4.

a) *(Bookwork.)* 'CS' stands for 'carrier sense'. It means: don't start transmission as soon as you have a packet to send; instead, listen to the channel first, and if it is busy then don't transmit but instead back off.

b) *(Bookwork.)* A packet which arrives at time t is successfully transmitted only if no others interfere with it, i.e. no others start in $[t-\Delta, t+\Delta]$. Since packet arrivals are Poisson, the number of packets which arrive in $[t, t+\Delta]$ is Poisson with parameter $\lambda\Delta$, so the probability that no packets arrive in $[t, t+\Delta]$ is $e^{-\lambda\Delta}$. Similarly, the probability that no packets arrive in $[t-\Delta, t]$ is $e^{-\lambda\Delta}$. Thus the probability of no interference is $e^{-2\lambda\Delta}$.

Thus the *rate* of successful transmissions is $\lambda e^{-2\lambda\Delta}$. The maximum achievable rate is

$$\max_{\lambda} \lambda e^{-2\lambda\Delta} = \frac{1}{2e\Delta}, \quad \text{attained at } \lambda = \frac{1}{2\Delta}.$$

If we had an omniscient scheduler it could schedule one packet every Δ, i.e. achieve a tranmission rate of $1/\Delta$. Thus the efficiency is

$$\text{efficiency} = \frac{1/2e\Delta}{1/\Delta} = \frac{1}{2e}.$$

c) Now, a packet which arrives at time t can only be dropped if some other packet started in $[t-\Delta, t]$. Any new packets in $[t, t+\Delta]$ are dropped immediately and do not interfere. As above, the packet loss probability is $e^{-\lambda\Delta}$. Carrying through the algebra above, the maximum possible efficiency is $1/e$.

d) *(Bookwork.)* The number of packets which attempt transmission in timeslot $[t, t+1]$ is $X_t = N_t + R_t$. That is, all newly arrived packets attempt transmission, and so do R_t of the B_t backlogged packets.

At time $t+1$, what does the backlog consist of? It consists of the old backlog, plus any new packets which attempted transmission but failed, minus the number of packets which are no longer part of the backlog i.e. which have succeeded in transmission. Consider three cases: if $N_t = 0$ then $B_{t+1} = B_t - 1_{X_t=1}$, since there are no new packets which enter the backlog; if $N_t = 1$ and $R_t = 0$ then $B_{t+1} = B_t$ since the new packet is transmitted successfully and no others attempt transmission; if $N_t > 1$ or if $N_t = 1$ and $R_t > 0$ then all the new packets attempt transmission and are blocked and so enter the backlog. These three cases cover all contingencies, and in each of them the suggested equation is valid.

$$\mathbb{P}(B_{t+1} - B_t = i) = \begin{cases} -1 & \text{with prob. } \mathbb{P}(N_t = 0, R_t = 1) = e^{-\lambda} b(1-p)^{b-1} p \\ 0 & \text{otherwise} \\ 1 & \text{with prob. } \mathbb{P}(N_t = 1, R_t \geq 1) = \lambda e^{-\lambda}[1-(1-p)^b] \\ 2 & \text{with prob. } \mathbb{P}(N_t = 2) = \lambda^2 e^{-\lambda}/2! \text{ etc.} \end{cases}$$

e) Let $Y_t = N_t + R_t$. This is the number of packets that are scheduled to attempt transmission in $[t, t+1]$. The actual number that attempt transmission is $X_t = \min(Y_t, 1)$, since no more than one of them actually attempts transmission. The same equation holds, with this new X_t.

$$\mathbb{P}(B_{t+1} - B_t = i) = \begin{cases} -1 & \text{with prob. } \mathbb{P}(N_t = 0, R_t \geq 0) = e^{-\lambda}[1 - (1-p)^b] \\ 0 & \text{otherwise} \\ 1 & \text{with prob. } \mathbb{P}(N_t = 1) = \lambda e^{-\lambda} \\ 2 & \text{with prob. } \mathbb{P}(N_t = 2) = \lambda^2 e^{-\lambda}/2! \text{ etc.} \end{cases}$$

5.

a) *(Bookwork.)* First, the relationship between window size and rate. If TCP has a window size w, this means that w packets are sent every RTT, and so the average transmit rate is $x = w/RTT$.

Now suppose drops are period at rate p, i.e. one drop every $1/p$ packets. Then TCP window size follows a periodic sawtooth, say of period T, going from window $w_{\min}$ to $w_{\max}$. During the window increase phase, TCP increases w by $1/w$ for every acnkowledgement, which occur roughly at rate $x = w/RTT$. Thus w increases at roughly rate $1/RTT$, thus

$$w_{\max} = w_{\min} + T\frac{1}{RTT}.$$

When drops occur, TCP cuts its window by half:

$$w_{\min} = \frac{1}{2}w_{\max}.$$

Solving these equations simultaneously,

$$T = wRTT2/3$$

where $w = (w_{\min} + w_{\max})/2$ is the average window size. Now, in a single sawtooth, the overall average transmit rate is $x = w/RTT$, so there are wT/RTT packets sent every sawtooth. By assumption there's one drop every $1/p$ packets. Thus

$$\frac{wT}{RTT} = \frac{1}{p}$$

which, rearranged, gives the throughput formula

$$x = \frac{\sqrt{3/3}}{RTT\sqrt{p}}.$$

b) Substituting in TCP's parameters,

$$\frac{1}{p\hat{w}} = \frac{\hat{w}}{2}$$

which gives

$$\hat{w} = \frac{\sqrt{2}}{\sqrt{p}} \quad \text{or} \quad x = \frac{\sqrt{2}}{RTT\sqrt{p}}.$$

This is has the same form as the TCP throughput equation but is out by a constant factor.

c) With $\alpha = \beta = 1$, the heuristic equation gives

$$x = \frac{\sqrt{A/B}}{RTT\sqrt{p}}.$$

We want this not to depend on RTT. Therefore we want

$$\frac{A}{B} = \kappa RTT^2$$

for some arbitrary constant κ. This can be allocated however we like to A and B, e.g. $A = \kappa RTT$, $B = 1/RTT$.

d) Let the reference RTT be RTT_0. Then, for fairness, we want

$$\frac{\sqrt{\kappa}}{\sqrt{p}} = \frac{\sqrt{2}}{RTT_0\sqrt{p}} \qquad \Longrightarrow \qquad \kappa = \frac{2}{RTT_0^2}.$$

e) We'll work with the generalized window control, with parameters A and B (and $\alpha = \beta = 1$). When the window is cut, window size decreases from $w_{\max} = x_{\max} RTT$ to $(1-B)w_{\max}$. To recover, it needs to regain $Bw_{\max}$. As above, the window size increases at steady rate A/RTT in the absence of drops. Thus, the time T to recover satisfies

$$T\frac{A}{RTT} = Bw_{\max} \qquad \Longrightarrow \qquad T = \frac{B}{A}w_{\max}RTT = \frac{B}{A}x_{\max}RTT^2.$$

For TCP, this time is $x_{\max}RTT^2/2$. For modified TCP, this time is $x_{\max}RTT^2 = x_{\max}/\kappa$. Observe that there is no RTT dependence for modified TCP.

The time taken to reach $x_{\max}$ starting from near-nothing is, as above,

$$T = \frac{w_{\max}RTT}{A} = \frac{x_{\max}}{B\kappa}.$$

To keep this small, we want B large.