WiFi Access Point as a Sensing Platform

Milad Heydariaan
Department of Computer Science
University of Houston
milad@cs.uh.edu

Omprakash Gnawali
Department of Computer Science
University of Houston
gnawali@cs.uh.edu

***Abstract*—The growth of the Internet of Things and the trends to deploy more sensors everywhere has led to search for cost effective ways to connect the devices to the Internet. In the most common network architecture for home IoT, the devices use low-power wireless or WiFi to connect to an AP and access the Internet backend. We present a study on the feasibility and efficiency of an alternate network architecture for home IoT. In the proposed architecture, sensors and devices are directly attached to WiFi access points and utilize the computing resources of WiFi APs. We design and implement several sensing applications based on the proposed architecture and report on the performance and limitations of the approach. We find that the proposed architecture allows WiFi APs to become the computational, networking, and storage host for sensing applications without degrading the AP's primary function of providing Internet access to the home users.**

I. Introduction

Although the economy of scale of the devices that comprise the Internet of Things in the homes and cities have made the first generation of smart home and smart city applications viable, the cost barrier of IoT is still slowing its growth. Most of these devices have components providing sensing, computation, memory, and communication capabilities (e.g., Intel Edison, Arduino Uno, TI LaunchPad, etc.). Although a temperature sensor used for indoor climate monitoring may be simple and inexpensive, by the time we add rest of the essentials for a sensing node, not only the cost may become high but also the power and other deployment constraints. Extreme constraints in the four resources mentioned has led to a lot of research projects. For example, power constraint has led to research on energy efficient communication systems across the layers of the stack [8], [5], [16], [4]. Constraints on CPU and memory resources has led to a number of algorithms for data processing. Constraints on communication has led to work on in-network data aggregation [10], [13] and compression [18].

Despite this research, it is still challenging to meet all the resource needs of a sensor application without introducing tradeoff on one of the resource dimensions. Using a fast CPU or a large memory on the sensor platform will increase the cost and power footprint. If the platform requires large amount of power, the node cannot be powered using a battery and may need to be line-powered. In most sensor deployment scenarios providing a power source to the sensor can be very challenging due to limitations of power infrastructure.

While the quest for universal sensing platform for home continues, the industry is offering a large number of customized sensing and control platforms and applications. Most of these home sensing and control have two architectures: (1) the end devices are equipped with 900 MHz radio using 802.15.4. These devices connect to a custom gateway with these radios and the gateway connects to the WiFi AP using a cable. (2) the end devices are equipped with 2.4 GHz radio using 802.11 and they directly communicate with the WiFi AP. The second architecture is not efficient due to the power footprint of the sensor nodes when they have WiFi chipset onboard. The first architecture requires additional gateway device at home. For a sensing application with a large number of sensors, especially when we need to deploy sensors certain locations, these architectures may be the right choice. For simple sensing application with just one or a few sensor or control devices, a simpler architecture may be more desirable.

Our solution of a simpler architecture is to use WiFi AP as a sensing platform. WiFi AP has sufficient computational, memory, networking, and power resources. Many APs also have peripheral interconnect interfaces to connect a sensor. Thus, the WiFi APs can play a more direct role to collect, process and forward the data. The main advantage of this approach is we eliminate other intermediate devices or additional equipments such as power and computation resources and this makes the sensor deployment simpler and more cost effective. Additionally, by scavenging a huge wasted computation resource, we use WiFi APs to do more than just a home WiFi AP for users to surf the web.

In order to effectively and safely use WiFi AP as a sensing platform, we need to find the opportunities and limitations of the system by studying the sensing application and WiFi performance. In our study we describe how the proposed architecture simplifies the design and makes the sensor deployment cost effective. We implement real-world sensor applications in our system. We also describe the limitations of the approach by answering the following questions:

- In which scenarios and applications we can use AP as a sensor platform?
- Which type of sensors can we integrate with a WiFi AP?
- How many sensors can we integrate with a WiFi AP?
- Can we use a WiFi AP both as a mean to access the Internet and a sensor platform at the same time?
- How much computation power is wasted on a WiFi AP and how much can we scavenge?

