CMPUT 313 Assignment 2

Stephen Jahns and Marek Kundera

Tuesday, March 13

Introduction

Choosing an appropriate backoff strategy for the Slotted ALOHA MAC layer protocol depends on a series of factors. In this report, we have simulated the Probabilistic Backoff (P), Interval-Based Backoff (I), and Truncated Binary Exponential Backoff (B) strategies. In addition, we have simulated the Time Division Multiplexing (T) protocol for additional comparison. Our simulation was based on several assumptions found in the problem specification and on a terse description of each protocol:

T: a station i only transmits at slot number $mN+i$, $m = 0, 1, 2, ... N$, where m is the cycle number, and N is the number of stations

P: a station will try to transmit immediately on generation, or retransmit with $1/N$ probability at each subsequent slot

I: a station will try to transmit immediately on generation, or retransmit at a slot 1 to N slots later, chosen randomly

B: a station will try to transmit immediately on generation, or retransmit at a slot 1 to 2 slots later, then 1 to 4 slots later, or eventually 1 to 2^i slots later, chosen randomly until the frame successfully transmits

To set some groundwork, we will begin by outlining our expectations with a preliminary analysis. Next we will briefly explain the main functionality and design of our `psim` simulator. Then we will reveal the simulation results with the help of some tables and graphs, paired with the parameters we used to generate the media. Finally, we will examine each protocol's performance by analyzing their average throughputs and the overall average per-frame slot time delays of the successfully transmitted frames. In addition, we will examine the fairness of each protocol by observing the number of undelivered frames under each protocol at the individual stations.

Time Division Multiplexing

Here we expect that as the probability of frame generation, p, increases, the average frame delay should increase. This is because in any given cycle of N slots, the probability of a frame being generated at that slot increases. Therefore as p increases, up to N frames might be generated in a cycle. On the other hand, the mean throughput should increase in direct proportion to p. As p increases, it is more likely frames will be available for transmission in a given cycle. In other words, it is more likely a station will not waste its designated slot time. Furthermore, since each station has a designated slot only it can transmit on, it is not possible for collisions to occur between the stations. A potential serious weakness of TDM is that if the load is not equally balanced between the stations on the bus, unnecessary delays may occur, especially if only a single station has a very large number of frames queued for a period of many slots. The busy station will experience delays of N slots between each frame transmitted, while all other slots are left unused, leaving the overall hub throughput very low and the delay unnecessarily high.

Probabilistic Backoff

Notice that for a probability of successful transmission of $1/N$, we need N stations to each have a frame queued for transmission, in order for one to *likely* transmit. We can build on this observation and hypothesize that at some p, every station will be generating frames faster than it can transmit them. And therefore, the throughput will level off in concert with the probability that *exactly one* station transmits, i.e. let event $ExactlyOneTx$ represent the case where exactly one station is transmitting. Thus, when N=20 stations, $Prob(ExactlyOneTx) = C(20,1) \times (1/20) \times (19/20)^{19} = 0.377$. For 20 stations, it is likely then that when $p = 0.377/20 = 0.019$, we can expect the throughput to begin leveling off and for the average frame delay to start increasing. This is because as p increases, frames are more likely to generate, while the throughput remains the same. And so, the queue can only get larger thus each frame must wait longer before moving to the front of the queue for transmission.

Interval-Based Backoff

In a similar fashion to the probabilistic backoff, as p is increased, we predict that some critical point will be reached where the rate of generation will be higher than the rate of transmission, and so the queues at each station will grow, increasing the average frame delay. Additionally, the throughput will level out as the probability of a collision stabilizes along with the rate of successful transmission. Consider an analogy of N=20 buckets. At each time slot, each station has p chance of adding a 'ball to one bucket'. Therefore, in the long run, the frames generated per slot is $N \times p$ while the rate of transmission can be 1 at best, but realistically much lower (let's say Z). It is easy to see that as p increases, eventually the rate of generation will be greater than the rate of transmission (or throughput), i.e. $N \times p \geq Z$. Z can be said to be equal to the probability that a bucket has exactly one ball. This isn't the same probability as above, and it won't be calculated as it would involve higher level probabilistic math. At this point however, the throughput will will level out and the average frame delay will increase as p increases.

