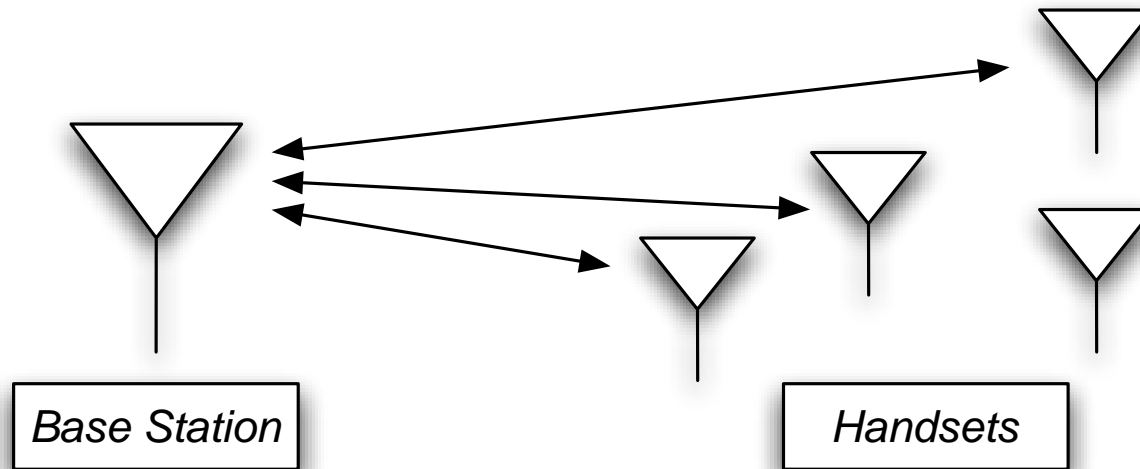


Signal Processing and Linear Systems I

Applications 2: Introduction to Spread Spectrum Communications

January 29, 2015

Cell Telephone



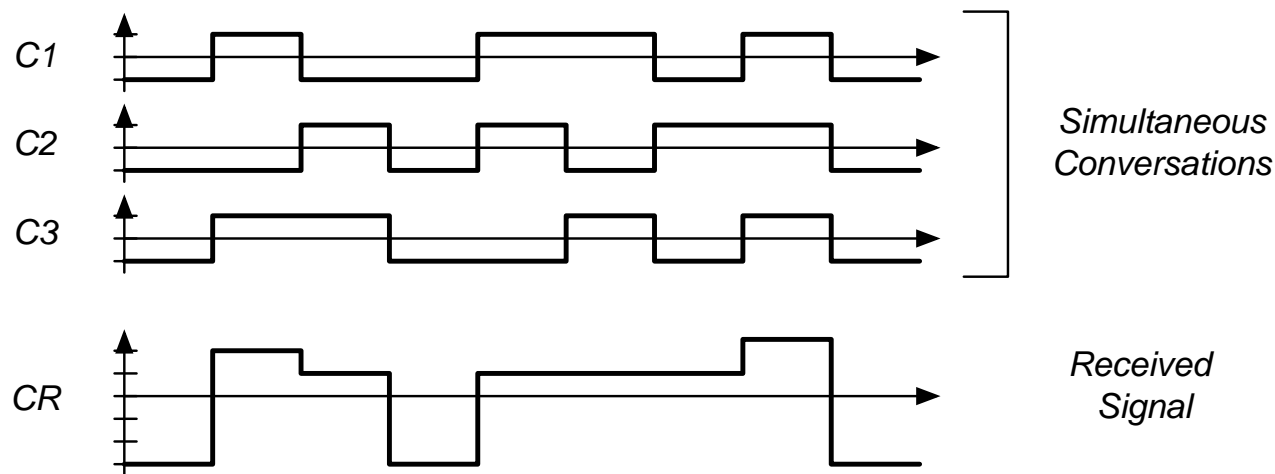
Base station talks to all of the handsets in the cell.

Simple solution, each link has a frequency. However, there are a limited number of frequencies.

We would like to have multiple connections for each frequency. How do we do this?

Sharing a Channel

If all of the handsets try to use the same frequency, the signals superimpose, and we can't sort them out.