Although the maker community has integrated devices with

OpenWRT APs, to the best of our knowledge there is no specific study on WiFi AP to show how specific tasks can interfere with each other with respect to resource sharing. In our study, we try to clarify how a WiFi AP may perform its primary service of Internet connectivity while also offering secondary service of sensing platform.

Our main contributions are:

- We survey the resources of APs available in the market to explore the capability of integrating various sensors to a WiFi AP.
- We explore the scenarios in which using WiFi AP as a sensing platform is reasonable by studying the limitations that our solution introduces for sensor deployment.
- We study the performance issues and resource interference when WiFi APs serve their primary function of Internet access and secondary function as a sensor platform.
- We evaluate the idea of WiFi AP as a sensing platform using real APs and physical prototyping of typical sensor applications integrated with the AP.

II. Related Work

A. WiFi AP-based Systems and Applications

Most of the previous studies came from hacking and maker community and proved the feasibility of various applications. WiFi APs like Fritz!Box [1] and many other similar products have already developed functionalities to re-use the AP resources for other applications. Chen et al. [6] and Ha et al. [9] tried to fill the gap between sensor networks and the Internet by implementing a gateway using OpenWRT, while others proposed and implemented various applications of sensor networks on top of OpenWRT. Kciuk [11] implemented two applications, mobile robot and building automation. Kim and Kim [12] implemented a home lighting control using an OpenWRT router and Arduino, then using a mobile phone they tried to control the home lighting system through the router. Rettig et al. [17] and Zinas et al. [21] build a bigger scale system, a geospatial sensor network and a meteorological sensor equipped with GPS, using an OpenWRT-installed router.

B. Commonly Used Embedded Computers

BeagleBone Black and Raspberry Pi are popular embedded computers with several GPIO pins and support different Linux distributions. Arduino is suitable for real-time applications but it has limited resources. Intel Edison is a newly introduced embedded computer with higher benchmark rank compared to its competitors [3]. LinkSprite pcDuino has the same capabilities of RPi but it can run Android ICS. Most of the above platforms are widely used as sensor platforms.

C. Performance Isolation and Resource Sharing

Ouyang et al. [15] presented a co-kernel that provides performance isolation for multiple OSs. Chiang et al. [7] proposed a scheduling system consist of three parts to mitigate the process interference: prediction of interference, scheduling and resource monitoring. Zhu and Tung [20] evaluated a scheduling model for the cloud which has the capability of predicting the impact of application performance in presence of resource interferences. Meng et al. [14] talked about a trustworthy cloud state monitoring which tries to prevent interference across different nodes.

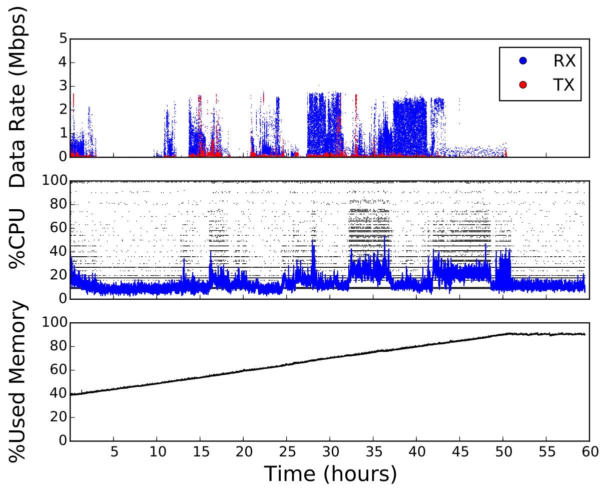


Fig. 1: 60 hours of resource monitoring on a home WiFi AP. The usage includes video streaming and web browsing. The results show that the CPU is idle most of the time.

TABLE I: Percentage of APs in the market that provide the given I/O capability.

USB	Serial	JTAG	Available GPIO
100%	85%	50%	20%

III. A Reality Check

We do a market survey on WiFi APs to find out if today's WiFi APs can run realistic sensing applications. Table II describes the required resources for a typical sensor deployment scenario, while Table III shows the requirements for different sensor applications. One big challenge is to provide all these resources, especially under different deployment scenarios. Furthermore, as shown in Fig. 1 we observed that a regular WiFi AP in home WiFi scenarios, including video streaming, is idle most of the time, meaning that the system resources are not completely utilized. According to Strategy Analytics, by the end of 2015 there were more than 521 million households using wireless APs to connect to the Internet [19]. We did a survey on the OpenWRT supported devices using the list of devices available on OpenWRT website to understand what capabilities popular WiFi APs have. We summarize the list of OpenWRT supported devices into 20 devices by choosing from different vendors. Most of WiFi APs have serial interface available for users to attach additional devices to the WiFi AP. According to Fig. 2 and Table I, most of the sensor deployment needs can be satisfied using a typical WiFi APs available in the market as we can connect a wide range of sensors directly to a WiFi AP.

