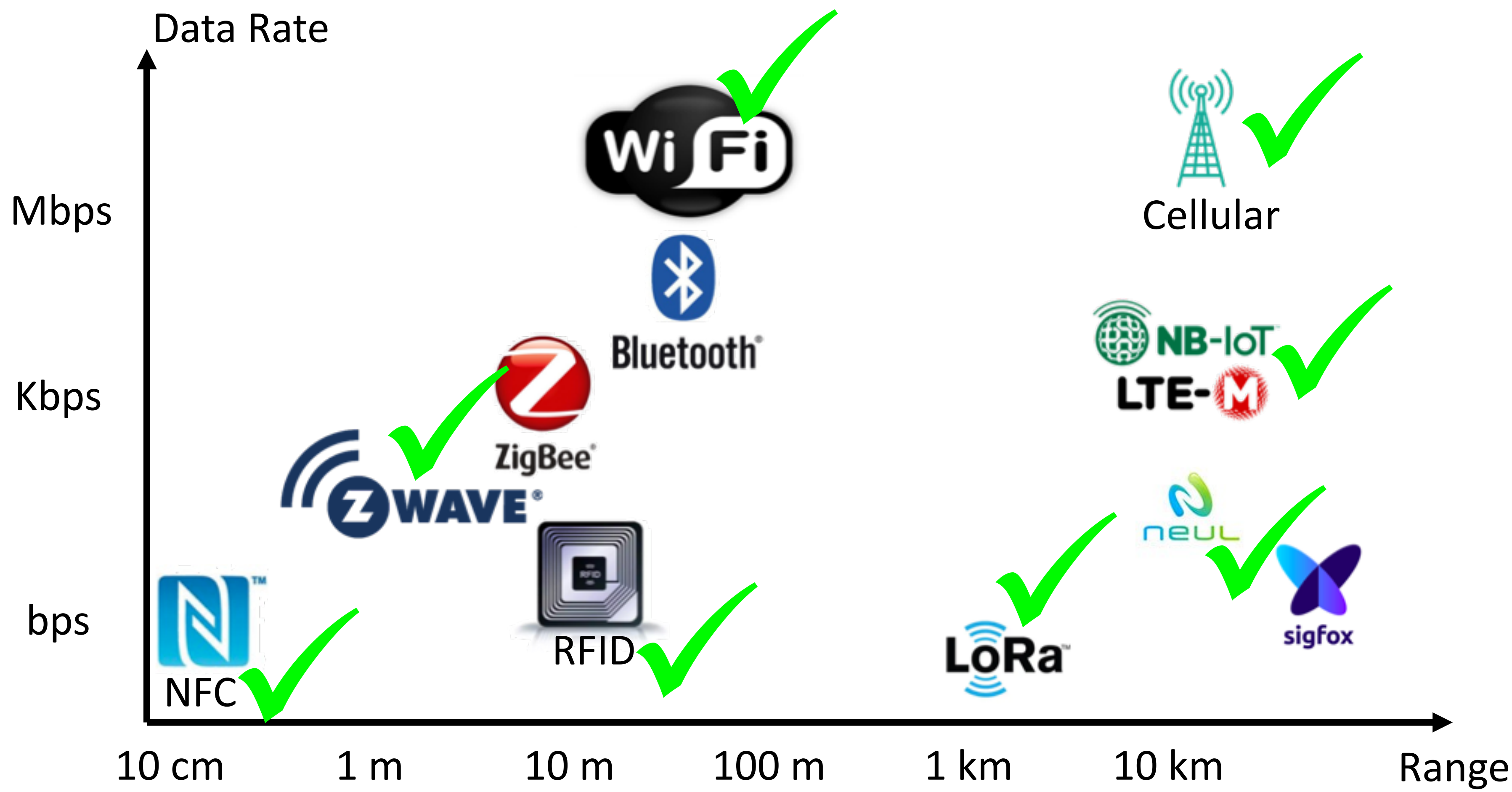
IoT Technologies



Bluetooth

Bluetooth®

Wearables

Smart Devices

Tracking

iBeacon

Bluetooth

Bluetooth

Bluetooth v4.0+: BLE
Bluetooth Low Energy

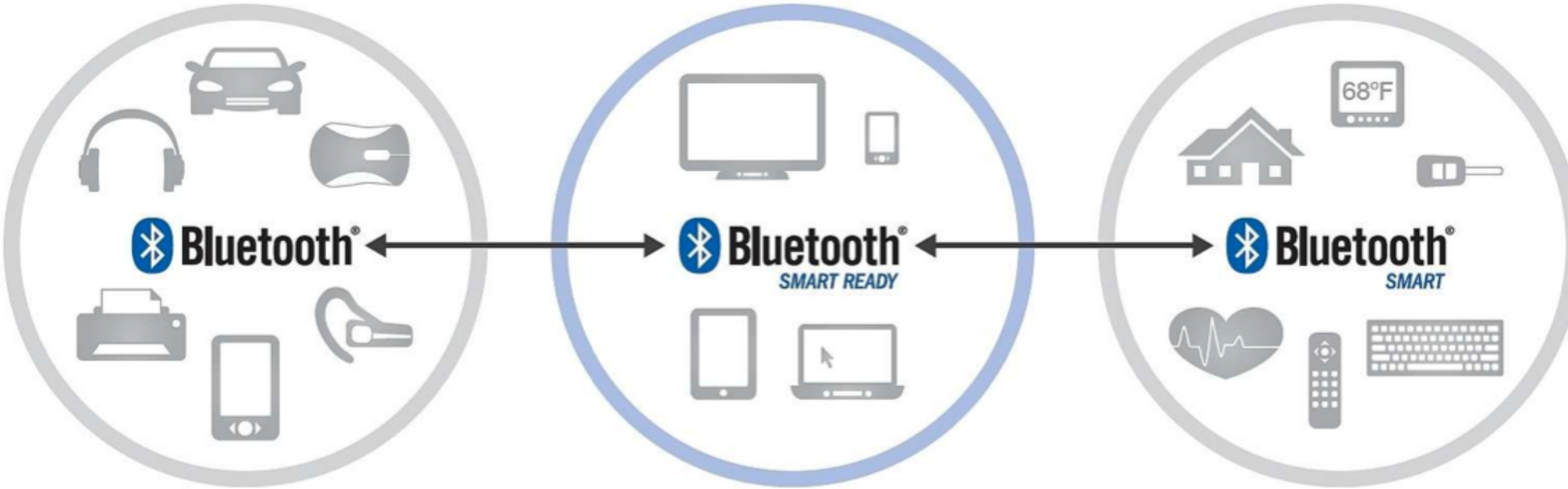
Bluetooth
Traditional wireless devices,
streaming rich content,
like video and audio