Truncated Binary Exponential Backoff

Truncated binary exponential backoff is more or less a fancy version of interval backoff where instead of a fixed interval size of N slots to randomly backoff for, the interval starts at 4 slots and doubles at every collision until 1024. Due to this property, if a single station has at least 9 consecutive transmission attempts with collision, the interval before the next transmission attempt could be as long as 1024 slots. During low loads on the bus, we should expect small delays as immediate transmission of a frame upon generation (if the transmission queue was previously empty) or after a successful transmission is allowed. During progressively higher loads, the relative performance between TBEB and IB and PB in in terms of hub throughput and average delivery delay may be difficult to predict.

Methods

The `psim` simulator was implemented in C++. It requires several parameters at the command line:

```
psim Protocol N p R T seed1 seed2 ... seedT
```

`Protocol` can taken on a value of `T`, `P`, `I`, or `B`, depending on what protocol is being simulated. The parameter `N` represents the number of stations that will be transmitting under the protocol. `p` represents the probability that a frame will be generated at each station. `R` represents the number of slots that the experiment will run, and is therefore the time-measure for the experiment. And `T` represents the number of trials the simulation will run, while the following `seed1...seedT` inputs are used to reseed the randomization functions for each trial.

For the purpose of this simulation, the values `N=20`, `R=50000`, and `T=5` will be fixed. In other words, we will simulate 5 trials where 20 stations transmit over a time period of 50,000 slots. The observations shown later are based on 5 trials that were seeded with values 1, 2, 3, 4, 5, respectively.

`psim` functions by simulating `T` independent trials of hub of `N` stations running for `R` time slots, using the random number generator seeds separately specified for each trial. At the beginning of each timeslot, each station generates a new frame on its transmission queue with probability `p`. Then, the delay value in time slots is incremented for each frame on each stations transmission queue. Then, the simulator determines which and how many stations will attempt transmission in that timeslot, according to the MAC protocol specified in `Protocol`. If there is only one station attempting transmission, the transmission is deemed successful and the frame is removed from that station's queue. If there are more than one transmissions, a `tx_collide(slot)` method is executed on each station that partly implements the station's backoff protocol as specified in `Protocol`.

The simulator outputs the parameters (minus the seeds) on the first line. After running all trials, it calculates and prints 95% confidence intervals for the overall average throughput in frames delivered per time slot, and delay in slots for each frame successfully delivered, on lines 2 and 3 respectively. Delay was calculated from the time of frame generation to the time of successful delivery in slots, in a matter that the minimum delay is 1 if the frame is generated in the same slot that it is delivered in. Confidence intervals were calculated using appropriate t-statistics for the number of trials from a file labelled `tvalues.dat`. Per-station statistics were output on the remaining lines, where each line started with `nN`, the id of the station, followed by confidence intervals for throughput and delay per frame delivered, and finally the ratio of undelivered frames to total frames generated by that station for each trial separately, all on the same line.

Experiments were automated using the BASH script `make_plot.sh` to run the simulator for each protocol over ranges of `p` values ranging from 0.00 up to 0.04. The scripts parsed data from the output into appropriately formatted data files that were sent to GNUPLOT for plotting.

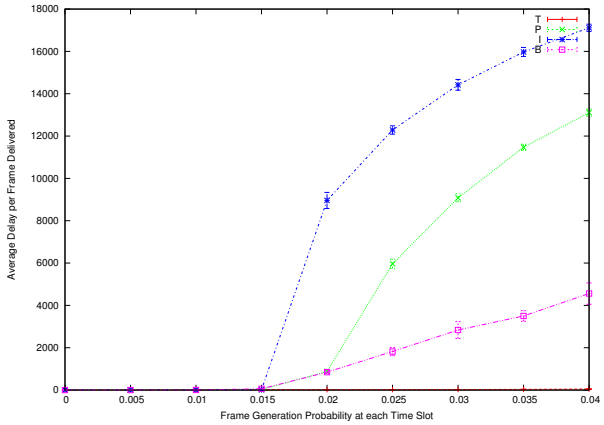

Figure 1: Average delay per frame under each protocol for p:(0,0.04; @0.005 increments). Used to illustrate what happens as p nears 0.04.