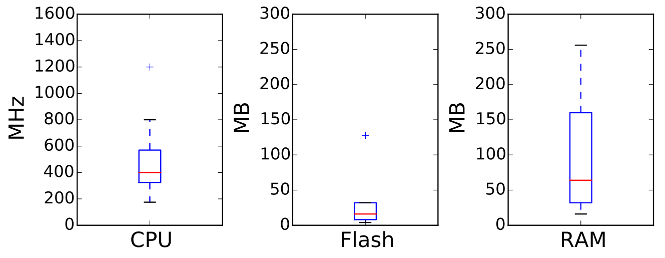


Fig. 2: Results of a survey on CPU, Flash, and RAM specification on OpenWRT supported WiFi APs in the market.

TABLE II: Sensor Deployment Requirements

Essential Resource	Description
Power Source	A power source for the sensor device
Processor	A processing unit to run essential commands and to process the data
Memory	A temporary storage for the collected data and a workspace for the processor
Long-term Storage	A place to store the data for a long time
Network Connectivity	To make the sensor platform accessible remotely and to transfer data to another server and for storage or processing

IV. System Design and Implementation

In this section, we present an overview of how we design the sensing platform on WiFi APs.

A. Overview

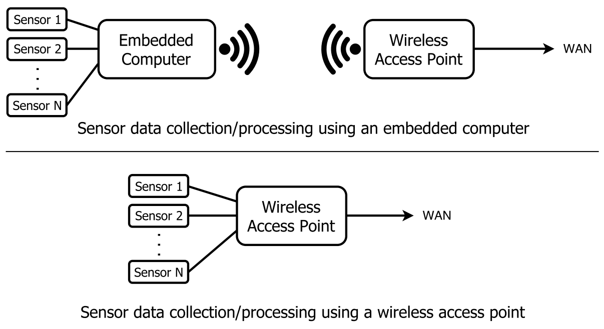
Designing and implementing a sensing platform on WiFi AP allows us to attach sensors directly to the WiFi AP without any other intermediate embedded computers as shown in Fig. 3. The WiFi AP is equipped with a software platform capable of running user space programs to communicate with digital interfaces of the WiFi AP, to collect data from the directly attached sensors and to send control commands to the sensors. The WiFi AP is also capable of processing the collected data and to forward the data to the cloud or a remote server through WAN interface.

Our experiment platform is a WiFi AP with OpenWRT installed as the firmware operating system. At least there are 1250 routers listed by the OpenWRT as the supported devices which shows the wide use of OpenWRT.

The WiFi AP is responsible for network connectivity (WLAN) and running various data collection, data processing

TABLE III: Requirements for Different Applications

	The QCN	Intrusion Detection	Monitor Indoor Climate	Indoor AQM	Fire Alarm System
Power Source	Low	Low	Low	Low	Low
Processor	Medium	High	Low	Low	Low
Memory	Medium	Large	Small	Medium	Small
Data Storage	V. Large	V. Large	Small	Medium	Small
Network Connectivity (Throughput)	Low	Medium	Low	Low	Low



Sensor 1
Sensor 2
Sensor N
Embedded Computer
Wireless Access Point
WAN
Sensor data collection/processing using an embedded computer
Sensor 1
Sensor 2
Sensor N
Wireless Access Point
WAN
Sensor data collection/processing using a wireless access point

Fig. 3: The diagram illustrates the difference between using a WiFi AP as the sensing platform with classic method of data collection, processing and forwarding.

and data forwarding tasks. In multiple applications and experiments we connect multiple sensors to the WiFi AP through USB port. We monitor system resources on WiFi AP while doing all these tasks as a part of our experiments. We also connected a laptop to the WiFi AP through WiFi to initiate network traffic and to run the measurement scripts.

