Lecture 3

Today: (1) Interference (from Lecture 2); (2) Trunking

- Reading for today’s lecture: 3.5, 3.6
- TA (Jordan Kemp) office hours: 9:00-noon, Fridays, in 2350 MEB. HW deadline will be postponed to 2:00pm, Friday. This is a permanent change.

0.1 Adjacent Channel Interference

Our standard, non-ideal radios do not perfectly filter out any out-of-band signals. Thus any signal that a mobile sends in another channel (besides its assigned channel) is interference at the BS w.r.t. the desired signal sent by another mobile in that channel. Each mobile’s receiver also must filter out out-of-band signals from the BS, which does send signals on all channels. One standard way of making this problem less difficult is to assign non-adjacent channels within each cell’s channel group.

Example: Assign $S = 50$ channels into groups for $N = 7$.

Solution: We should have about $k = 50/7 \approx 7$ channels per group. For group 1, use forward channels $\{1, 8, 15, 22, 29, 36, 43, 50\}$. For group $i$, $i = 2, \ldots, 7$, let the channels for group $i$ consist of $\{i, i + 7, i + 14, i + 21, i + 28, i + 35, i + 42\}$.

Because of the spacing in channels used in one group, the specifications for transmit and receive filters will be less strict, and can be more effective at reducing adjacent channel interference.

There is still the near-far effect. If a TX near the BS is producing just a little bit of out-of-band noise, it might swamp out the desired signal transmitted by a TX far away to the same BS.

One solution is power control, i.e., reducing the power level TXed by mobiles close to the BS, since a high TX power is not necessary in that case. This reduces their out-of-band noise as well. This means that antennas far away must transmit larger power than those nearby. Compared to a mobile transmitting full power all the time, power control extends battery life, and generally reduces even co-channel interference on the reverse channel. However, power control requires, well, control. Controlling a mobile means communication from the BS to the mobile to inform it whether to increase or decrease its power, which then requires data overhead.
Tight power control is particularly required in CDMA systems, in which the “near-far problem” is even more of a problem.

1 Trunking

Trunking refers to sharing few channels among many users. Let $U$ be the number of users, and $C$ be the number of channels. Each user requires a channel infrequently, so a dedicated channel for each user is not required. But, the request for a channel happens at random times, and so for any $C < U$, it is possible that there will be more requests than channels.

- **Erlang**: A “unit” of measure of usage or traffic intensity. One Erlang is the traffic intensity carried by one channel that is occupied all of the time. 0.1 Erlang is the same channel occupied only 10% of the time.

- **Average holding time**: Average call duration, denoted $H$.

- **Call rate**: Average number of calls per unit time, denoted $\lambda$. Typically taken to be at the busiest time of day.

- **Total offered traffic intensity**: The total amount of traffic users request of the system, denoted $A$.

- **Grade of Service** (GOS): The probability an offered call will be blocked (and thus not served, or carried by the system).

Rappaport presents that an average user will request (offer) this much traffic, $A_u = \lambda H$. For example, if a user makes on average, two calls per hour, and that call lasts an average of 3 minutes, $A_u = \frac{2}{60 \text{ min}} \cdot 3 \text{ min} = 0.1$ Erlang. (Check your units!)

Then, to compute the total offered traffic intensity, and the total offered traffic intensity per channel (denoted $A_c$),

$$A = U A_u, \quad A_c = A/C$$

For the above example, assume that there are 1000 users and 200 channels. Then $A = 1000(0.1) = 100$, and $A_c = 100/200 = 0.5$.

Note that $A_c$ is a measure of the efficiency of the utilization of the channels.

**How should we design our system?** Obviously, $A_c$ should be less than one ($A < C$); or we’ll never satisfy our call demand. But how should we set $U$, $A_u$, $C$ to satisfy our customers?

First choice: what do we do when a call is offered (requested) but all channels are full?

- **Blocked calls cleared**: Ignore it!

