Computer Science & Engineering Department

Wifi in your system

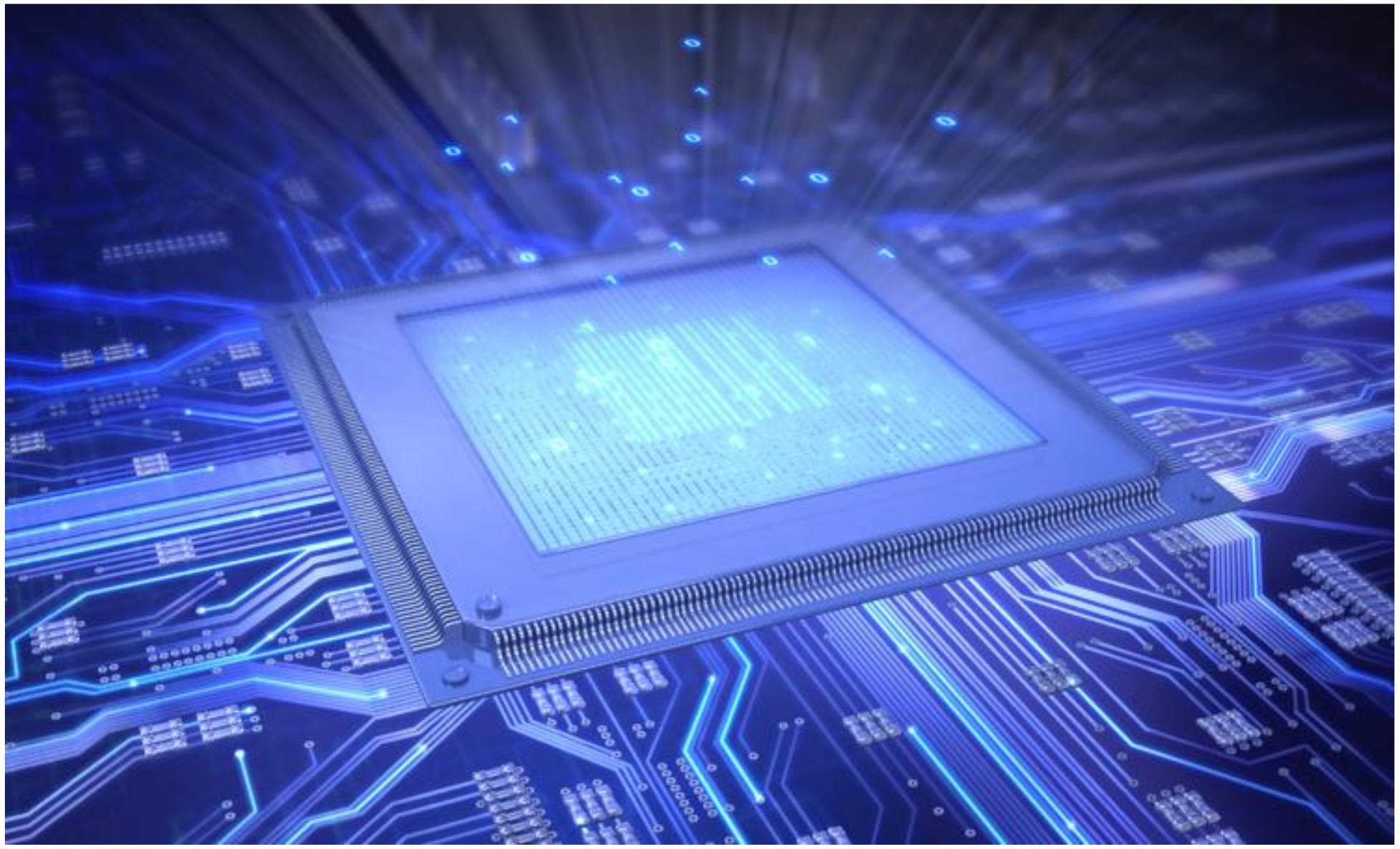
Mihai Bărbulescu

16.04.2020

University Politehnica of Bucharest

“The fundamental cause of the trouble is that in the modern world the stupid are cocksure while the intelligent are full of doubt.”

– Bertrand Russell (a.k.a Bertie)

