Lecture-28: TCP Vegas

1 TCP-Vegas

We now consider another variation of TCP called TCP-Vegas. TCP-Vegas uses queueing delay to infer congestion in the network. We denote the observed round trip time RTT by $T_r(t)$ and the estimated propagation delay by $T_{pr}(t)$ for source r at time t. We assume that the propagation delay of the route is equal to the smallest RTT seen by the source, i.e. $T_{pr} \triangleq \inf_{s \leqslant t} T_r(s)$ This is a reasonable assumption if we assume that queues empty occasionally in which case the only delay is the propagation delay. Any excess delay above this amount would be queueing delay and we denote it $T_{qr}(t) \triangleq T_r(t) - T_{pr}(t)$. The objective of the algorithm is to calculate the value of window size such that there is minimal queueing delay. When this occurs, the rate of generation of packets x_r is equal to the available capacity $\min_{\ell \in \mathcal{L}_r} c_\ell$ on the route r. We now study the details of this algorithm.

1.1 A delay based algorithm

Let $W_r(t)$ be the window size of source r at time t. If there was no queueing delay, the throughput for source r at time t would be approximately given by $e_r(t) \triangleq \frac{W_r(t)}{T_{pr}}$. We call this expected or desired throughput. However, the actual throughput is the number of packets that make it successfully to the destination in a fixed amount of time. To calculate this quantity, the source r sends a marked packet and waits for it to be acknowledged. The duration of time that it takes for this event to occur is the round-trip time RTT denoted $T_r(t) = T_{pr}(t) + T_{qr}(t)$. Suppose during this time, the source receives $S_r(t)$ acknowledgments, then the actual throughput for source r at time t is estimated as $a_r(t) \triangleq \frac{S_r(t)}{T_r(t)}$. If there is no loss in the system, then $S_r(t) = W_r(t - T_r)$. Whenever we receive the acknowledgment for a marked packet at some time t, we have two values
(a) the expected throughput $e_r(t)$ and
(b) the actual throughput $a_r(t)$.
If $a_r(t) < e_r(t)$, it means that our transmission rate is too high, so we should cut down the window size. On the other hand, if $a_r(t) > e_r(t)$, it means that the estimate of the available rate is too low and we should increase the window size. Formally, we define constants α and β, with $\alpha \leqslant \beta$ and proceed as follows.

- If $(e_r(t) - a_r(t)) \in [\alpha, \beta]$, then do nothing. This means that our estimate of the throughput is fairly accurate, and everything is as it should be.
- If $(e_r(t) - a_r(t)) > \beta$, decrease the window size by 1 for the next RTT. This means that our estimate is too high and the window size must be reduced.
- If $(e_r(t) - a_r(t)) < \alpha$, increase the window size by 1 for the next RTT. This means that our estimate is too low and the network can support more than we think.

Note that both the increase and decrease of window size are linear in nature. Also, the algorithm uses the usual slow start mechanism, with exponential growth initially. The behavior of TCP-Vegas under ideal conditions would look something like Figure 4.3.

1.2 A resource allocation algorithm

We interpret TCP-Vegas as a resource allocation algorithm in the utility maximization framework. We assume $\alpha = \beta$ and the propagation delay is estimated accurately, i.e., for source r, we have $T_{pr}(t) = T_{pr}$

for all times t. We denote the equilibrium window size as $\hat{W}_r$ and the equilibrium queueing delay as $\hat{T}_{qr}$ for source r. At equilibrium, the difference between the estimated and the actual throughput remains unchanged as $\hat{e}_r - \hat{a}_r = \alpha$, with the window size $\hat{W}_r$ and the number of acknowledgements $\hat{S}_r$ received in an RTT being the same as $\hat{W}_r = \hat{S}_r$. Then

$$\frac{\hat{W}_r}{T_{pr}} - \frac{\hat{W}_r}{T_{pr} + \hat{T}_{qr}} = \alpha.$$

At equilibrium, the transmission rate $\hat{x}$ is approximately $\hat{x}_r = \frac{\hat{W}_r}{T_{pr}+\hat{T}_{qr}}$, to rewrite the above equation as $\alpha \frac{T_{pr}}{\hat{x}_r} = \hat{T}_{qr}$. Now that we know what the equilibrium transmission rate looks like, let us study what the equilibrium queueing delay $\hat{T}_{qr}$ would look like. At link ℓ, the equilibrium queue length is denoted by $\hat{b}_\ell$ and link capacity by c_ℓ, then the equilibrium queueing delay is $\frac{\hat{b}_\ell}{c_\ell}$. So we have $\hat{T}_{qr} = \sum_{\ell \in \mathcal{L}} R_{\ell,r} \frac{\hat{b}_\ell}{c_\ell}$.

