An Early Resource Characterisation of Wi-Fi Sensing on Residential Gateways

Min, Chulhong; Alloulah, Mohammed; Kawsar, Fahim

DOI
10.1145/3276774.3276789

Publication date
2018

Document Version
Final published version

Published in
BuildSys'18

An Early Resource Characterisation of Wi-Fi Sensing on Residential Gateways

Chulhong Min
Nokia Bell Labs
Cambridge, United Kingdom
chulhong.min@nokia-bell-labs.com

Mohammed Alloulah
Nokia Bell Labs
Cambridge, United Kingdom
mohammed.alloulah@nokia-bell-labs.com

Fahim Kawsar
Nokia Bell Labs and TU Delft
Cambridge, United Kingdom
fahim.kawsar@nokia-bell-labs.com

ABSTRACT

Recent research has successfully shown brand new models with Wi-Fi signals explaining space dynamics, assessing social environments, and even tracking people's posture, gesture and emotion. However, these models are seldom used in real execution and operating environments, i.e., on residential gateways with networking tasks. In this paper, we present the first, albeit preliminary, measurement study of common Wi-Fi sensing models on a residential gateway. This investigation aims to understand the performance characteristics, resource requirements, and execution bottlenecks for Wi-Fi sensing when being used in parallel with communication tasks. Based on our findings, we propose two optimisation techniques - i) dynamic sampling and ii) dynamic planning of inference execution - for optimum Wi-Fi sensing performance without compromising the quality of communication service. The results and insights lay an empirical foundation for the development of optimisation methods and execution environments that enable sensing models to be more readily integrated into next-generation residential gateways.

CCS CONCEPTS

• **Information systems → Information systems applications**;

KEYWORDS

Wi-Fi Sensing, Residential Gateways, Runtime Adaptation

ACM Reference Format:
Chulhong Min, Mohammed Alloulah, and Fahim Kawsar. 2018. An Early Resource Characterisation of Wi-Fi Sensing on Residential Gateways. In *The 5th ACM International Conference on Systems for Built Environments (BuildSys '18), November 7–8, 2018, Shenzen, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3276774.3276789

1 INTRODUCTION

Recent compelling research has reimagined a commodity Wi-Fi device as a multi-purpose sensor capable of turning radio signals into a rich source of computational information explaining space dynamics, assessing the social environment and even tracking people's posture, gesture and emotion [1, 2, 4, 6–9]. A common intuition is that multi-path Wi-Fi signals emitted from a transmitter are affected by human body and movement before reaching a receiver, i.e., human body reflects, scatters, and blocks the signals. The signal patterns can be learned to model a variety of motion-based human activities. Conventional approaches seek to associate these signal patterns acquired from channel state information (CSI) with human activities through training classifiers on top of often bespoke features, e.g. statistical distributions and Doppler variations.

Although these approaches demonstrated the potential of Wi-Fi sensing in a brand new class of applications, they are seldom used in real execution environments, i.e., on a residential gateway, and in real operating environments, i.e., when the gateway is equipped with networking tasks. We argue for unleashing the true potential of Wi-Fi signals as a general-purpose human sensing modality. We need to turn our attention to developing a sound understanding of runtime behaviour of these models on a residential gateway, the natural, economically practical, and privacy-preserving execution environment of these class of models. One may argue for leveraging the cloud server to process the CSI data, but it is not practical considering the data volume; e.g., 0.54 million numbers are generated every second at 1 kHz of CSI sensing (See Section 2.1 for the details.) It also naturally brings the privacy issues.

To this end, in this paper, we present a measurement study of common shallow and deep Wi-Fi sensing models for human occupancy and physical activity detection on a representative residential gateway platform. We systematically explore the runtime behaviour of the platform under various communication and sensing workloads to uncover the performance characteristics, resource requirements, and execution bottlenecks for Wi-Fi sensing models.

Based on our findings, we propose two optimisation techniques for Wi-Fi sensing on commodity residential gateways. The first technique dynamically shapes the sampling rate of the sensing pipeline based on resource utilisation, leveraging a combination of sparse and downsampling mechanism. The second technique dynamically schedules the inference execution to achieve optimum Wi-Fi sensing performance without compromising the quality of communication service. The results and insights of this study offer an empirical foundation for the faster integration of optimised Wi-Fi sensing pipeline into next-generation residential gateways.