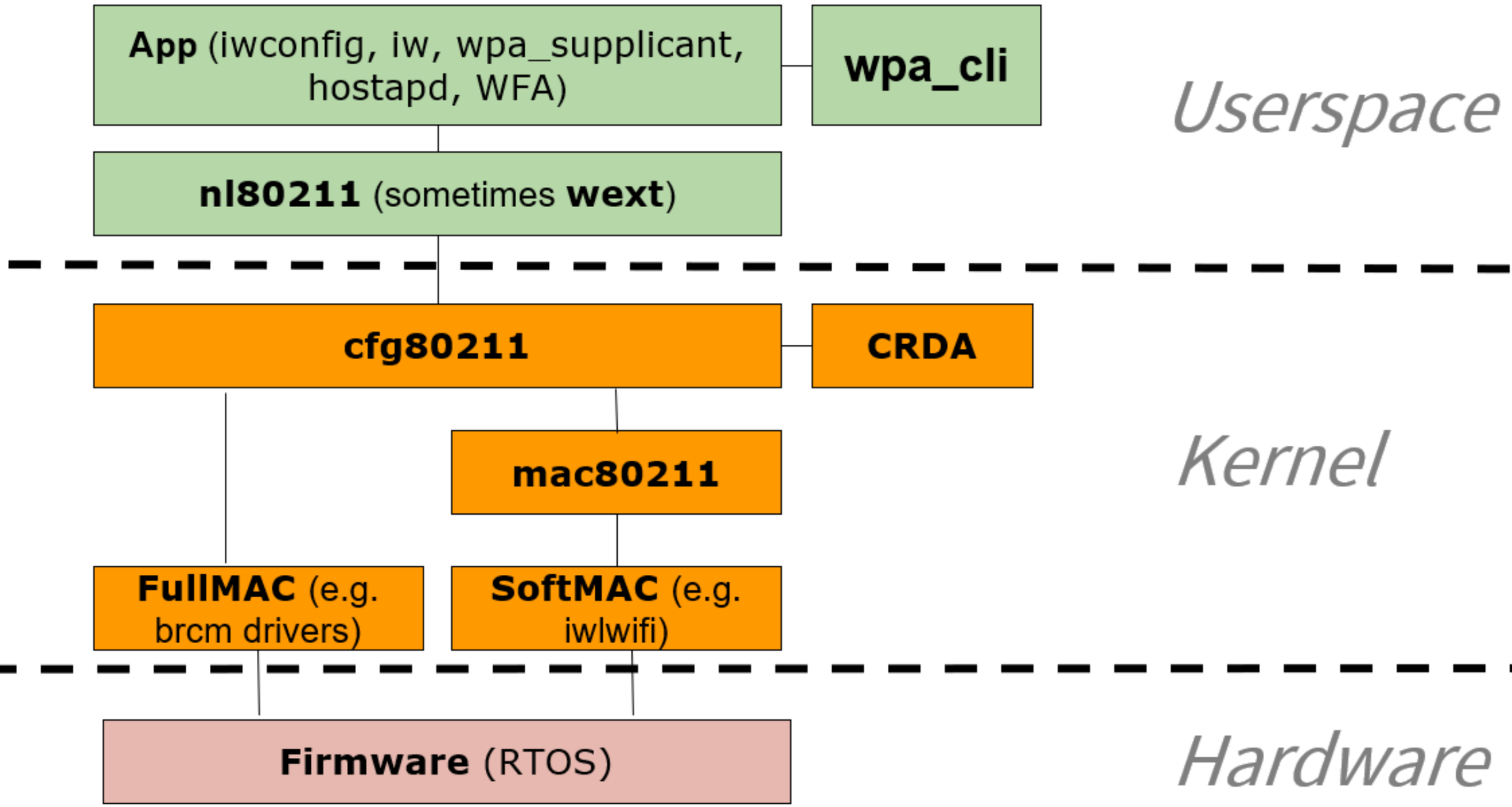


Components

Wifi networking stack

- Userspace configuration
- Driver
- Hardware (firmware, ASIC)
- And ... Protocol design/Standard

Linux stack


App (iwconfig, iw, wpa_supplicant, hostapd, WFA)
wpa_cli
nl80211 (sometimes wext)
cfg80211
CRDA
mac80211
FullMAC (e.g. brcm drivers)
SoftMAC (e.g. iwlwifi)
Firmware (RTOS)
Userspace
Kernel
Hardware

Mac80211 – wrapper for all hardware

- **Probe** interface (PCIe scan or USB – udev events usually)
- **Remove Ethernet header** and add LLC & dot11 header
- Fill (SA , DA, RA, ToDS/FromDS SEQ_CTRL)
- Fill (CONTROL , DURATION)
- **Rate adaption** (based on RX-VECTOR feedback from hardware) ⟶ set rate (legacy, HT, VHT, MCS, BW)
  - Sample algorithms: AARF (wifi dongles), Minstrel (Linux), iwl-mwm-rs (Intel cards)
- **Flag frame for encryption**
  - *“Software”* using CPU (e.g. ARM Crypto extensions)
  - Offload to firmware

What were those acronyms from previous slide?

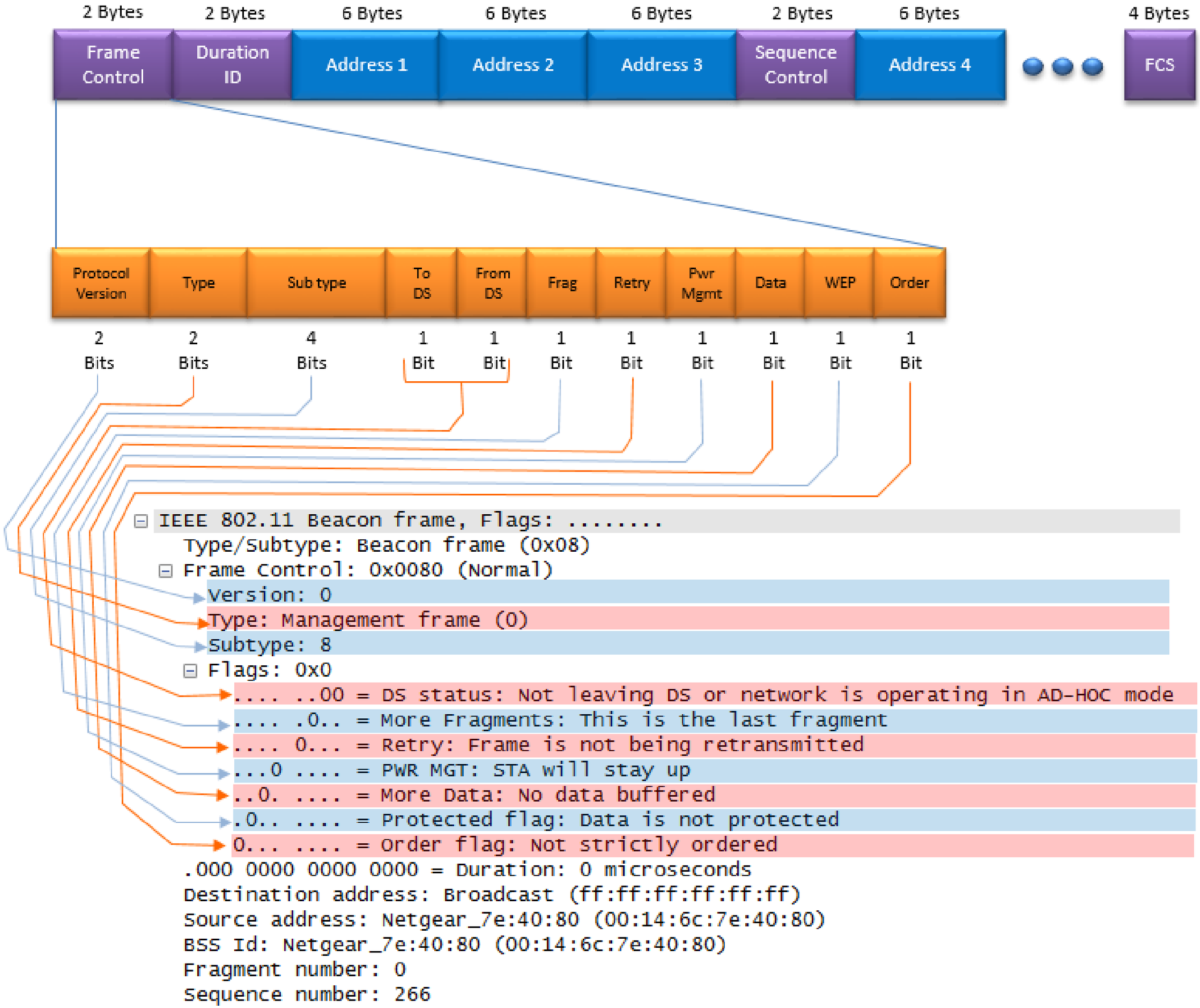
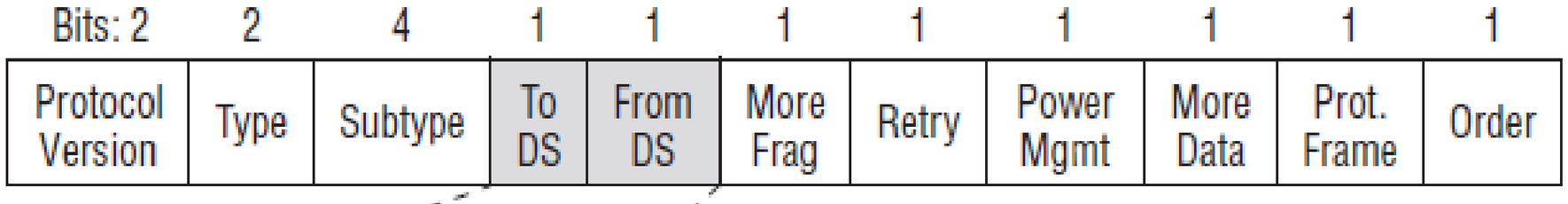