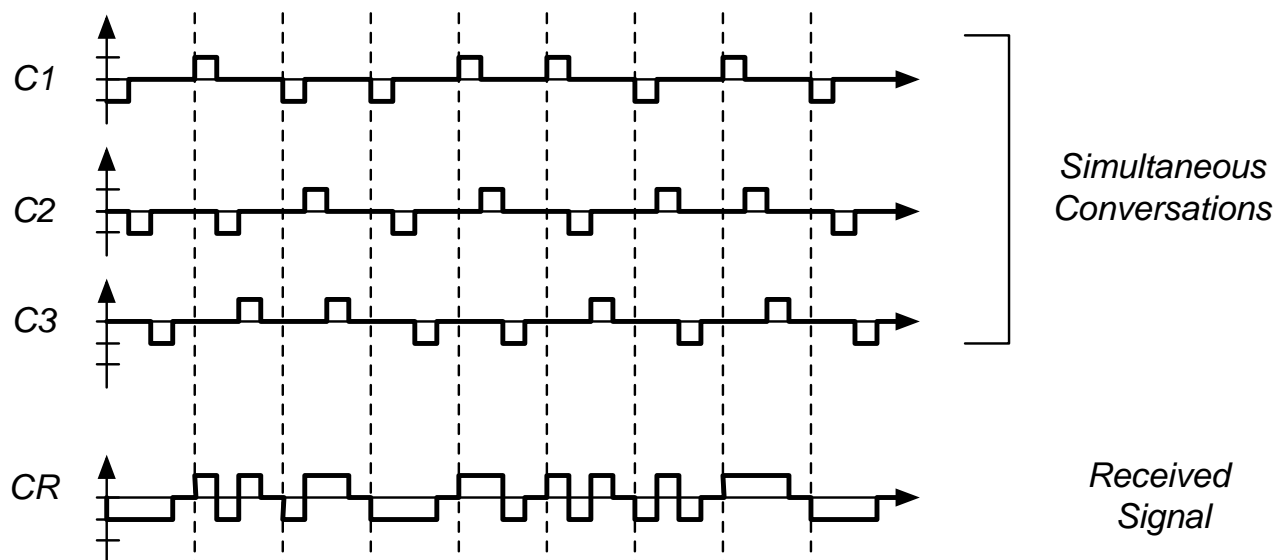
Two methods (out of many) for sharing the channel are

- Time division multiplexing (TDMA)
- Code division multiplexing (CDMA)

Time Division Multiplexing

Each transmit bit is divided into several subintervals.

Each handset only talks during its allotted times.



Here there are four time slots. C1 only transmits during the first time slot, C2 the second time slot, etc.

There are a limited number of time slots, limiting the number of simultaneous users. We could add one more user, but the second would fail.

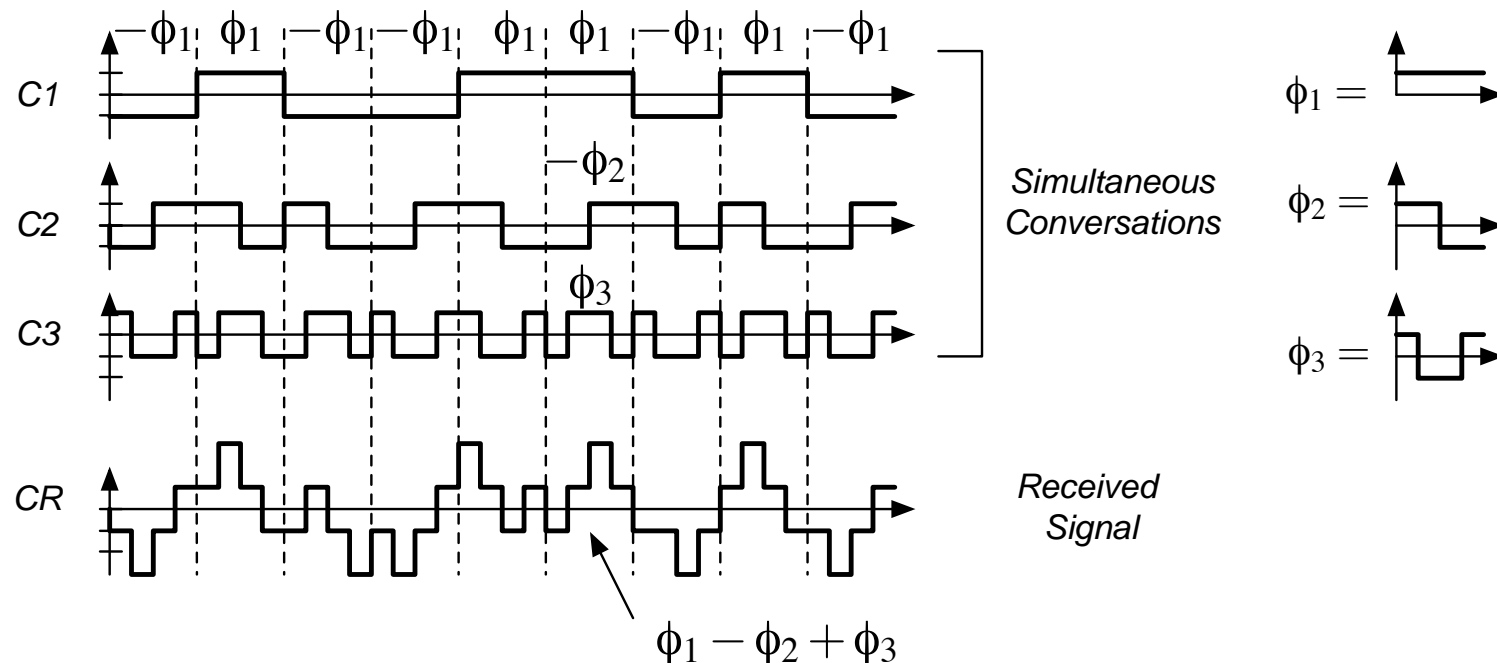
The received signal bandwidth is increased by a factor of four in this case (the spectrum is spread by that factor).

