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 $mN+i$, $m = 0, 1, 2, \ldots N$, 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.

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.

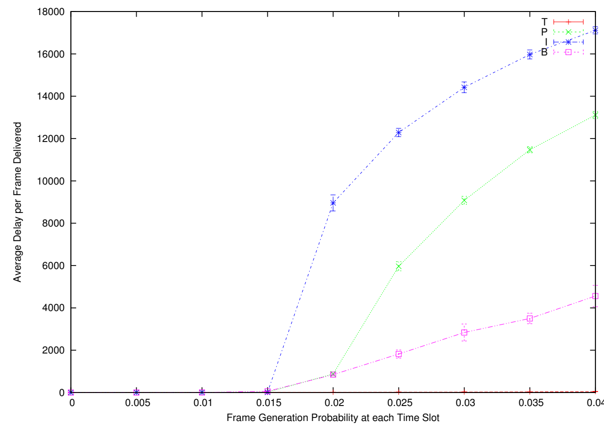

Figure 1: Average delay per frame under each protocol for p:(0,0.04; @0.005 increments). Used to illustrate what happens as p nears 0.04.

Truncated Binary Exponential Backoff

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.

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

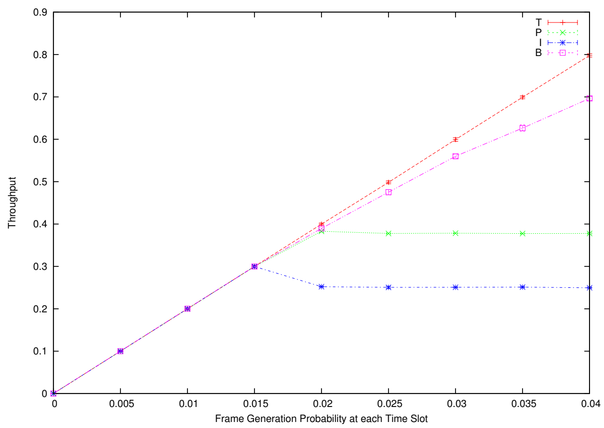

Figure 2: Mean throughput of all stations under each protocol for p:(0,0.04; @0.005 increments). Used to illustrate what happens as p nears 0.04.

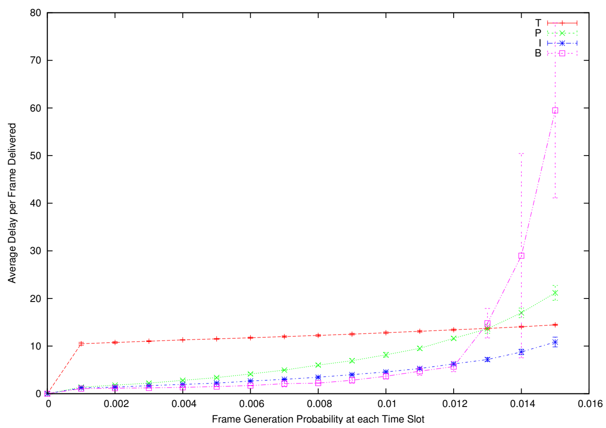

Figure 3: Average delay per frame under each protocol for p:(0,0.015; @0.001 increments). Used to illustrate what happens as p nears 0.

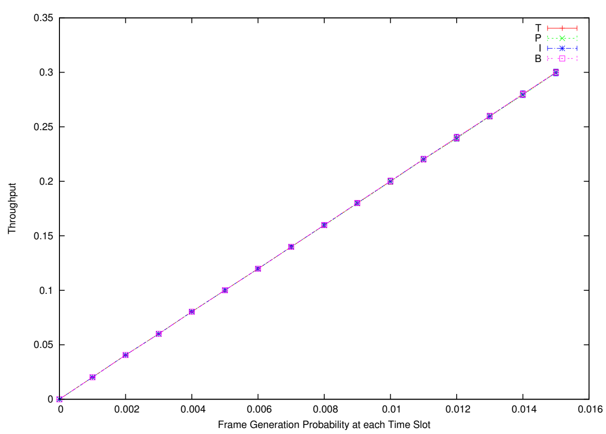

Figure 4: Mean throughput of all stations under each protocol for p:(0,0.015; @0.001 increments). Used to illustrate what happens as p nears 0.