2 STUDY PRELIMINARIES

2.1 Wi-Fi Sensing Primer

As depicted in Figure 1, the Wi-Fi sensing model consists of a pair of transmitter and receiver devices in the environment. There are many paths by which electromagnetic energy travels between the

BuildSys '18, November 7–8, 2018, Shenzen, China

ACM ISBN 978-1-4503-5951-1/18/11…$15.00
https://doi.org/10.1145/3276774.3276789

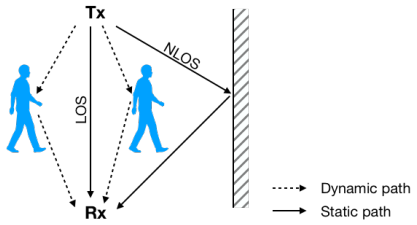
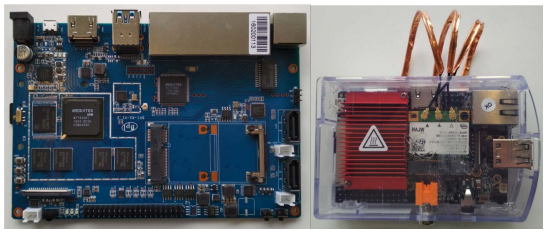


Figure 1: Wi-Fi signal propagation characteristics.



(a) Banana Pi R2 **(b) HummingBoard i2eX**

Figure 2: Hardware devices.

transmitter and receiver. When people move, they disturb the multipath profile in the environment, which is the linear superposition of a number of paths. For instance, Figure 1 shows two static paths: the direct line-of-sight (LOS) and reflected non-line-of-sight (NLOS) path. When a human subject walks from left to right in the figure, a dynamic path is *modulated* by this movement. We build sensing models by analysing their temporal patterns at the receiver.

For each transmitter-receiver pair, the superposition of multipaths in the time domain is described by a N_{sc}-dimensional frequency-domain CSI H corresponding to a sampling of OFDM subcarriers across the bandwidth.[1] As such, the transmitted signal X can be related to the received signal Y through this input-output channel response relationship according to $Y = HX$. A MIMO system generalises this input-output relationship for N_{tx} transmitters and N_{rx} receivers. For instance, if we have 3 transmitters and 3 receivers, the channel is described as a $3 \times 3 \times 30$ tensor of complex numbers each representing the amplitude and phase response of the respective subcarriers. Typically these CSI tensors are modelled with a purpose-built signal processing pipeline to detect a variety of human activities induced by kinetic movements.

2.2 Representative Hardware

For the benchmark study, we used two commodity hardware devices, Banana Pi R2[2] and HummingBoard-i2eX[3]. Figure 2 shows the hardware devices and Table 1 details their configurations. We selected Banana Pi R2 for the networking benchmark as it is an open-source router and already used by a number of commercial third-party home-network routers.

We used HummingBoard-i2eX for the sensing benchmark as it supports CSI reading and also has the similar processing capability to Banana Pi R2. CSI is captured on commodity wireless NICs as

[1] e.g. $N_{sc} = 30$ for the widely used Intel 5300 chip.
[2] http://www.banana-pi.org/r2.html
[3] https://www.solid-run.com/nxp-family/hummingboard/

	Banana Pi R2	HummingBoard i2eX
CPU	MediaTek MT7623N, Quad-core ARM A7	iMX6 D, Dual core 1.0GHz ARM A9
Memory	2GB DDR3	1GB DDR3
Ethernet	MT7530, $1 \times$ *WAN* and $4 \times$ *LAN*	AR8030, $1 \times$ *Ethernet*
Wi-Fi	MT6625L	Intel 5300

Table 1: Specification of the hardware used in the study.

estimated by the physical layer, but the access to raw data from the application layer is highly limited. The most common way is to use Intel Wi-Fi Wireless Link 5300 and the CSI tool with a custom modified firmware and open source Linux wireless driver [5], and it is supported by HummingBoard-i2eX.

For resource monitoring, we used atop, a light-weight resource monitoring tool. We measured CPU and memory usage of the device at the interval of one second.

2.3 Representative Sensing Models

We developed two sensing models, one for *occupancy detection*, i.e., number of people, and the other for *activity recognition*, i.e., standing, sitting, or walking. These contexts are simple but provide instrumental cues to a broad range of home applications including smart energy metering, elderly monitoring, and intrusion detection.