Bluetooth Smart Ready
Devices that connect with both
The center of your wireless world

Bluetooth Smart
Sensor devices,
sending small bits of data,
using very little energy

Bluetooth

Bluetooth v4.0+: BLE
Bluetooth Low Energy



Bluetooth
Traditional wireless devices,
streaming rich content,
like video and audio

Bluetooth Smart Ready
Devices that connect with both
The center of your wireless world

Bluetooth Smart
Sensor devices,
sending small bits of data,
using very little energy

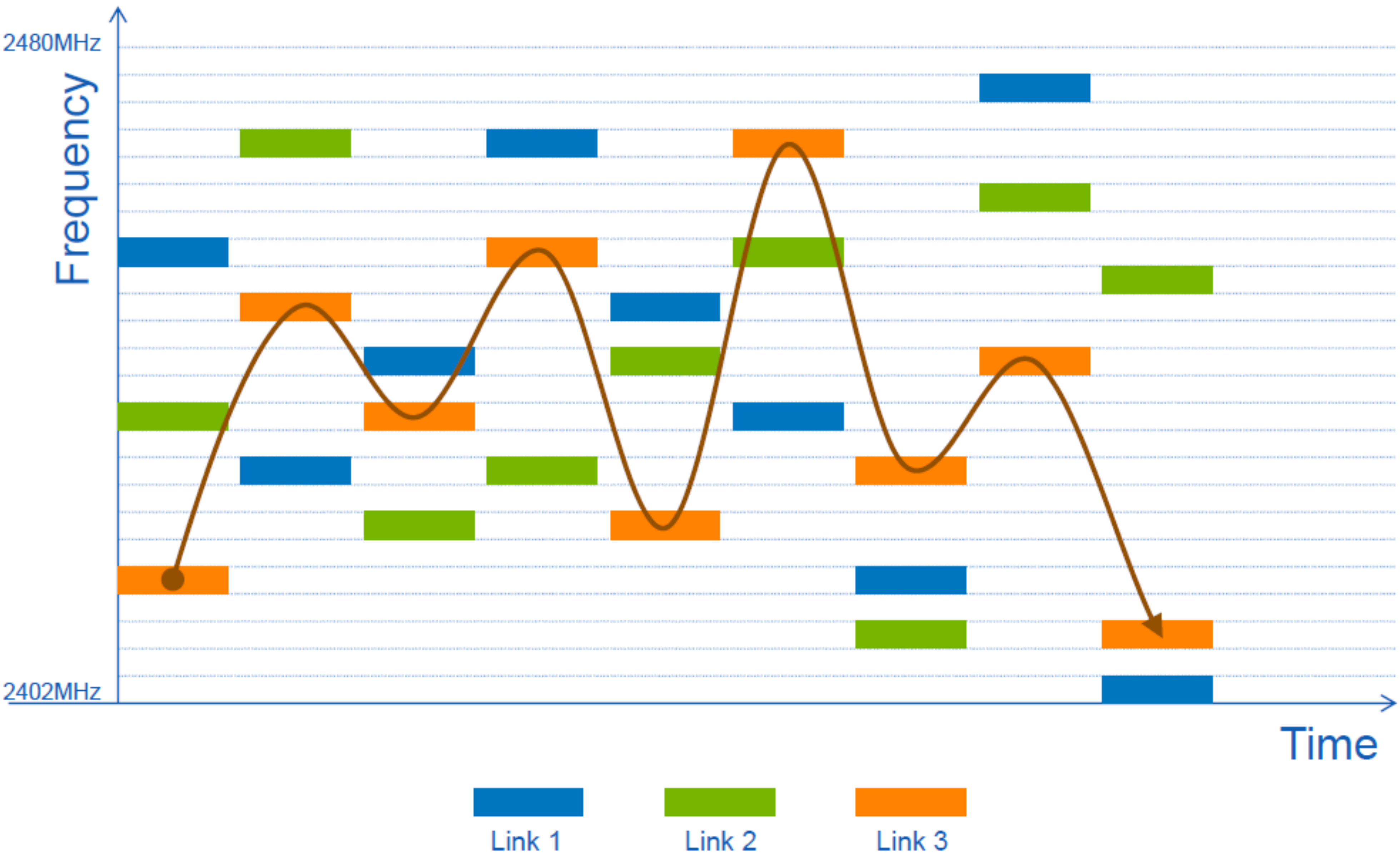
If your product bears this logo…	It's compatible with products bearing any of these logos…
Bluetooth SMART READY	Bluetooth SMART READY; Bluetooth; Bluetooth SMART
Bluetooth	Bluetooth SMART READY; Bluetooth
Bluetooth SMART	Bluetooth SMART READY

Bluetooth vs. Bluetooth Low Energy

	Classic Bluetooth	Bluetooth Low Energy
• Frequency:	2400 MHz – 2483.5 MHz	2400 MHz – 2483.5 MHz
• Bands:	79 channels (1 MHz sep.) 32 for device discovery	40 channels (2 MHz sep.) 3 for device discovery
• Connection Setup:	100 ms	$\approx$3ms
• Power:	1	20 to 100x lower power
• Range:	up to 150m	up to 50m
• Data Rate:	2-3 Mbps	200 Kbps – 1Mbps
• Modulation:	GFSK	GFSK
• FHSS:	1600 hops/sec 625μsec (dwell time) Pseudo Random Seq.	Longer dwell time max (400 msec)