B. WiFi Access Point

WiFi AP - Buffalo Router: We used Buffalo WZR-HP-G300NH2 (Atheros AR7242@400MHz processor, 64MB RAM, 32MB flash) with OpenWRT as our experiment platform. This USB Serial interface on the AP facilitates communication with sensors.

OpenWRT: OpenWRT is an open source embedded operating system based on GNU/Linux, supporting more than 1250 different devices and one of the most popular WiFi AP OSs. In our experiments we used Attitude Adjustment (12.09) and Chaos Calmer (15.05) versions.

C. Sensors Connected to the WiFi AP

We connect multiple sensors to Arduino Uno using GPIO pins, running I2C, SPI or Serial protocol depending on the sensor. Then, we connect the Arduino Uno device to the WiFi AP. This configuration is an instantiation of the proposed architecture and used only as a rapid prototyping platform. From the perspective of the WiFi AP, Arduino Uno is accessed as the /dev/ttyX file system, similar to how a directly connected USB sensor device would appear to the system. In the real-world scenarios, sensors can be connected directly to the WiFi AP without Arduino.

Using this approach, we integrated the following sensors to the AP to demonstrate the feasibility of implementing real-world applications and to study how they interfere with primary use of the WiFi AP.

- **Accelerometer**: We used LSM303 3-axis accelerometer with 400 Hz sampling rate implement an instance of the Quake-Catcher Network (QCN) [2], a real-world application.
- **PIR (Motion) Sensor**: Using a motion sensor, capable of detecting motions through interception of the warmth of

the body, we implemented occupancy detection, surveillance, and intrusion detection systems.

- **USB Camera**: We connected a USB video class camera to the WiFi AP to implement an application of surveillance and intrusion detection.
- **Temperature/Humidity Sensor**: We used AM2302, a temperature and humidity sensor, to implement indoor climate monitoring application.
- textbfPM2.5 Sensor: We implemented an air quality monitoring system using PM2.5 Optical Dust Sensor (GP2Y1010AU0F).
- **Gas Sensors**: We used CNG (MQ-4), LPG (MQ-6), CO (MQ-7), and Hydrogen (MQ-8) sensors to implement indoor climate monitoring and alarm systems.

V. Evaluation

Next we evaluate the effectiveness of the proposed approach to use WiFi AP as a sensing platform.

A. Metrics

Feasibility: We measure program size and difficulty of implementation of sensor applications on the APs.

Network and Device I/O Performance: We measure network throughput, device I/O performance as read/write speed and round-trip latencies.

System Resources: We use CPU and memory utilization to characterize the impact of using AP as the sensing platform.

We do not consider energy consumption in this study but consider how many sensor devices the AP can support.

B. Measurement Tools

Network Performance: We use iPerf 3.0.1 and GNU Wget 1.16.1 for network performance measurements. We configure Wget to download 50KB files. We configure ping to use 56 bytes of payload for RTT measurement.

System Resources Monitoring: We used "top", a tool on OpenWRT to monitor CPU and memory utilization every second on OpenWRT. We created Ardunio programs and Python scripts to measure I/O metrics of our study.

C. Feasibility of Real-world Applications of Sensor Networks

To test the feasibility of running real-world sensor network applications on WiFI APs, we implemented five applications on the WiFi APs and monitored their performance. Results from Table IV indicate that implementation of real-world sensor applications on a WiFi AP is practical.

The Quake-Catcher Network: By using an accelerometer connected to Arduino Uno, we implemented an instance of the USB sensor used by the QCN. This represents how a WiFi AP can turn into a data collection node for the QCN.

Intrusion Detection and Surveillance: We connected a USB camera and a motion sensor to the WiFi AP to build a room surveillance system streaming video continuously and detecting motions. The WiFI AP processes the video in real-time. A motion triggers video recording and background subtraction across the frames.

TABLE IV: Feasibility of Real-world Sensor Applications

	Program Size	**Difficulty of Implementation**
Surveillance	1870 bytes	Medium
Indoor Climate Monitoring	1140 bytes	Easy

Indoor Climate Monitoring: We connected a temperature and humidity sensor to Arduino Uno then to the WiFi AP. The Arduino Uno sends the sensor data to OpenWRT over serial interface. Another program on OpenWRT, collects and forwards the data to a remote server.

