Computer Networks - CS 204

Rishit Saiya - 180010027, Assignment - 4

April 24, 2020

1

We use the idea of Conservation of Data with the factor of transmission keeping in mind. The required data rate (D) is as follows:

$$D = N \times 1000 \, \frac{bits}{packet} \times \frac{1}{100} \, \frac{packet}{seconds}$$

or,

$$D = 10 \times N \, \frac{bits}{second}$$

With the reference to the literature, we know that using the implementation of Unslotted/Pure ALOHA, the available data rate is 18% of 56,000 $\frac{bits}{second}$. Hence, finally equating both,

$$10 \times N \, \frac{bits}{second} = 0.18 \times 56,000 \, \frac{bits}{second}$$

or,

$$N = 1008$$

It is to be noted that, we used 18% for the pure ALOHA (which is maximum possible), so the resulting N is also maximum value.

2

With pure ALOHA, transmission can be started immediately whereas with the slotted ALOHA, transmission has to wait to the next slot.

3

Given that, the efficiency of slotted ALOHA is Np$(1 - p)^{(N-1)}$, where N is the number of Active nodes and the p is the probability of transmission.

Let us assume the Efficiency to be E.

$$E = Np(1-p)^{N-1}$$

We see that, efficiency is dependent upon the parameter p, so in order to find the maximum efficiency, we need to derive the value of p such that E will be maximum. We do the following:

$$\frac{dE}{dp} = 0$$

$$(1-p)^{N-1} - p(N-1)(1-p)^{N-2} = 0$$

$$1 - p = p \times (N-1)$$

or,

$$p = \frac{1}{N}$$

Maximum E is attained at E(p = $\frac{1}{N}$).

$$E = (1 - \frac{1}{N})^{N-1}$$

As N approaches ∞, we calculate E there.

$$E_{max} = \lim_{N\to\infty} E$$

$$E_{max} = \lim_{N\to\infty} (1 - \frac{1}{N})^{N-1}$$

$$E_{max} = \exp(\lim_{N\to\infty} (1 - \frac{1}{N} - 1) \times (N-1))$$

$$E_{max} = 1/e$$

4

For pure ALOHA, we follow the similar process as above. Here the Efficiency is given as follows:

$$E = Np(1-p)^{2(N-1)}$$

We see that, efficiency is dependent upon the parameter p, so in order to find the maximum efficiency, we need to derive the value of p such that E will be maximum. We do the following:

$$\frac{dE}{dp} = 0$$

$$(1-p)^{2(N-1)} - p(2(N-1))(1-p)^{2(N-1)-1} = 0$$

$$1 - p = p \times (2N-2)$$

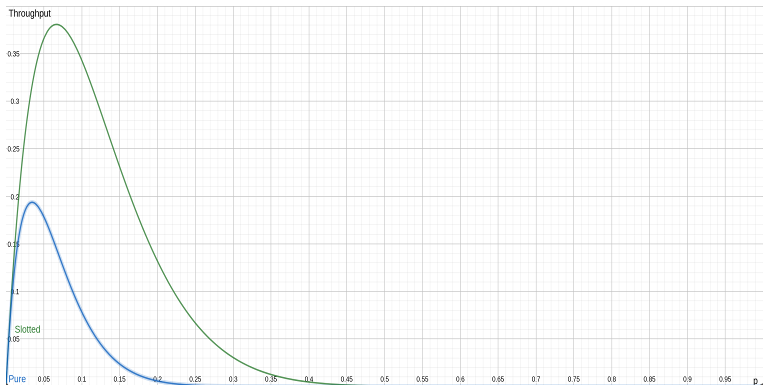

Figure 1: Pure ALOHA vs. Slotted ALOHA, N=15

or,

$$p = \frac{1}{2N-1}$$

Maximum E is attained at E(p = $\frac{1}{2N-1}$).

$$E = N(\frac{1}{2N-1})(1 - \frac{1}{2N-1})^{2(N-1)}$$

As N approaches ∞, we calculate E there.