At link ℓ, if equilibrium link load $\hat{y}_\ell \triangleq \sum_{k \in \mathcal{S}} R_{\ell,k} \hat{x}_k < c_\ell$, then there is no queueing delay, i.e., $\hat{b}_\ell = 0$ in this case. That is, $\hat{b}_\ell > 0$ only if $\hat{y}_\ell \geqslant c_\ell$. We note that the aggregate equilibrium transmission rate of all flows using link ℓ cannot exceed the link capacity c_ℓ, we cannot possibly have that $\hat{y}_\ell > c_\ell$. Therefore, if $\hat{b}_\ell > 0$, then $\hat{y}_\ell = c_\ell$. Thus, the equilibrium conditions for all sources $r \in \mathcal{S}$ and links $\ell \in \mathcal{L}$, are

$$\alpha \frac{T_{pr}}{\hat{x}_r} = \sum_{\ell \in \mathcal{L}} R_{\ell,r} \frac{\hat{b}_\ell}{c_\ell}, \qquad \hat{y}_\ell = \sum_{k \in \mathcal{S}} R_{\ell,k} \hat{x}_k, \qquad \frac{\hat{b}_\ell}{c_\ell}(\hat{y}_\ell - c_\ell) = 0.$$

However, these are the Karush-Kuhn-Tucker conditions for the utility maximization problem

$$\hat{x} \triangleq \arg\max \left\{ \sum_{r \in \mathcal{S}} U_r(x_r) : x \in \mathcal{D} \right\}, \qquad U_r(x_r) \triangleq \alpha T_{pr} \ln x_r, \qquad \mathcal{D} \triangleq \left\{ x \in \mathbb{R}_+^{\mathcal{S}} : y \leqslant c \right\},$$

where $\hat{p}_\ell \triangleq \frac{\hat{b}_\ell}{c_\ell}$ is the equilibrium price of link $\ell \in \mathcal{L}$. Thus, TCP-Vegas is weighted-proportionally fair. If we let each flow have a different value of α, i.e., we associate α_r with route r, then the equilibrium rates will maximize $\sum_{r \in \mathcal{S}} U_r(x_r)$ where $U_r(x_r) \triangleq \alpha_r T_{pr} \ln x_r$ for each source $r \in \mathcal{S}$.

1.3 Relation to dual algorithms and extensions

We now consider the relationship between TCP-Vegas and dual algorithms. A weighted proportionally fair dual algorithm would use the controller obtained by substituting $U_r(x_r) \triangleq w_r \ln x_r$ as the utility function, which yields

$$x_r = (U_r')^{-1}(q_r) = \frac{w_r}{q_r}, \qquad \dot{p}_\ell = h_\ell(y_\ell - c_\ell)_{p_\ell}^+.$$

To avoid confusion, we note that $w_r \triangleq \alpha_r T_{pr}$ is the weight assigned to source r and is unrelated to the window size $W_r(t)$. If we choose $h_\ell = \frac{1}{c_\ell}$, then the price function p_ℓ of a link ℓ becomes the queueing delay $\frac{b_\ell}{c_\ell}$ experienced by packets using that link ℓ, which when aggregated over all links traversed over the route r

$$q_r(t) \triangleq \sum_{\ell \in \mathcal{L}} R_{\ell,r} p_\ell(t) = \sum_{\ell \in \mathcal{L}} R_{\ell,r} \frac{b_\ell(t)}{c_\ell} = T_{qr}(t),$$

and added to a constant propagation delay T_{pr}, is the feedback $T_r(t) = T_{pr} + T_{qr}(t)$ that is used in the Vegas algorithm.