Indoor Air Quality Monitoring and Occupancy Detection: By connecting CO detector, temperature, humidity, PM2.5 and PIR motion sensors to Arduino Uno, we implemented an indoor climate and occupancy monitoring system. The AP then forwards sensor data to a remote server.

Fire Alarm System: The CO sensor and a PM2.5 sensor connected to Arduino Uno sends sensor data to the AP. A Python script on the AP triggers the alarm when density of smoke is higher than a threshold.

D. Number of Sensors

One concern about the proposed architecture is scaling in terms of number of physical sensors supported per AP. In one experiment, we connected 10 Arduino Uno devices equipped with AM2302 humidity/temperature sensors to a WiFi AP using 2 passive USB hubs and collected data from those sensors, with no external power other than the one powering the WiFi AP. The AP successfully collected and processed sensor readings from each Arduino Uno at 1 Hz. We believe the system can support even more sensors per AP if necessary.

E. Impact of Resource Sharing on Network Performance

We study how the AP's network handling capability is impacted as it provides different resources to the sensing application under these scenarios:

No Data Processing: Baseline, normal operation of AP.

High Memory Usage: We run a user space program that allocates up to 80% of memory increasingly.

High CPU: We run a program that has an infinite loop causing 100% CPU utilization.

High CPU and High Memory Usage: This scenario runs the previous two programs at the same time causing high memory and CPU load on the AP.

High Volume of I/O Interrupt Requests: We created a data flow with maximum data rate possible (115200 bps) with our platform over WiFi AP's USB serial interface.

Results of our experiments suggest that the network performance is not affected by our sensor data collection and data processing. Fig. 4 and 5 show no significant transition between the two cases of not using any resources on the WiFi AP by user space programs and the other cases of excessive use of system resources. Figure 6 also supports this conclusion: use of excessive system resources has no effect on the network throughput. We observed no impact even when

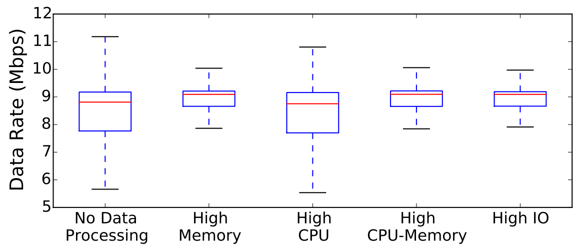


Fig. 4: Throughput (data rate) seems not affected by the excessive use of CPU, memory and I/O

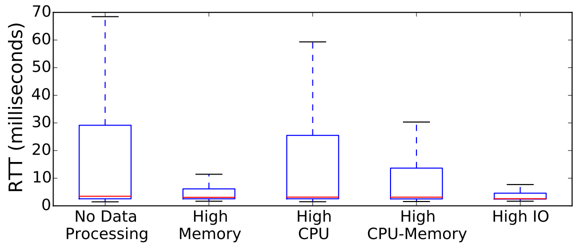


Fig. 5: Delay (RTT) seems not affected significantly by the excessive use of CPU, memory and I/O

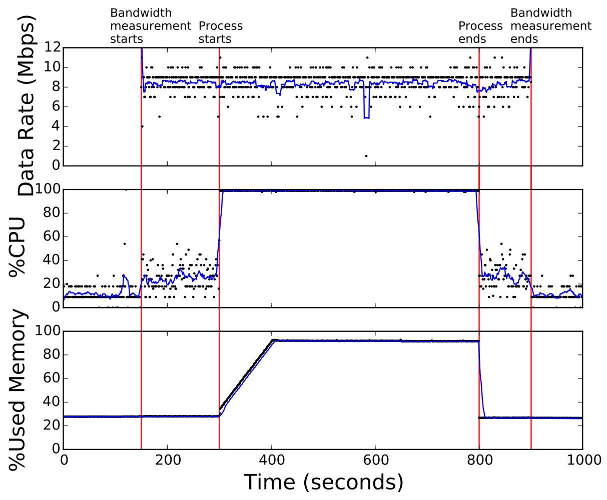


Fig. 6: System resources and throughput while utilizing 100% of CPU and consuming memory with a small linear growth and limited by a cap. No impact on throughput can be observed.

the AP was doing Full HD video streaming. Thus the network performance is not affected by sensor data processing, likely due to performance isolation provided by the operating system.