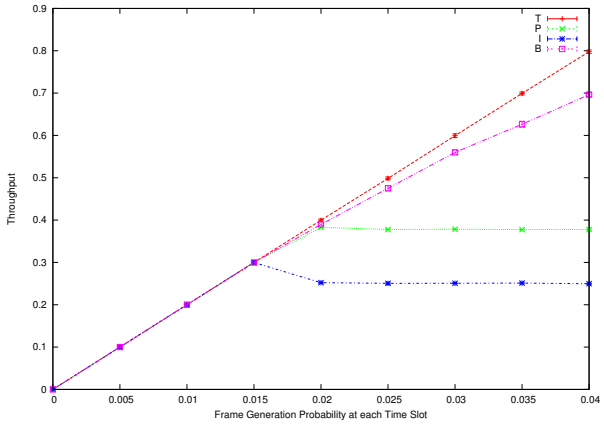

Figure 2: Mean throughput of all stations under each protocol for p:(0,0.04; @0.005 increments). Used to illustrate what happens as p nears 0.04.

Results

For ranges of frame generation probabilities per station per slot ranging from 0.0 to 0.04, plots of average delay in slots per frame successfully delivered and of throughput in number of frames per timeslot were obtained.

In Figure 1, we see that over larger values of p ranging up to 0.04, there is a clear separation in delay levels for each protocol. Interval backoff produces the largest delays, followed by probabilistic backoff. Truncated binary exponential backoff produces the least delays of the Slotted ALOHA MAC protocols. Time division multiplexing, however, provides delivery delays at $p = 0.04$ that are a tiny fraction of the delays of the ALOHA protocols. Similar results exist for throughput at higher loads (p approaching 0.04) as seen in Figure 2. TBEB provides the high-

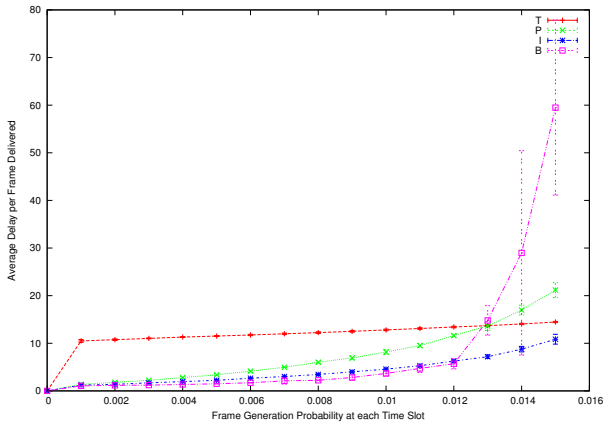

Figure 3: Average delay per frame under each protocol for p:(0,0.015; @0.001 increments). Used to illustrate what happens as p nears 0.

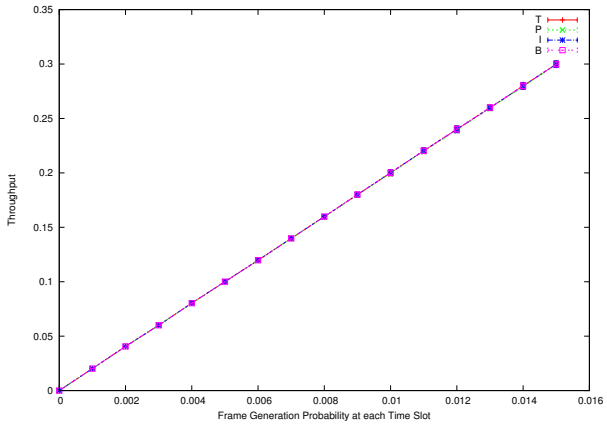

Figure 4: Mean throughput of all stations under each protocol for p:(0,0.015; @0.001 increments). Used to illustrate what happens as p nears 0.