FIGURE 3.20 802.11 MAC addressing



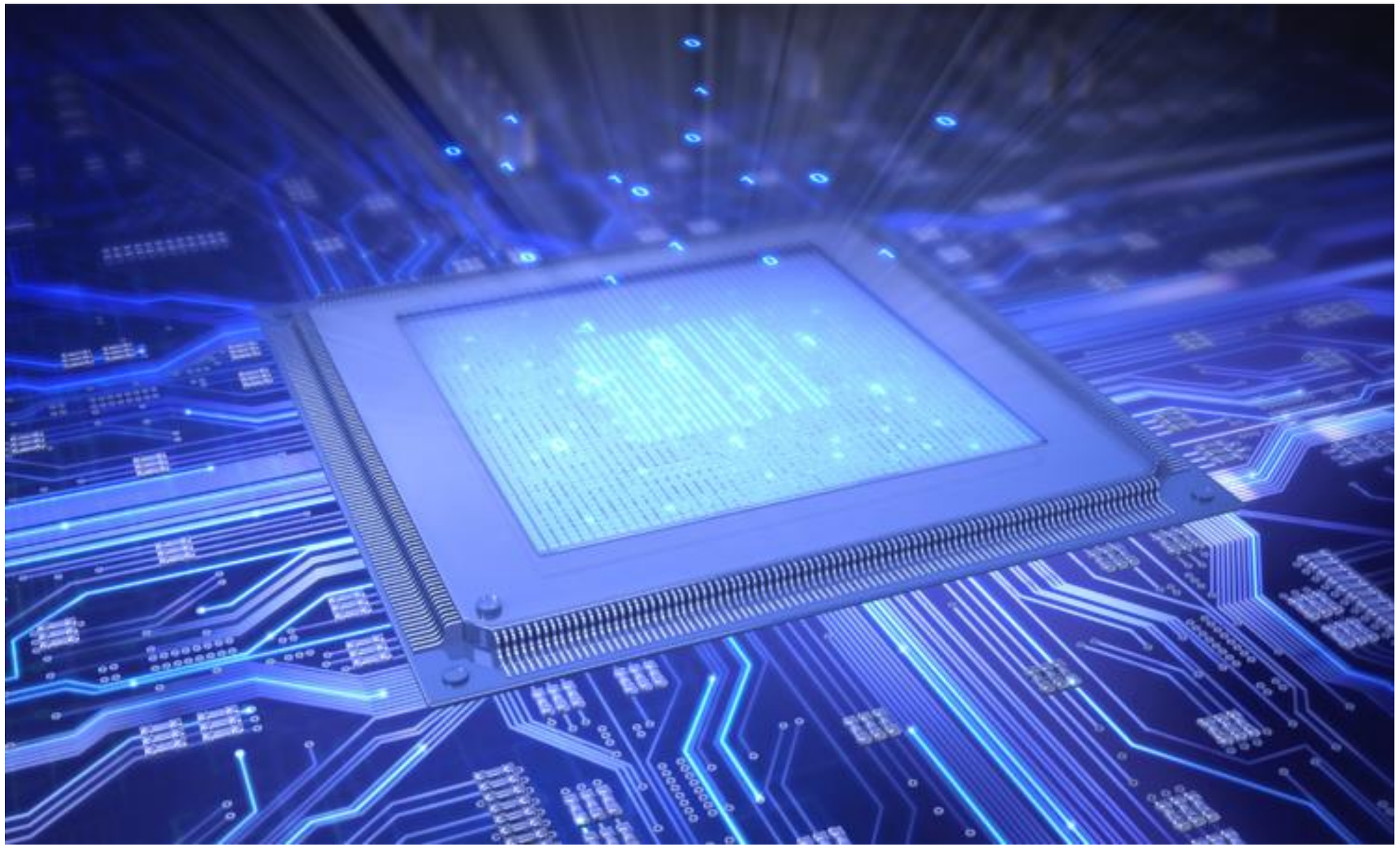
To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	RA = DA	TA = SA	BSSID	N/A
0	1	RA = DA	TA = BSSID	SA	N/A
1	0	RA = BSSID	TA = SA	DA	N/A
1	1	RA	TA	DA	SA

- SA = MAC address of the original sender (wired or wireless)
- DA = MAC address of the final destination (wired or wireless)
- TA = MAC address of the transmitting 802.11 radio
- RA = MAC address of the receiving 802.11 radio
- BSSID = L2 identifier of the basic service set (BSS)

- 802.11 frames typically use only 3 of the MAC address fields
- Frames send within WDS requires all 4 MAC address fields.

Ok, and firmware?

- Setup hardware registers
- DMA frame to/from Linux queues
- Keep statistics
- Reports to mac80211 RX info (e.g. RSSI) for rate adaption & carrier sense
- Implement DCF or EDCA
- Take care of bureaucracy such as L2 simple ACK/BlockACK, TSF, DTIM



Hardware exploits

Biztos, hardware?? Nem érdekes

- GoogleProjectZero project: BCM4339 exploit
- ROM - used to store firmware code
- RAM - data processing (heap, WiFi structures etc., mac80211 data, whatever) - firmware is **downloaded by the driver!!**
- Wi-Fi management frames encode most of their information in Information Elements (IEs)- structured as TLVs:
  - Cisco CCKM or 802.11r FT is trigger
  - Information embedded in management frames was vulnerable, could trigger a stack overflow → **CVE-2017-6957**

Oh … I want to read

https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html

https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html

https://bugs.chromium.org/p/project-zero/issues/detail?id=1051

Linux (and Android) was easy because

- You can get **ioctl** access - http://androidxref.com/7.1.1_r6/xref/system/sepolicy/ioctl_macros
- BCM provides free access to debug tools: https://android.googlesource.com/platform/hardware/broadcom/wlan/+/master/bcmdhd/dhdutil/Android.mk

