WiFresh: Age-of-Information from Theory to Implementation

Igor Kadota, Muhammad Shahir Rahman, and Eytan Modiano

***Abstract*—Emerging applications, such as smart factories and fleets of drones, increasingly rely on sharing time-sensitive information for monitoring and control. In such application domains, it is essential to keep information fresh, as outdated information loses its value and can lead to system failures and safety risks. The Age-of-Information is a performance metric that captures how fresh the information is from the perspective of the destination.**

In this paper, we show that as the congestion in the wireless network increases, the Age-of-Information degrades sharply, leading to outdated information at the destination. Leveraging years of theoretical research, we propose WiFresh: an unconventional architecture that achieves near optimal information freshness in wireless networks of any size, even when the network is overloaded. Our experimental results show that WiFresh can improve information freshness by two orders of magnitude when compared to an equivalent standard WiFi network. We propose and realize two strategies for implementing WiFresh: one at the MAC layer using hardware-level programming and another at the Application layer using Python.

***Index Terms*—Age of Information, Implementation, Software Defined Radio, Wireless Networks, Optimization**

I. Introduction

Emerging applications will increasingly rely on sharing time-sensitive information for monitoring and control. Examples are abundant: monitoring mobile ground-robots in automated fulfillment warehouses at Alibaba and Amazon [2]; collision prevention applications for vehicles on the road [3]; path planning, localization and motion control for multi-robot formations using drones [4] and using ground-robots [5]; multi-drone system for exploration of subterranean environments [6]; multi-robot simultaneous localization and mapping (SLAM) using drones [7] and using ground-robots [8]; real-time surveillance system using a fleet of ground-robots [9]; and data collection from sensors, drones and cameras for agriculture using the Azure FarmBeats IoT platform [10]. In such application domains, it is essential to *keep information fresh*, as outdated information loses its value and can lead to *system failures* and *safety risks*.

The various time-sensitive applications in [2]–[10] *are all implemented using the IEEE 802.11 standard (WiFi)*. WiFi is an attractive choice for it is low-cost, well-established, and immediately available in drones [4], computing platforms running the Robot Operating System (ROS) [9], sensors that measure soil temperature, pH, and moisture [10], and in the Raspberry Pis used in this paper. Moreover, as showcased by these various implementations, *small-scale underloaded* WiFi networks are capable of supporting time-sensitive applications. Two main shortcomings of WiFi, or any other wireless technology employing First-Come First-Served (FCFS) queues and Random Multiple Access mechanisms, are scalability and congestion.

Igor Kadota is with the Department of Electrical Engineering, Columbia University, New York, NY, 10027. Muhammad Shahir Rahman and Eytan Modiano are with the Laboratory for Information and Decision Systems (LIDS), Massachusetts Institute of Technology, Cambridge, MA, 02139.
E-mail: igor.kadota@columbia.edu, shahir@mit.edu, and modiano@mit.edu

This paper was presented in part at ACM MobiCom 2020 [1].

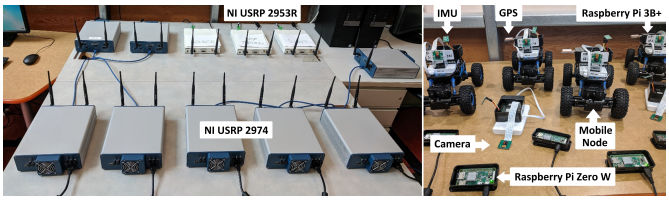


Fig. 1: On the LHS: Software-Defined Radio testbed. On the RHS: Raspberry Pi testbed with cameras, GPS receivers, and IMU sensors.

Our contribution: 1) Leveraging years of theoretical research on Age-of-Information, we propose WiFresh: an unconventional network architecture that scales gracefully, achieving near optimal information freshness in wireless networks of any size, even when the network is overloaded. 2) We propose and realize two strategies for implementing WiFresh: WiFresh Real-Time, which is designed to maximize performance, and is implemented at the *MAC layer* in a network of eleven FPGA-based Software Defined Radios (Fig. 1) using hardware-level programming; and WiFresh App, which is designed to lower the barriers to adoption, and is implemented at the *Application layer*, without modifications to lower layers of the networking protocol stack, in a network of twenty five Raspberry Pis (Fig. 1) using Python 3. The WiFresh App runs over UDP and standard WiFi, making it easy to integrate into applications that are implemented using WiFi such as [2]–[10]. 3) Our experimental results in Sec. V show that the more congested the network, the more prominent is the superiority of WiFresh when compared to WiFi. In particular, we show that under high load, WiFresh can improve information freshness by two orders of magnitude when compared to an equivalent standard WiFi network.