Occupancy detection: We employ a light-weight shallow classifier borrowing from [3]. The model collects CSI data at 500 Hz and takes a 10-second window of data as an input. Then, it extracts temporal variations, e.g., variations of distances between consecutive samples, and applied for linear discriminant analysis (LDA).

Activity recognition: We employ a purpose-built deep neural network model, consisting of three convolution layers, three LSTM layers, and one softmax output layer. The model samples CSI data at 1 kHz and segments the stream into 3 second-window frames. Then, it passes a time series of amplitude and phase values to the first convolution layer. The softmax layer outputs the probability of the activity classes. The model is processed every 10 seconds.

It is important to note that we do not argue that the described models are representative for each context. Our goal is to cover different workload characteristics of Wi-Fi sensing rather than optimising the accuracy. Thus, we intentionally chose different types of pipelines, i.e., one with the statistical features and a shallow classifier and the other with raw CSI data and deep learning network.

3 BENCHMARK STUDIES

Our first set of experiments are designed to assess the runtime performance of a representative residential gateway, i.e., Banana Pi R2 under various networking loads. Next, we look at how the platform behaves under sensing tasks, e.g., end-to-end execution of occupancy and activity detection models. We expect, the outcome of these experiments will uncover the feasibility of running sensing tasks on residential gateways in parallel to networking tasks.

3.1 Benchmark for Networking Workloads

We study CPU and memory usage on Banana Pi R2, the representative residential gateway, under various communication workloads.

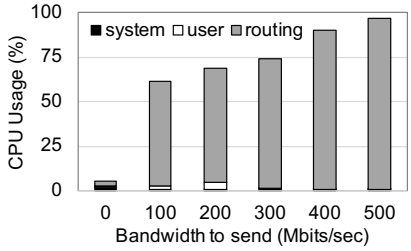


Figure 3: CPU usage under various networking workloads

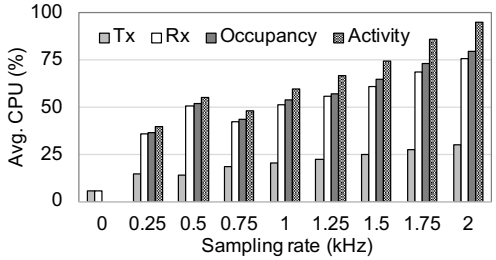


Figure 4: CPU usage under various sensing workloads

Experimental setup: To systematically explore the impact of communication workloads on the resource usage, we synthetically generate the workloads using iPerf [4], a network speed test tool. For routing tasks, we connected one laptop to Banana Pi R2 via a WAN port and the other laptop via a LAN port. We configured iPerf server on the WAN-connected laptop, i.e., listening on a network port, and iPerf client on the LAN-connected laptop, i.e., transmitting the data. To ensure the consistency of the networking behaviour between two different platforms, we also conducted the benchmark study on HummingBoard-i2eX. However, we found out that its routing capacity is much limited compared to that of Banana Pi R2 even though they have similar performance capacity. We omit the result on HummingBoard-i2eX due to page limit.

Results and implications: We measure CPU and memory usage while increasing the bandwidth to send from the client. Figure 3 shows the CPU usage of components on a single CPU core; *system* and *user* are when CPU is running kernel code and user-level code, respectively and *routing* is when the kernel is servicing network routing requests. The results show that the overall CPU usage significantly increases as the network traffic to route increases. It implies that home routers consume nontrivial CPU resource even for the communication task only. The CPU usage of system and user is insignificant when the bandwidth exceeds 400 Mbits/sec, because, the priority for routing interrupt request (IRQ) is higher than that of system and user. Interestingly, even with the full bandwidth, we can observe high CPU usage only on a single core, which means the other three cores are mostly idle. We conjecture that routing mechanism is not optimised yet on Banana Pi R2. Interestingly, different from CPU usage, the memory usage hardly changes, i.e., mostly remains around 370 MB regardless of the bandwidth.

3.2 Benchmark for Sensing Workloads

We then investigate the resource usage of Wi-Fi sensing on HummingBoard i2eX. For the workload, we consider the tasks for sending and receiving wireless packets at different sampling rates. These packets are then passed to different sensing models to assess the overall resource footprint of the board.