Higher data rates can be obtained by allocating one user several time slots.

Identifying Individual Channels

What we'd like is a "tag" that allows each channel to be selected, while ignoring all of the others.

If $\phi_1(t)$ is the tag, or code for one channel, we send a sequence of $(-\phi_1(t), +\phi_1(t), +\phi_1(t), \dots)$ to represent the sequence $(-1, +1, +1, \dots)$



The received signal at one interval contains these superimposed codes from each channel, weighted by channel gain, transmit power, etc

$$y(t + kT) = \sum_n D_{n,k} \phi_n(t)$$

$D_{n,k}$ is the information from the n^{th} channel at the k^{th} sample.

How do we choose the code waveforms to make it easy to find the signal of interest, and ignore all the others?

Orthogonal Codes

What we want is to choose the $\phi_n(t)$ so that we can extract $D_{n,k}$ (the information we are trying to receive) by a simple matched filter,

$$\begin{aligned}\tilde{D}_{n,k} &= \frac{1}{T} \int_0^T \phi_n(t) y(t + kT) dt \\ &= \frac{1}{T} \int_0^T \phi_n(t) \sum_m D_{m,k} \phi_m(t) dt \\ &= \sum_m D_{m,k} \frac{1}{T} \int_0^T \phi_n(t) \phi_m(t) dt\end{aligned}$$

If we choose

$$\frac{1}{T} \int_0^T \phi_n(t) \phi_m(t) dt = \begin{cases} 1 & \text{if } n=m \\ 0 & \text{otherwise} \end{cases}$$

Then we only get the term we get is the term we want,

$$\tilde{D}_{n,k} = D_{n,k}$$

and we ignore all of the other channels!

We want to choose $\phi_n(t)$ to be *orthogonal* signals.