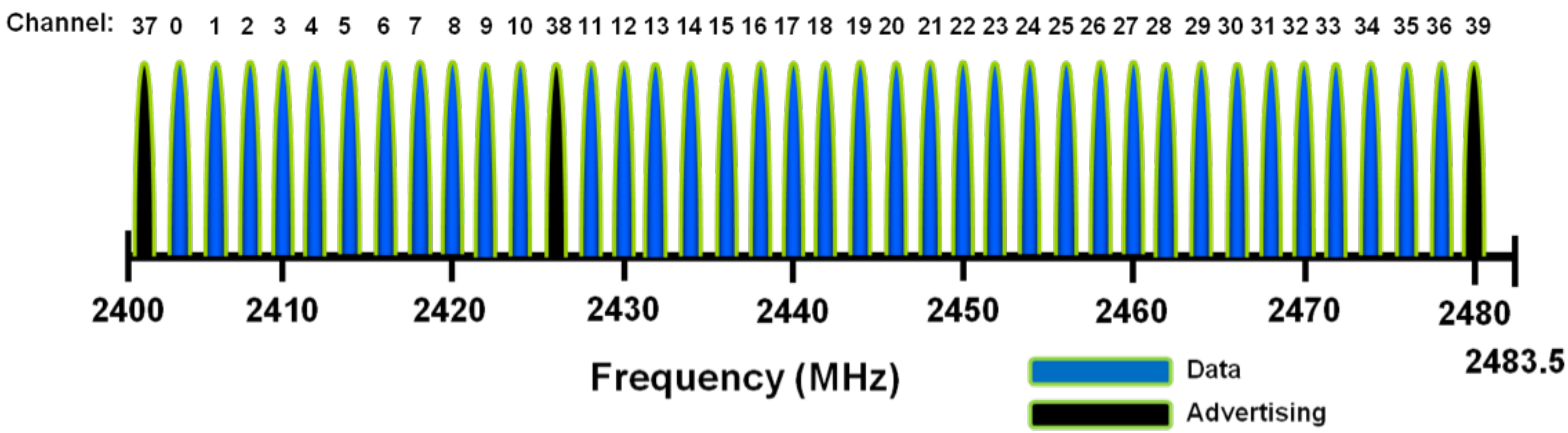
Bluetooth Low Energy Frequency Hopping

Frequency hopping: $f_{n+1} = (f_n + hop) \bmod 37$



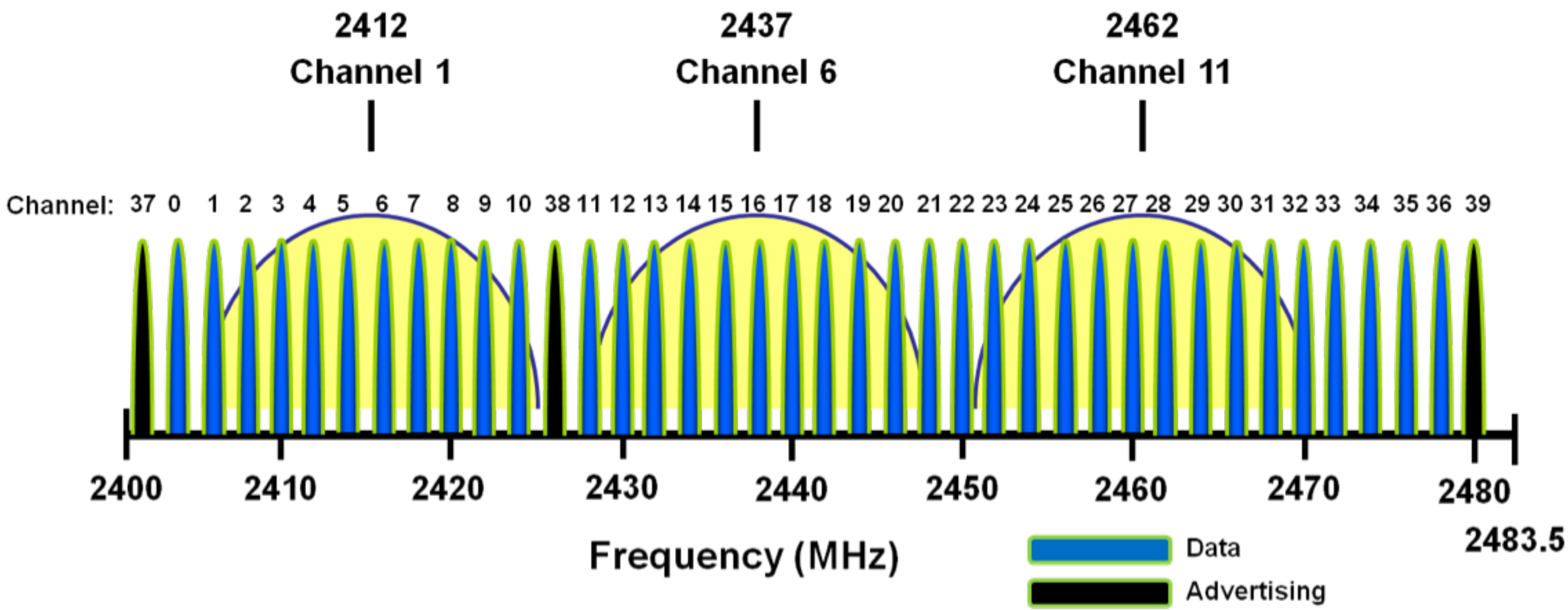
Bluetooth Low Energy Frequency Hopping

Frequency hopping: $f_{n+1} = (f_n + hop) \bmod 37$



Bluetooth Low Energy Frequency Hopping

Frequency hopping: $f_{n+1} = (f_n + hop) \bmod 37$



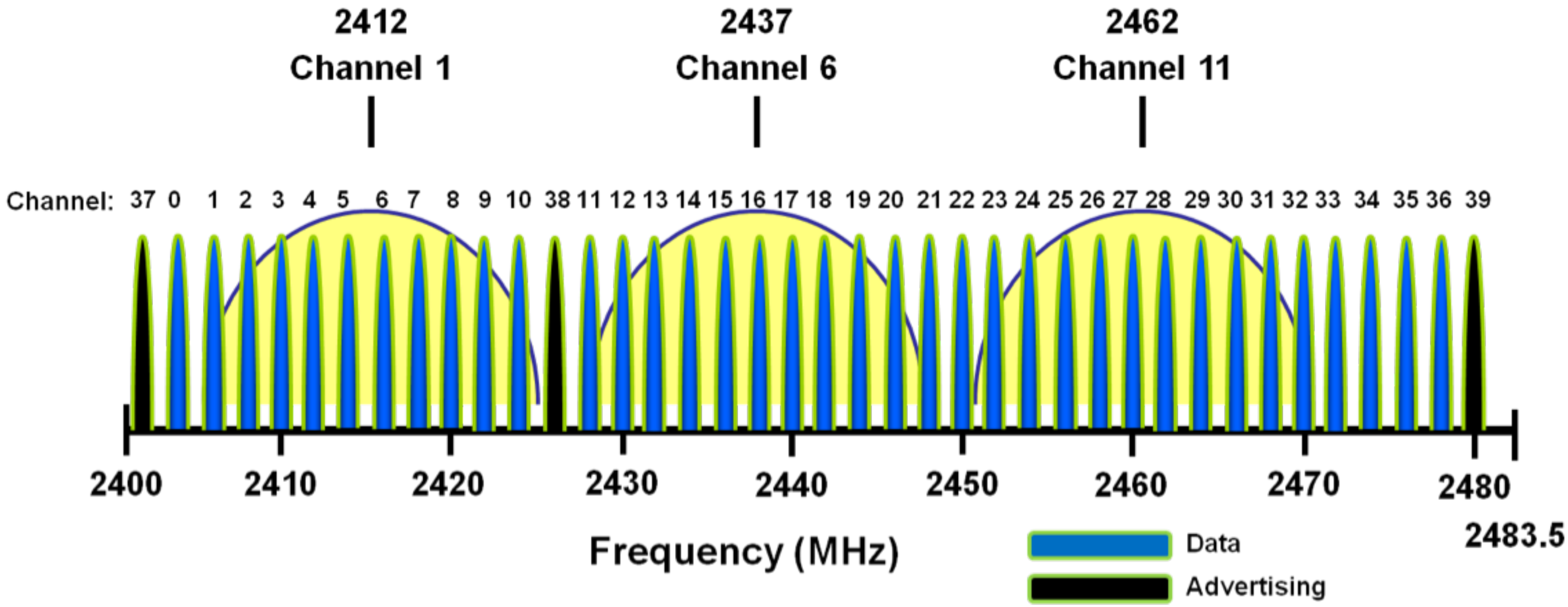
What about Interference from WiFi?

