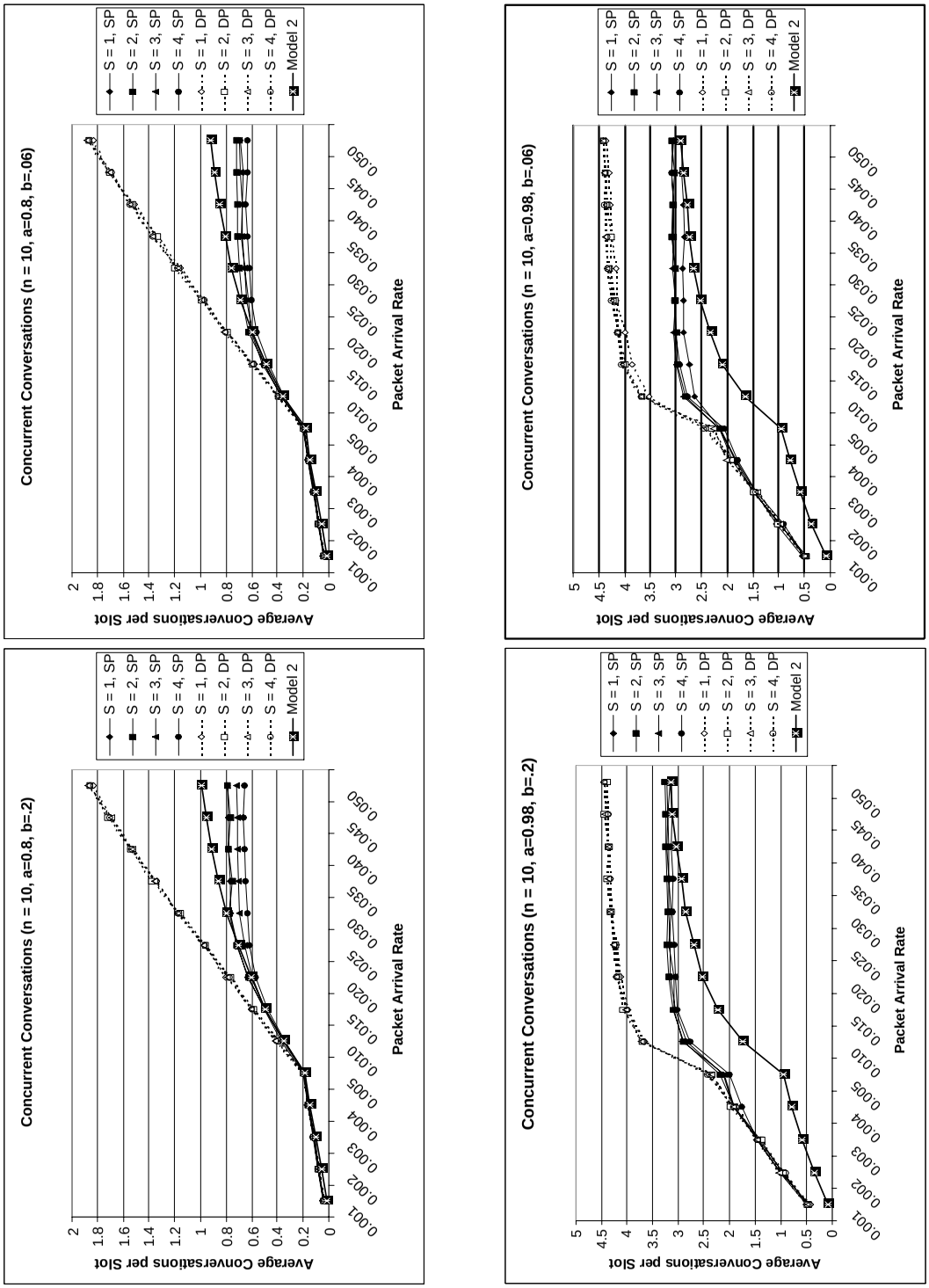

Figure 4: Concurrent Conversations, $N = 10$

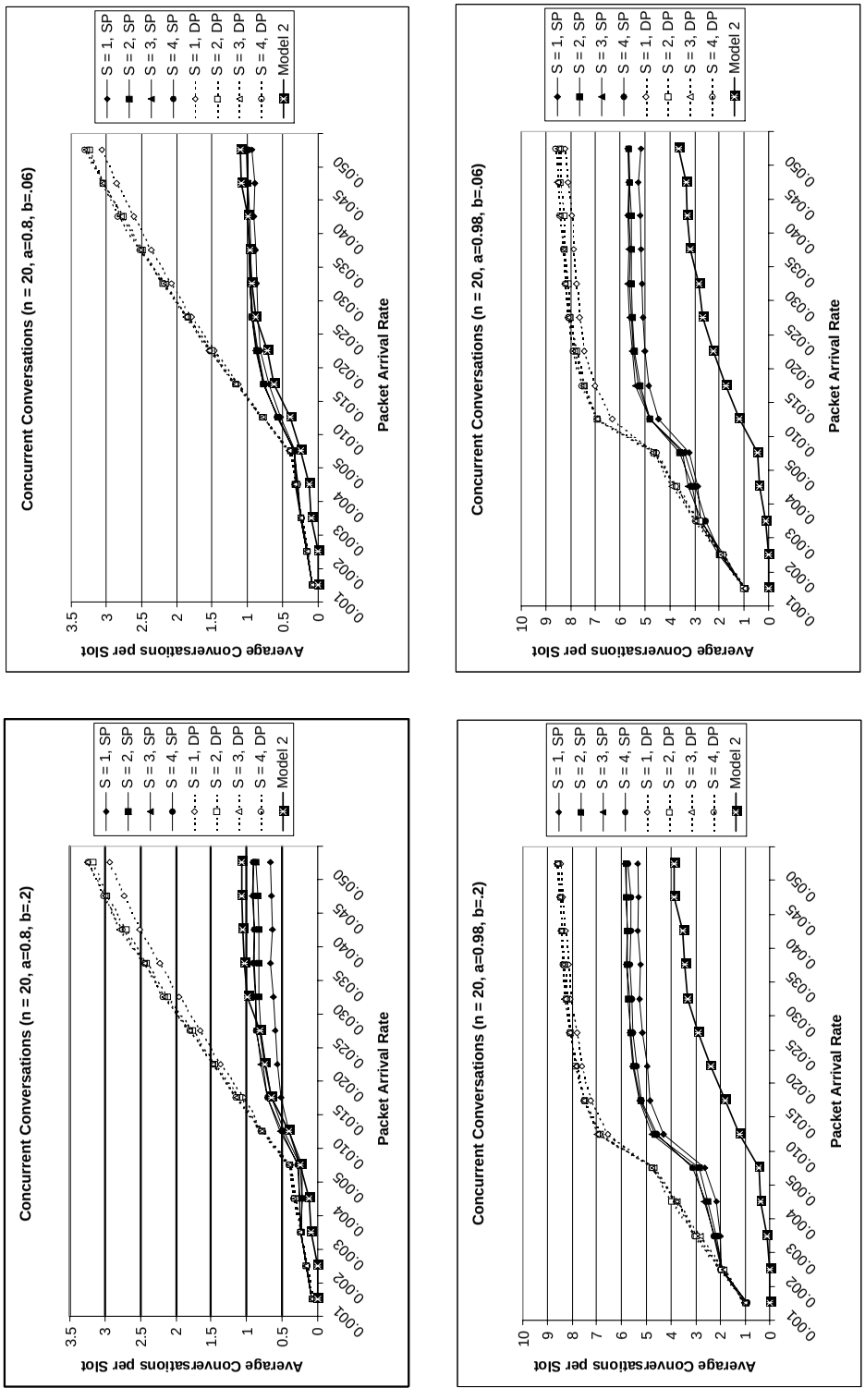

Figure 5: Concurrent Conversations, $N = 20$

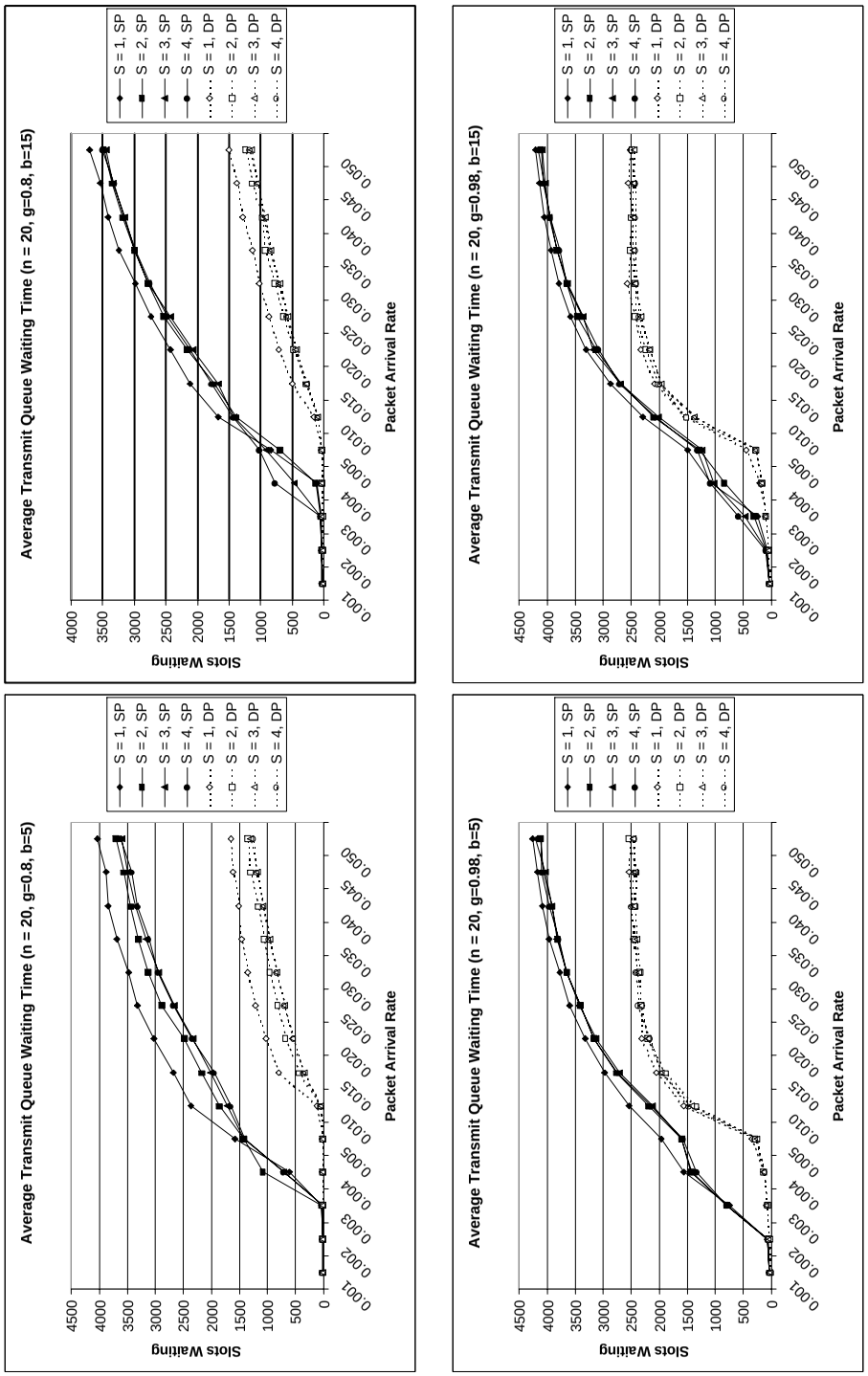

Figure 6: Mean Waiting Time, $N = 20$

geometric. Backoff periods were exponential in the simulations and geometric in the Markov model. The reason for the difference in Backoff distributions were because of modelling problems of an exponential distribution.

There were two Markov models. Model 1 had nodes exiting Backoff go to Idle mode. Model 2 had nodes existing Backoff go to Ready mode. Model 2 was closer to simulation than