Same research was carried on iOS

https://googleprojectzero.blogspot.com/2017/09/over-air-vol-2-pt-1-exploiting-wi-fi.html

https://googleprojectzero.blogspot.com/2017/10/over-air-vol-2-pt-2-exploiting-wi-fi.html

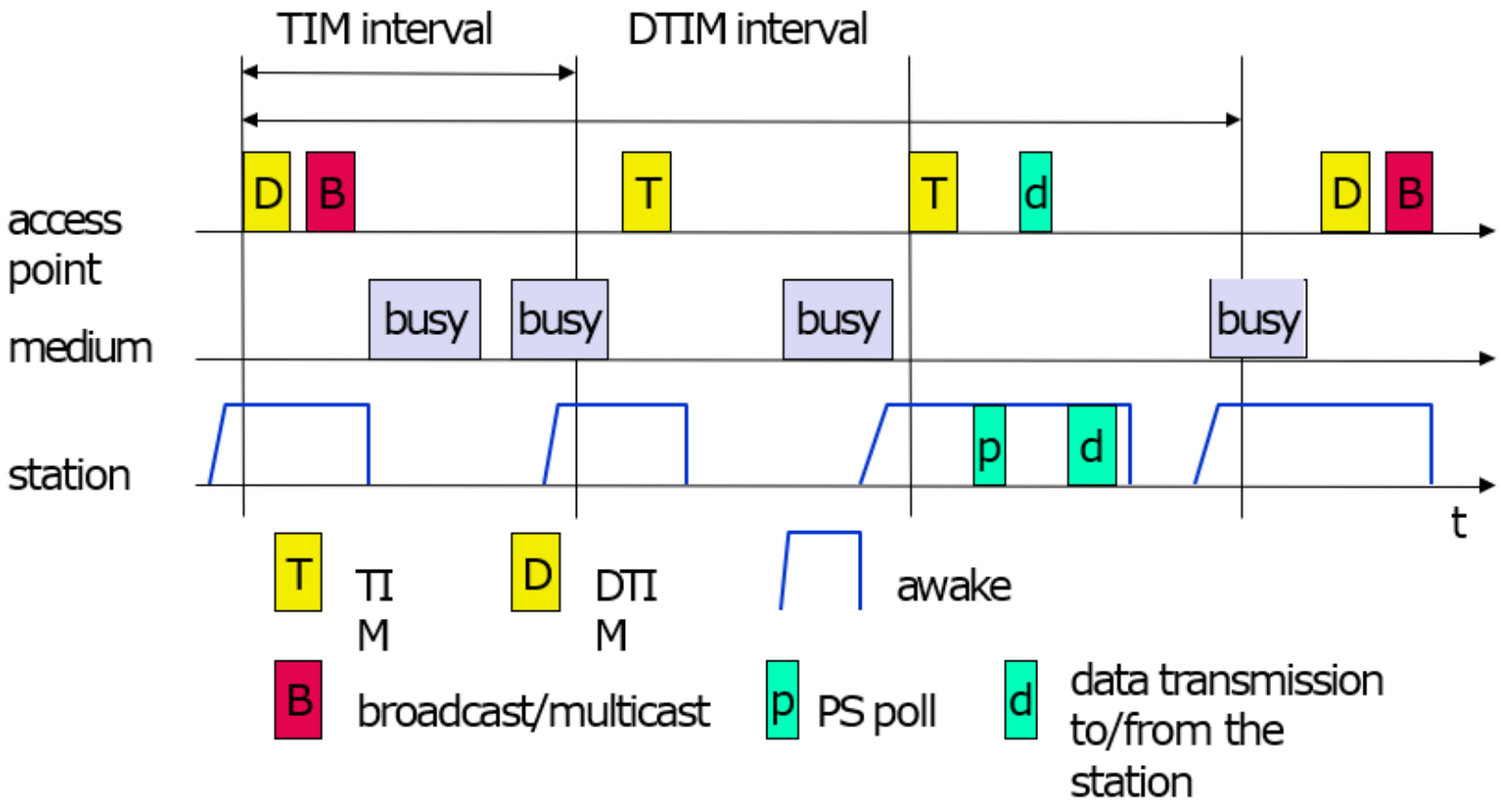
https://googleprojectzero.blogspot.com/2017/10/over-air-vol-2-pt-3-exploiting-wi-fi.html

Biztos, hardware?? Nem érdekes (cont.)

- Buffer overflow can be triggered in Realtek Wi-Fi chips, no user interaction needed
- The flaw dates to version 3.10.1 of the Linux kernel released in 2013
- CVE-2019-17666
- Affected device must be in radio range of malicious device
- Simply add malicious code in vendor-specific area of WiFi beacons and trigger buffer overflow in kernel
- On paper, [this] is an overflow that should be exploitable. Worst-case scenario, [this] is a denial of service; best scenario, you get a shell.