Use Adaptive Frequency Hopping!

Bluetooth Low Energy Frequency Hopping

Use Adaptive Frequency Hopping!

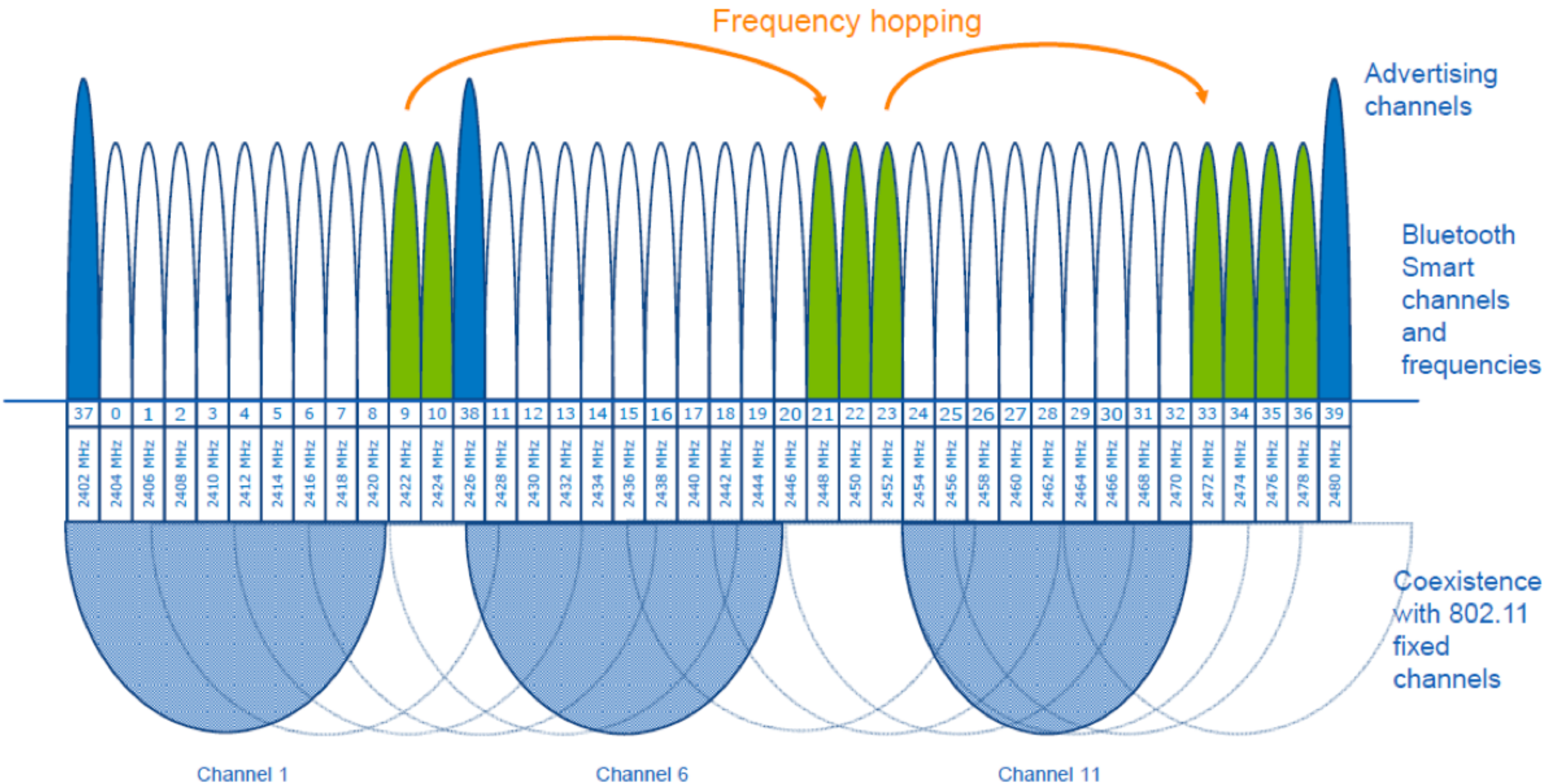
Avoid bad channels by remapping them to other channels.



Bluetooth Low Energy Frequency Hopping

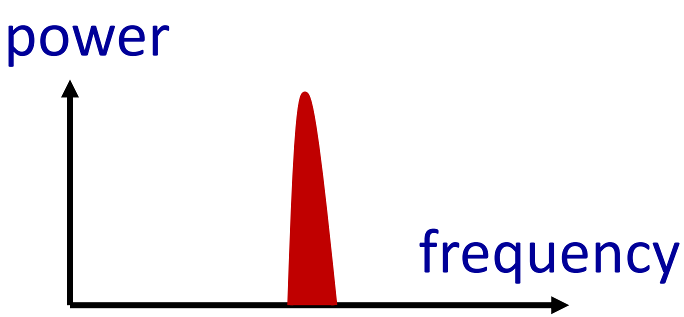
Use Adaptive Frequency Hopping!

Avoid bad channels by remapping them to other channels.



