Santa Clara University Internet of Things Research Lab SIOTLAB
Internet of Things Research Lab
Department of Computer Science and Engineering
Santa Clara University

ACM International Symposium on Mobility Management and Wireless Access (MobiWac 2021)

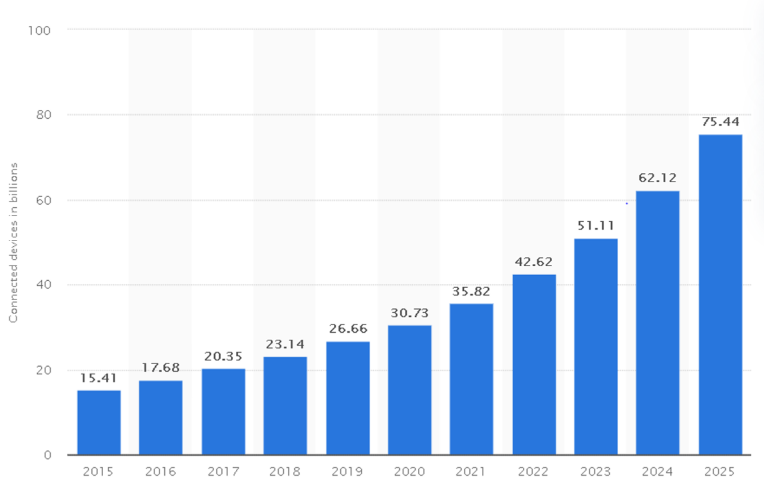
FLIP: A Framework for Leveraging eBPF to Augment WiFi Access Points and Investigate Network Performance

Jaykumar Sheth, Vikram Ramanna, and Behnam Dezfouli

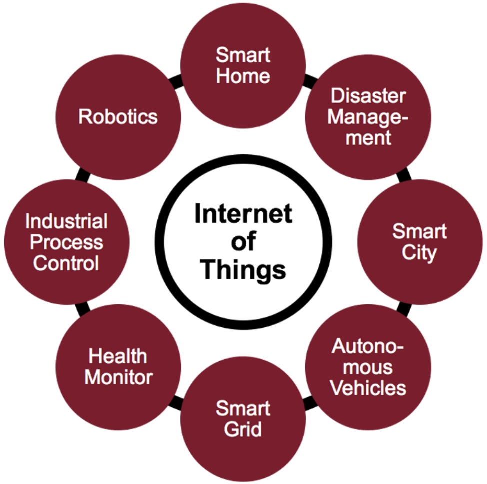
November 2021

Ubiquity of Internet of Things (IoT) Devices

- There will be more than 75 Billion connected devices by 2025 [1]



Number of connected devices worldwide 2015-2025

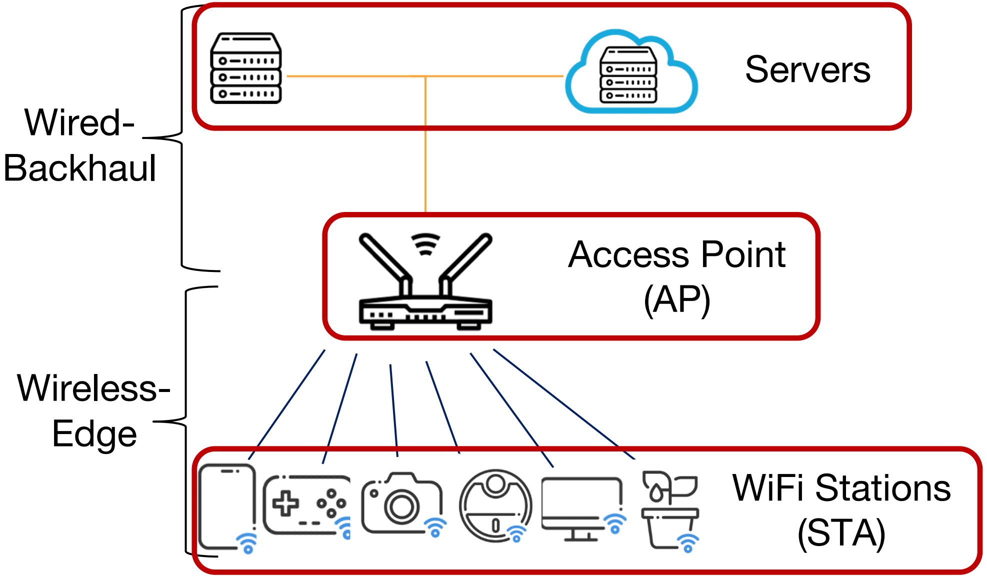


Application domains of IoT

To offer ease of deployment and support mobility, most of these applications rely on wireless communication

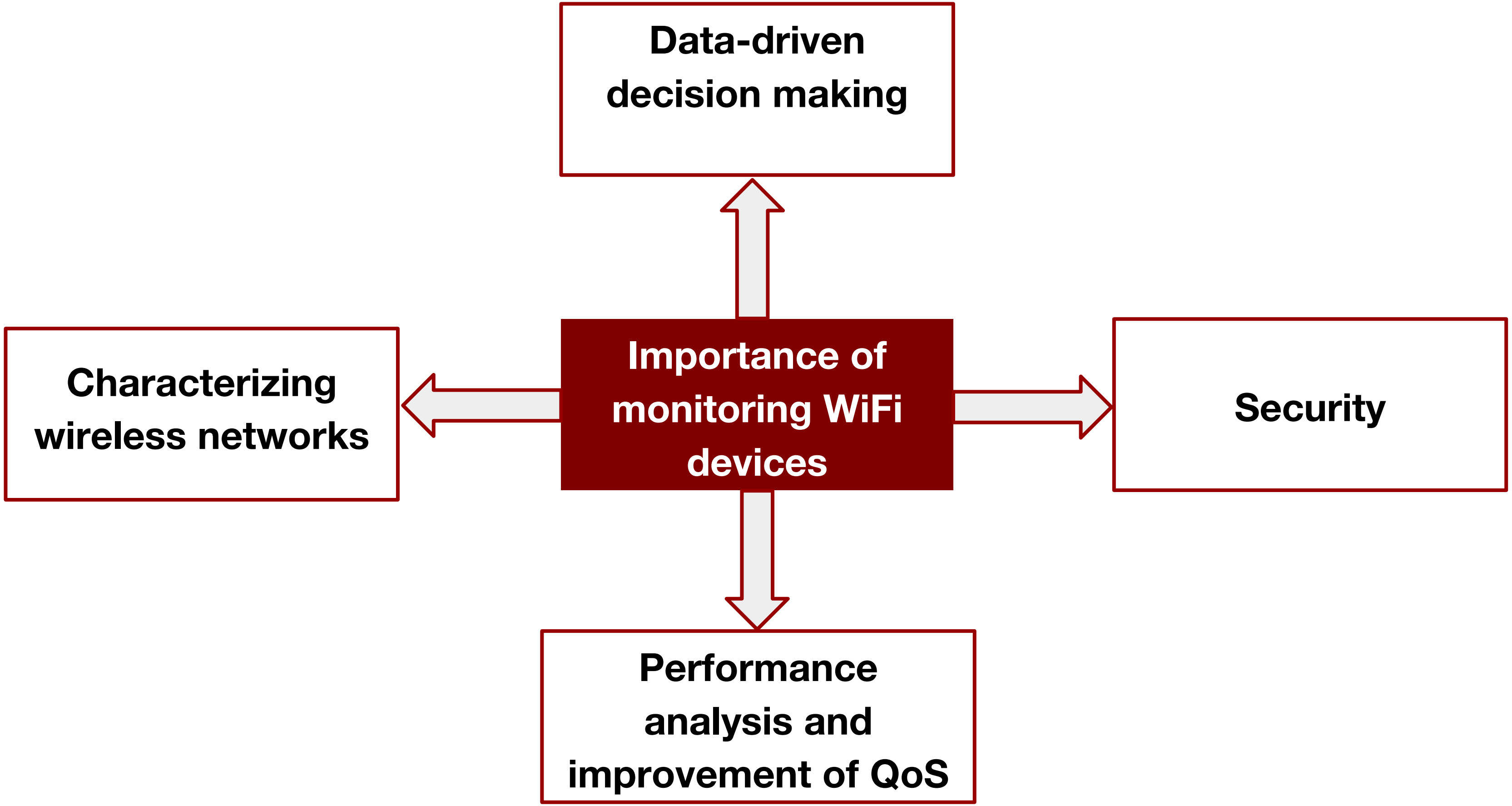
WiFi-based IoT Networks

- **Increasing adoption of WiFi**
  - Higher data rate vs. Bluetooth and Zigbee
  - Unlicensed spectrum
  - WiFi devices are becoming less expensive (i.e., <$3 per module)
  - Ubiquity of existing WiFi deployments
    - Residential
    - Enterprise
    - Industrial