F. Impact of Network Traffic on Data Processing

We study how AP network traffic impacts sensor data processing latency on the AP. We ran a simple background subtraction algorithm on static frames of a video stored on the AP disk. We measured the execution time of this process in two scenarios, with and without presence of network traffic. Fig. 7 shows a considerable increase in execution time when AP is handling network traffic. Thus, the proposed architecture may not be appropriate for applications that require real-time CPU-intensive sensor data processing.

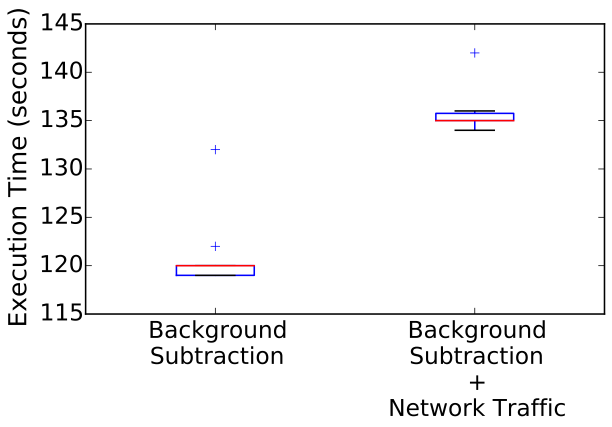


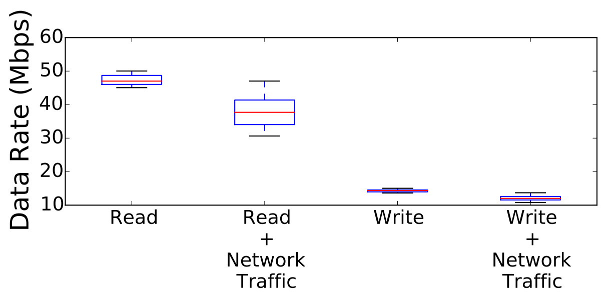
Fig. 7: Video processing takes longer when the AP handles network traffic because OpenWRT gives higher priority to the network operations over a long running process.

G. Impact of Network Traffic on Device I/O Performance

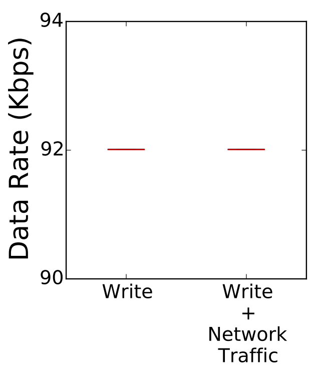
We design an experiment to study if WiFi AP handling excessive network traffic impacts device (e.g. interface to the sensors) I/O performance. We compare the I/O performance in two scenarios, without any network traffic and with excessive network traffic by downloading large files from a laptop connected to the AP. We repeated the experiments for two different types of I/O operations, USB mass storage I/O and serial interface I/O measuring read and write speed by reading or writing several bytes from or to the disk/Arduino. To measure delay (RTT), the AP sends a packet of unit size (1 byte) to Arduino and reply back the same byte to the AP. Fig. 8a shows that the network traffic degrades the device I/O performance slightly, but Fig. 8b shows that it is not true for Serial communication, because of lower throughput of the Serial interface compared to USB disk. Fig. 8c also shows that the network performance has no impact on Serial communication delay.

VI. Discussion of Limitations

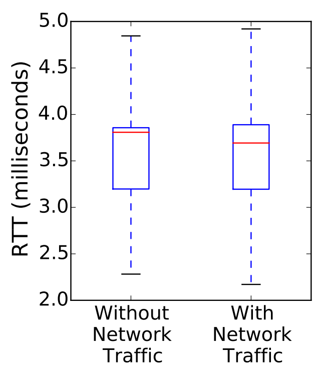
A WiFi AP at home is not necessarily deployed in places where the sensors need to be placed for a sensing application. In such cases, long cables may be necessary to connect the sensors that are placed in strategic sensing locations to the WiFi AP. This is often unwieldy. Most consumers do not have the skill to update AP firmware to support sensing applications. The AP manufacturers could facilitate by pre-installing sensing software on the APs. Our performance measurements were focused on network level metrics. Future work could also investigate the impact of sensing on latency-sensitive application performance.