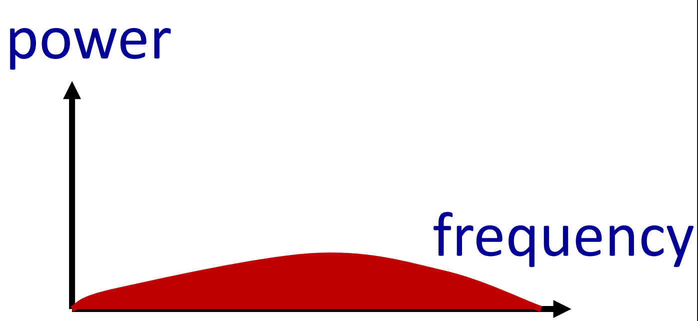
Spread Spectrum Technology

DSSS:
Direct Sequence Spread spectrum



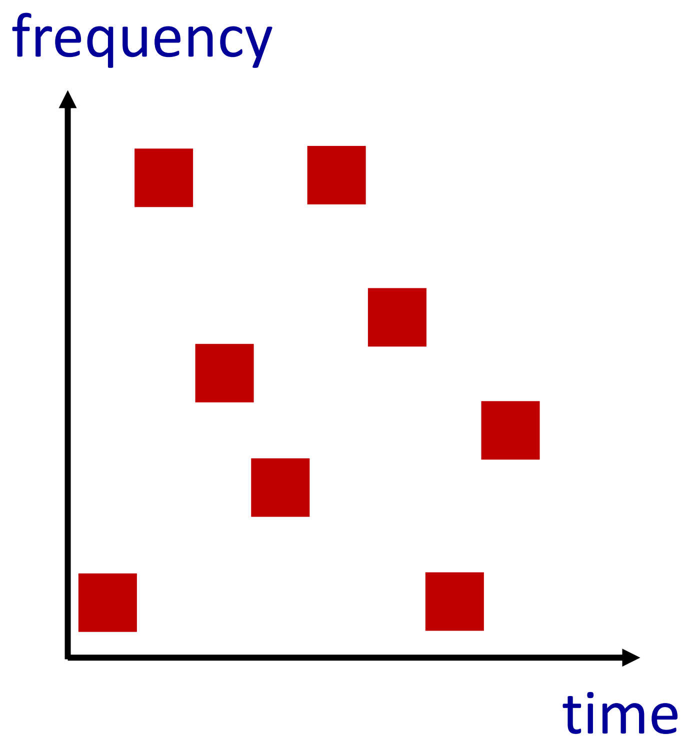
×

PN Sequence (Chip)