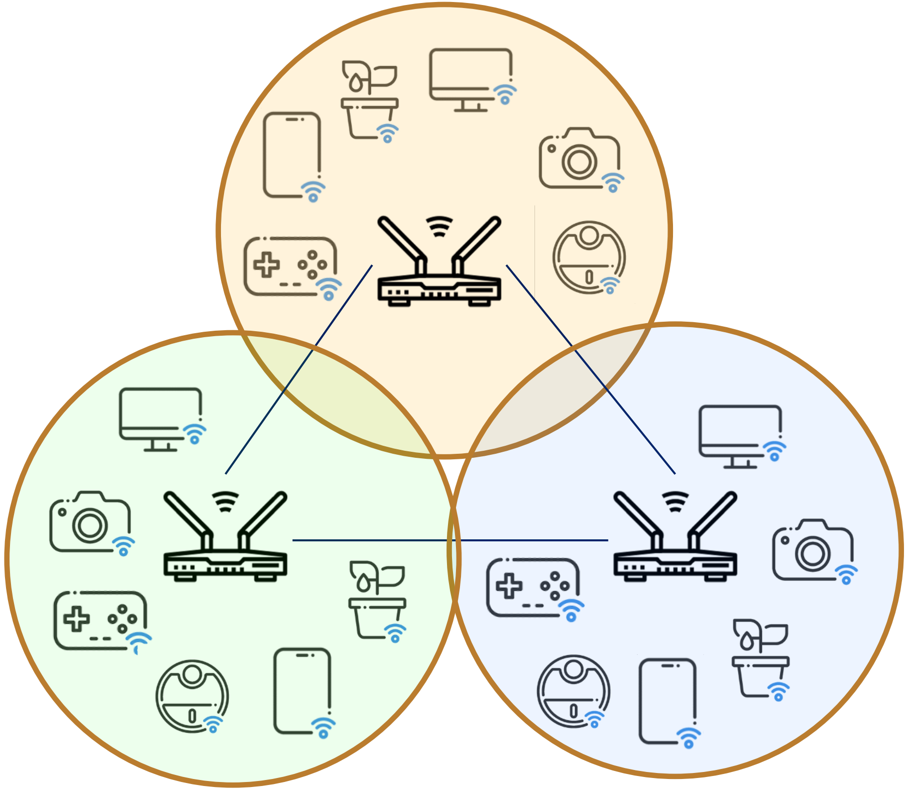
Components of a typical IoT network

Monitoring WiFi Networks



Importance of Statistics Collection on the AP

- Global view of the network
- Centralized vs. Distributed
  - Collecting data at each STA is infeasible due to hardware and software constraints
  - Gathering data from all devices has communication overhead
- All packets pass through the AP
- APs can easily coordinate with other APs in Extended Service Set (ESS)



Existing Methods: Sniffer-based Approaches

- Using sniffers to capture network traffic has been widely used for network monitoring
  - Manufacturers have adopted this method in commercial deployments as well

Cisco Meraki MR18

Arista C-230

- Additional cost
  - as they require dedicated hardware and software resources
- Each incoming packet requires the operating system to process an additional packet

Existing Methods: Network Management Protocols

- Protocols commonly used for obtaining network statistics
  - SNMP (Simple Network Management Protocol)
  - NETCONF
  - NetFlow
- Imposes a high overhead on network throughput [2, 3]
  - Hence, the rate of network monitoring is considerably lower
- For example, in [4, 5], SNMP logs are polled every 5 minutes, and in [6], monitoring data is collected once every 30 minutes

Existing methods: Linux Tools

- Utilities allow the collection of monitoring data from Linux networking devices
  - ethtool
  - iwpriv
  - debugfs

- Developed for debugging or configuration, not for an efficient and high-rate collection of monitoring data

- The statistics reported are not extensive
  - In order to add a debug statement
    - In debugfs: associated kernel modules must be modified
    - In case of ethtool an ioctl call needs to be added

```
siotlab1@siotlab1-laptop:~$ ethtool -S wlp3s0
NIC statistics:
     rx_packets: 60474
     rx_bytes: 7444249
     rx_duplicates: 1
     rx_fragments: 60470
     rx_dropped: 1
     tx_packets: 68879
     tx_bytes: 103833581
     tx_filtered: 0
     tx_retry_failed: 0
     tx_retries: 793
     sta_state: 0
     txrate: 0
     rxrate: 0
     signal: 0
     channel: 5765
     noise: 161
     ch_time: 258790
     ch_time_busy: 25789
     ch_time_ext_busy: 18446744073709551615
     ch_time_rx: 16834
     ch_time_tx: 8596
     tx_pkts_nic: 68936
     tx_bytes_nic: 103981330
     rx_pkts_nic: 944415285
     rx_bytes_nic: 3685929749
     d_tx_pkts_BE: 39496
     d_tx_pkts_BK: 29410
     d_tx_pkts_VI: 0
     d_tx_pkts_VO: 30
     d_tx_bytes_BE: 60839171
     d_tx_bytes_BK: 43137700
     d_tx_bytes_VI: 0
     d_tx_bytes_VO: 4459
     d_tx_mpdus_queued_BE: 0
     d_tx_mpdus_queued_BK: 0
     d_tx_mpdus_queued_VI: 0
     d_tx_mpdus_queued_VO: 28
     d_tx_mpdus_completed_BE: 37
     d_tx_mpdus_completed_BK: 2
     d_tx_mpdus_completed_VI: 0
     d_tx_mpdus_completed_VO: 30
```

Energy Monitoring and Switching Delay in WiFi Networks

- Two of the most important statistics
  - **Energy consumption of connected stations**
    - In low-power stations, WL-NIC accounts for more than 50% of the station's total energy budget [7, 8, 9]
  - **Downlink delivery delay at the AP**
    - Delay experienced at APs is more than 60% of the total communication delay between a station and server [10, 11, 12]
    - AP's traffic intensity
    - Queuing disciplines used at various layers
    - Airtime utilization by other APs and stations
    - Access category of flows

Kernel Functionality Modifications

- **Native kernel modifications:**
  - Change kernel source code
  - Wait several years for the new kernel version to become a commodity

- **Modification via kernel modules**
  - Fix it up regularly, as every kernel release may break it
  - Risk corrupting your Linux kernel due to lack of security boundaries

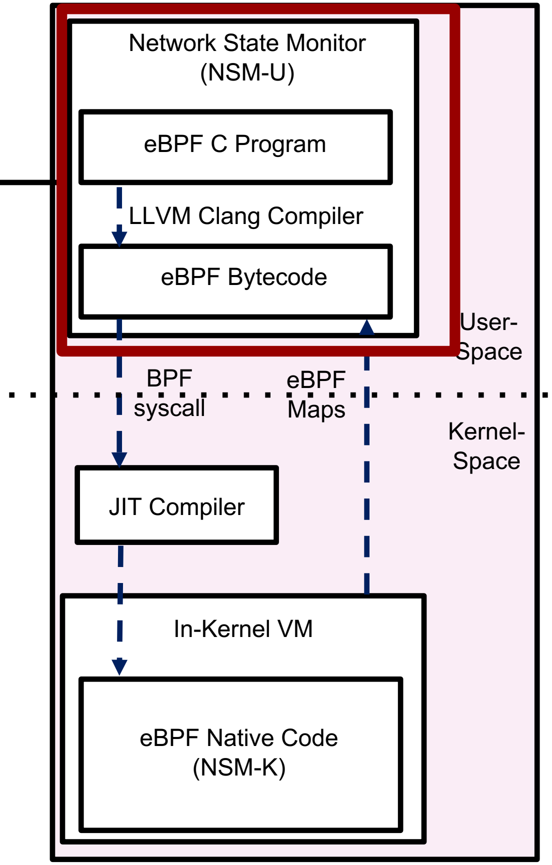
Kernel Functionality Modifications

eBPF

- Provides a sandboxed environment (in-kernel virtual machine) which runs programs injected from user space
- Fundamentally the same as BPF (Berkley Packet Filter)
- With additionally instructions and register to widen the scope apart from network traffic filtering
- Add additional capabilities to the operating system at runtime **without requiring to change kernel source code or load kernel modules**

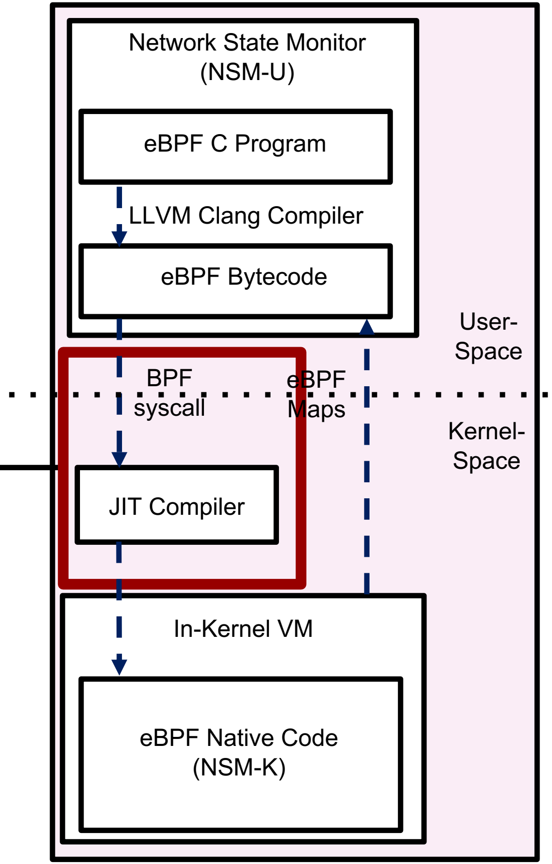
Network State Monitor: eBPF (Extended Berkeley Packet Filter)