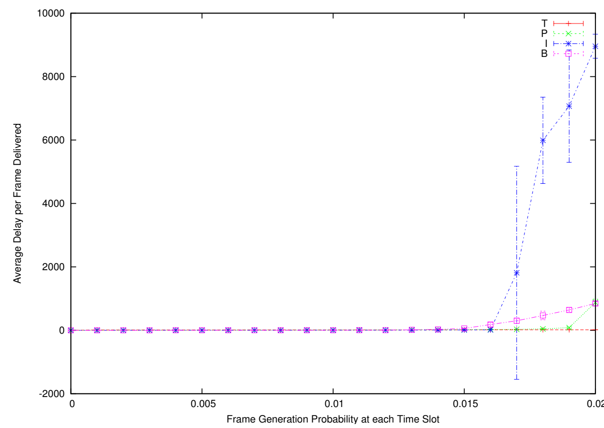

Figure 5: Average delay per frame under each protocol for p:(0,0.02; @0.001 increments). Used to illustrate where protocols begin to diverge significantly. Especially, protocol I at p=0.015.

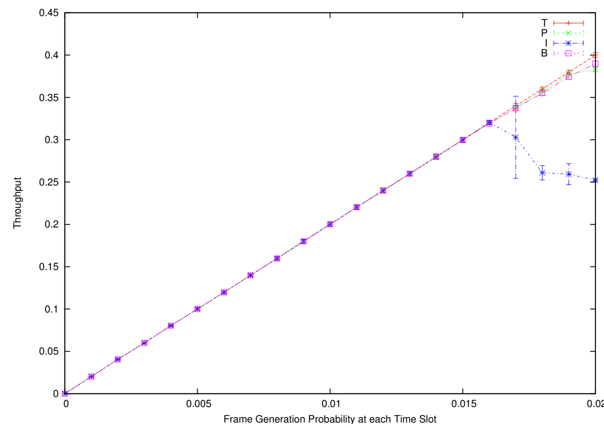

Figure 6: Mean throughput of all stations under each protocol for p:(0,0.02; @0.001 increments). Used to illustrate where protocols begin to diverge significantly. Especially, protocol I at p=0.016.

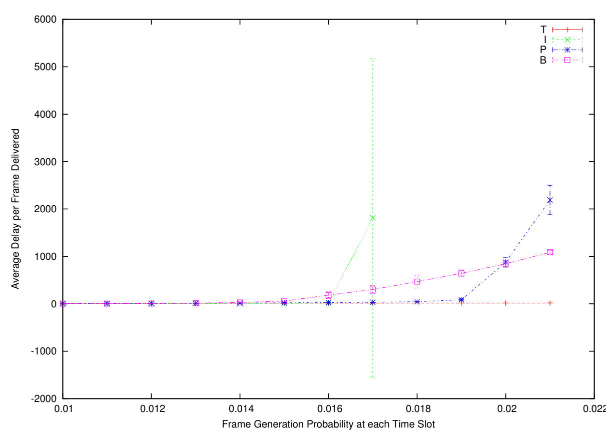

Figure 7: Average delay per frame under each protocol for p:(0.01,0.021; @0.001 increments). Used to illustrate the points where the protocols intersect.

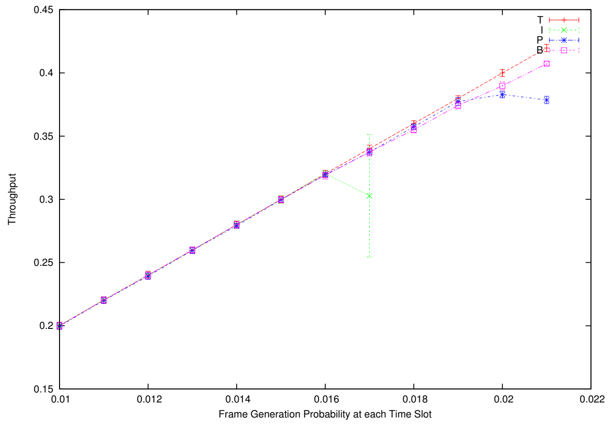

Figure 8: Mean throughput of all stations under each protocol for p:(0.01,0.021; @0.001 increments). Used to illustrate the points where the throughput for protocols I and P begins to level off.