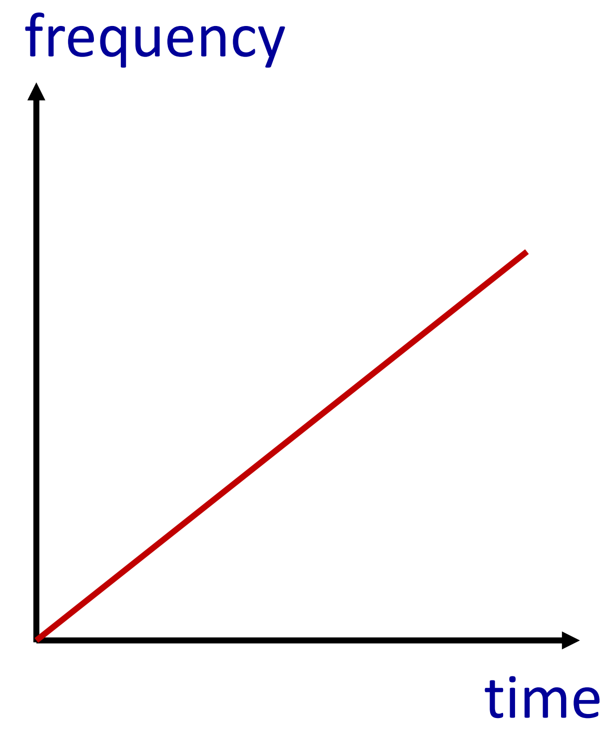
3G, 802.11b, GPS, Military

FHSS:
Frequency Hopping Spread spectrum



Bluetooth, Military

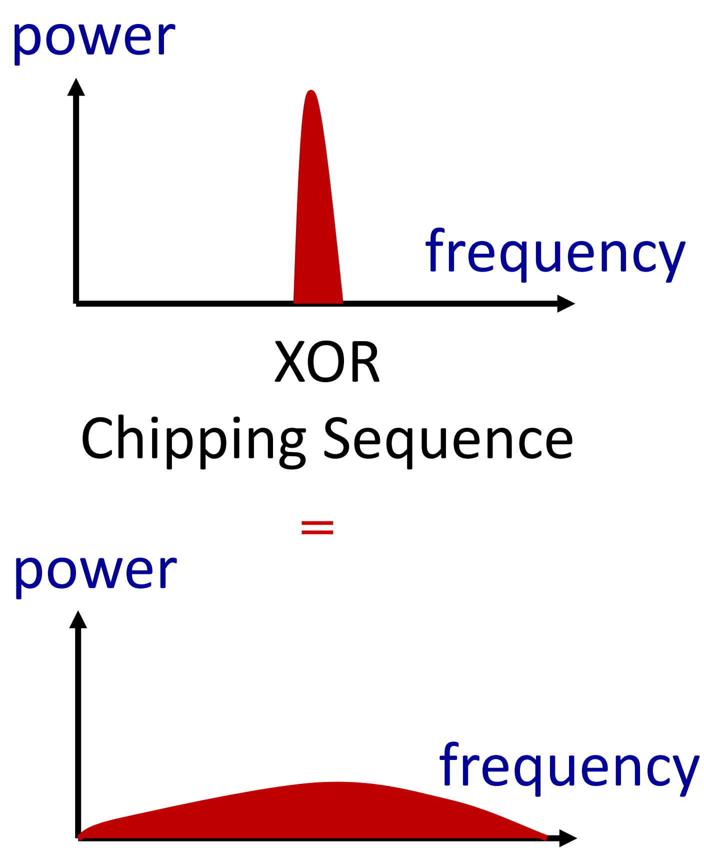
CSS:
Chirp Spread Spectrum

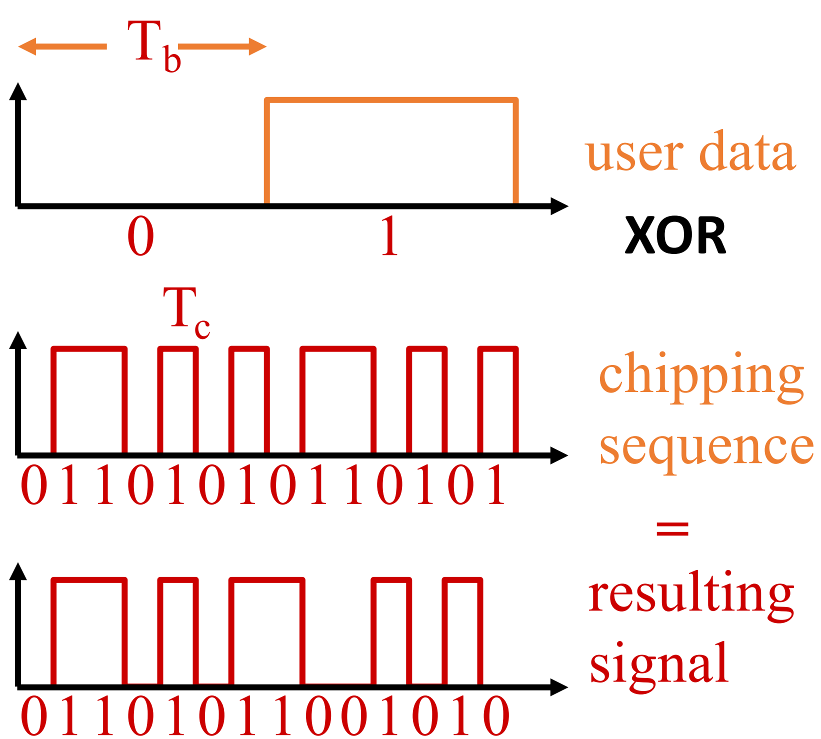


LPWAN, Radar

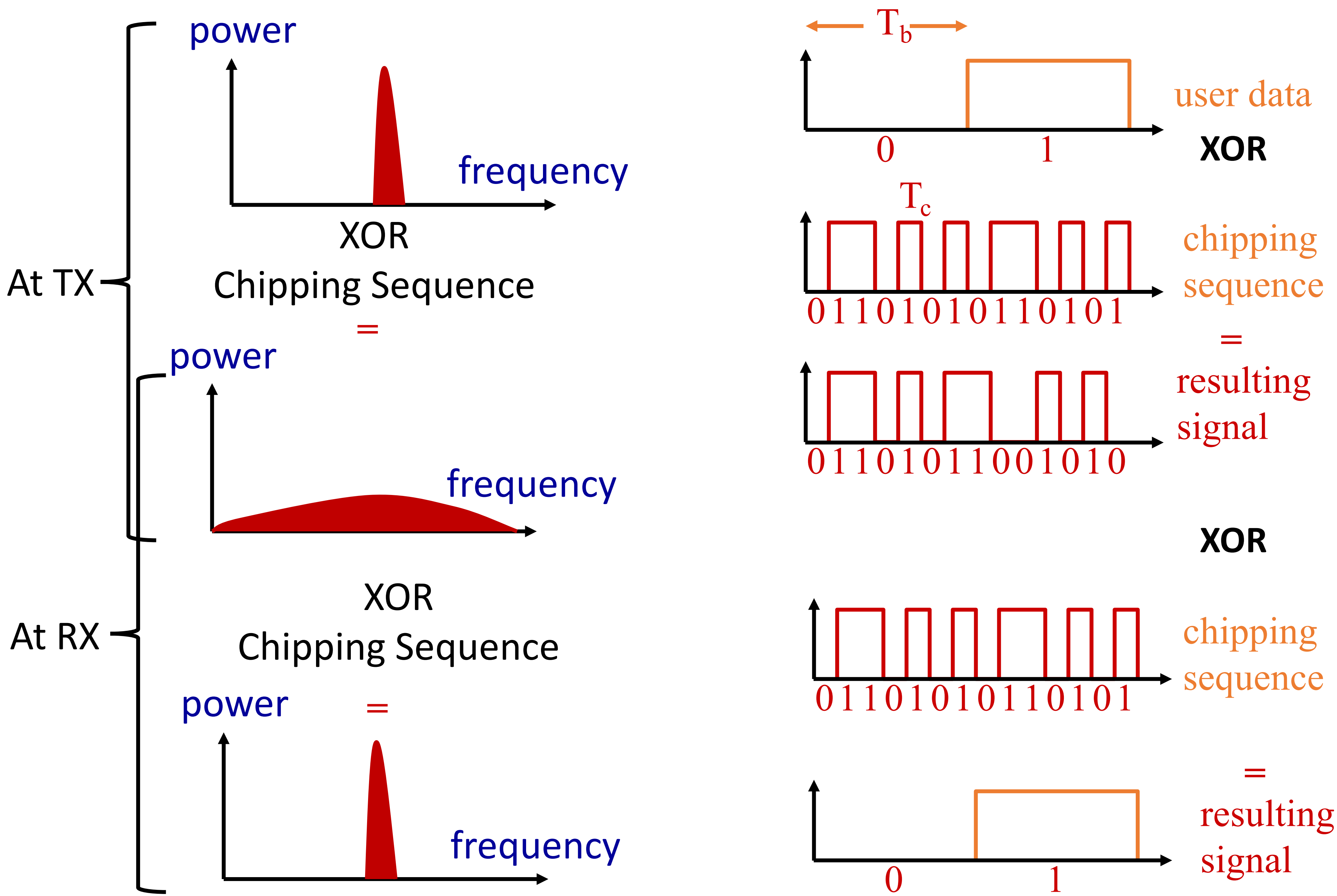
DSSS: Direct Sequence Spread Spectrum

- **Problem:** frequency dependent fading & interference can wipe out narrow band signals for duration of the interference
- **Solution:** spread the narrow band signal into a broad band signal using a special code
- XOR the signal with PN sequence (chipping sequence)