- eBPF programs are **compiled utilizing the LLVM-clang compiler**
- This produces an **object file containing BPF bytecode**



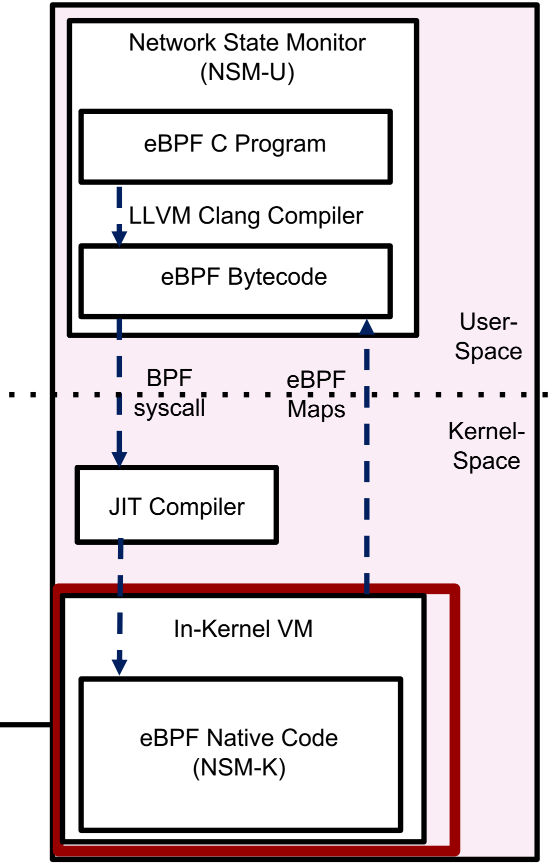
Network State Monitor: eBPF (**Extended Berkeley Packet Filter**)

- **BPF() syscall** loads the bytecode into the kernel for the final stage
- **Verifier** checks for correctness and safety
- **(Just-In-Time) JIT compiler** translates BPF instructions into native machine code



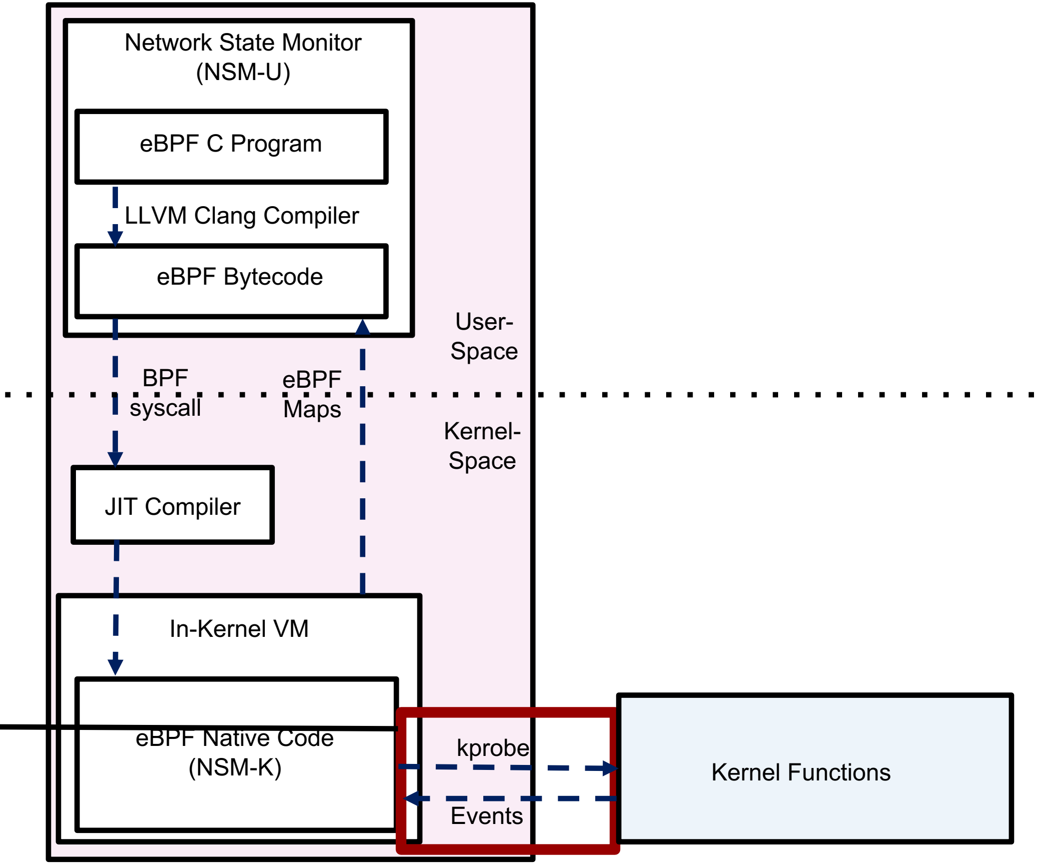
Network State Monitor: eBPF (**Extended Berkeley Packet Filter**)

- This native machine code is executed in an in-kernel virtual machine and thus, reduces context-switching overhead



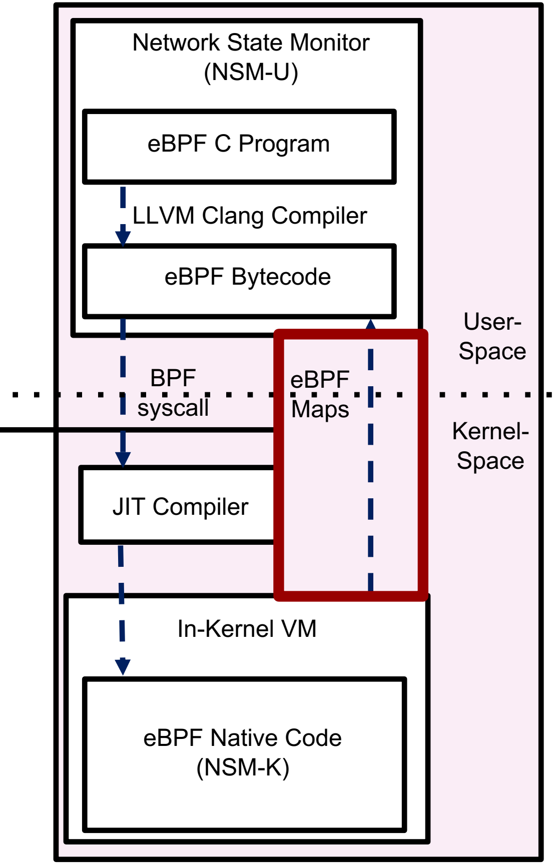
Network State Monitor: eBPF (**Extended Berkeley Packet Filter**)

- *kprobe* instruments the beginning of a function's execution with the breakpoint instruction (e.g., int3 for x86 platforms)
  - On occurrence of an event, i.e., when a particular kernel function is called
  - The eBPF program is executed
  - The kernel function runs normally
- Allows accessing kernel function arguments and data structures