$$E_{max} = \lim_{N\to\infty} E$$

$$E_{max} = \lim_{N\to\infty} N(\frac{1}{2N-1})(1 - \frac{1}{2N-1})^{2(N-1)}$$

$$E_{max} = \frac{1}{2} \times \exp(\lim_{N\to\infty}(1 - \frac{1}{2N-1} - 1) \times 2(N-1))$$

$$E_{max} = 1/2e$$

5

The graphs are as follows:

The efficiencies of Slotted and Pure ALOHA for N=15, is given in Figure 1.
The efficiencies of Slotted and Pure ALOHA for N=25, is given in Figure 2.
The efficiencies of Slotted and Pure ALOHA for N=35, is given in Figure 3.
For all given N, Slotted ALOHA gives higher efficiency than the Pure ALOHA. Furthermore, we observe that as the value of N increases, the graph becomes narrower. As the N is increasing, the area under the curves decreases but still it is to be noted that efficiencies remain constant irrespective of the values of N.

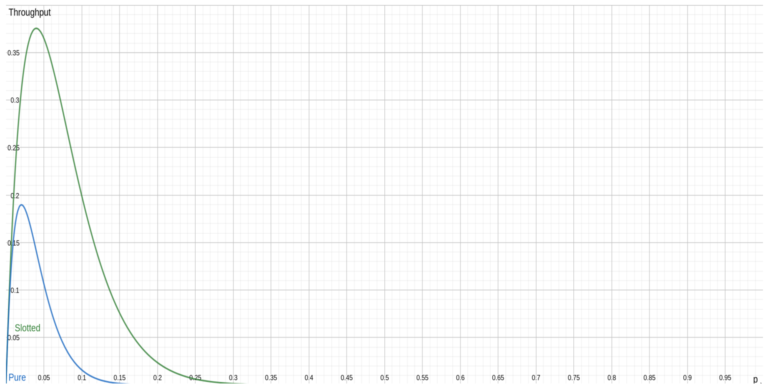

Figure 2: Pure ALOHA vs. Slotted ALOHA, N=25

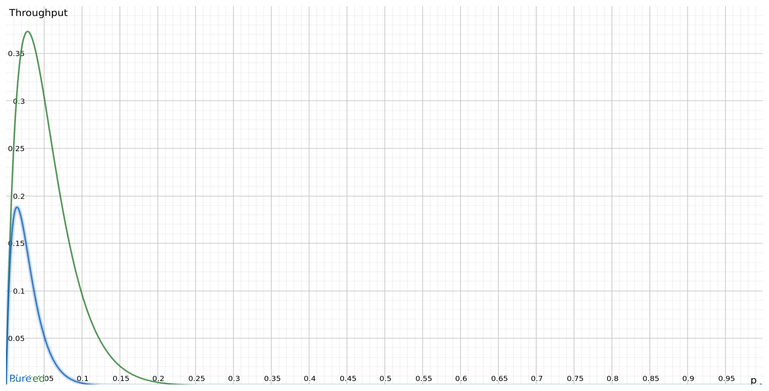

Figure 3: Pure ALOHA vs. Slotted ALOHA, N=35

6

In accordance with the Binary Exponential Backoff Algorithm, if the node is chosen after 5^{th} collision, then K is chosen from the set $\{0,1,2,...2^5-1\}$ and the corresponding delay would be K $\times$ 51.2 μsec.

$$P(K=4)=\frac{1}{32}$$

Where P is the probability that a node chooses K = 4, after the 5^{th} collision in CSMA/CD. To calculate the delay, we just have to do the following:

$$data_{delay}=4\times 512\,bits$$

or,

$$t_{delay}=\frac{4\times 512}{10\times 10^6}\,\frac{bits}{\frac{bits}{second}}$$
$$t_{delay}=204.8\,\mu sec$$

7

Given that G = 1001, and D = 101110, we calculate G expression for the sake of completeness which is given as follows:

$$G=1\times x^3+0\times x^2+0\times x^1+1\times x^0$$

or,

$$G=x^3+1$$

Clearly, from above G expression, r = 3.
The bit pattern D+r pattern becomes 10110000. To calculate the R, we do the following process.

Now, divide and XOR the message with divisor polynomial bits. Make resultant remainder to 5 bit again and that is the CRC send along with the message.

The R = 100 in this case.