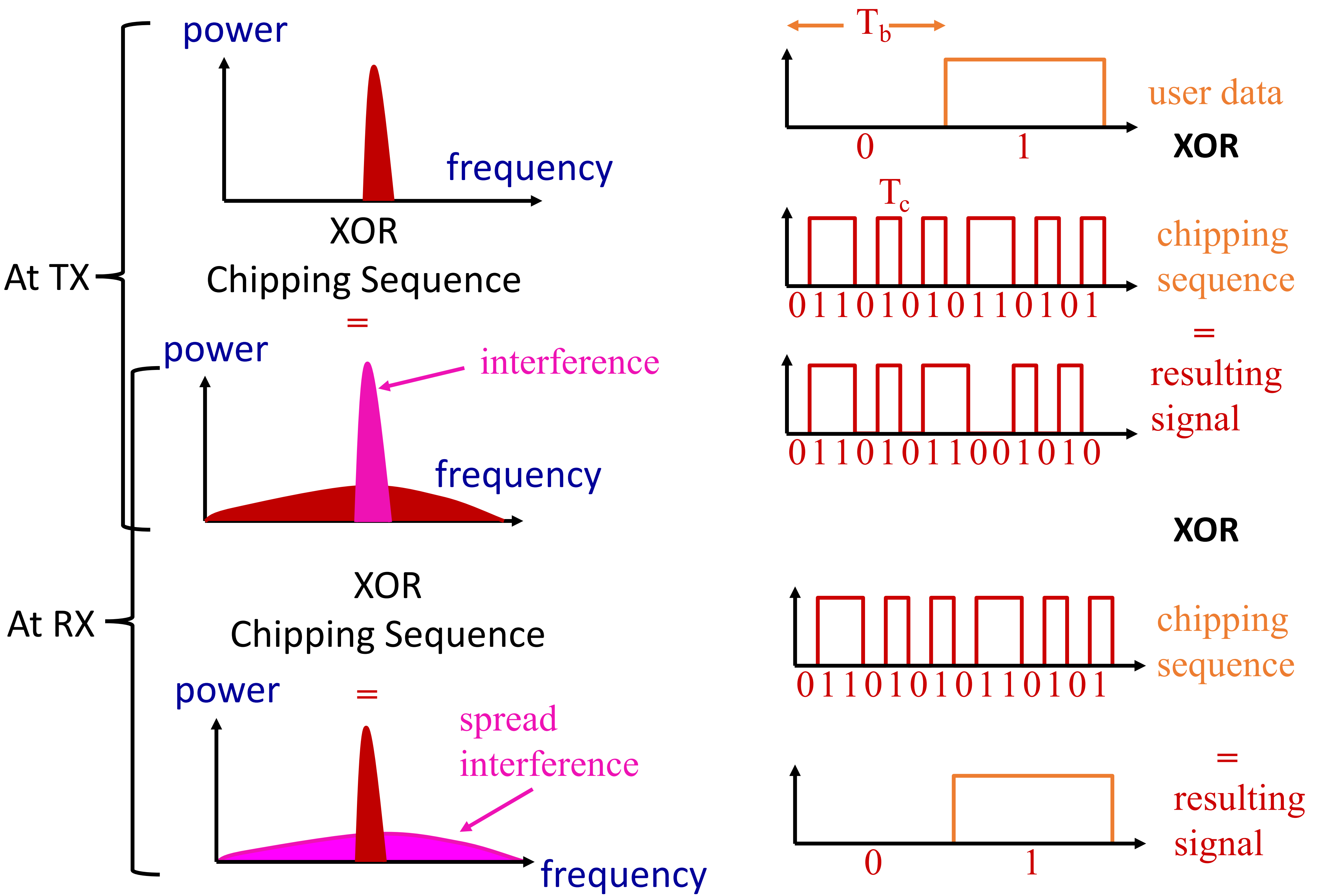




DSSS: Direct Sequence Spread Spectrum



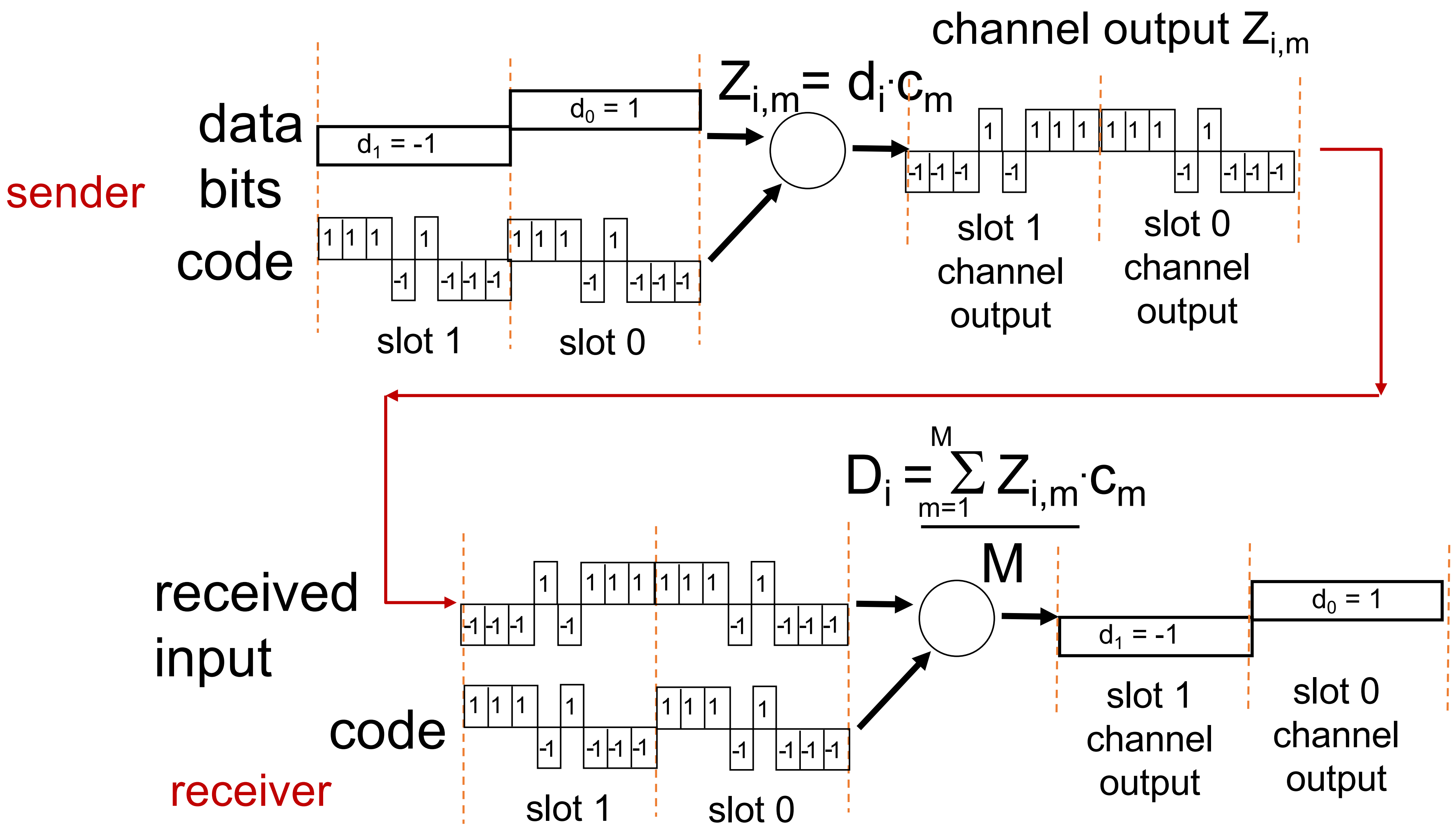
DSSS: Direct Sequence Spread Spectrum



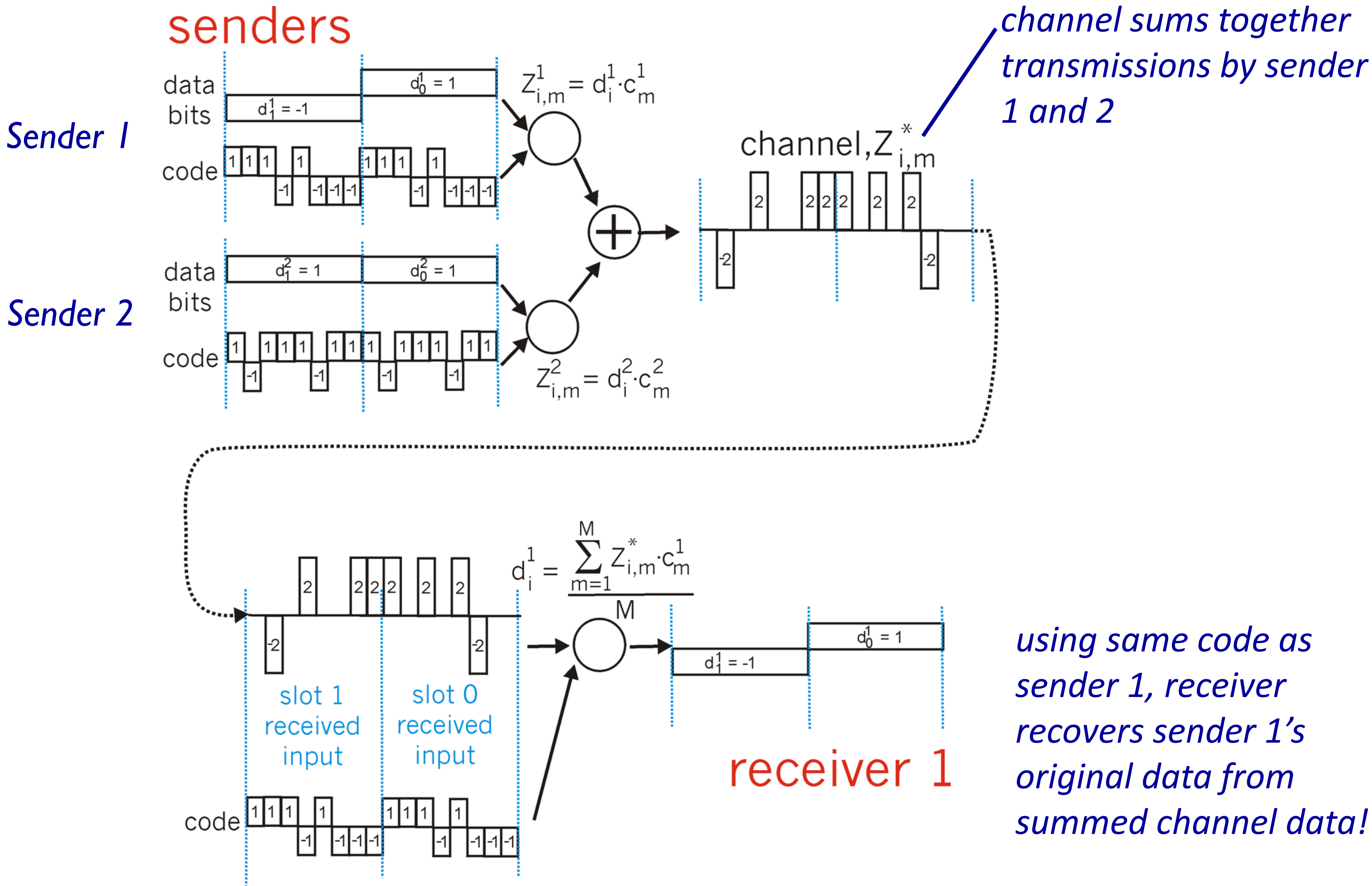
Code Division Multiple Access (CDMA)

- DSSS enables multiple users to transmit at the same time using CDMA
- unique "code" assigned to each user; i.e., code set partitioning
  - all users share same frequency, but each user has own "chipping" sequence (i.e., code) to encode data
  - allows multiple users to "coexist" and transmit simultaneously with minimal interference (if codes are "orthogonal")
- *encoded signal* = (original data) X (chipping sequence)
- *decoding:* inner-product of encoded signal and chipping sequence

CDMA encode/decode



CDMA: two-sender interference



Code Division Multiple Access (CDMA)

- Ideally, need codes to have good:

Auto-correlation properties: $c_i(t) \cdot c_i(t) = 1$

Cross-correlation properties: $c_i(t) \cdot c_j(t) = 0$ for $j \neq i$

$$\left(\sum_i h_i d_i(t)\, c_i(t) \right) \cdot c_i(t) = h_i d_i(t)$$

- Need orthogonal codes:

For N users, length of code is exponential in N → 2^{N-1}

- Example of good codes: Gold Codes, Walsh Codes

DSSS: Direct Sequence Spread Spectrum

DSSS enables decoding at very low SNR

- Transmit: $bit \times c(t)$

- Receiver: $h \times bit \times c(t) + n(t)$

- Decode: $\sum_{t=1}^{M} h \times bit \times c(t) \times c(t) + n(t) \times c(t)$

$$= M \times h \times bit + \sum_{t=1}^{M} n(t) \times c(t)$$

$$= M \times h \times bit + n'(t)$$

DSSS: Direct Sequence Spread Spectrum

DSSS enables decoding at very low SNR

$$n(t) \sim N(0, \sigma)$$

$$n'(t) \sim N\left(0, \sqrt{M}\sigma\right)$$