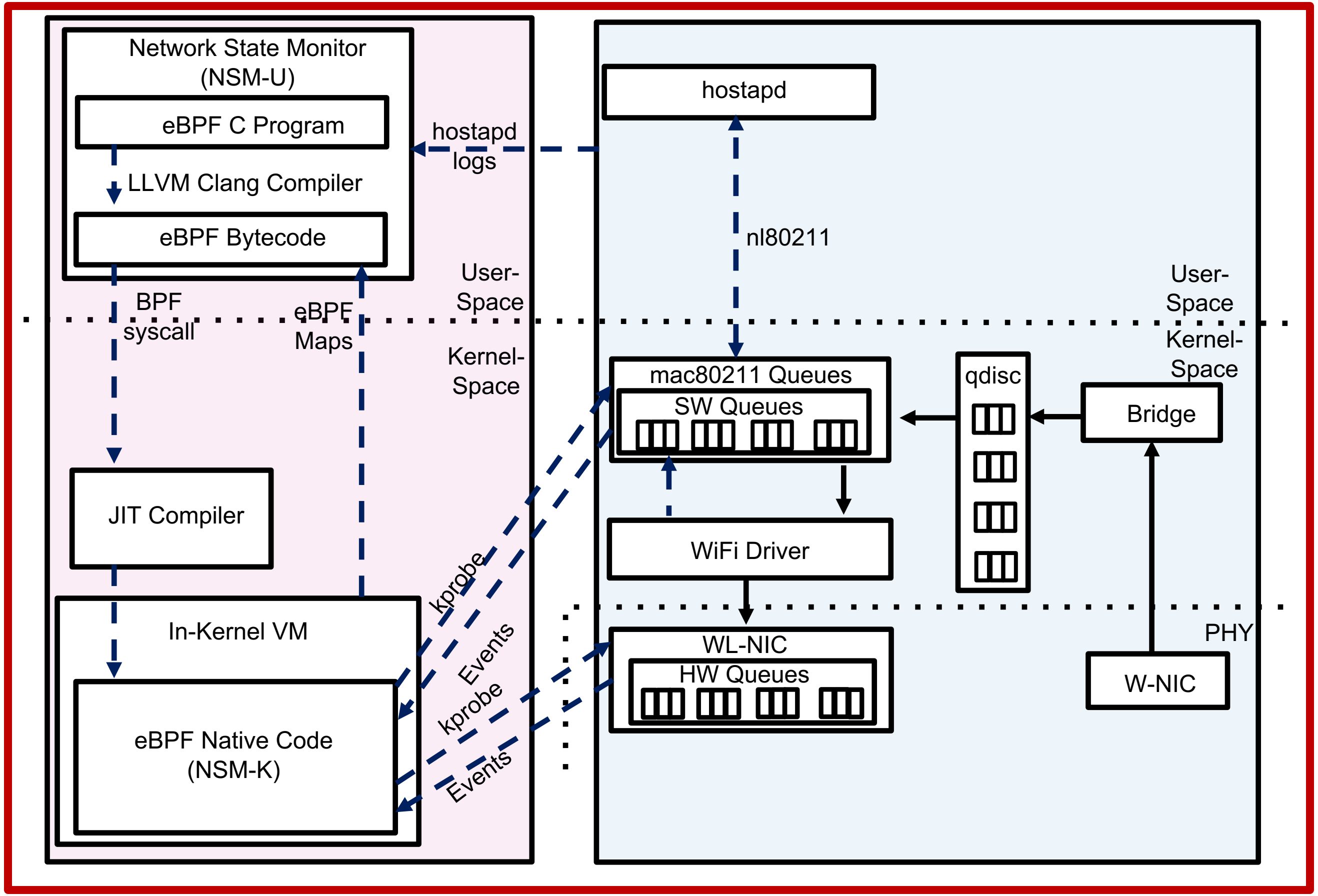
Network State Monitor: eBPF (**Extended Berkeley Packet Filter**)

- To store and transfer the collected information to the user-space
- Populated in-kernel by your custom eBPF program
- Can be read from user-level code asynchronously
- Uses shared-memory ring buffer



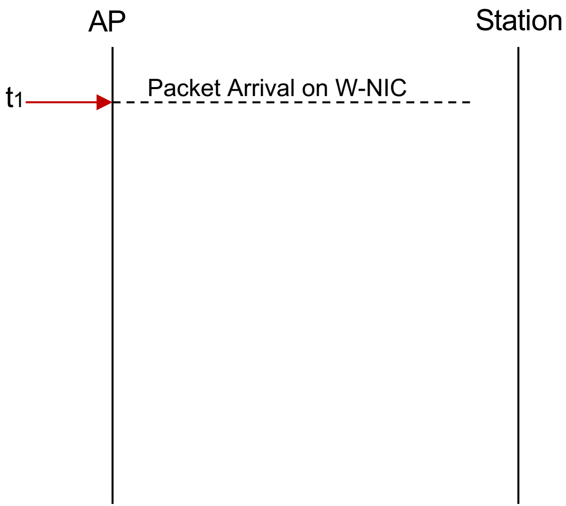
Network Stack Monitoring via eBPF

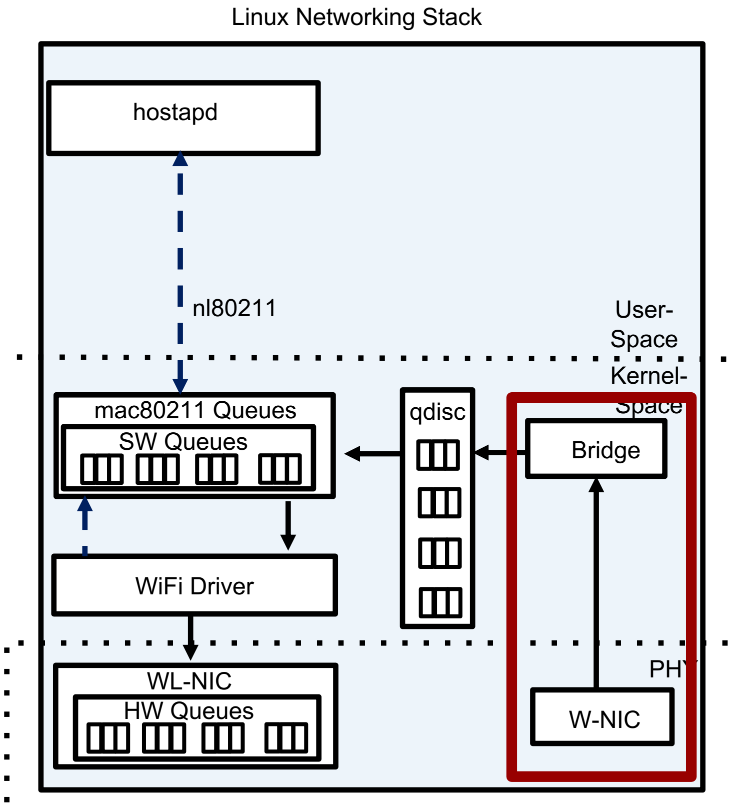
- Utilizing the eBPF technology, NSM taps into multiple modules in the networking stack to record measurements



Monitoring the AP's Networking Stack: Delay Analysis

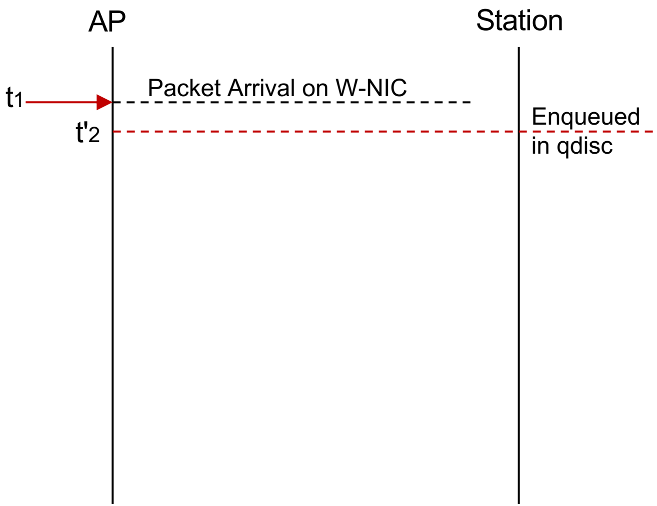
- A packet arrives at the AP from the W-NIC (wired interface)
- Bridge acts as interface between the wired and wireless subsystems

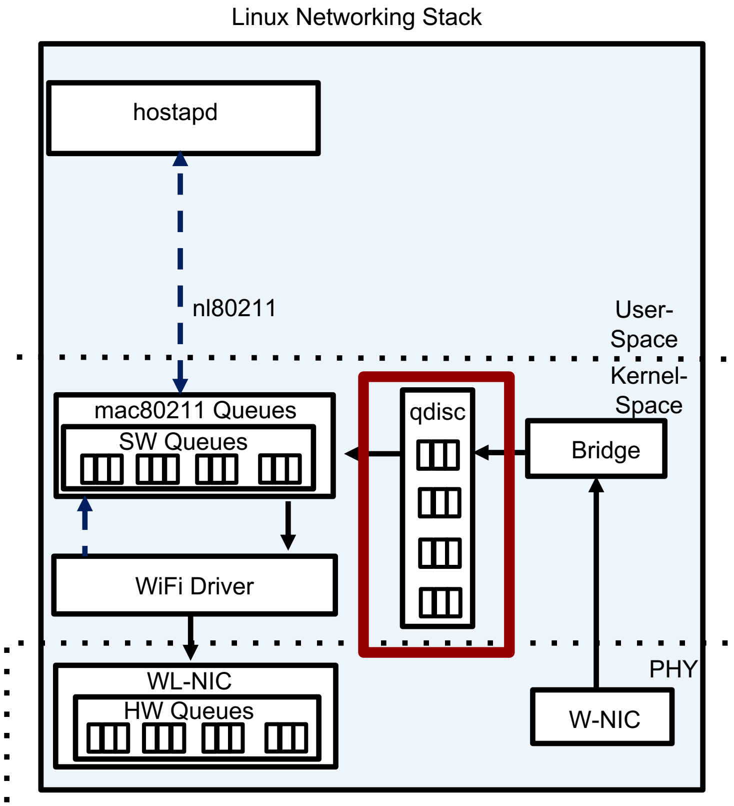




Monitoring the AP's Networking Stack: Delay Analysis

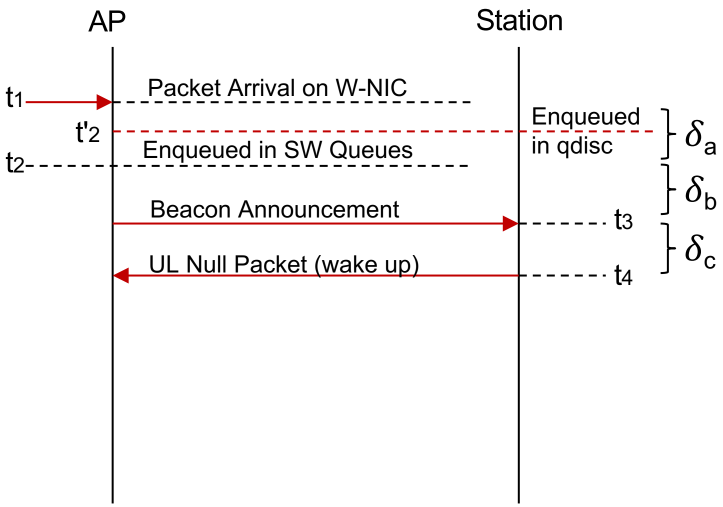
- Queuing disciplines (qdiscs) provide a flexible traffic control framework
- Efficiency of the qdisc layer depends on
  - Packet scheduling method
  - Size of the queues
  - Rate of incoming traffic

