# Experiment Intuition & Finding

## Figure 8

Implementation:

1. As wait is a build in var, we need to modify red.tcl and set wait\_ to corresponding value

Algorithm for wait:

if (count1 \* p < 1.0)

p = 0.0;

else if (count1 \* p < 2.0)

p /= (2.0 - count1 \* p);

else

p = 1.0;

Algorithm for no wait:

if (count1 \* p < 1.0)

p /= (1.0 - count1 \* p);

else

p = 1.0;

Intuition

1. wait ensure after a drop packet, proceeding p packet will not be drop
2. wait increase the spacing between drop
3. Pr(x=n) = (2-2p)/2 \* (2-3p)/(2-2p)\*…..\*p/(2-np) = p/2

Observation

1. Delay increase, Throughput increase for wait = true
2. Fit our assumption as Pr(x=n) decrease

## Figure 9

Implementation

To implement geometric random variable, modify\_p() is modified as follow

REDQueue::modify\_p(double p, int count, int count\_bytes, int bytes,

int mean\_pktsize, int wait, int size, int geometric) {

//adding geometric interdrop periods as test

if(geometric) {

p\*=p;

printf("Geometric on\n");

}

Intuition

1. For geometric random variable, according to section 7, method 1, the next drop prob = current drop prob \*p
2. It is not fair to busty traffic as the probability of dropping nearby packet is high

Observation

1. Huge delay, high throughput

## Figure 10

Implementation

As gentle\_ is a build in var, , we need to modify red.tcl and set wait\_ to corresponding value

if (gentle && v\_ave >= th\_max) {

// p ranges from max\_p to 1 as the average queue

// size ranges from th\_max to twice th\_max

p = v\_c \* v\_ave + v\_d;

}

Intuition

1. Higher throughput and higher delay if 2 Max Threshold < Queue Size
2. If Max Threshold = Queue Size, then it will be forced to drop tail, then the throughput and delay should be the same in this situation.

Observation

1. Delay is higher for throughput is low (gentle = true)