- Transmit: $bit \times c(t)$
- Receiver: $h \times bit \times c(t) + n(t)$
- Decode: $\sum_{t=1}^{M} h \times bit \times c(t) \times c(t) + n(t) \times c(t)$

$$= M \times h \times bit + \sum_{t=1}^{M} n(t) \times c(t)$$

$$= M \times h \times bit + n'(t)$$

DSSS: Direct Sequence Spread Spectrum

DSSS enables decoding at very low SNR

$$n(t) \sim N(0, \sigma)$$

$$n'(t) \sim N\left(0, \sqrt{M}\sigma\right)$$

- Transmit: $bit \times c(t)$
- Receiver: $h \times bit \times c(t) + n(t)$
- Decode: $= M \times h \times bit + n'(t)$

$$\text{SNR Before} = \frac{|h|^2}{\sigma^2} \qquad \text{SNR After} = \frac{|Mh|^2}{M\sigma^2} = M \times \frac{|h|^2}{\sigma^2}$$

SNR Increased By M times

DSSS: Direct Sequence Spread Spectrum

DSSS enables decoding at very low SNR

- GPS uses DSSS with code length $M = 1023$
- GPS uses BPSK: Can be decode well at $SNR > 6\text{ dB}$
- GPS signals can be decoded if received at SNR:

$$6\text{ dB} - 10\log_{10} M = -24\text{ dB}$$

- GPS signals come from satellites → typically received below the noise floor.

DSSS enables decoding signals buried below the noise floor

DSSS: Direct Sequence Spread Spectrum

DSSS enables decoding at very low SNR

- GPS uses DSSS with code length $M = 1023$

GPS receivers sometimes use a single bit ADC to sample the signal and yet can still decode correctly. How come?

Quantization SNR:

$6\ \mathrm{dB} \times$Quantization bits

$= 6\ \mathrm{dB}$

$\gg$

Thermal SNR:

$-23\ \mathrm{dB}$

Summary

IoT:

- LPWAN: LoRA (Lec. 11)
- Backscatter Communication: RFIDs, Miller code, full duplex (Lec. 12)
- Bluetooth (Lec. 13)
- Spread Spectrum: DSSS, FHSS, CSS (Lec. 11, 12, 13)