To illustrate the concept of *information freshness*, which is formally defined in Sec. III, consider a monitoring system composed of a remote monitor, a wireless base station (BS) and N mobile nodes. Each node $i \in \{1, 2, \cdots, N\}$ moves with an average velocity of v_i meters per second, generates status information from time to time, and sends this information to the remote monitor via the wireless base station. Status information can include the node's current position, inertial measurements, and pictures of the environment. The remote

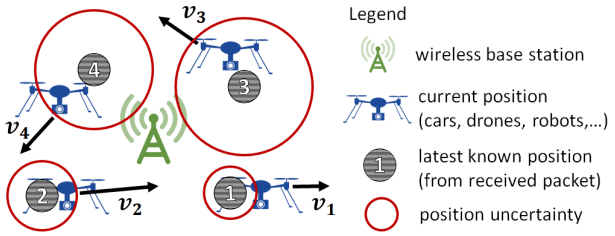


Fig. 2: Illustration of a monitoring system.

TABLE I: Measurements of the average age $\Delta_i(t)$ (in seconds) for networks with different number of sources N.

N	2	8	12	20	24
WiFi UDP	0.34	0.32	0.32	22.43	34.09
WiFi ACP	0.99	0.97	0.88	5.76	6.91
WiFresh App	0.29	0.35	0.39	0.49	0.54

monitor keeps track of the information, and is particularly interested in the position of the nodes. Assume that at time t, the latest packet received by the remote monitor from node i had information about its position at time $\tau_i(t)$. Then, the quantity $\Delta_i(t) := t - \tau_i(t)$ captures how fresh the position information is at time t. We refer to $\Delta_i(t)$ as *Age-of-Information (AoI)* and say that the remote monitor has *stale information* when $\Delta_i(t)$ is large, and *fresh information* when $\Delta_i(t)$ is small. In particular, an age of $\Delta_i(t) = 2$ seconds represents that at time t the remote monitor knows the location of node i two seconds ago. Hence, the *uncertainty about node i's position at time t* is captured by the quantity $v_i\Delta_i(t)$, as illustrated in Fig. 2, and a *large age corresponds to a large uncertainty*.

In Table I, we consider a sequence of networks with increasing size N and display the average age $\Delta_i(t)$ in seconds, where the average is taken over time t and over all the N sources. Each *source* is a Raspberry Pi (shown in Fig. 1) generating position information using the Stratus GPYes 2.0 u-blox 8 GPS receiver and generating inertial measurements using the Pololu MinIMU-9 v5 sensor. The *wireless base station* is a Raspberry Pi receiving data from the N sources. We compare networks using three different architectures, namely:

1) WiFi UDP, which is UDP over standard WiFi;
2) WiFi Age Control Protocol (ACP), which uses the Transport layer protocol developed in [11] to *control the packet generation rates* at the sources in order to minimize AoI;
3) WiFresh App, which is one of the WiFresh implementations proposed in this paper.

Additional details about the experimental setup are provided in Sec. V-C.

WiFi UDP. The measurements in the *first row* of Table I show that as the number of sources N in the WiFi UDP network increases, the *network becomes overloaded and the average age $\Delta_i(t)$ degrades sharply*. The average age for $N = 12$ sources is 0.32 seconds, while for $N = 20$ sources is 22.43 seconds, which means that the information at the remote monitor is (on average) 22 seconds old. Naturally, this staleness directly affects the capability of the monitoring system of tracking the current position of the source nodes, especially for source nodes that are moving fast.

WiFi ACP. The *second row* of Table I shows that the average age $\Delta_i(t)$ for WiFi ACP with $N = 20$ sources is 5.76 seconds. By controlling the packet generation rates at the sources using the protocol developed in [11], ACP *improves the average age by a factor of four* when compared with WiFi UDP. Notice that a high packet generation rate may overload the network and lead to a high average age, while a low packet generation rate may result in infrequent information updates at the destination, which may also lead to a high average age. The ACP dynamically adapts the packet generation rates at the sources in order to drive the network to the point of optimal information freshness. This point of minimum AoI is illustrated in Fig. 4.

WiFresh App. The *third row* of Table I shows that the average age $\Delta_i(t)$ for WiFresh with $N = 20$ sources is 0.54 seconds. WiFresh *improves the average age by a factor of forty* when compared with WiFi UDP. Experimental results in Sec. V show that this improvement increases for larger N. The superior performance of WiFresh is due to the combination of three elements:

- **Polling Multiple Access mechanism** that *prevents packet collisions*, allowing for efficient resource allocation among sources, which is critical in congested networks and in networks with large number of sources N;
- **Max-Weight (MW) policy** that determines the sequence of sources to poll in order to keep the age of each source as low as possible and, thus, *optimize information freshness* in the network. Notice that the Polling mechanism is used to support the MW policy; and
- **Last-Come First-Served (LCFS) queues** that prioritize the *packet with lowest delay*, leading to sources that always transmit the freshest packets to the base station.