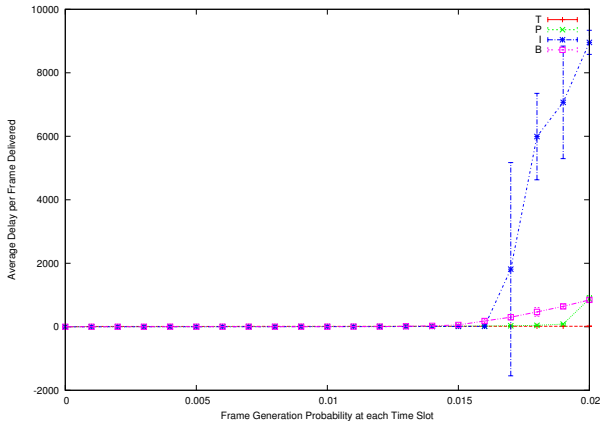

Figure 5: Average delay per frame under each protocol for p:(0,0.02; @0.001 increments). Used to illustrate where protocols begin to diverge significantly. Especially, protocol I at p=0.015.

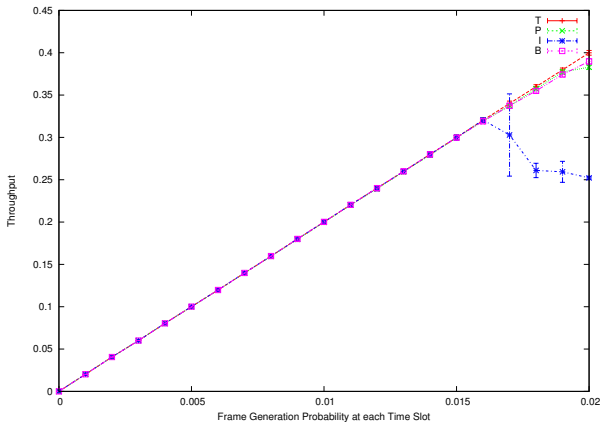

Figure 6: Mean throughput of all stations under each protocol for p:(0,0.02; @0.001 increments). Used to illustrate where protocols begin to diverge significantly. Especially, protocol I at p=0.016.

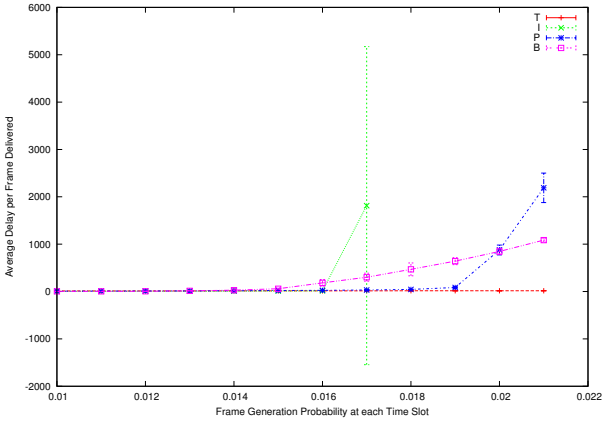

Figure 7: Average delay per frame under each protocol for p:(0.01,0.021; @0.001 increments). Used to illustrate the points where the protocols intersect.