- **Blocked calls delayed**: Postpone it!
1.1 Blocked calls cleared

1. Call requests are a Poisson process. That is, the times between calls are exponentially distributed, and memoryless.

2. Call durations are also exponentially distributed.

3. Rather than a finite number $U$ of users each requesting $A_u$ traffic, we set the total offered traffic as a constant $A$, and then let $U \to \infty$ and $A_u \to 0$ in a way that preserves $UA_u = A$. This is the “infinite number of users” assumption that simplifies things considerably.

These assumptions, along with the blocked calls cleared setup of the system, leads to the Erlang B formula:

$$GOS = P[\text{blocking}] = \frac{A^C/C!}{\sum_{k=0}^{C} A^k/k!}$$

Since $C$ is very high, it’s typically easier to use Figure 3.6 on page 81. By setting the desired GOS, we can derive what number of channels we need; or the maximum number of users we can support (remember $A = UA_u$); or the maximum $A_u$ we can support (and set the number of minutes on our calling plans accordingly).

1.2 Blocked calls delayed

Instead of clearing a call; put it in a queue (a first-in, first-out line). Have it wait its turn for a channel. (“Calls will be processed in the order received”). There are now two things to determine

1. The probability a call will be delayed (enter the queue), and

2. The probability that the delay will be longer than $t$ seconds.

The first is no longer the same as in (1); it goes up, because blocked calls aren’t cleared, they “stick around” and wait for the first open channel.

Here, we clarify the meaning of GOS for a blocked calls delayed system. Here it means the probability that a call will be forced into the queue AND it will wait longer than $t$ seconds before being served (for some given $t$).

We need a couple additional assumptions:

1. The queue is infinitely long. In a computer system, this translates to infinite memory.

2. No one who is queued gives up / hangs up (rather than wait).
With these assumptions, we can derive the Erlang C formula, for the probability that a call will be delayed:

$$P[\text{delay} > 0] = \frac{AC}{AC + C! (1 - A/C) \sum_{k=0}^{C-1} \frac{A^k}{k!}}$$  \hspace{1cm} (2)

It is typically easiest to find a result from Figure 3.7, on page 82. Once it enters the queue, the probability that the delay is greater than \( t \) (for \( t > 0 \)) is given as

$$P[\text{delay} > t | \text{delay} > 0] = \exp \left( -\frac{C - A}{H} t \right)$$  \hspace{1cm} (3)

The two combined are needed to find the marginal (overall) probability that a call will be delayed AND experience a delay greater than \( t \), the event that we are quantifying in GOS.

$$GOS = P[\text{delay} > t] = P[\text{delay} > t | \text{delay} > 0] P[\text{delay} > 0] = P[\text{delay} > 0] \exp \left( -\frac{C - A}{H} t \right)$$  \hspace{1cm} (4)

**Example: \( N = 7 \) cell cluster**

A 7 cell cluster (with \( N = 7 \)) has 28 MHz allocated to it for forward channels and each channel is 200 kHz. Assume blocked-called-delayed and a probability of delay of 1%, and each user makes one 10 minute call every 3 hours. (a) What is the number of users that can be supported? (b) What is \( P[\text{delay} > 10] \) seconds? (c) What if it was a blocked-calls-cleared system with QOS of 1%?

**Solution:**

- **(a)** With 20 channels, and probability of delay of 1%, looking at figure 3.7, we see \( A \) is about 11. With \( 11 = A_u U \) and \( A_u = 10/(3 \times 60) = 1/18 \), we have that \( U = 102 \approx 200 \). But this is per cell. So there can be \( 7(200) = 1400 \) users in the 7 cells. (b) In each cell, \( C = 20, \ A = 11, \ H = 600 \) sec. So \( P[\text{delay} > t] = (0.01) \exp[-(20 - 11)(10)/600] = 0.01 \exp(-0.15) = 0.0086 \). (c) From Figure 3.6, \( A \approx 12 \), so \( U = 216 \), for a total of 1512 total users.

**1.3 Discussion**

What are the problems or benefits we see from the assumptions we’ve made? Are call requests “memoryless”? Is the exponential interarrival time assumption accurate? When catastrophic events occur, or major news breaks, what happens? How should a communications system be designed to handle these cases?