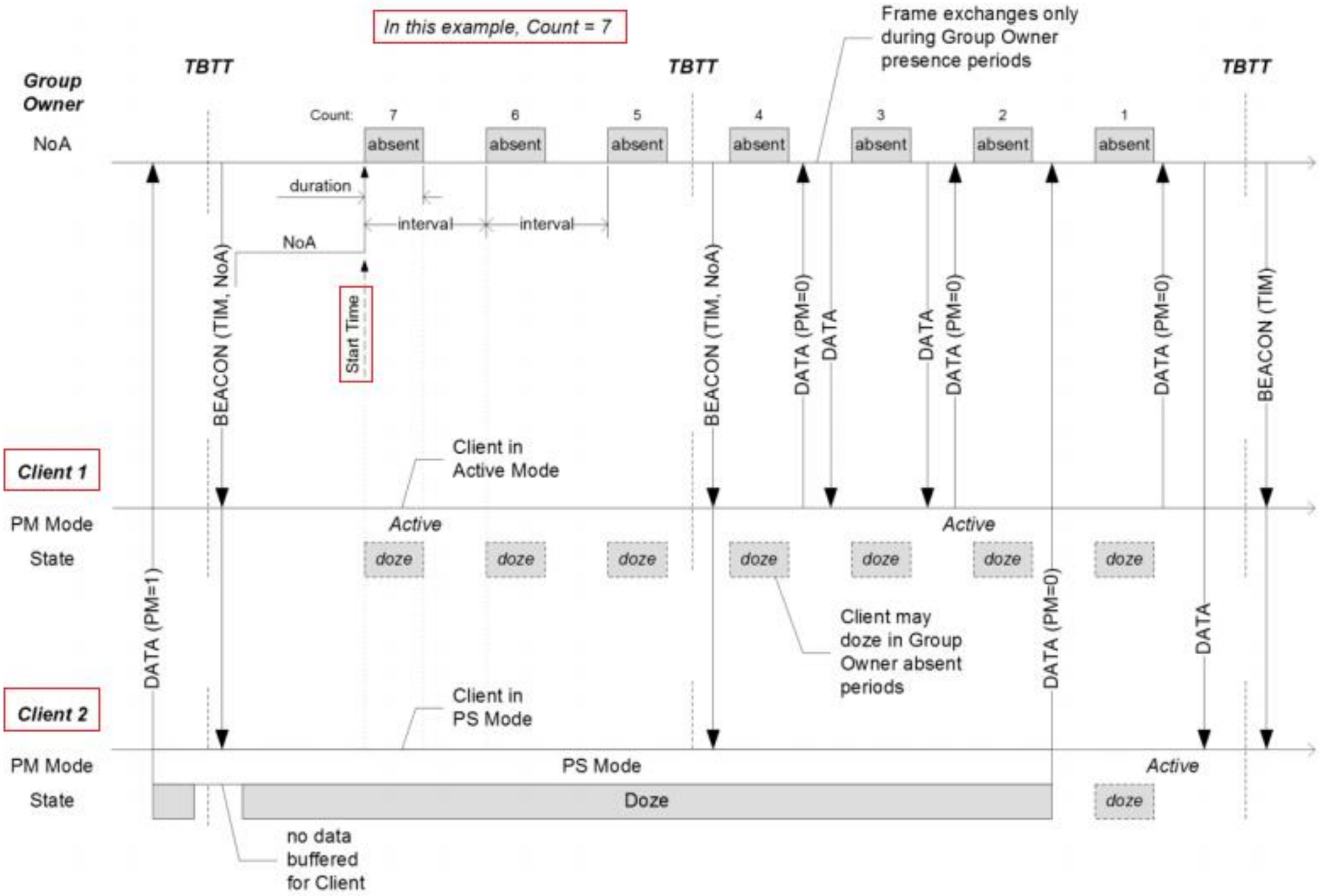
- More reading about rtlwifi issue: https://arstechnica.com/information-technology/2019/10/unpatched-linux-flaw-may-let-attackers-crash-or-compromise-nearby-devices/
- https://lkml.org/lkml/2019/10/16/1226

Biztos, hardware?? Nem érdekes (cont.)

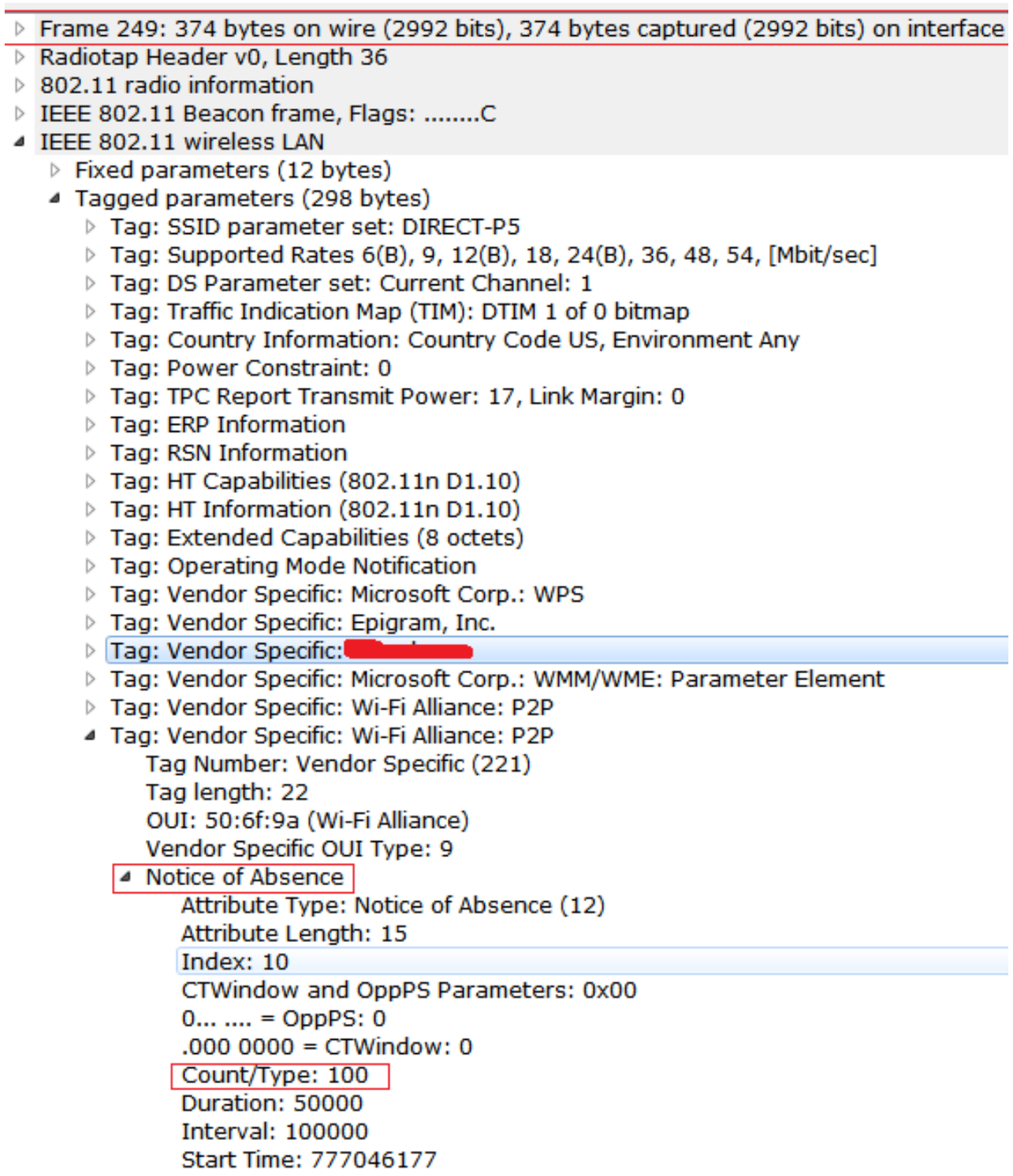


- Issue resides in beacons sent in wifi direct
- This picture shows normal flow AP-STA
- More about tim/dtim: https://blogs.arubanetworks.com/industries/802-11-tim-and-dtim-information-elements/

