1. Consider a single TCP Tahoe connection with a single source; this connection has a packet loss whenever window is 16. What is the range in which its window size oscillates? If the RTT =1 sec and the optimal window =16, what is the utilization of the bottleneck link when the source has a window size \(w\)? Use this to compute the mean utilization of the bottleneck if it only carries one source.

**Answer:** the window oscillates between 1 and 16. Slow Start threshold = 8. The max throughput is 16pkts/sec (full pipe). If window = \(w\), the throughput is \(w\). Average throughput is obtained by computing the average window over the slow start and congestion periods of the repetitive cycle. One obtains approximately 9.25 pkts/sec (roughly, half of the max throughput)

2. Consider a CDMA system. Why is power control critical to prevent interference between users? Is power control equally critical in GSM to prevent users to interfere with each other?

**Answer:** In CDMA all users share the same spectrum. A receiver is immune from interference from other channels if these are perfectly orthogonal and their spreading sequences are synchronized. Without perfect orthogonality/synchronization there is interference among users in the same cell. The interference is proportional to the tx power of the interferer. Power control is thus required to adjust the transmitting power of a mobile to a level that does not interfere with a neighbor receiving a signal from another transmitter. In GSM, each user channel has its own time slot. No interference between users. So, power control is not required in the data phase.

3. Consider a UMTS system based on WCDMA with chip rate = 4 Mchips/sec. How many simultaneous mobile users with 64Kbps each can the system support? What spreading ratio would they require? Now, assume these users after acquiring the channel are all transmitting data (as in a wireless LAN). Would it make sense to use the 802.11 MAC protocol? Or can you think of a better protocol?

**Answer:** The system can support 64 users, with orthogonal codes and a spreading ratio of 64 chips per bit. To get this result, note that if a code uses N chips per bit, the associated rate is given by \(R = \frac{4}{N}\) Mbps. Thus, \(N = \frac{4}{R}\) chips/bit = 64. The spreading ratio than is 64 chips/bit. The number of users is the number of leaves in an orthogonal frequency tree with 6 levels (since with 6 levels we get code length = 64). The number of leaves is the same as the number of chips per bit = 64 (by construction).
After code assignment, the users are orthogonal and have a dedicated channel each. No need to do per packet channel access and acquisition as in 802.11. The MAC layer may be a point to point protocol with optional ARQ.

4. In class we discussed the issue of fairness across TCP connections. In studying the “fairness diagram” used to prove fairness, we recognized that if two TCP connections have different propagation delay, they will not converge to “fair share”. Can you explain why? If all the connections sharing a bottleneck knew the “average” RTT among them, could you suggest a method for fair share? Comment on the difficulty to share this information among the connections.

Answer: recall that in congestion avoidance each connection increases its window by one unit every RTT. This is unfair if RTTs are different. Naturally, if a connection has RTT$_{\text{min}} = 100\text{ms}$, say, and has learned that the average is RTT$_{\text{avg}} = 50\text{ms}$, it simply increases the window by 2 units after an RTT. Unfortunately, it is difficult to disseminate the RTT$_{\text{avg}}$ to all the involved flows. Suppose a very sophisticated router (in the bottleneck) could intercept flows, figure out RTT for individual flows and send out the RTT$_{\text{avg}}$ info to the flows involved. Still, how can the router determine it is in the bottleneck? In practice, the fair share across TCP connections with different prop delays has been achieved only with per flow queues, forcing min-max bandwidth share across the flows. XCP unique resource shuffling among flows achieves the same results without per flow queuing (but it requires per packet state).