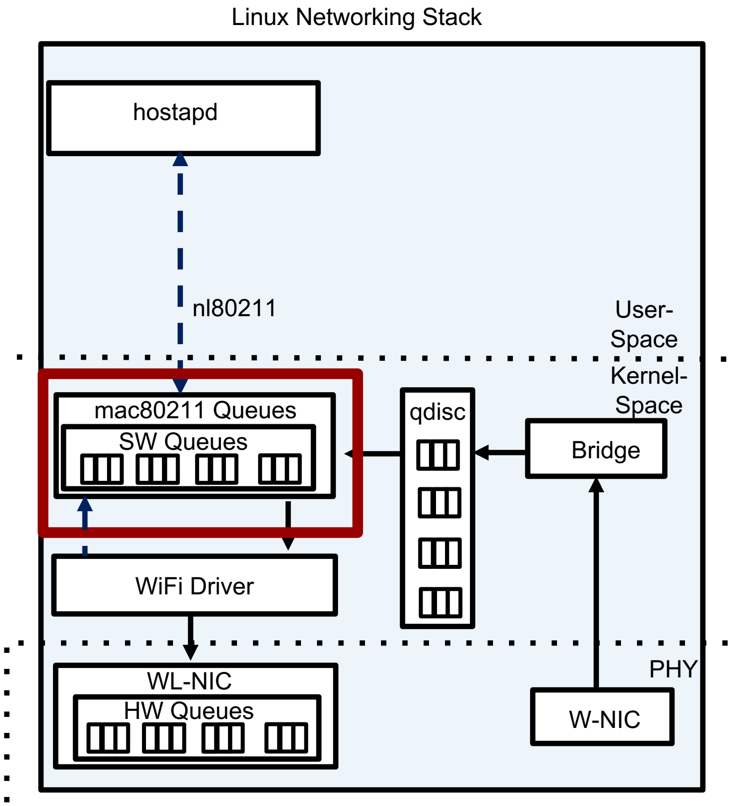




Monitoring the AP’s Networking Stack: Delay Analysis

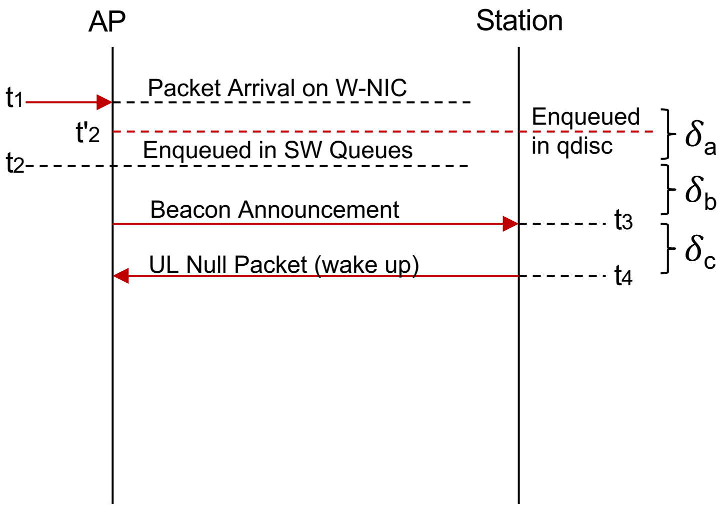
- Provides a unified interface between the driver, qdisc, and hostapd.
- Administers the MLME functions for SoftMAC drivers
  - Building MAC headers
  - Assigning sequence numbers

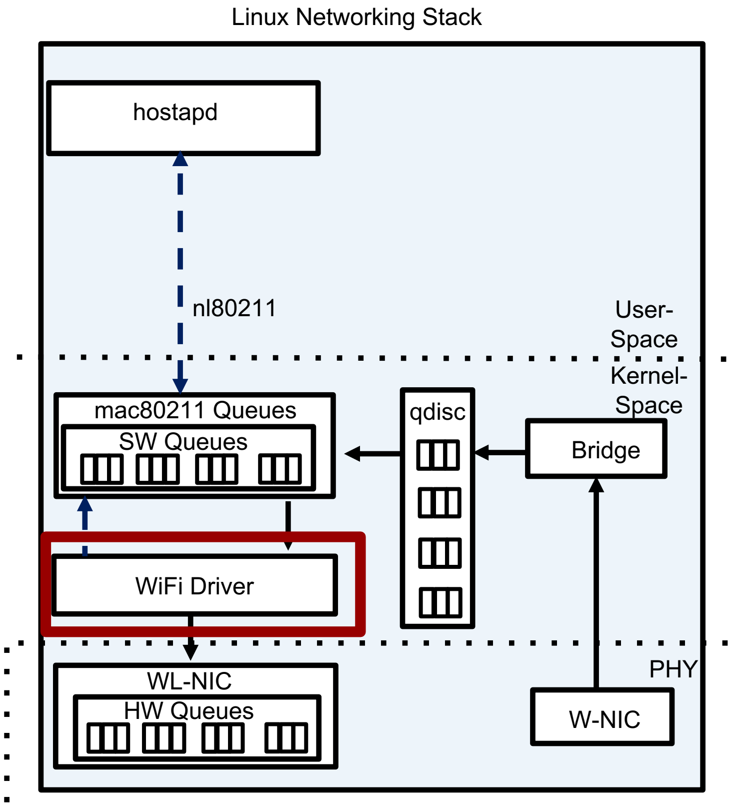




Monitoring the AP's Networking Stack: Delay Analysis

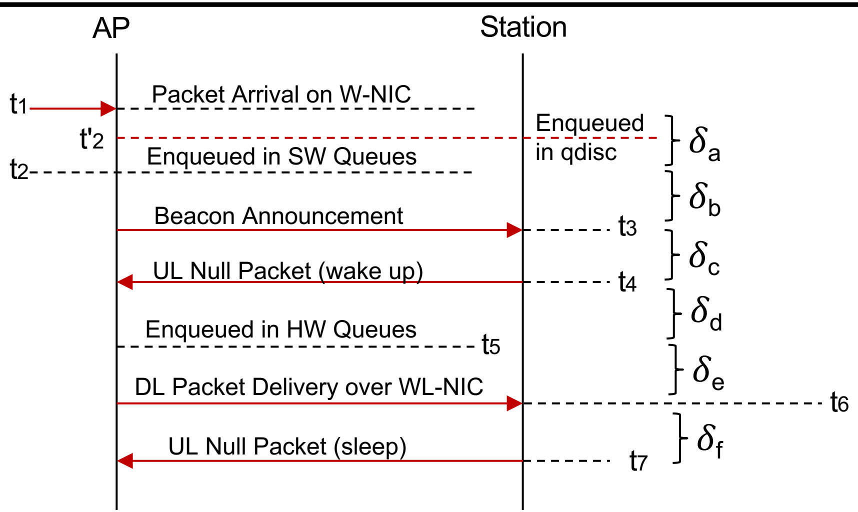
- WiFi driver is responsible for packet aggregation
- Transfers the packets to the WL-NIC for transmission on the channel.