In WiFresh App, these three elements are implemented at the Application layer in a network of Raspberry Pis. In WiFresh Real-Time, these three elements are implemented at the MAC layer in a network of SDRs. The choice of each of these elements is underpinned by theoretical research. In [12]–[14], the LCFS queue was shown to be the optimal queueing discipline in terms of AoI in different settings. In [15]–[18], the authors developed performance guarantees for the MW policy in different settings.

Scalability. Neither WiFi UDP nor WiFresh attempt to control the packet generation rate at the sources. Hence, when the number of sources N increases to the point that the cumulative packet generation rate exceeds the capacity of the network, both WiFi UDP and WiFresh become overloaded and the number of backlogged packets at the sources grows rapidly. For WiFi UDP, a large backlog in the First-Come First-Served (FCFS) queues leads to high packet delay and, thus, to high average age, as observed in Table I. In contrast, *WiFresh scales gracefully, even when the network is overloaded*, with average age increasing linearly with N. In [18, Chapter 3], the authors derived a lower bound on the achievable AoI performance in wireless networks and concluded that the average age cannot

scale better than linearly on the network size N.

The remainder of this paper is organized as follows. In Sec. II, we describe related work. In Sec. III, we discuss the impact of the multiple access mechanism, transmission scheduling policy, and queueing discipline on AoI. In Sec. IV, we describe the design and implementation of WiFresh Real-Time and WiFresh App. In Sec. V, we evaluate the performance of WiFresh in networks with increasing load and increasing size. The paper is concluded in Sec. VI.

II. Related Work

The Age-of-Information was recently proposed in [19] and has been receiving increasing attention in the literature for its application in communication systems that carry time-sensitive data. The AoI captures how fresh the information is *from the perspective of the destination*, in contrast to the well-established packet delay that represents the latency of *a particular packet*. Most papers on AoI focus on theory and a few consider system implementation.

Theoretical research. The problem of optimizing AoI has been addressed in a variety of contexts. Queueing Theory is used in [12], [19], [20] to characterize the AoI performance of various important queueing systems. Information Theory is used in [21], [22] for designing source and channel coding schemes to improve AoI. Optimization of scheduling policies in communication networks is considered in [16], [17], [23]–[29]. Different applications of AoI in sensor networks, cellular networks and vehicular networks are analyzed and/or emulated in [30]–[33]. This list of works is not exhaustive. For a more comprehensive list of theoretical works we refer the reader to [18], [34].

Systems research. A few papers [1], [11], [35], [36] have implemented AoI-based systems. In [35], the authors consider a source-destination pair transmitting packets over the Internet and measure the AoI for different packet generation rates. In [36], the authors consider a vehicular network and develop an Application layer algorithm that adapts the packet generation rates at the sources to improve information freshness. This algorithm is validated using the ORBIT testbed with wireless sources employing WiFi, in particular the IEEE 802.11a standard. In [11], the authors consider an Internet-of-Things network and develop a Transport layer protocol named Age Control Protocol (ACP) that adapts the packet generation rates at the sources in order to optimize for information freshness. This protocol is validated using ten sources connected via WiFi to the Internet and sending packets to a destination in another continent. In Sec. V, we implement ACP and evaluate its performance against WiFresh. Notice that [11], [35], [36] address the problem of controlling the packet generation rates at the sources in order to optimize information freshness.

In this paper, we develop and implement a network architecture that is optimized across the queueing discipline, the multiple access mechanism, and the transmission scheduling policy. A simplified version of WiFresh Real-Time was introduced in our poster [1]. The main differences between [1] and this paper are with respect to the scope and depth.

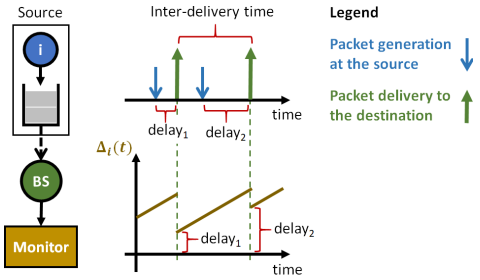


Fig. 3: AoI evolution for a single information source sending packets to the remote monitor via the BS.

This paper presents an in-depth description of both WiFresh Real-Time and WiFresh App. Moreover, it provides extensive experimental results using two testbeds with different sizes and different traffic loads.

III. Background on AoI

Consider a communication network in which packets are time-stamped upon arrival. Naturally, the higher the time-stamp, the fresher is the information contained in a packet. Let $\tau_i(t)$ be the time-stamp of the *freshest packet received by the destination from source i by time t*. The AoI associated with source i is defined as $\Delta_i(t) := t - \tau_i(t)$. The AoI measures the time elapsed since the generation of the freshest packet received by the destination from source i. The value of $\Delta_i(t)$ increases linearly in time while no fresher packet from source i is received, representing the information getting older. At the moment a *fresher* packet is received by the destination, the value of $\tau_i(t)$ is updated and $\Delta_i(t)$ *decreases to the packet delay*. The evolution of the age process is illustrated in Fig. 3.