8

Given that G = 10011, and D = 1010101010, we calculate G expression for the sake of completeness which is given as follows:

$$G=1\times x^4+0\times x^3+0\times x^2+1\times x^1+1\times x^0$$

or,

$$G=x^4+x+1$$

Clearly, from above G expression, r = 4.
The bit pattern D+r pattern becomes 10101010100000. To calculate the R, we do the following process.

Now, divide and XOR the message with divisor polynomial bits. Make resultant remainder to 5 bit again and that is the CRC send along with the message.

The R = 00100 in this case.

9

(a) Clearly, from G , r = 4.
The bit pattern D+r pattern becomes 10010101010000. To calculate the R, we do the following process:

Now, divide and XOR the message with divisor polynomial bits. Make resultant remainder to 5 bit again and that is the CRC send along with the message.

The R = 00000 in this case.

(b) Clearly, from G , r = 4.
The bit pattern D+r pattern becomes 01011010100000. To calculate the R, we do the following process:

Now, divide and XOR the message with divisor polynomial bits. Make resultant remainder to 5 bit again and that is the CRC send along with the message.

The R = 01111 in this case.

(c) Clearly, from G , r = 4.
The bit pattern D+r pattern becomes 10101000000000. To calculate the R, we do the following process:

Now, divide and XOR the message with divisor polynomial bits. Make resultant remainder to 5 bit again and that is the CRC send along with the message.

The R = 01001 in this case.

10

2-D Parity is the generalization of the simple 1-D parity scheme using the following steps:

1. Generate a M $\times$ N matrix of bits.
2. Add a (even or odd) parity bit to each row and to each column depending upon the parity scheme chosen.

Since we are using an even parity scheme, we obtain the matrix as given in Table 1.

	Bit Pattern				**Parity Bits**
Bit Pattern	1	1	1	0	**1**
	0	1	1	0	**0**
	1	0	0	1	**0**
	1	1	0	1	**1**
Parity Bits	**1**	**1**	**0**	**0**	**0**

Table 1: Fields of 2-D Even Parity Scheme for the given Bit Pattern

	Bit Pattern				**Parity Bits**
Bit Pattern	1	1	1	0	**1**
	0	1	1	0	**0**
	1	0	0	1	**0**
	1	1	0	1	**1**
Parity Bits	**1**	**1**	**0**	**0**	**0**

Table 2: Sender's Bit Pattern with corresponding Parity Bits

11

Consider a case, where the sent bit pattern is as given in Table 2. Now consider that the received bit pattern is as given in Table 3.

When the receiver receives an erroneous frame with Data bits and the parity bits together, the parity bits in the frame will be the ones which were computed by the sender using the actual error-free data. So, now when the frame arrives at the receiver, it simply calculates the number of 1's in every row and every column, and then compares the same with the corresponding parity bits (at every row and every column) as sent by the sender. Whenever there is a mismatch in the number of 1s and the value of the parity bit, it knows that there is an error in that particular row/column. To be sure about the particular bit index, it further looks at the parity bit corresponding row and column, i..e, if the bit has an index (i,j), then for error to happen at that index, the corresponding row as well as column parity

	Bit Pattern				**Parity Bits**
Bit Pattern	1	1	1	0	**1**
	0	$\underline{\boldsymbol{\mathit{0}}}$	1	0	**0**
	1	0	0	1	**0**
	1	1	0	1	**1**
Parity Bits	**1**	**1**	**0**	**0**	**0**

Table 3: Receiver's Bit Pattern with 1 error and corresponding Parity Bits

	Bit Pattern				**Parity Bits**
Bit Pattern	1	1	1	0	**1**
	0	***0***	1	0	**0**
	1	***1***	0	1	**0**
	1	1	0	1	**1**
Parity Bits	**1**	**1**	**0**	**0**	**0**

Table 4: Receiver's Bit Pattern with 2 error and corresponding Parity Bits

as computed by the receiver will be different from the parity bits sent by the sender.

As for the above example is concerned, the bit at bit_pattern[2][2] (Underlined bit) was the error and it was detected and corrected.

Consider that Table 4 as the receiver's bit pattern with 2 errors. It has errors at bit_pattern[2][2] & bit_pattern[3][2].

Clearly, this mechanism can detect those errors but cannot correct them.