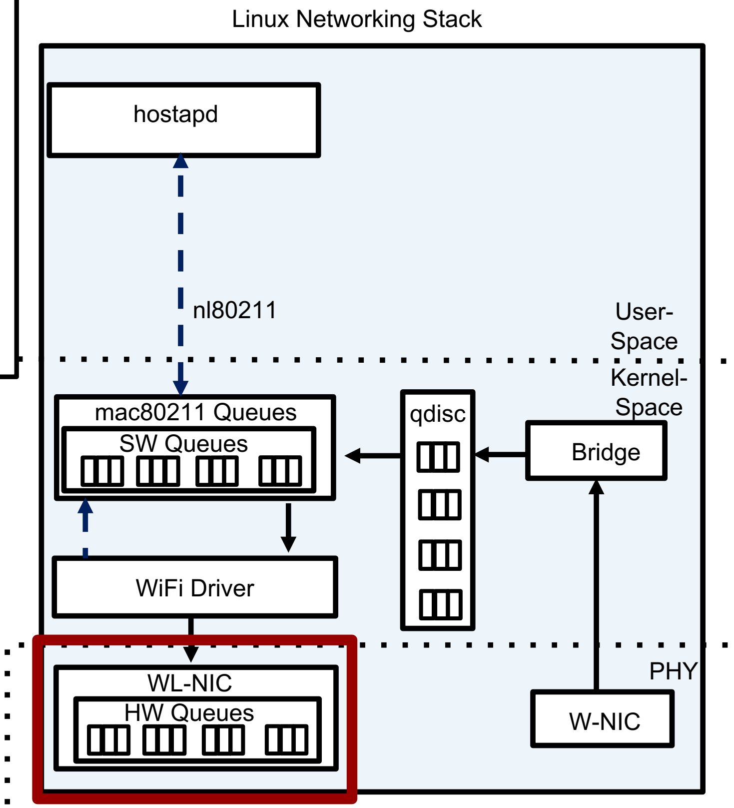




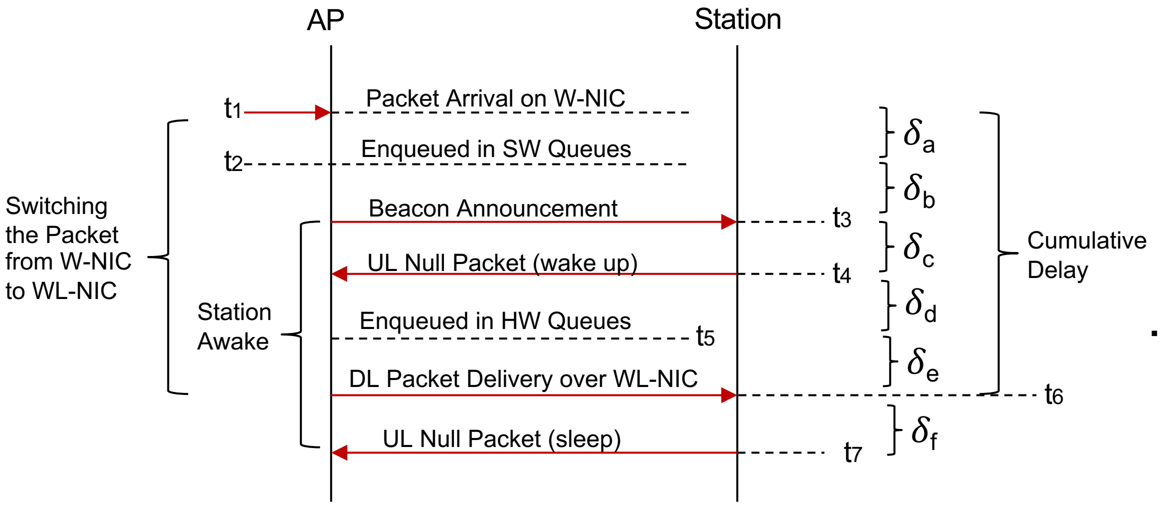
Monitoring the AP's Networking Stack: Delay Analysis

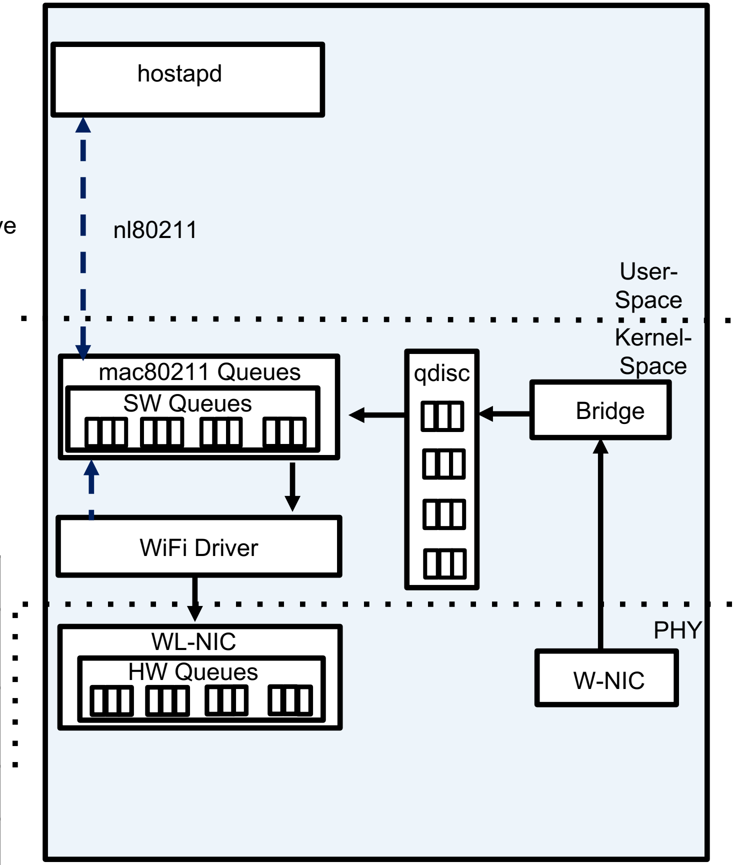
- Responsible for contending for channel access and transmitting a packet
- Includes five hardware (HW) queues
  - Four for each access category (Voice, Video, Best-effort, Background)
  - One for management and control packets
- Each queue contends for channel access
- Once a HW Queue obtains access to the channel for transmission,
  - It transmits the packet to the station and generates an interrupt that triggers the dequeuing of packets from the SW Queues





Downlink Delivery Delay Components





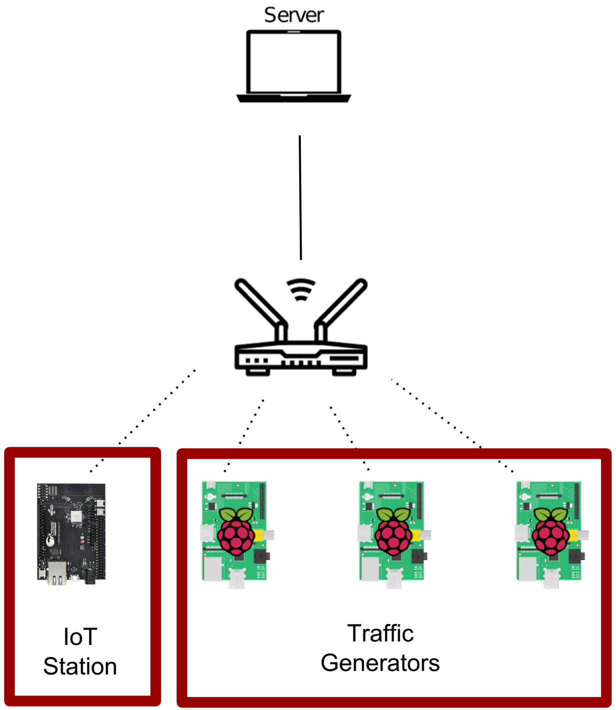
δ_a	Time spent by a packet in qdisc
δ_b	Interval between the insertion of a packet into the SW Queues and the next beacon announcement
δ_c	Time duration between the beacon announcement with TIM bit set and the NULL packet received by the AP
δ_d	Delay incurred in mac80211's SW Queues, when the HW Queues are full
δ_e	Amount of time spent in the HW Queues when contending for channel access

Testbed Setup

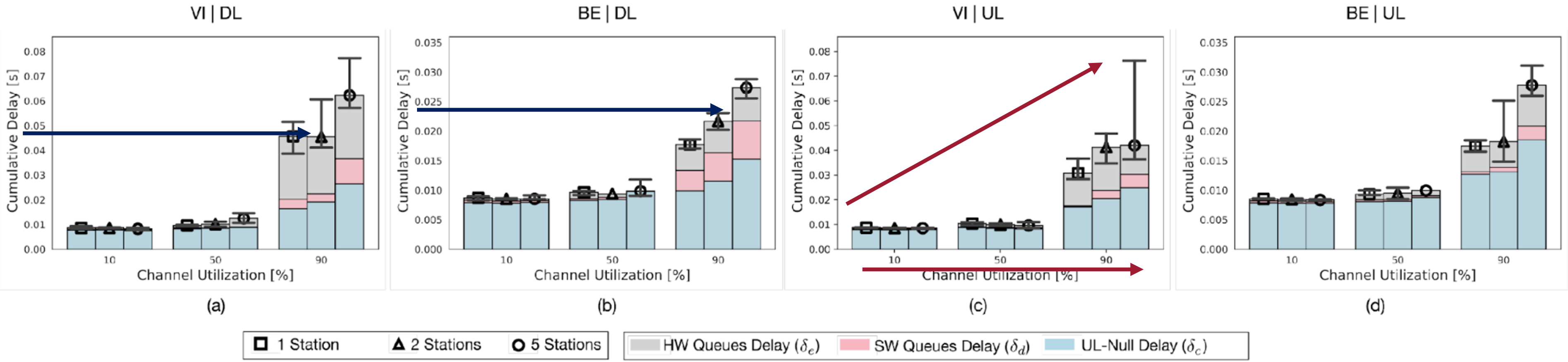
- A workstation (connected to the AP via the wired interface)
  - Sends packets to the IoT Station