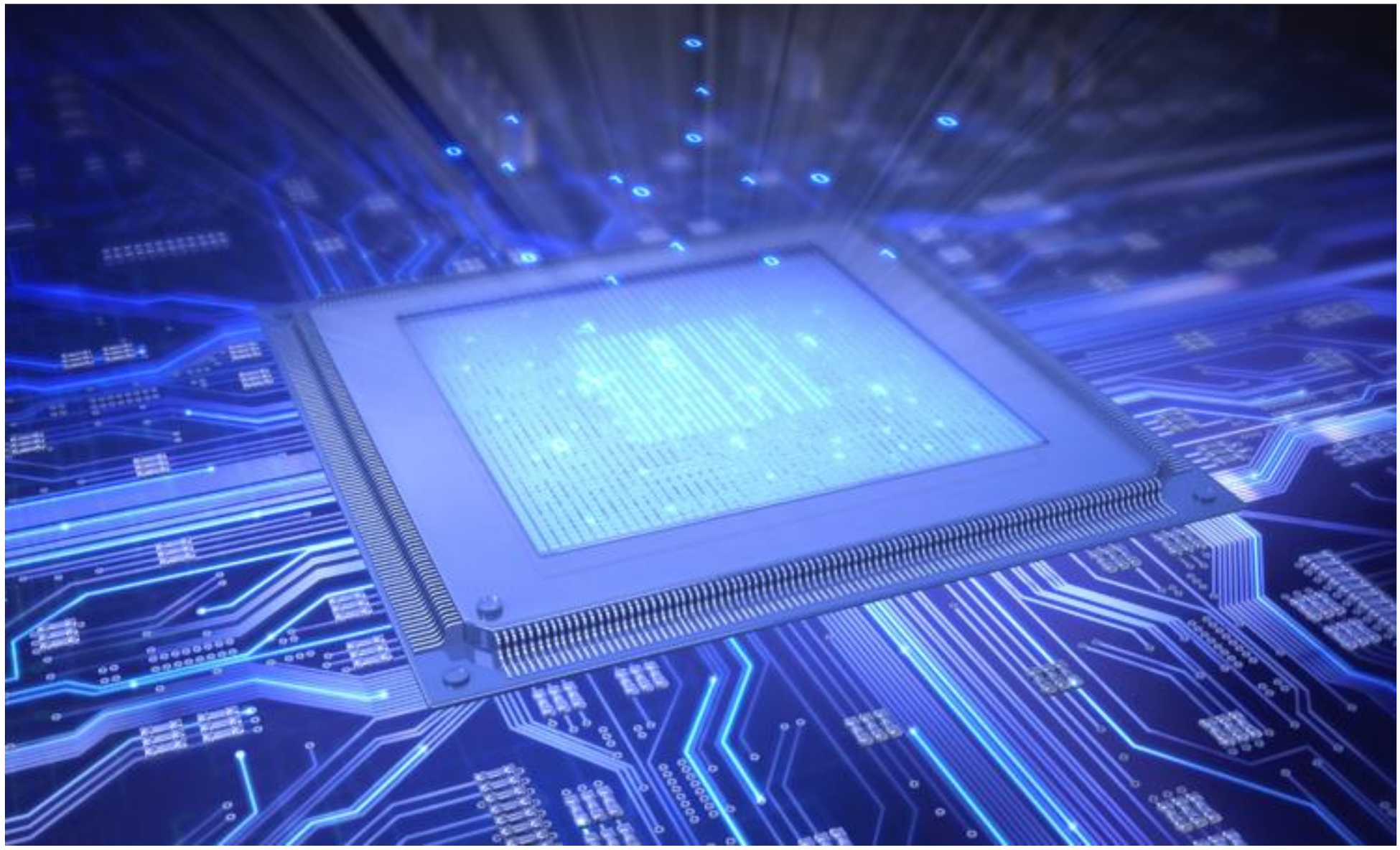
Biztos, hardware?? Nem érdekes (cont.)



Biztos, hardware?? Nem érdekes (cont.)



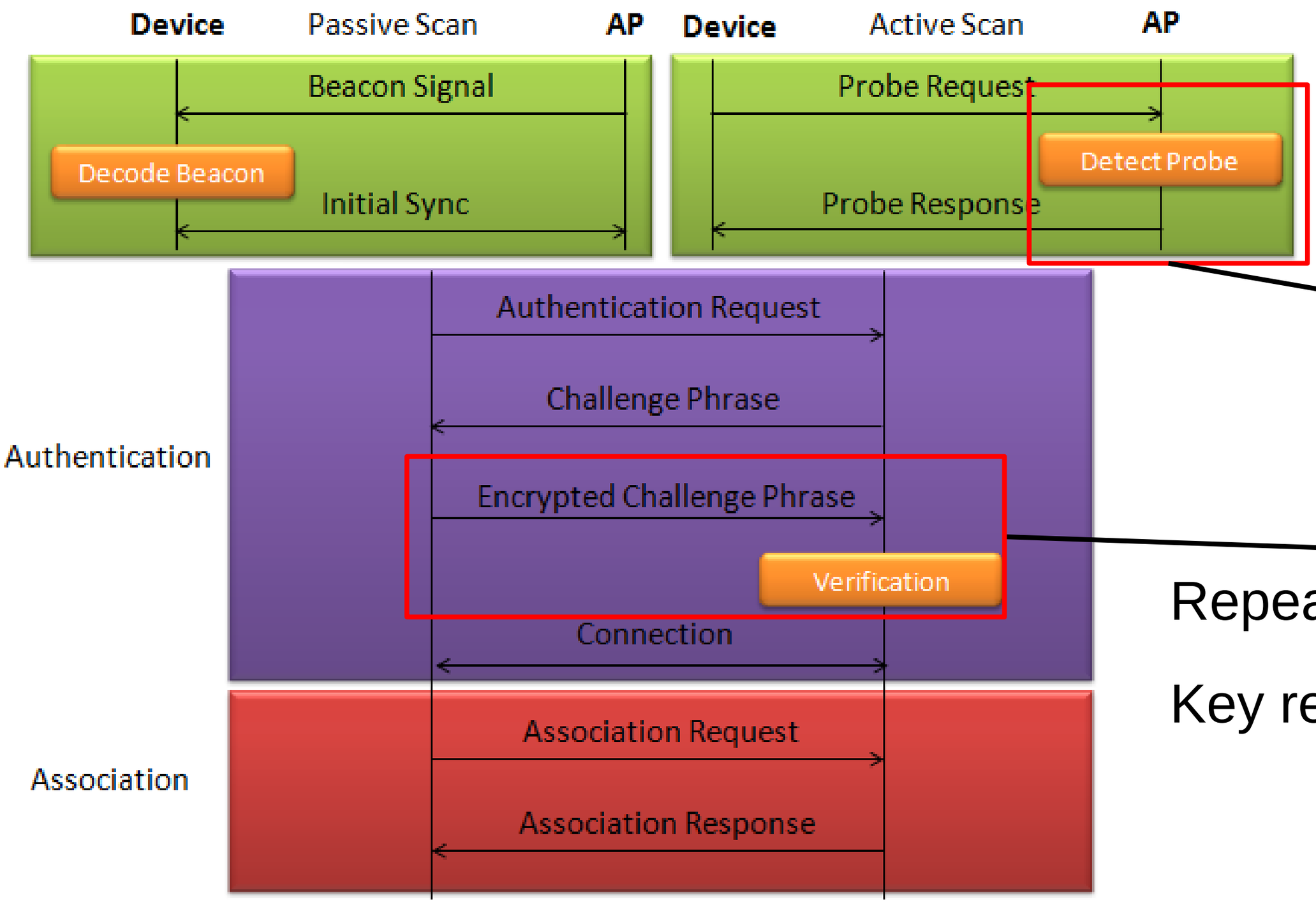
- Issue resides in beacons sent in wifi direct
- This picture shows a capture where you add your vendor specific bytes (just be sure to respect standard) and thus you trigger buffer overflow
- Source of pics: https://praneethwifi.in/2019/12/07/p2p-power-saving-mechanism-part-2-notice-of-absence-noa/