There are many possible choices. For example

$$\phi_n(t) = \cos(n\omega_0 t)$$

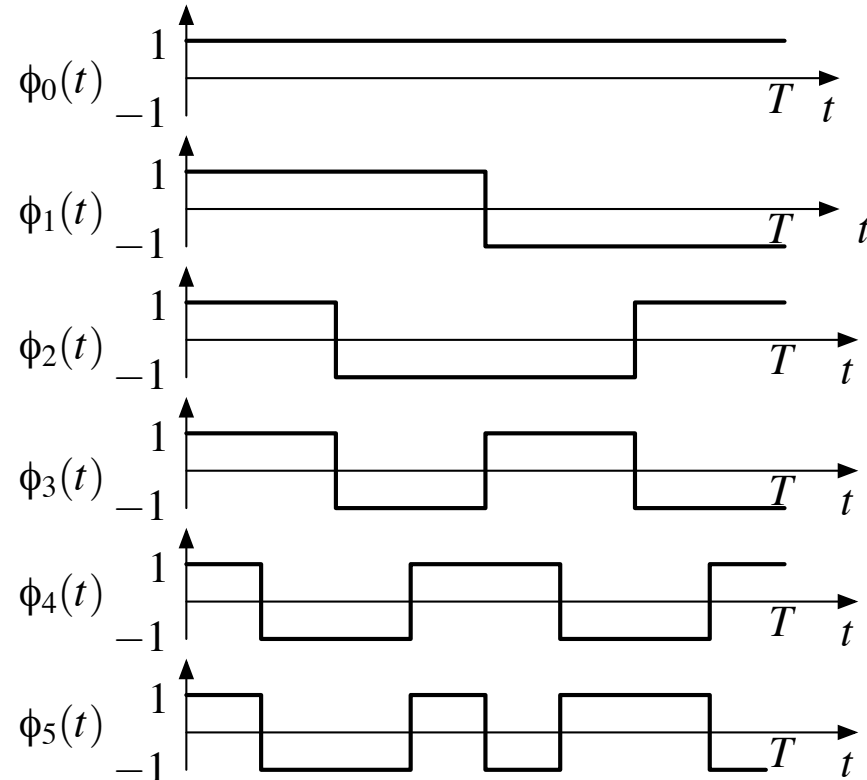
where $\omega_0 = \frac{2\pi}{T}$. The $D_{n,k}$ are then the coefficients of the cosine Fourier series of the signal. This effectively shifts each channel to a different frequency!

However, what we *really* want is something very easy to compute ...

Hadamard Waveforms

Hadamard waveforms are an orthogonal set made up of only ± 1 's.

The first couple of Hadamard signals are (in sequency order):



The signs of the Hadamard signals can be generated (in a different order) by defining

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

and then defining

$$H_4 = \begin{pmatrix} H_2 & H_2 \\ H_2 & -H_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}$$

and similarly for H_8, H_{16}, \dots

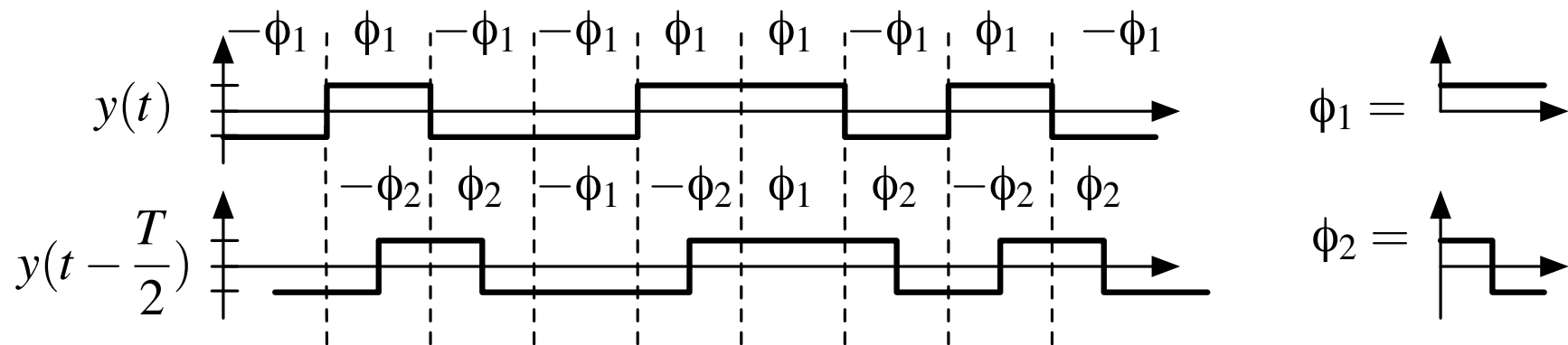
It is easy to show that the columns of H_N are orthogonal, and hence that the Hadamard functions with these signs are orthogonal (try this!).

We can tag each channel by assigning it a Hadamard waveform $\phi_n(t)$.

We can continue to add users by adding new codes (at some point we run out of bandwidth, though).