- Additional stations (Traffic Generators) introduce concurrent network traffic

- The intensity of the traffic generated => Channel Utilization (CU)

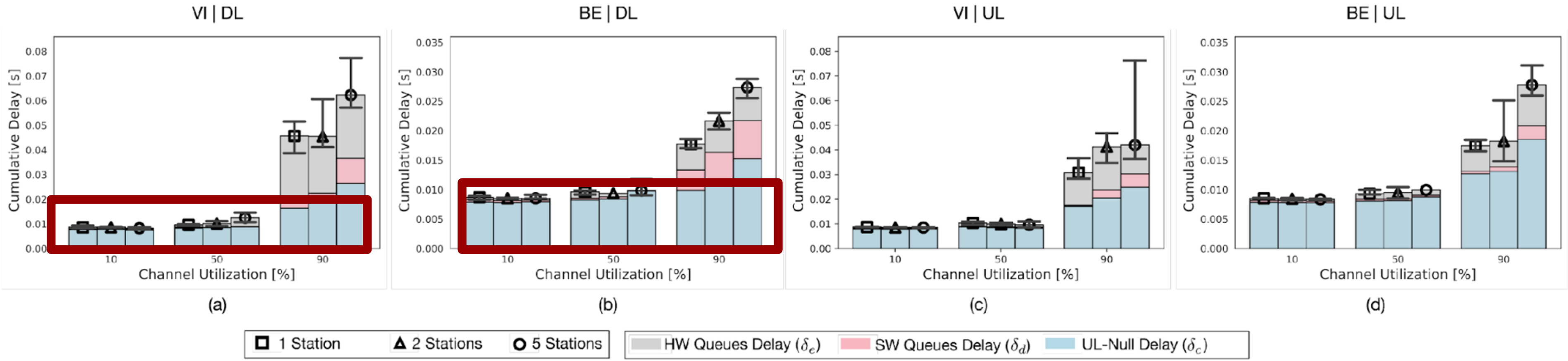


Cumulative Delay: Channel Utilization



- Cumulative delay of IoT traffic increases as CU intensifies.
- However, we observed that for a particular CU intensity, the values of packet delivery delays vary depending
  - on the access category
    - As Figure (b) shows, the median cumulative delay in the presence of 90% best-effort CU is 27 ms
    - As the results of Figure (a) suggest, the cumulative delay with 90% video CU is 61 ms **(i.e., 125% higher)**

Cumulative Delay: NULL Packets



- δc accounts for a significant portion of the total delay
  - Specifically, as channel contention due to UL or DL traffic escalates
- This is because NULL packets are regular data packets belonging to best-effort access category
  - Therefore, they need to contend with concurrent traffic
- Additionally, even when the CU level is low, **Null packets are sent at least 7 ms after the beacon**
  - We observed that this is due to the guard times employed by the station around each beacon reception instance

Energy Monitoring in WiFi Networks

Very accurate measurements, but expensive

1300$

Keysight 34465A

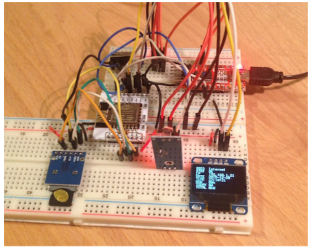
$3500

Keithley 7510

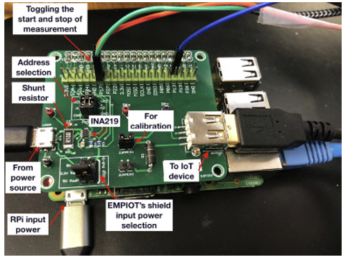
$800

Monsoon power meter

Relatively cheaper, but infeasible to connect to large number of IoT devices



IoT Energy Monitor

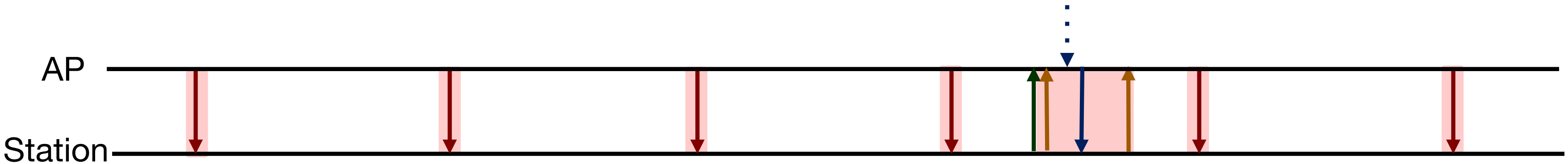


EMPIOT

FLIP utilizes the measurements from the AP's networking stack, hence, does not require any additional hardware

Energy Monitoring in WiFi Networks

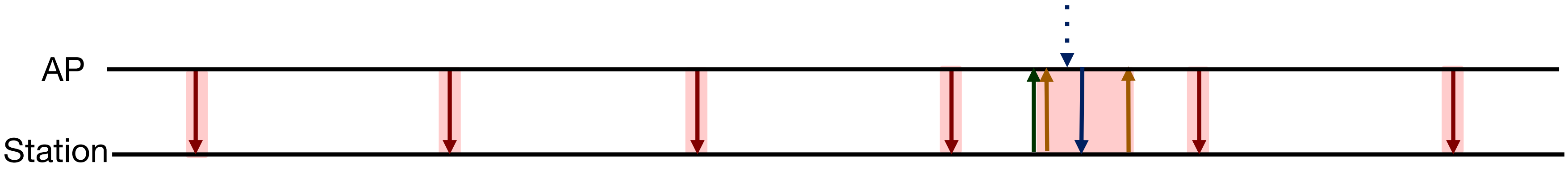
- With PSM,
  - Station sends a **PSPOLL packet to the AP to express its transition into awake mode**
  - until the **more-data field is set to '0'** in the data packet sent by the AP
- With APSM,
  - the station informs the AP about these transitions by the **power-save-bit inside NULL packets**
    - **'0': waking up**
    - **'1': transitioning into sleep state**



FLIP framework, NSM is capable of keeping track of the **timestamp, type, sub-type, direction, and the power-save bit of each packets exchanged with the AP**

Energy Monitoring in WiFi Networks: Beacons

- Stations do not inform the AP when they wake up for beacon reception
  - We **rely on the driver** to obtain **beacon transmission events**
- Additionally, some IoT devices may wake only to receive some beacons (Listen Interval)
  - We obtain the **Listen Interval of each station from hostapd**



Calculating Duty cycle

Total duration spent in awake mode

$$\boxed{\Psi} = (\boxed{d_b} \times \boxed{c_b}) + \boxed{\lambda}$$

Station awake time inferred from power-save-bit and more-data fields

Station's wake-up duration per beacon instance

Number of beacons

Total duration spent in awake mode

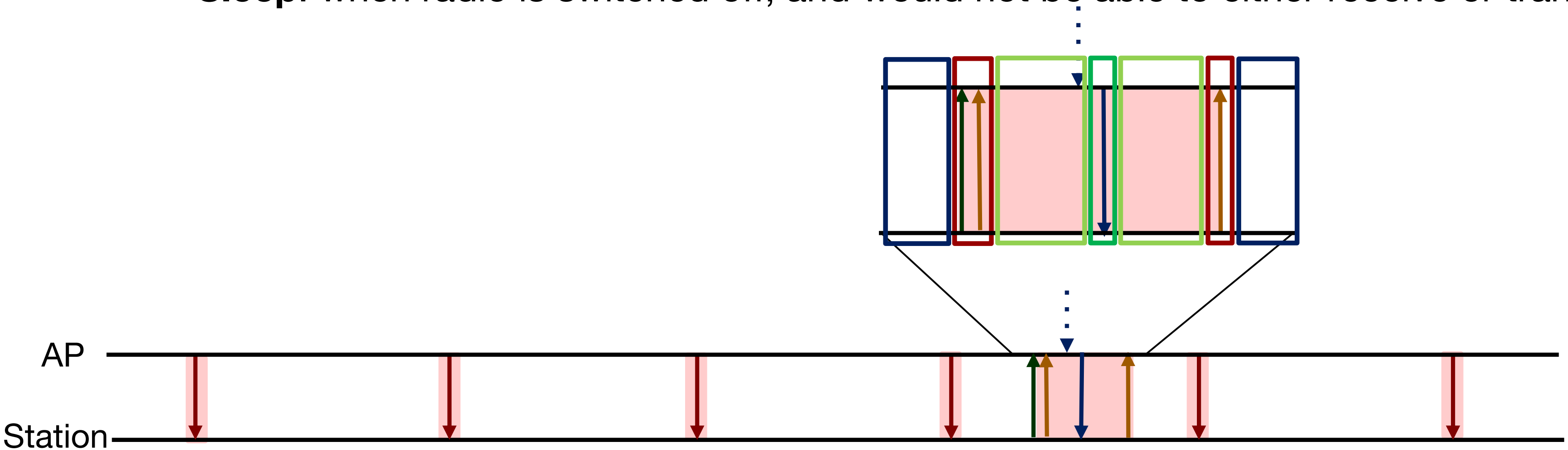
Duty cycle

$$\boxed{\mathcal{D}} = \frac{\boxed{\Psi}}{\boxed{t_n - t_m}}$$

Monitoring interval

Operational Modes

- Radio's operational modes:
  - **Transmission**: when the device is transmitting packets
  - **Reception**: when the device is receiving packets
  - **Idle (a.k.a., idle listening)**: when the device is ready to receive packets
  - **Sleep**: when radio is switched off, and would not be able to either receive or transmit