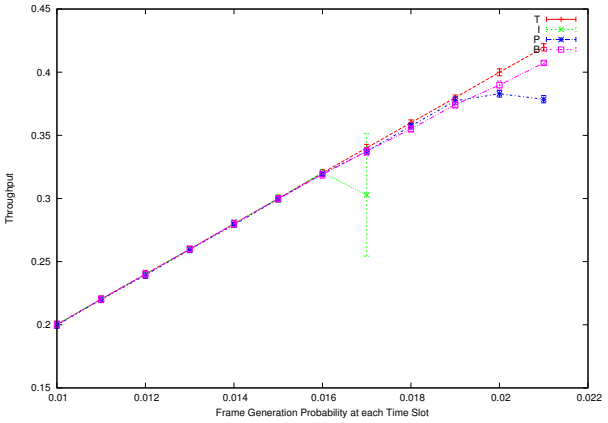

Figure 8: Mean throughput of all stations under each protocol for p:(0.01,0.021; @0.001 increments). Used to illustrate the points where the throughput for protocols I and P begins to level off.

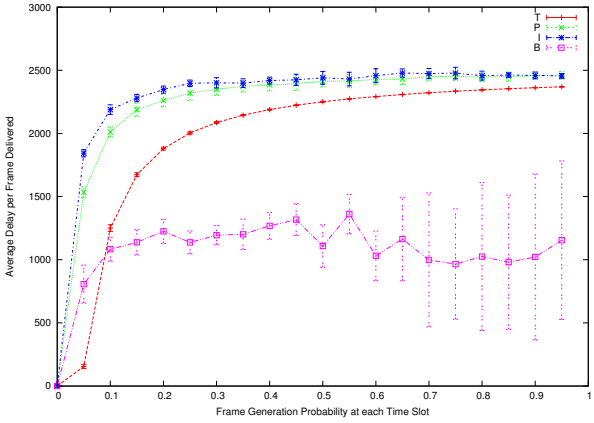

Figure 9: Average delay per frame delivered on hub for full range of per slot, per station frame generation probability, tested for each MAC protocol $R = 5000$ slots, $T = 5$ trials.

est throughput at high load, followed by IB and PB. TDM provides a higher throughput at high load than the ALOHA protocols, but by a much smaller margin (10% higher than TBEB) than exhibited by the difference in delays.

At smaller loads (approx. $p < 0.013$), the frame delivery delays for TDM become larger than that of the ALOHA protocols (Figure 3). The relative delays between the different ALOHA protocols also changes. For $p < 0.012$, TBEB has the smallest delay, but the delay increases strongly above that value. Interval backoff has the smallest delays for $p \in (0.012, 0.016)$, after which the delay for IB jumps up sharply (Figure 5). For $p > 0.016$, TDM has less delay than any of the ALOHA protocols, and PB has the smallest delay of the ALOHA protocols for $p \in (0.016, 0.02)$, after which the delay jumps to above that of TBEB (Figure 7). TBEB has the least delay of the ALOHA protocols for $p > 0.02$.

All protocols produce very similar throughputs at low load for $p \in (0, 0.016)$ (see Figures 4 and 6), where all throughputs appear to increase linearly with frame generation probability. Above $p = 0.016$, IB throughput drops fairly quickly by about 20% and flattens to a throughput of about 25% for all higher loads. Above $p = 0.02$, PB throughput drops a little and flattens to about 40% for all higher loads.

For the range of p values that could be tested using `R=50000` (higher p values resulted in at least polynomially higher simulation run times as the frame queues became heavier loaded), we did not see a point where TBEB throughput levels off. Running the simulator under the same conditions as before but with `R=1000` and `p` ranging from 0 to 1, TBEB throughput