Experimental setup: We configure two HummingBoards, one as a transmitter and the other as a receiver. We transmit and receive wireless packets via Intel 5300 in 5 GHz carrier and 40 MHz bandwidth. We vary the packet transmission rate (sampling rate) as it is an essential part of sensing pipeline and the resource usage of subsequent processing. We then connect the two representative sensing models for occupation detection and physical activity recognition independently, i.e., we first test the occupancy detection model to assess its resource footprint, and then repeat the experiment with the activity recognition model.

Results and implications: We measure CPU and memory usage while increasing the sampling rate of CSI packets. Figure 4 shows the results; we omit the memory usage because it does not increase much even at the maximum rate. HummingBoard uses a considerable amount of CPU even for CSI reading only. With 1 kHz, one of the widely used parameters in the literature, a transmitter and a receiver use 20% and 51% of CPU, respectively. Interestingly, a receiver uses much more CPU than a transmitter, around 2 to 3 times primarily due to the Linux CSI Tool [5] intercepting, and parsing CSI data. We also notice the resource footprint of models increasing gradually with higher sampling rates. For simplicity, we only look at their execution behaviour without assessing classification performances. Naturally, shallow occupancy detection model demands lower CPU cycles than the deep activity detection model. However, as the sampling rate reaches over 1kHz, the resource requirements of these models are reasonably high.

3.3 Key Takeaways

Mainly, there are two critical takeaways from our benchmark studies. Firstly, residential gateways use a non-trivial amount of resources for network services and Wi-Fi sensing. Secondly, since the resource demand of network processing changes dynamically depending on ongoing traffic, runtime adaptation of CSI sampling and inference execution schedule are needed for integrating Wi-Fi sensing on residential gateways. In the next section, we reflect on these issues by offering two optimisation techniques.

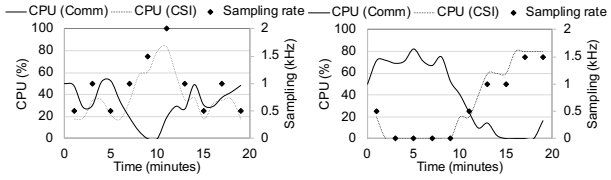
4 OUTLOOK

Based on our findings, we propose potential ways for *dynamic runtime adaptation* of Wi-Fi sensing. The key idea is to obtain multiple processing options for a sensing model and dynamically select the best one based on the available resources. In this section, we present two techniques that can generally be applied without any knowledge and modification of inference pipelines, *dynamic sampling* and *resource-aware execution planning*. Then, we demonstrate the early assessment of our techniques on HummingBoard with the simulated CPU usage.

4.1 Dynamic Sampling

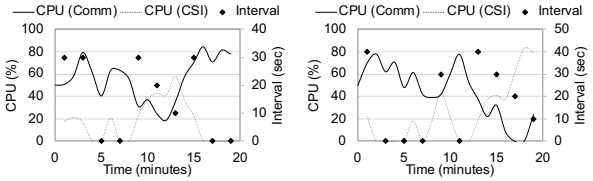
The sampling rate of CSI sensing determines how densely and frequently the model senses human body and movement. Generally,

[4] https://iperf.fr



(a) Occupancy detection (b) Activity recognition

Figure 5: Behaviour of dynamic sampling



(a) Occupancy detection (b) Activity recognition

Figure 6: Behaviour of resource-aware execution planning

the high sampling rate is known to be advantageous to detect fine-grained activities, e.g., 2.5 kHz for speed modelling of limbs [7], but also causes high processing costs due to a large amount of data to process. Thus, a simple but effective way for the optimisation is to *downsample* CSI sensing from a transmitter, i.e., choosing the maximum sampling rate in a way of not compromising the quality of the communication tasks. It may sacrifice the accuracy of the model to some extent but guarantees seamless context monitoring. Downsampling is effective especially when multiple receivers receive and process broadcast packets from a single transmitter as it can easily control the resource usage for Wi-Fi sensing in the whole network. Another alternative is *sparse sampling*, which is conducted on a receiver side by selectively discarding the received packets. For example, when a transmitter sends wireless packets for CSI sensing at 1 kHz, a resource-scarce receiver can uniformly discard three-quarter packets and process the inference pipeline with 250 Hz CSI data. The dynamic sampling technique is for the sensing models which are robust to different sampling rates, e.g., the ones using statistical features. We can also assume that the model developers provide multiple models with different sampling rates for runtime adaptation.

