# 1 Opt-Ack Attack

By the definition and algorithm with attacks, we know we want to implement Opt-ack ideas to prevent attacks.

```
1: maxwindow − 65535 ∗ 2
wscale
2: n − |{v1, . . . , vn}|
3: for i − 1 . . . n do
4: connect(mss,wscale) to vi , get isni
5: acki − isni + 1 ; wi − mss
6: end for
7: for i − 1 . . . n do
8: send vi data request { http get, ftp fetch, etc. . . }
9: end for
10: while true do
11: for i − 1 . . . n do
12: acki − acki + wi
13: send ACK for acki to vi { entire window}
14: if wi < maxwindow then
15: wi − wi + mss
16: end if
17: end for
18: end while
```

It is possible to modify TCP to eliminate this undesirable behavior entirely, without requiring assumptions of any kind about receiver behavior

Also from the paper, we have three main attacks that a it is possible to modify TCP to eliminate this undesirable behavior entirely, without requiring assumptions of any kind about receiver behavior.

# 2 Implication

$$\mathcal{T}_{\max} = \beta \times 65535 \times 2^{\text{wscale}} \times \frac{1}{mss} + \frac{1}{40}$$

Practical limits appeared in those maxium number of victims. Therefore it is hard to implementing the attack accurately predict which segments the victim is sending and ensure that the corresponding ACKs arrive at the correct time.

$$\text{processing delay} = \frac{\lfloor \text{ cwnd } / \text{mss} \rfloor \times (40 + \text{mss})}{\text{TargetBandwidth}}$$

## 3 Attack evaluation

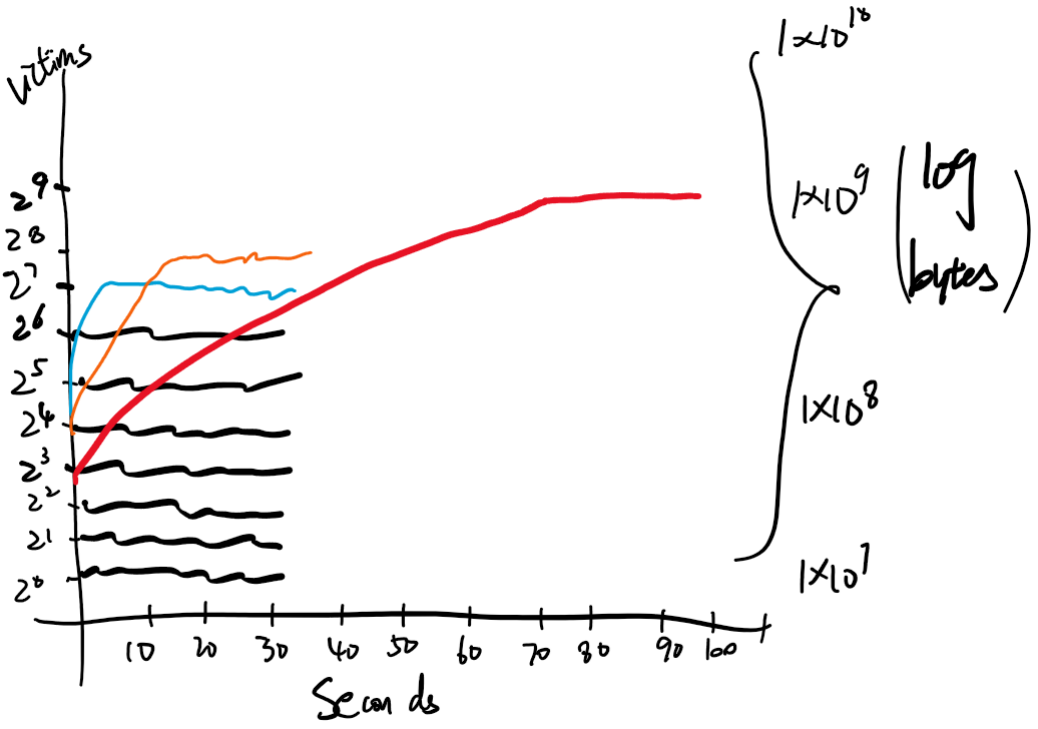


Figure 1: Hypothetical Simulation Results

There was issue when impleting the final coding part, but per my analysis part, i predict the hypothetical simulation results will behave like the graph that i drawing above. Where as we can see, when range from $2^0$ to $2^6$ Victims, their log bytes increase roughy linearly. After $2^7$ victims, it starts to have a rapid growth after the seconds start, then approaches linear. As for $2^9$ victims, the log bytes goes much rapid and approaches roughly $1 * 10^9$ log bytes.