	1	2	3	4	5
n1	0.006471	0.000503	0.000989	0.000505	0.001422
n2	0.000987	0.002000	0.002017	0.002999	0.001004
n3	0.000501	0.001001	0.000988	0.001035	0.000000
n4	0.000000	0.000501	0.000985	0.000481	0.000000
n5	0.000000	0.000489	0.000985	0.000986	0.000000
n6	0.000000	0.000517	0.000000	0.000000	0.001496
n7	0.002455	0.002026	0.000000	0.001485	0.000484
n8	0.001047	0.000000	0.000000	0.000000	0.002499
n9	0.000000	0.000000	0.002534	0.003348	0.000000
n10	0.000000	0.001004	0.000991	0.000000	0.001492
n11	0.000000	0.000495	0.003854	0.001487	0.001453
n12	0.001505	0.001004	0.002004	0.001037	0.000000
n13	0.002463	0.000000	0.003085	0.000000	0.000513
n14	0.002552	0.001035	0.000503	0.004805	0.003038
n15	0.002014	0.000000	0.001000	0.000490	0.002535
n16	0.000000	0.000496	0.001001	0.000506	0.000000
n17	0.000000	0.000514	0.004460	0.001044	0.000978
n18	0.000000	0.000499	0.001999	0.000000	0.000512
n19	0.000000	0.003044	0.000986	0.000000	0.002092
n20	0.000501	0.000507	0.000000	0.000494	0.000504

Table 1: Ratios of undelivered to total generated frames, per station, per trial using time division multiplexing, $p = 0.04$, $R = 50000$ slots, $T = 5$ trials. Each column corresponds to a single trial, each row corresponds to a single station.

	1	2	3	4	5
n1	0.508249	0.581281	0.539540	0.536810	0.536203
n2	0.519899	0.544526	0.561975	0.563332	0.536524
n3	0.481934	0.525169	0.486970	0.501256	0.544828
n4	0.545960	0.516467	0.479901	0.471414	0.493199
n5	0.566901	0.565152	0.511707	0.531282	0.539249
n6	0.535802	0.483191	0.508449	0.525551	0.479410
n7	0.545680	0.511827	0.541955	0.522461	0.505500
n8	0.533605	0.493834	0.534942	0.541357	0.506216
n9	0.530633	0.490852	0.550930	0.542089	0.502367
n10	0.559280	0.528793	0.560729	0.510886	0.522739
n11	0.518182	0.546937	0.552566	0.542323	0.528358
n12	0.518127	0.529263	0.535873	0.511548	0.521608
n13	0.472489	0.473892	0.547284	0.516274	0.522795
n14	0.499205	0.546417	0.497184	0.520791	0.538313
n15	0.545084	0.524158	0.527061	0.527906	0.573762
n16	0.503090	0.526585	0.511013	0.571001	0.580835
n17	0.519297	0.563939	0.516014	0.499746	0.511922
n18	0.508870	0.549527	0.542843	0.530334	0.513667
n19	0.545858	0.515464	0.560582	0.538499	0.518900
n20	0.495988	0.556727	0.498999	0.507109	0.526804

Table 2: Ratios of undelivered to total generated frames, per station, per trial using probabilistic backoff, $p = 0.04$, $R = 50000$ slots, $T = 5$ trials. Each column corresponds to a single trial, each row corresponds to a single station.

	1	2	3	4	5
n1	0.682902	0.686082	0.700204	0.690148	0.673203
n2	0.679692	0.689398	0.697190	0.661972	0.675635
n3	0.716981	0.682341	0.659226	0.673226	0.666492
n4	0.713648	0.716935	0.692615	0.673288	0.685700
n5	0.677587	0.694077	0.702149	0.672148	0.694153
n6	0.688391	0.706201	0.702863	0.679980	0.655243
n7	0.658811	0.664447	0.666136	0.721930	0.687562
n8	0.671120	0.683063	0.664175	0.705179	0.707486
n9	0.691258	0.666667	0.675761	0.691448	0.709483
n10	0.691358	0.669888	0.666159	0.695152	0.701021
n11	0.686657	0.681751	0.690379	0.676530	0.679303
n12	0.689154	0.651744	0.681750	0.700640	0.704902
n13	0.715681	0.692882	0.686462	0.690809	0.677016
n14	0.680972	0.706897	0.681089	0.699294	0.698234
n15	0.686216	0.662398	0.699356	0.691370	0.658612
n16	0.682927	0.714427	0.695309	0.712512	0.700500
n17	0.647456	0.703210	0.685269	0.705513	0.722195
n18	0.670229	0.667469	0.682620	0.694542	0.698836
n19	0.698437	0.688623	0.689368	0.699150	0.673657
n20	0.672286	0.701729	0.689209	0.674254	0.662021

Table 3: Ratios of undelivered to total generated frames, per station, per trial using interval backoff, $p = 0.04$, $R = 50000$ slots, $T = 5$ trials. Each column corresponds to a single trial, each row corresponds to a single station.