WPA Epic

Known WiFi security protocols

- **WEP** - *sept 1999* - should be abandoned :)
  - Aircrak-ng tutorial using ARP + IVs
- **WPA** (or WPA-PSK) - 802.11i - around 2003 - as w/a to WEP
  - TKIP for encryption, PSK for getting keys
- **WPA2** - 802.16i, 2004 first rel, 2006 - mainstream
  - AES (because US Gov), PSK for key sharing
  - KRACK & other board people
- **WPA3** - Jan 2018 - fresh!!
  - Should replace “safely” key generation



https://www.sharetechnote.com/html/WLAN_Protocol.html

Known WPA2 vulnerabilities - BruteForce

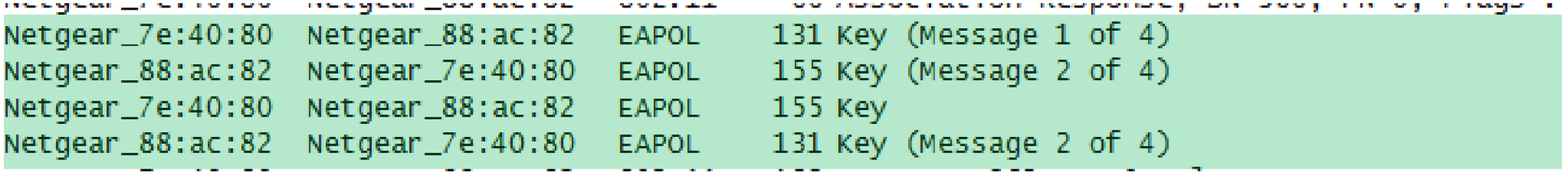
- Exploits below 4-way handshake when associating

Netgear_7e:40:80	Netgear_88:ac:82	EAPOL	131	Key (Message 1 of 4)
Netgear_88:ac:82	Netgear_7e:40:80	EAPOL	155	Key (Message 2 of 4)
Netgear_7e:40:80	Netgear_88:ac:82	EAPOL	155	Key
Netgear_88:ac:82	Netgear_7e:40:80	EAPOL	131	Key (Message 2 of 4)

- If you have a Linux try this at home, let me know results: https://www.aircrack-ng.org/doku.php?id=cracking_wpa

Known WPA2 vulnerabilities - KRACK

- Exploits below 4-way handshake when associating



- Trick the vulnerable client to reinstall key already in use
- Force reset of packet numbers - go to a rogue AP
- Wifi has loses ⟶ retransmission is carried by AP
- Entire research at https://www.krackattacks.com/
- Catastrophic because this is design protocol issue, not hardware, nor crappy vendor

WPA2 hobby cracks

- Prof Bill Bunachan implements KRACK using a RPI and 10$ wifi transceiver to crack a PBKDF2-SHA1, without need to capture entire association process
- Random guy on forum reads about WPA3 founds a way to crack WPA2

Source1: https://medium.com/@billatnapier/the-beginning-of-the-end-of-wpa-2-cracking-wpa-2-just-got-a-whole-lot-easier-55d7775a7a5a

Source 2: https://hashcat.net/forum/thread-7717.html

WPA3

- Replace PSK for getting key with Simultaneous Authentication of Equals (SAE) and 128-bit encryption in personal
- For enterprise: 256-bit GCMP
- Key derivation and confirmation: 384-bit-HMAC with SHA384
- Key establishment and authentication: ECDH + ECDSA using a 384-bit elliptic curve
- Robust management frame protection: BIP-GMAC-256

WPA3 – DragonFly Attack – not a joke

Dragonblood: A Security Analysis of WPA3's SAE Handshake

Mathy Vanhoef
New York University Abu Dhabi
Mathy.Vanhoef@nyu.edu

Eyal Ronen
Tel Aviv University and KU Leuven
eyal.ronen@cs.tau.ac.il

ABSTRACT

The WPA3 certification aims to secure Wi-Fi networks, and provides several advantages over its predecessor WPA2, such as protection against offline dictionary attacks and forward secrecy. Unfortunately, we show that WPA3 is affected by several design flaws, and analyze these flaws both theoretically and practically. Most prominently, we show that WPA3's Simultaneous Authentication of Equals (SAE) handshake, commonly known as Dragonfly, is affected by password partitioning attacks. These attacks resemble dictionary attacks and allow an adversary to recover the password by abusing timing or cache-based side-channel leaks. Our side-channel attacks target the protocol's password encoding method. For instance, our cache-based attack exploits SAE's hash-to-curve algorithm. The resulting attacks are efficient and low cost: brute-forcing all 8-character lowercase password requires less than 125$ in Amazon EC2 instances. In light of ongoing standardization efforts on hash-to-curve, Password-Authenticated Key Exchanges (PAKEs), and Dragonfly as a TLS handshake, our findings are also of more general interest. Finally, we discuss how to mitigate our attacks in a backwards-compatible manner, and explain how minor changes to the protocol could have prevented most of our attacks.

design and implementation flaws. For instance, when verifying the assumptions made by the formal proof of the SAE handshake [59], we discovered both timing and cache-based side-channel vulnerabilities in its password encoding method. We empirically confirmed all our findings against both open source and recently-released proprietary implementations of WPA3.