Let us study the source rates used in Vegas more closely to create a fluid model equivalent. From the algorithm description for $\alpha = \beta$, it follows that the increase in the congestion window $W_r(t)$ under the Vegas algorithm is equal to

$$-\operatorname{sign}\Big(\frac{W_r(t)}{T_{pr}} - \frac{W_r(t)}{T_{pr} + T_{qr}(t)} - \alpha\Big),$$

where $\text{sign}(z) = -1\mathbb{1}_{\{z<0\}} + \mathbb{1}_{\{z>0\}}$. Using the approximation $x_r(t) \triangleq \frac{W_r(t)}{T_{pr}+T_{qr}(t)}$, we can rewrite the conditions as

$$-\text{sign}\Big(x_r(t)T_{qr}(t) - \alpha T_{pr}\Big).$$

Denoting the queue length and queueing delay at link ℓ at time t as $b_\ell(t)$ and $p_\ell(t) \triangleq \frac{b_\ell(t)}{c_\ell}$, we have $T_{qr}(t) \triangleq \sum_{\ell\in\mathcal{L}} R_{\ell,r}\frac{b_\ell(t)}{c_\ell} = \sum_{\ell\in\mathcal{L}} R_{\ell,r}p_\ell(t)$. We have used $p_\ell(t)$ to denote the queueing delay at link ℓ, which acts as the price function for TCP Vegas. Combining the above expressions, the condition for increase/decrease of congestion window becomes

$$-\text{sign}\Big(x_r(t)\sum_{\ell\in\mathcal{L}} R_{\ell,r}p_\ell(t) - \alpha T_{pr}\Big).$$

The window control algorithm can be written as

$$W_r(t+T_r(t)) = \Big[W_r(t) + \frac{1}{T_{pr}+T_{qr}(t)}\text{sign}(\alpha T_{pr} - x_r(t)T_{qr}(t))\Big]^+_{W_r(t)}.$$

Thus, we can now write down the differential equations describing the TCP-Vegas algorithm as

$$\dot{p}_\ell(t) = \frac{1}{c_\ell}(y_\ell - c_\ell)^+_{p_\ell}, \quad \dot{W}_r(t) = \Big[\frac{\text{sign}(\alpha T_{pr} - x_r(t)T_{qr}(t))}{T_{pr}+T_{qr}(t)}\Big]^+_{W_r}, \quad x_r(t) \triangleq \frac{W_r(t)}{T_{pr}+T_{qr}(t)}, \quad T_{q,r}(t) \triangleq \sum_{\ell\in\mathcal{L}} R_{\ell,r}p_\ell(t).$$

The above is not the same as the dual algorithm that we derived before. However, the price update dynamics are the same as the price update for the dual algorithm. Each source r, attempts to increase or decrease the rate based on whether x_r is less than or greater than $\alpha\frac{T_{pr}}{T_{qr}(t)}$. Denoting the equilibrium values of rate and queueing delay for route $r \in \mathcal{S}$ and price for link $\ell \in \mathcal{L}$ as $\hat{x}_r, \hat{T}_{qr}, \hat{p}_\ell$ respectively, it is clear the source attempts to drive the system towards

$$\hat{x}_r = \alpha\frac{T_{pr}}{\hat{T}_{qr}} = \alpha\frac{T_{pr}}{\sum_{\ell\in\mathcal{L}} R_{\ell,r}\hat{p}_\ell} = \frac{\alpha T_{pr}}{q_r},$$

which is the desired source behavior for a dual congestion controller. Thus, one can interpret TCP-Vegas as an algorithm that approximates the dual congestion control algorithm.

1.4 FAST TCP

A modification of TCP-Vegas called FAST-TCP has been suggested for very high-speed networks. In FAST-TCP, the window size is increased or decreased depending upon how far the window size is from a desired equilibrium point. The fluid model describing the protocol is

$$\dot{p}_\ell(t) = \frac{1}{c_\ell}(y_\ell - c_\ell)^+_{p_\ell}, \quad \dot{W}_r(t) = \gamma_r(\alpha_r - x_r(t)T_{qr}(t))^+_{W_r}, \quad x_r(t) = \frac{W_r(t)}{T_{pr}+T_{qr}(t)}, \quad T_{q,r}(t) \triangleq \sum_{\ell\in\mathcal{L}} R_{\ell,r}p_\ell(t),$$

where α_r determines the desired equilibrium point and γ_r is a scaling constant. Replacing the sign function in TCP-Vegas with the difference between the current operating point and the desired equilibrium allows FAST-TCP to rapidly approach the desired equilibrium point.