Synchronous Detection

Hadamard decoding fails if it is not synchronized

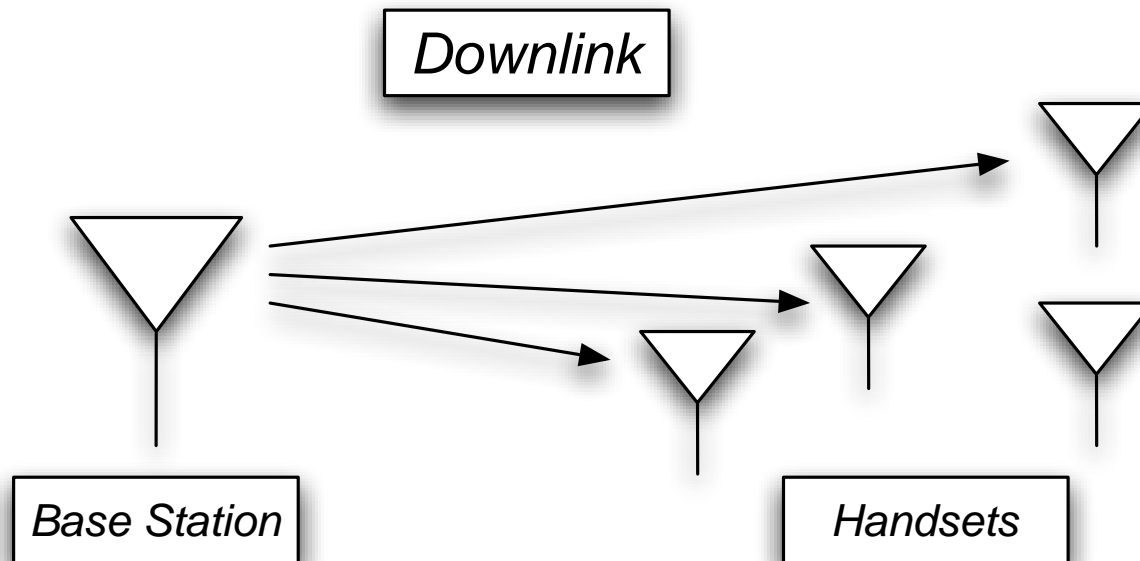


Without a delay, the signal decodes to channel 1.

With a delay, the decoding is completely different.

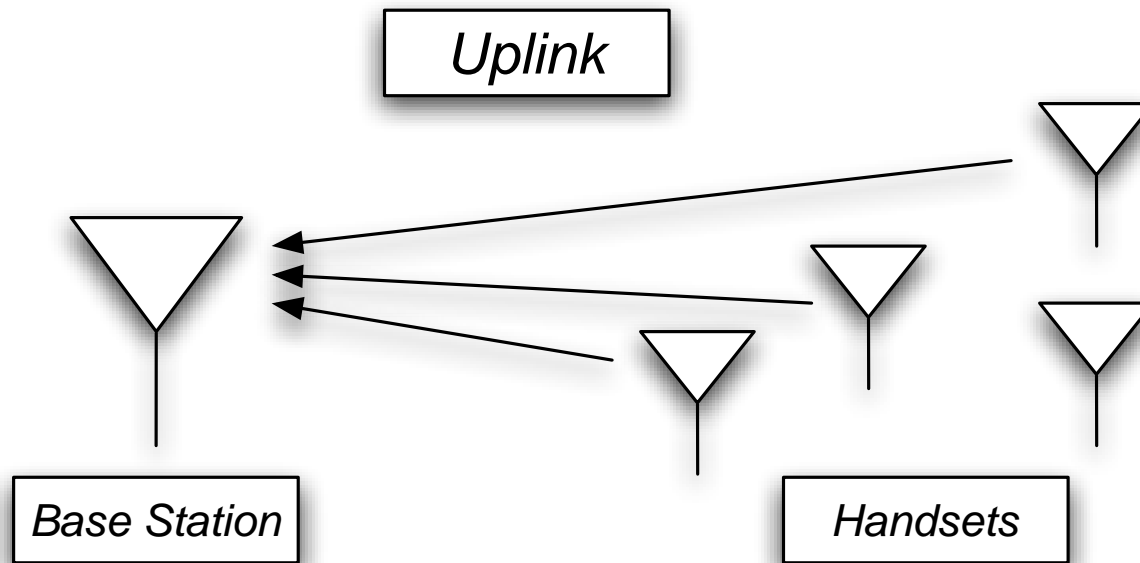
The Hadamard codes only work if all of the signals are synchronized.