All combined, our work resulted in the following contributions:

- We provide a self-contained and high-level description of WPA3 and its SAE handshake (Section 2 and 3).
- We show that the anti-clogging mechanisms of SAE is unable to prevent denial-of-service attacks (Section 4). In particular, by abusing the overhead of SAE's defenses against already-known side-channels, a resource-constrained device can overload the CPU of a professional Access Point (AP).
- We present a dictionary attack against WPA3 when it is operating in transition mode (Section 5). This is accomplished by trying to downgrade clients to WPA2. Although WPA2's 4-way handshake detects the downgrade and aborts, the frames sent during the partial 4-way handshake provide enough information for a dictionary attack. We also present a downgrade attack against SAE, and discuss implementation-

https://papers.mathyvanhoef.com/dragonblood.pdf

WPA3 – DragonFly Attack – not a joke

- Dragonforce - experimental tool that takes the information to recover from the timing attacks and performs a password partitioning attack.
- Dragonslayer - implements attacks against EAP-P (Extensible Authentication Protocol-Password)

"Nearly all of our attacks are against SAE's password encoding method, i.e., against its hash-to-group and hash-to-curve algorithm. Interestingly, a simple change to this algorithm would have prevented most of our attacks," the researchers say

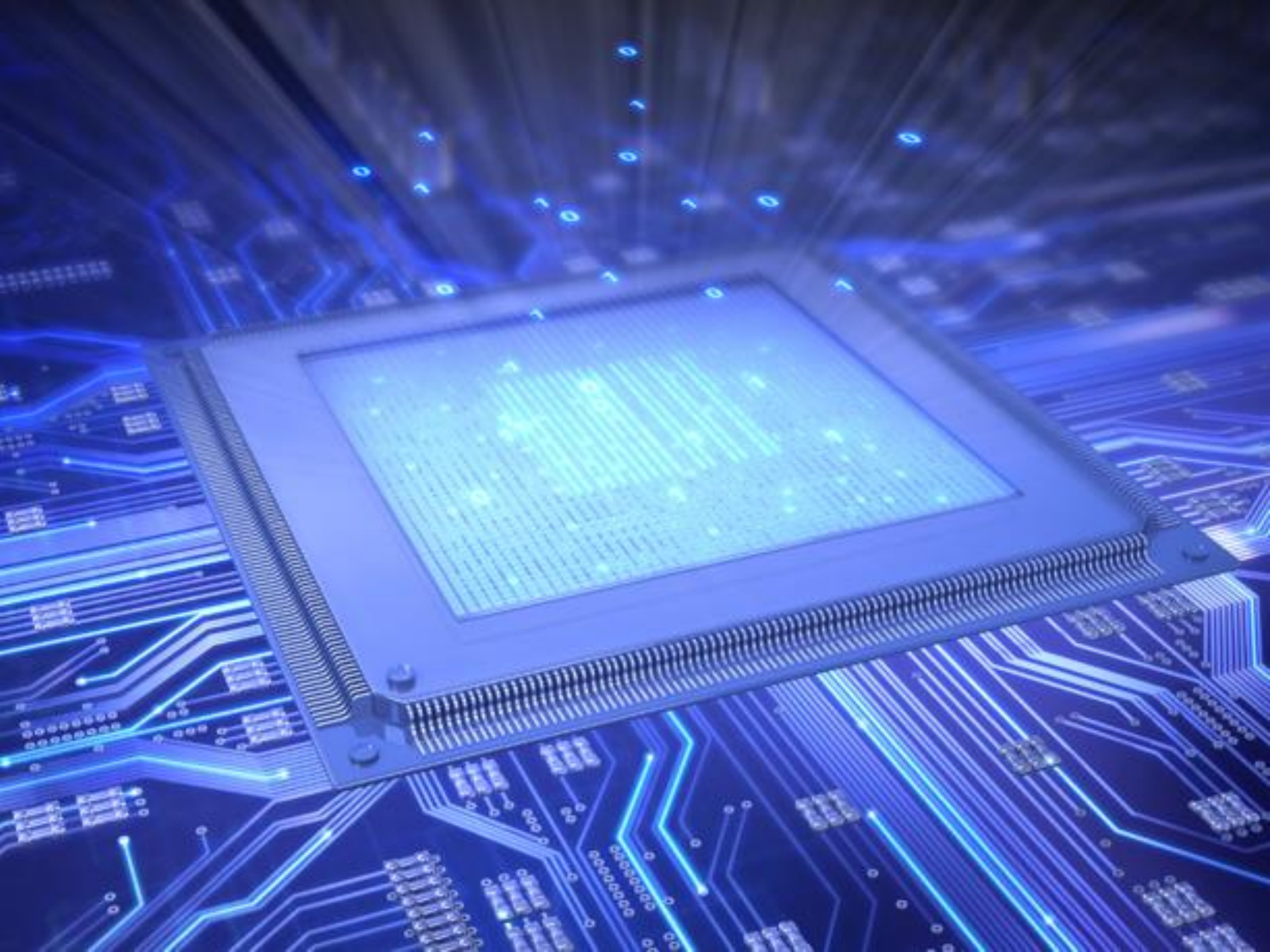
WPA3 other weaknesses

https://thehackernews.com/2019/04/wpa3-hack-wifi-password.html?m=1

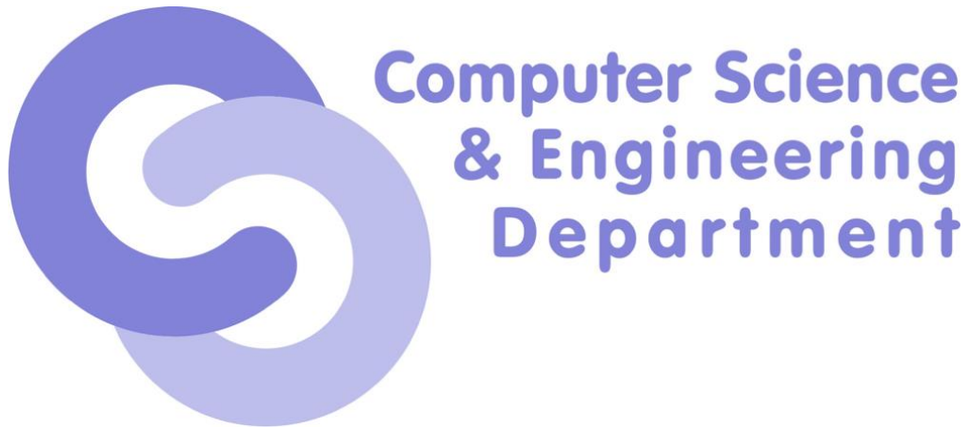
https://medium.com/asecuritysite-when-bob-met-alice/wpa-3-dragonfly-out-of-the-frying-pan-and-into-the-fire-35240aef4376

Now you care? (instead of conclusions)

- Yes, I should care about who makes the hardware and how
- Yes, I should care who develops protocols
- Yes, I should read some papers
- Yes, Wi-Fi is worse than loses: MAC headers are always open
- Yes, capturing wifi from my neighbours can be fun



Thank you for your attention.



mbarbulescu@upb.ro