	1	2	3	4	5
n1	0.077351	0.118118	0.425962	0.002500	0.052495
n2	0.125247	0.062253	0.008973	0.386563	0.084267
n3	0.061347	0.106634	0.146091	0.035376	0.079079
n4	0.053150	0.073552	0.000000	0.316767	0.063099
n5	0.115442	0.031403	0.084800	0.093294	0.018945
n6	0.290680	0.310675	0.084221	0.040675	0.016035
n7	0.285861	0.006979	0.057115	0.171313	0.364198
n8	0.143426	0.054381	0.311475	0.043000	0.161356
n9	0.016907	0.008755	0.156189	0.109576	0.201082
n10	0.043048	0.133580	0.099445	0.224008	0.052000
n11	0.090303	0.126420	0.418695	0.242541	0.003605
n12	0.080145	0.000000	0.000000	0.339708	0.086697
n13	0.185393	0.203203	0.003885	0.103681	0.031915
n14	0.049010	0.190793	0.000000	0.144299	0.212091
n15	0.240223	0.082819	0.314556	0.121367	0.091960
n16	0.150581	0.078744	0.427317	0.024150	0.287167
n17	0.016312	0.204412	0.003458	0.018482	0.377398
n18	0.119456	0.179462	0.059233	0.029340	0.068982
n19	0.135551	0.203085	0.072709	0.002467	0.083416
n20	0.118554	0.331832	0.097561	0.129732	0.238192

Table 4: Ratios of undelivered to total generated frames, per station, per trial using truncated binary exponential backoff, $p = 0.04$, $R = 50000$ slots, $T = 5$ trials. Each column corresponds to a single trial, each row corresponds to a single station.

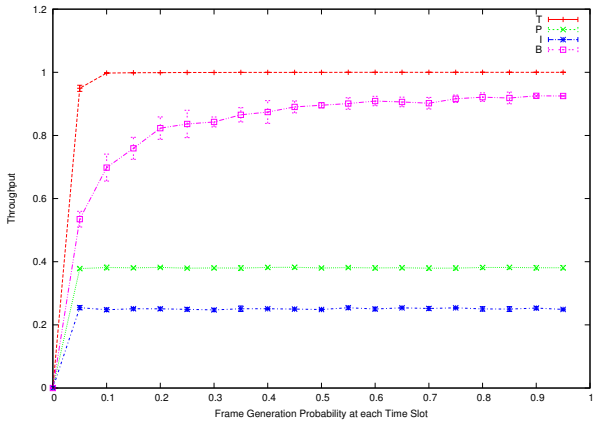

Figure 10: Average throughput on hub for full range of per slot, per station frame generation probability, tested for each MAC protocol $R = 5000$ slots, $T = 5$ trials

appears to increase steadily until about 78% at about $p = 0.5$, and the same plot for `R=5000` shows TBEB throughput leveling off at about 93% (Figure 9). At `R=5000`, IB and PB delays appear to level off at the same values as for `R=50000`.

In Figure 9, we also see that with sufficiently high load, TBEB yields a lower average delay than T

In order to determine the relative fairness of each protocol with respect to individual stations competing for transmissions on the hub, we measured per trial, per station ratios of undelivered frames to total frames generated at that station. For this metric, if the ratio is 0 at the end of the trial, all frames were successfully delivered, and if the ratio is 1, 0 frames were successfully delivered. Simulations were run for each of the protocols, and results backoff protocols are listed in Tables 1 - 4 . In Table 1, the ratios do not grow much larger than one for any station on any trial, meaning that all stations were able to transmit nearly all their frames by the end of the trial. In Table 2, for PB we see that the ratios for each station never stray below 0.45 or above 0.6 within a trial. Table 3 shows similar results for interval backoff. In Table 4, within a single trial, the ratios for each station range from 0 (all generated frames successfully delivered) to 0.43 (nearly half of all frames unsuccessfully delivered, left in queue at end of trial).