Figure 5a and 5b show the behaviour of the dynamic shaping of CSI sampling for the two sensing models, i.e., the dynamically selected sampling rate (single dots) and corresponding CPU usage (dotted line); a solid line represents the simulated CPU usage of networking tasks. The results show that the technique maximises the sampling rate with given CPU availability. The CPU usage of network tasks (solid line) increases, i.e., as the available CPU decreases, the technique chooses the lower sampling rate to reduce the CPU usage of CSI sensing, but the maximum one among possible rates. The whole CPU usage (sum of a solid line and dotted line) remains between 90% and 100%.

4.2 Resource-aware Execution Planning

Another opportunistic way for the sensing model-opaque adaptation is to change the interval of pipeline execution dynamically. Frequent pipeline execution guarantees shorter latency of context updates but also increases the overall resource use. In the tolerable range of the interval defined by the application, *resource-aware execution planning* dynamically chooses the minimum interval of the pipeline execution to be constrained without compromising the quality of networking.

Figure 6a and 6b show the behaviour of the resource-aware execution planning for the two sensing models. Similarly, the technique chooses a longer execution interval (but the shortest one that makes the whole CPU usage less than 100%.), when the available CPU decreases, i.e., when the CPU use of networking tasks increases. Note that, here, longer execution interval means lower CPU use.

4.3 Discussion and Future Work

To build and evaluate the end-to-end runtime adaptation system, we further need to consider several more aspects. First, for the system development, we need a technique that predicts the short-term future communication bandwidth, e.g., for next 5 seconds. Based on it, we need a scheduling algorithm that chooses the best execution parameters for Wi-Fi sensing, i.e., sampling rate and execution interval, while considering two optimisation techniques at the same time. Second, regarding the evaluation, our preliminary study was limited to the synthetically generated networking workload and the analysis of the resource usage. We plan to validate our system on top of the real networking workloads and investigate the impact of the adaptation on the quality of the networking and sensing services. We leave those as future work.

REFERENCES

[1] Fadel Adib, Hongzi Mao, Zachary Kabelac, Dina Katabi, and Robert C. Miller. 2015. Smart Homes That Monitor Breathing and Heart Rate. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 837–846.

[2] Mohammed Abdulaziz Aide Al-qaness, Fangmin Li, Xiaolin Ma, and Guo Liu. 2016. Device-Free Home Intruder Detection and Alarm System Using Wi-Fi Channel State Information. *International Journal of Future Computer and Communication* 5, 4 (2016), 180.

[3] Simone Di Domenico, Mauro De Sanctis, Ernestina Cianca, and Giuseppe Bianchi. 2016. A Trained-once Crowd Counting Method Using Differential WiFi Channel State Information. In *Proceedings of the 3rd International on Workshop on Physical Analytics (WPA '16)*. ACM, New York, NY, USA, 37–42.

[4] Biyi Fang, Nicholas D. Lane, Mi Zhang, Aidan Boran, and Fahim Kawsar. 2016. BodyScan: Enabling Radio-based Sensing on Wearable Devices for Contactless Activity and Vital Sign Monitoring. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16)*. ACM, New York, NY, USA, 97–110.

[5] Daniel Halperin, Wenjun Hu, Anmol Sheth, and David Wetherall. 2011. Tool Release: Gathering 802.11n Traces with Channel State Information. *ACM SIGCOMM CCR* 41, 1 (Jan. 2011), 53.

[6] Sameera Palipana, Piyush Agrawal, and Dirk Pesch. 2016. Channel State Information Based Human Presence Detection Using Non-linear Techniques. In *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments (BuildSys '16)*. ACM, New York, NY, USA, 177–186.

[7] Wei Wang, Alex X. Liu, Muhammad Shahzad, Kang Ling, and Sanglu Lu. 2015. Understanding and Modeling of WiFi Signal Based Human Activity Recognition. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking (MobiCom '15)*. ACM, New York, NY, USA, 65–76.

[8] Yang Xu, Wei Yang, Jianxin Wang, Xing Zhou, Hong Li, and Liusheng Huang. 2018. WiStep: Device-free Step Counting with WiFi Signals. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 172 (Jan. 2018), 23 pages.

[9] Mingmin Zhao, Fadel Adib, and Dina Katabi. 2016. Emotion Recognition Using Wireless Signals. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking (MobiCom '16)*. ACM, New York, NY, USA, 95–108.