Fine when the basestation is talking to the handsets (the downlink). The basestation can synchronize all the signals before transmitting.



Does not work well when the handsets are talking to the basestation (the uplink).

Each channel has it's own delay:



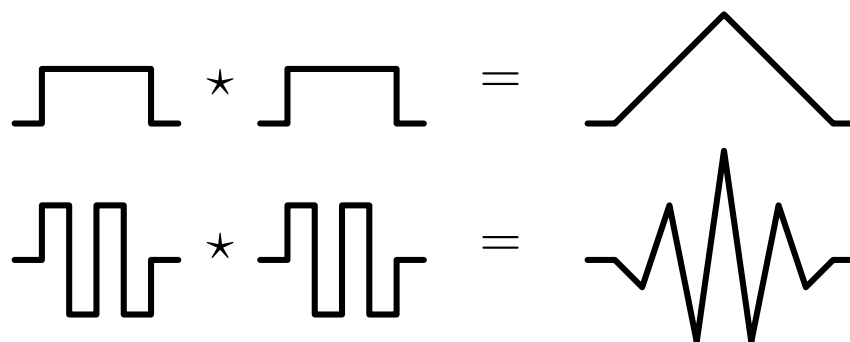
Even if we could synchronize to one channel, we would be decoding the other channels with delays, and these interfere with our channel.

Uplink Codes

Ideally, we would like a code that

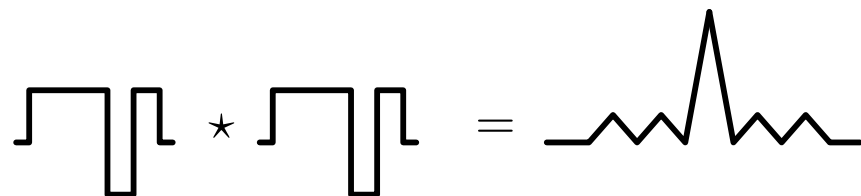
- Is orthogonal, so that we can isolate a particular channel
- Has a sharp autocorrelation, so that we can accurately identify the delay of a channel, and decode it properly.

Unfortunately, Hadamard codes have broad autocorrelations:



Not good for estimating delays!

We would like autocorrelations like the Barker codes from last week,

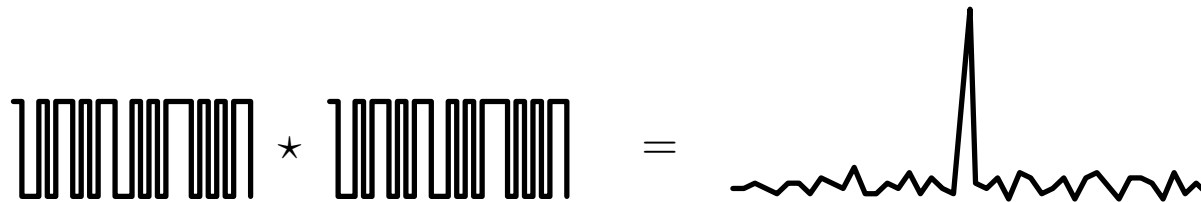

$$\text{Barker Code} \star \text{Barker Code} = \text{Autocorrelation}$$

However, there are no orthogonal sets of signals with the autocorrelation properties of Barker codes.

Pseudo-Noise Codes

Pseudo-noise codes approximate the ideal sharp auto-correlation, orthogonal codes. These are deterministic codes that approximate the characteristics of a noise sequence.

They are ± 1 with probability of $1/2$, and have the run lengths of -1 's and 1 's of a random sequence.



At zero shift, the product of the two is a constant "1", and the autocorrelation is "N". At other shifts, the product of each interval is just as likely to be $+1$ as -1 , so the autocorrelation is small.