(a) USB disk I/O performance



(b) Arduino Serial I/O performance



(c) Arduino Serial I/O delay

Fig. 8: Impact of network traffic on I/O performance. Network traffic degrades the USB disk IO performance but it has no impact on Serial communication performance because the Serial interface has lower throughput compared to USB flash disk.

VII. Conclusions

A WiFi AP has lower amount of resources compares to the other sensing platforms such as Raspberry Pi and BeagleBone Black. This limits the software development on a WiFi AP to require the developer to use cross- compilation technologies instead of compiling directly on the AP. Moreover, some applications may run slower on a WiFi AP compared to other platforms. We observed the network performance is not affected by system performance metrics of CPU, memory and I/O usage; On the other hand, presence of network operations affects the performance of data processing applications in which they require excessive use of CPU and memory.

VIII. Acknowledgment

We thank Shengrong Yin for technical help and Profs. Christoph Eick, Weidong Shi, and Yunsoo Choi for discussions in the context of a DHS-funded project under contract no. D14PC00202. This work was partially supported by the National Science Foundation under grant no. IIS-1111507.

References

[1] Avm fritz!box. http://en.avm.de/products/fritzbox/.

[2] The quake-catcher network. http://qcn.stanford.edu.

[3] Single board computer benchmarks. https://learn.sparkfun.com/tutorials/single-board-computer-benchmarks.

[4] M. Buettner, G. Yee, E. Anderson, and R. Han. X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. In *SenSys 2006*.

[5] P. Buonadonna and G. Tolle. MultihopLQI. Available at: http://www.tinyos.net/tinyos-1.x/tos/lib/MultiHopLQI, 2004.

[6] Y.-C. Chen, C.-C. Chuang, R.-I. Chang, J.-S. Lin, and T.-C. Wang. Integrated wireless access point architecture for wireless sensor networks. In *ICACT 2009*.

[7] R. C. Chiang and H. H. Huang. Tracon: interference-aware scheduling for data-intensive applications in virtualized environments. In *ICHPC-NSA 2011*, page 47. ACM, 2011.

[8] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. In *SenSys 2009*.

[9] M. Ha, S. H. Kim, H. Kim, K. Kwon, N. Giang, and D. Kim. Snail gateway: Dual-mode wireless access points for wifi and ip-based wireless sensor networks in the internet of things. In *CCNC 2012*.

[10] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A scalable and robust communication paradigm for sensor networks. In *MOBICOM 2000*.

[11] M. Kciuk. Openwrt operating system based controllers for mobile robot and building automation system students projects realization. In *REM 2014*.

[12] C. G. Kim and K. J. Kim. Implementation of a cost-effective home lighting control system on embedded linux with openwrt. *Personal and ubiquitous computing*, 18(3):535–542, 2014.

[13] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *OSDI '02*.

[14] S. Meng, A. K. Iyengar, I. M. Rouvellou, L. Liu, K. Lee, B. Palanisamy, and Y. Tang. Reliable state monitoring in cloud datacenters. In *CLOUD 2012*.

[15] J. Ouyang, B. Kocoloski, J. R. Lange, and K. Pedretti. Achieving performance isolation with lightweight co-kernels. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing*, pages 149–160. ACM, 2015.

[16] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *SenSys 2004*.

[17] A. J. Rettig, S. Khanna, D. Heintzelman, and R. A. Beck. An open source software approach to geospatial sensor network standardization for urban runoff. *Computers, Environment and Urban Systems*, 2014.

[18] C. Sadler and M. Martonosi. Data compression algorithms for energy-constrained devices in delay tolerant networks. In *SenSys 2006*.

[19] D. Watkins. Global broadband and wlan (wi-fi) networked households forecast 2010-2019, 2015.

[20] Q. Zhu and T. Tung. A performance interference model for managing consolidated workloads in qos-aware clouds. In *CLOUD 2012*.

[21] N. Zinas, S. Kontogiannis, G. Kokkonis, and C. Pikridas. A novel microclimate forecasting system architecture integrating gps measurements and meteorological-sensor data. In *BCI*. ACM, 2013.