19005010Y SAMBIT BEHERA

Q1 We need to design a variant of TCP protocol that can subtly detect if there is a TCP Reno in the flow and than adjust its protocol to behave like TCP Reno, from initially being TCP Vegas. Here is the following protocol -

- Initially ~~act~~ the protocol starts off as TCP vegas, with the corresponding values for α, β determined experimentally previously.

$$Diff = \frac{Cwind}{BaseRTT} - \frac{Cw}{RTT} \propto \text{queuing delay}$$

In congestion Avoidance,

If $diff < \alpha \Rightarrow$ ~~Cw~~ Cw increases by 1MSS per RTT

If ~~$\beta < di$~~ $\alpha < Diff < \beta \Rightarrow$ cw is not modified

we introduce a γ, $\gamma > \beta$ such that,

~~$< Diff < \beta$~~

~~If $\beta < Diff < \gamma$~~

- If $Diff > \gamma \Rightarrow$ TCP protocol starts behaving like TCP Reno [changing protocol] and AI starts taking place again.

If $\beta < Diff < \gamma \Rightarrow$ cw decreases by 1MSS per RTT

~~Reasong~~ Reasoning is that, if all flows in the network are TCP Vegas, then after Diff crosses β, each flow starts decreasing its cw, resulting in decrease in congestion at the router and hence fall in value of Diff.

If, the value of Diff crosses the new threshold γ, we infer that there is some greedy Reno flow in the network, which is increasing the congestion and queuing delay at the routers.

As, there is no entry/exit during the flow, congestion neither increases or drops suddenly, and hence value of Diff is maintained in a continous fashion.

Hence, our algorithm ensures the conditions as follows -

a) End-to-End congestion control -

From the design of the ~~algorith~~ protocol, it is wholly based on measurements at the sender/receiver and doesnot require explicit special information from routers.

b) All TCP flows have the same RTT. Hence Diff value will be almost the same for all the flows.

According to the protocol, it starts off as TCP Vegas, and if all flows are TCP Vegas, then after Diff crosses β, each flow starts decreasing their congestion window, resulting in decrease in congestion at the router and lowering of Diff value altogether. Hence this condition is satisfied.

c) If there is some TCP Reno flow in the network, when the protocol has Diff value over β, it decreases congestion window, still the queuing delay at routers increases resulting in a higher Diff value. On crossing this threshold γ we are inferring that there is a TCP Reno in the network, and protocol switches to TCP Reno.

d) Congestion window for our protocol is based on the value of Diff.

$$\text{Diff} = \frac{CW}{BaseRTT} - \frac{CW}{RTT} = \frac{CW\left(\overbrace{RTT - BaseRTT}^{\text{Queuing delay}}\right)}{RTT \times BaseRTT}$$

Hence, <u>$\text{Diff} \propto \text{Queuing delay}$</u>

Hence, our protocol implicitly infers that there is queuing delay in the network.

Packet loss is inferred on getting 3DUP ACKs (3 duplicate acknowledgements) which is common to both TCP Vegas and TCP Reno.

Q2 For each flow i, congestion window size at time $t = w_i(t)$

and each flow has $RTT = T_i$

Hence, instantaneous bitrate $= \frac{w_i(t)}{T_i} = \frac{df_i}{dt}$ —(1)

For a particular link, bandwidth is C bits/sec

Because cw is incremented by AI, we have

$$\frac{dw_i(t)}{dt} = \frac{1}{T_i}$$

$$w_i(t) - w_0(t) = \frac{t - t_0}{T_i}$$ t_0 = when $w_i(t)$ is reset

$$w_i(t) = \frac{t - t_0}{T_i}$$ t = for any time t

But,

$$\sum \frac{w_i(t)}{T_i} = C$$

$$\Rightarrow \sum \frac{(t_f - t_0)}{T_i^2} = C$$

$$\Rightarrow (t_f - t_0)\left(\frac{1}{T_1^2} + \frac{1}{T_2^2}\right) = C$$

$$\Rightarrow \boxed{t_f - t_0 = \frac{C T_1^2 T_2^2}{T_1^2 + T_2^2} = K}$$ (suppose) some const.

t_f = time of loss

Now, total amount of data transmitted for each flow in the time period is

$$\int_{t_0}^{t_f} df_i = \int_{t_0}^{t_f} \frac{w_i(t)}{T_i}\, dt$$ from (1)

$$\Rightarrow f_i = \int_{t_0}^{t_f} \frac{(t - t_0)}{T_i^2}\, dt = \frac{1}{2T_i^2}(t_f - t_0)^2$$

Hence,

$$f_i = \frac{1}{2T_i^2}(t_f - t_i)^2 \qquad t_f - t_1 = \frac{cT_1^2 T_2^2}{T_1^2 + T_2^2}$$

$$f_1 = \frac{c^2 T_1^2 T_2^4}{2(T_1^2 + T_2^2)^2} \qquad f_2 = \frac{c^2 T_1^4 T_2^2}{2(T_1^2 + T_2^2)^2}$$

Also, $f_i = \frac{K}{2T_i^2} \Rightarrow f_i \propto \frac{1}{T_i^2}$

We can observe that total amount of data transmitted is inversely proportional to the square of RTT.

The flow which has lower RTT value will have higher amount of data transmission.

$$f_i \propto \frac{1}{RTT_i^2}$$

From the above observation, we can conclued that TCP RENO is not fair, as ~~diff~~. two flows with different RTTs do not get the same bandwidth. Flow with lower RTT has higher bandwidth value.