Discussion

Our discussion of the simulation results will cover 3 metrics: delay, throughput, and fairness.

Delay

Initially we kept p within the realm of the assignment specifications, i.e. $p \leq 0.04$. In Figure 1, notice that the delay for the protocols have deviated significantly. In terms of this metric we can say that under the T protocol, the average amount of time a frame had to wait before being transmitted was the least. For the backoff strategies we see protocols performed from better to worst in order B, P, I. At p=0.04 frame generation rate the Interval Based protocol performed the worst, with the average frame waiting around 17,000 time slots before transmitting.

Throughput

Fairness

In our simulation, the rate of frame generation for each station is equal, so there is equal demand for the channel from all stations. Therefore, a fair MAC protocol should provide approximately equal access to each station over the course of a single trial (if a trial is sufficiently long). In order to quantify this, we recorded the ratio of undelivered frames to total generated frames for each station for each trial run by the simulator. If each station is given fair access to the channel, these ratios should be approximately for each station. For time division multiplexing, probabilistic backoff, and interval backoff, this is what we observed (Table 1).

Time division multiplexing is by its definition clearly a fair protocol, as each station is granted exactly equal access to the channel in the form of its unique assigned slot. No collisions are possible. This assertion is backed up by our observation in Table 1 that all stations were able to transmit nearly all their frames by the end of each trial. The stations that did not transmit all of them likely had generated the frames soon before the end and had not yet had time to empty their queue.

Probabilistic backoff was also observed to be a fair protocol, in Table 2 we see all the ratios hovering in a reasonably small interval between 0.45 and 0.6, meaning that all stations were able to successfully transmit approximately half their generated frames by the end of each trial. The fairness in PB is due how the backoff procedure has no memory; after a collision, each consecutive step is evaluated independently for retransmission with a probability of $\frac{1}{N}$. There is no means by which one station could gain some sort of advantage over another, and there is no period of slots in which a station is not probabilistically considered for retransmission.

Similar results to probabilistic backoff were observed for interval backoff in Table 3, only with what appears to be a higher average of undelivered frames ratios over all stations. With the number of stations as used in our experiments ($N = 20$), IB is fair because after a collision, the maximum interval in slots before the next transmission attempt by that station is the number of stations on the bus. This backoff period is small enough that does not appear to significantly disadvantage a station after a transmission; its not long enough for luckier stations to 'race ahead' and deliver a bunch of frames in quick succession during that backoff period.

However, for truncated binary exponential backoff, considerable differences between the end of trail ratios of undelivered frames to total generated frames were observed between stations (Table 4). For instance, in trial 3, stations n4, n12, and n14 were able to transmit all of their generated frames, but stations n1, n11, and n16 all had over 40% of their generated frames remaining untransmitted on the queue at the end of the trial. This can be exponential growth of the backoff period when multiple successive collisions are experienced by a station; after 9 collisions the backoff interval between attempts can be as large as 1024 slots. If a station is unlucky enough to sustain so many collisions, it will be put at a serious disadvantage as compared to other stations that were luckier, as many new frames will be generated and queued during the extended backoff periods. Luckier stations will instead be able to transmit many frames almost as soon as they were generated, as a significant portion of the stations may be stuck waiting in an extended backoff period, leaving the channel clear.

Conclusions

Rehash discussion?

(Under these applcations, use this protocol! Under others, use that!)

—— NB - One thing to note could be limitations of our simulation - a global probability constant for frame generation assumes that all stations are providing the same level of load to the hub, which may not necessarily be true and could possibly have significant repercussions on fairness, ie if most stations are pretty quiet, but one is SUPER active, could it drown out the others in the various protocols in a way that is somehow unfair? ——