Operational Modes

- Radio’s operational modes:
  - **Transmission**: when the device is transmitting packets
  - **Reception**: when the device is receiving packets
  - **Idle (a.k.a., idle listening)**: when the device is ready to receive packets
  - **Sleep**: when radio is switched off, and would not be able to either receive or transmit

$$\underbrace{\boxed{\Psi_{tx}}}_{\text{Time during which STA is transmitting packets}} = \sum_{\forall p \in \mathrm{P}} \frac{\boxed{l_p}}{\boxed{r_p}}$$

Length (bits) of packet p

Physical-layer transmission rate (bps) of packet p

Operational Modes

- Radio's operational modes:
  - **Transmission**: when the device is transmitting packets
  - **Reception**: when the device is receiving packets
  - **Idle (a.k.a., idle listening)**: when the device is ready to receive packets
  - **Sleep**: when radio is switched off, and would not be able to either receive or transmit

$$\boxed{\mathcal{E}_{rx}} = \boxed{p_{rx}} \times (\boxed{\Psi - \Psi_{tx}})$$

Energy consumed by a STA while receiving packets

Power consumption of idle and reception modes

Time during which STA is receiving packets or idle listening

Each devices' data-sheet provides the energy consumption of these operational modes

Operational Modes

- Radio’s operational modes:
  - **Transmission**: when the device is transmitting packets
  - **Reception**: when the device is receiving packets
  - **Idle (a.k.a., idle listening)**: when the device is ready to receive packets
  - **Sleep**: when radio is switched off, and would not be able to either receive or transmit

$$\boxed{\mathcal{E}_{sleep}} = \boxed{p_{sleep}} \times \boxed{(t_n - t_m - \Psi)}$$

Energy consumed by a STA while in sleep state

Power consumption of idle and reception modes

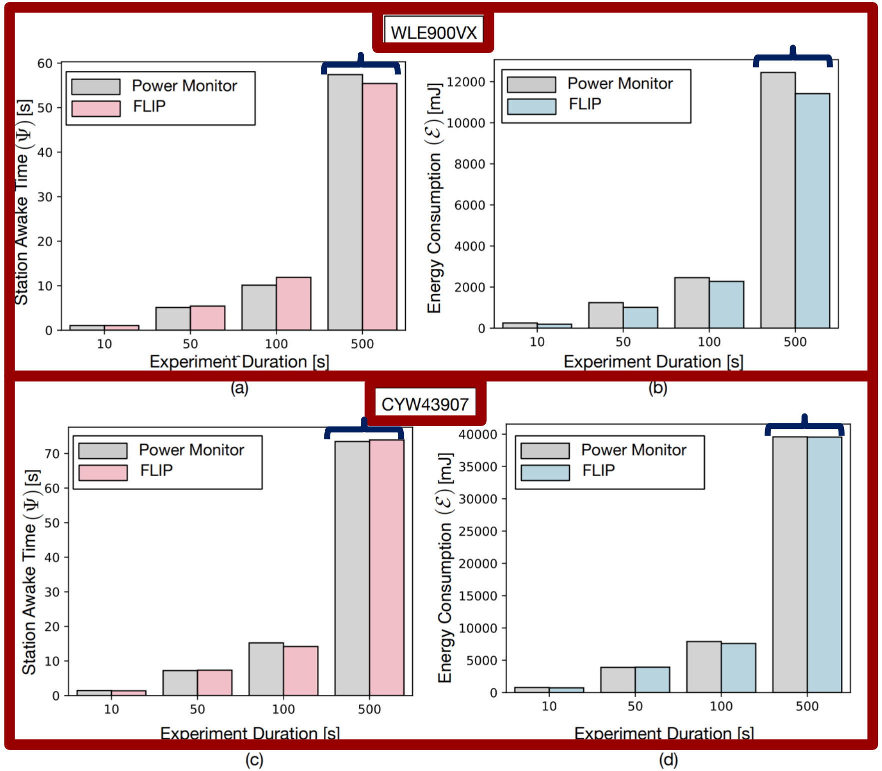
Time during which STA is receiving packets or idle listening

Operational Modes

- Radio’s operational modes:
  - **Transmission**: when the device is transmitting packets
  - **Reception**: when the device is receiving packets
  - **Idle (a.k.a., idle listening)**: when the device is ready to receive packets
  - **Sleep**: when radio is switched off, and would not be able to either receive or transmit

$$\mathcal{E} = \overbrace{\boxed{p_{rx} \times (\Psi - \Psi_{tx})}}^{\text{Energy consumed by a STA while receiving packets}} + \overbrace{\boxed{p_{tx} \times \Psi_{tx}}}^{\text{Energy consumed by a STA while receiving packets}} + \overbrace{\boxed{p_{sleep} \times (t_n - t_m - \Psi)}}^{\text{Energy consumed by a STA while in sleep state}}$$

Calculate Energy Consumption of Associated Clients



- We compare it with commercial power monitoring tool
- Parameters
  - Station awake time
  - Energy consumption
- Robustness
  - Across multiple platforms
  - Over longer monitoring interval
- Measurement error of FLIP is within 9% of the baseline

Conclusion

- We studied the internals of the WiFi networking stack
- We demonstrated how various components of this stack operates
- We then leveraged eBPF to augment WiFi AP to monitor
    - Packet switching delay
    - Energy consumption of stations
- The empirical studies of this paper show how the proposed framework can be used to investigate the operation of WiFi networks.
- FLIP framework can also be easily extended to be used to collect monitoring data and develop methods that react to network dynamics

Jaykumar Sheth
jsheth@scu.edu

Vikram Ramanna
vramanna@scu.edu

Behnam Dezfouli
bdezfouli@scu.edu

References

1. [Online] Statista Research Department, https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
2. Silva, F. A., Ruiz, L. B., Braga, T. R. M., Nogueira, J. M. S., & Loureiro, A. A. F. (2005, August). Defining a Wireless Sensor Network Management Protocol. In *LANOMS* (pp. 39-50).
3. Dezfouli, B., Esmaeelzadeh, V., Sheth, J., & Radi, M. (2018). A review of software-defined WLANs: Architectures and central control mechanisms. IEEE Communications Surveys & Tutorials, 21(1), 431-463.
4. Balazinska, M., & Castro, P. (2003, May). Characterizing mobility and network usage in a corporate wireless local-area network. In Proceedings of the 1st international conference on Mobile systems, applications and services (pp. 303-316).
5. Zhou, M., Sui, K., Ma, M., Zhao, Y., Pei, D., & Moscibroda, T. (2016, June). Mobicamp: A campus-wide testbed for studying mobile physical activities. In Proceedings of the 3rd International on Workshop on Physical Analytics (pp. 1-6).
6. Lyu, F., Ren, J., Cheng, N., Yang, P., Li, M., Zhang, Y., & Shen, X. (2019, May). Big data analytics for user association characterization in large-scale wifi system. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-6). IEEE.
7. Bujnowski, A., Osinski, K., & Wtorek, J. (2017, July). A navigation device utilizing body communication channel for mobile wearable systems. In 2017 10th International Conference on Human System Interactions (HSI) (pp. 25-30). IEEE.
8. Milenkovic, A., Milenkovic, M., Jovanov, E., Hite, D., & Raskovic, D. (2005, March). An environment for runtime power monitoring of wireless sensor network platforms. In Proceedings of the Thirty-Seventh Southeastern Symposium on System Theory, 2005. SSST'05. (pp. 406-410). IEEE.
9. Sun, L., Deng, H., Sheshadri, R. K., Zheng, W., & Koutsonikolas, D. (2016). Experimental evaluation of WiFi active power/energy consumption models for smartphones. IEEE Transactions on Mobile Computing, 16(1), 115-129.
10. Høiland-Jørgensen, T., Hurtig, P., & Brunstrom, A. (2015). The Good, the Bad and the WiFi: Modern AQMs in a residential setting. Computer Networks, 89, 90-106.
11. Pei, C., Zhao, Y., Chen, G., Tang, R., Meng, Y., Ma, M., ... & Pei, D. (2016, April). WiFi can be the weakest link of round trip network latency in the wild. In IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications (pp. 1-9). IEEE.
12. Pei, C., Zhao, Y., Liu, Y., Tan, K., Zhang, J., Meng, Y., & Pei, D. (2017, June). Latency-based WiFi congestion control in the air for dense WiFi networks. In 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS) (pp. 1-10). IEEE.