Model 1. Model 2 is reasonably accurate for shorter packet lengths. As the packet length increases, the model becomes less accurate. This difference is extenuated by an increase in the number of nodes.

The Markov models did not account for queuing. A node that goes in to Backoff mode with an RTR may then become ready with a new RTR for a different node than the previous RTR. The behavior is random. In simulation, a node that becomes ready and then backs off will retry missed RTRs in a round-robin fashion. In simulation, the RTR rate is data driven. If a node is backlogged, it will always have an RTR ready, thus $\lambda \rightarrow 1$ under load. The Markov model attempted to emulate this behavior with the parameter γ, being the rate at which RTRs met with success if received.

References

[1] Leonard Kleinrock. *Queueing Systems*. Wiley, New York, 1975-76.

[2] J.J. Metzner. On Improving Utilization in ALOHA Networks. *IEEE Transactions on Communications*, COM-24(4):447–8, Apr 1976.

[3] E.S. Souza and J.A. Silvester. Spreading Code Protocols for Distributed Spread-Spectrum Packet Radio Networks. *IEEE Transactions on Communications*, 36(3):272–81, Mar 1988.

[4] A. Tzamaloukas and J.J. Garcia-Luna-Aceves. Receiver-Initiated Channel-Hopping for Ad-Hoc Networks. *INFOCOM 2000*, 2000.