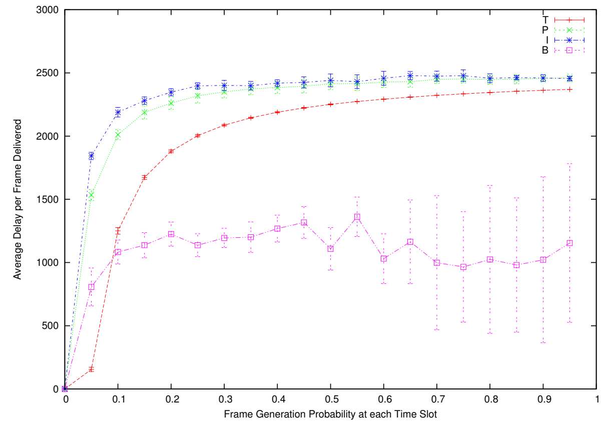

Figure 9: Average delay per frame delivered on hub for full range of per slot, per station frame generation probability, tested for each MAC protocol $R = 5000$ slots, $T = 5$ trials

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 highest 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 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). At `R=5000`, IB and PB delays appear to level off at the same values as for `R=50000`.

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

TODO figure TODO!!! Running with R=1000 shows TDM delay jumping after t=0.1 to above that of TBEB...

TODO - TDM delay should hypothetically spike when the per-station throughput (which should approach 1/N at sufficiently high load) is lower than the generation probability...

TODO Large load TBEB delay looks to be about half that of all the others... Also error increases with higher load – somehow related to unfairness?

TODO make distinction between Average Throughput used in most plots vs average per-station throughput

TODO - unfairness -¿ look at ratio of undelivered/-total generated frames at very high load for TBEB -¿

	1	2	3	4	5
n1	0.900000	0.547619	0.860000	0.968750	0.787234
n2	1.000000	0.468085	1.000000	0.220339	0.615385
n3	0.840909	0.394737	0.000000	1.000000	1.000000
n4	0.195652	0.000000	0.027778	0.411765	0.000000
n5	0.545455	0.023810	0.407407	0.000000	1.000000
n6	0.860465	0.634146	0.037037	0.954545	0.687500
n7	1.000000	0.731707	0.000000	0.125000	0.000000
n8	0.097561	1.000000	0.585366	0.000000	1.000000
n9	0.163265	0.511628	0.000000	0.925000	1.000000
n10	0.159091	1.000000	0.692308	0.488372	0.024390
n11	0.230769	0.000000	0.770833	1.000000	0.142857
n12	0.000000	0.961538	0.974359	0.386364	0.025641
n13	0.955556	0.025000	0.545455	0.060606	0.957447
n14	0.000000	0.026316	0.000000	0.846154	0.795455
n15	0.800000	0.954545	0.057143	0.342857	0.545455
n16	0.000000	1.000000	1.000000	0.815789	0.111111
n17	0.918919	0.000000	0.534884	1.000000	0.068182
n18	0.189189	0.725000	0.052632	0.000000	0.037736
n19	0.000000	0.666667	0.962963	0.119048	1.000000
n20	0.860465	0.276596	0.903226	0.481481	0.173913

Table 1: 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

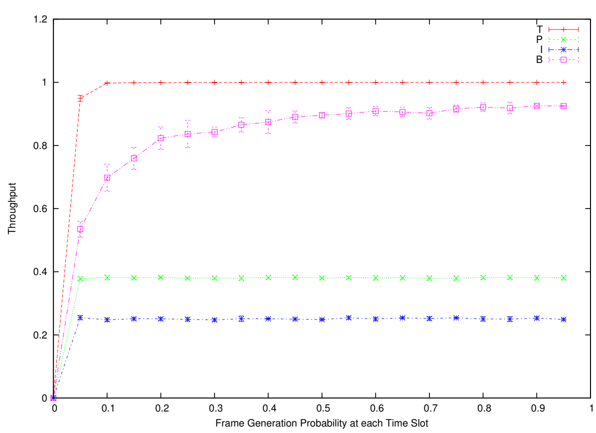

Figure 10: Average throughput on hub for full range of per slot, per station frame generation probability, tested for each MAC protocol $R = 5000$ slots, $T = 5$ trials

looks like some stations get to deliver most frames while others might not get any through on some trials....

-¿ probabalistic unfairness explanation - a couple 'lucky' stations get through early, while others keep colliding - lucky stations keep transmitting , unlucky ones have large (1024) slot backoff, never get to transmit

-¿ Plot/Figure -¿ Maybe just a table of undelivered/genrated ratios for station and trial -¿ point out that on some trials, some stations have 1, others are as low as 0.3

Describe results TODO: tables of undelivered/generated

Discussion

Explain results, contrast with expectations

—— 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? ——

Conclusions

Rehash discussion?

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