They are not completely orthogonal, although

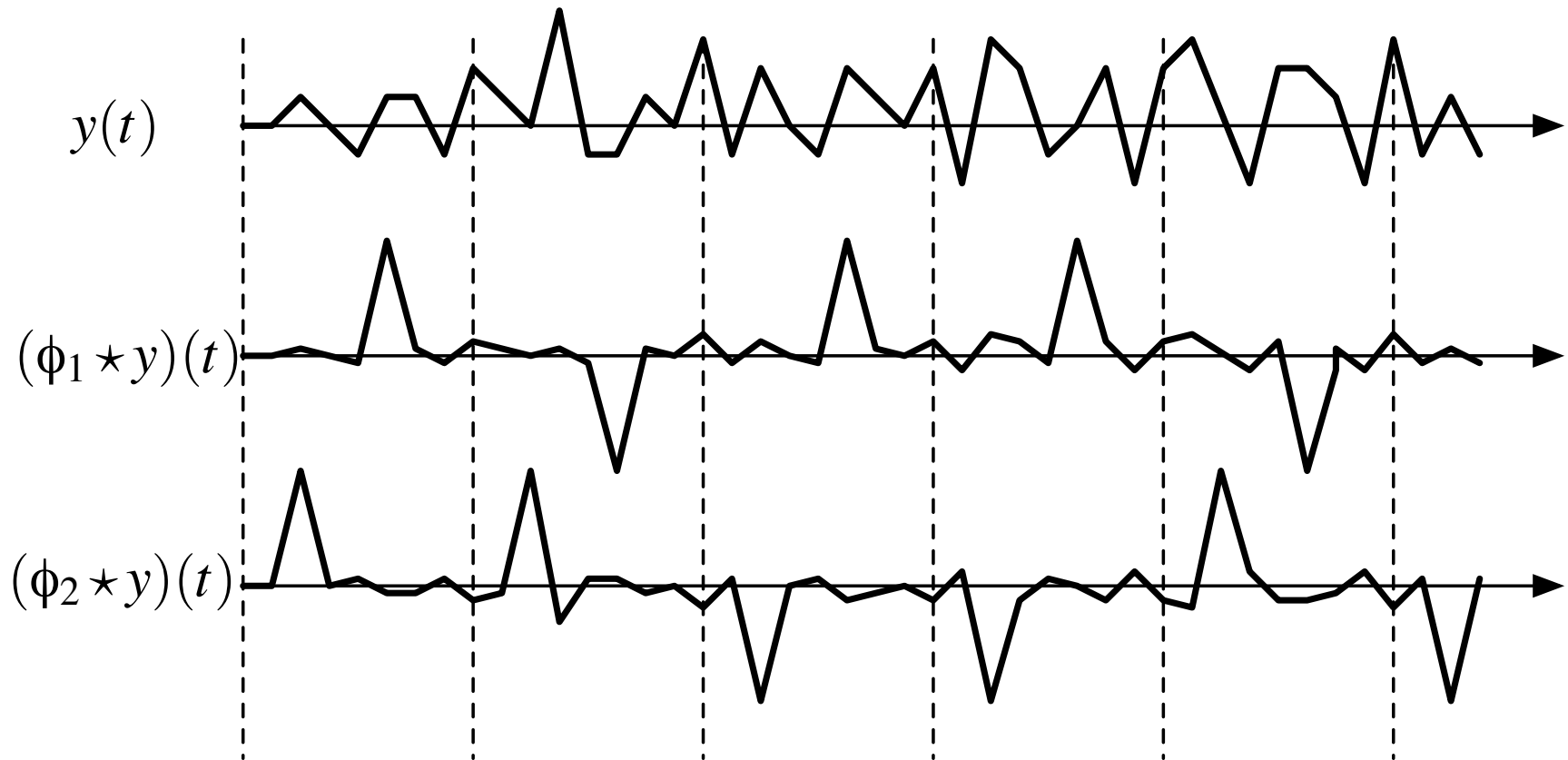
$$\frac{1}{T} \int_0^T \phi_n(t) \phi_k(t) dt$$

should be small (again, ± 1 equally likely, so the integral will be small).

Other channels appear as noise-like interference.

If we correlate the received signal with $\phi_n(t)$ we get a large signal for the n^{th} channel, with a peak at the delay for the n^{th} channel.

Adding users looks like an increased background noise level, which softly degrades performance.



Each channel properly decoded, with its own delay.

Other channel just adds noise.

Summary

Orthogonal codes allow users to share a channel.

Hadamard codes work well when the channels can be synchronized, such as when the basestation is talking to multiple handsets.

For unsynchronized channels, we want both orthogonality, and good autocorrelations. Psuedo-random codes are a good approximation.

We can continue to add users by handing out more codes, with a soft degradation of performance. Other users look like an increased noise level.