The time-average expected age associated with source i is given by $\int_{t=0}^{T} \mathbb{E}[\Delta_i(t)]dt/T$. From Fig. 3, we can see that to keep the information at the destination as fresh as possible, i.e. minimize the time-average expected AoI in the network, it is necessary to simultaneously provide: i) low packet delay; ii) high data throughput; and iii) service regularity[1]. To minimize AoI, we consider the network as a whole and optimize the system across the queueing discipline, the multiple access mechanism, and the transmission scheduling policy. Next, we discuss each of them in detail.

A. Queueing Discipline

The queueing discipline employed at the sources is central for minimizing AoI. In this section, we compare FCFS and LCFS queues and evaluate their performance in terms of AoI. FCFS queues are widely deployed in communication systems and they are the basis for other disciplines such as Priority Queueing and Fair Queueing. FCFS queues transmit packets in order of arrival, meaning that the *freshest packet is always placed at the tail of the queue*. Under heavy loads, the FCFS queue is often backlogged and the freshest packet has to wait for a long queueing delay before being delivered

[1]It is important to emphasize the difference between delivering packets regularly and providing a minimum throughput. In general, a given minimum throughput can be achieved even if long periods with no delivery occur, as long as those are balanced by short periods of consecutive packet deliveries.

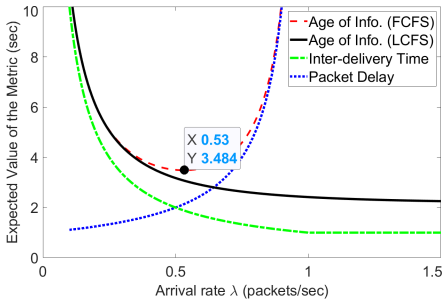


Fig. 4: Expected delay, interdelivery time and average age of an M/M/1 queueing system with service rate of $\mu = 1$.

to the destination. The high queueing delay leads to stale information at the destination and to high age. This effect is more prominent for large FCFS queues, as discussed in Sec. V-A.

LCFS queues are often considered in the AoI literature [12]–[14], [16], [28], [37], but they are not commonly deployed in communication systems. LCFS queues place the *most recently generated packet at the Head-of-Line (HoL)*, leading to sources that transmit the freshest packet first, which makes LCFS queues ideal for applications that rely on the knowledge of the *current* state of the system, i.e. applications that need fresh information at the destination. Under heavy loads, the LCFS queue is frequently replacing its HoL packet with fresher packets. We expect that the higher the packet generation rate at the sources, the lower the average age at the destination, regardless of the queue backlog. LCFS queues are not commonly found in communication systems. Not surprisingly, LCFS is not one of the queueing discipline (qdisc) options in Linux nor in the Software Defined Radios (SDRs) we utilized for implementing WiFresh. In both cases, the standard queueing discipline is FCFS.

Comparing FCFS and LCFS. Consider an M/M/1 queueing system with infinite queue size, fixed packet service rate of $\mu = 1$ packet per second and variable packet generation rate λ, employing either FCFS or LCFS discipline. In Fig. 4, we display the time-average expected age for FCFS and LCFS, the expected packet delay and the expected interdelivery time. The analytical expressions for the AoI associated with FCFS and LCFS queues were obtained in [19] and [12], respectively, and the expressions for packet delay and interdelivery time can be found in [38].

Choice of LCFS for WiFresh. From Fig. 4, we can see that the minimum time-average age for FCFS queues is achieved at moderate loads, in particular $\lambda/\mu \approx 0.53$, while for LCFS queues the higher the packet generation rate λ, the lower the average age. In addition, LCFS outperforms FCFS for every packet generation rate λ. As discussed in Sec. II, LCFS was shown to be the optimal queueing discipline in different settings including single queue systems [12], [13], [28], single-hop wireless networks [37] and multi-hop wireless networks [14]. Thus, we propose to use the LCFS discipline in WiFresh.

Effect of dropping packets. LCFS queues transmit the freshest packet first. Notice that when a packet with *older information* is delivered to the destination after a packet with *fresher information*, the freshness of the information is not affected and, thus, the value of $\Delta_i(t)$ *remains unchanged*. Hence, if packets with older information were dropped at the source as soon as a fresher packet arrived to the LCFS queue, the information freshness at the destination and the evolution of $\Delta_i(t)$ over time would not be affected. It follows that, *from the perspective of AoI, a LCFS queue is equivalent to a head-drop FCFS queue of size* 1 *packet, in which only the freshest packet is kept*. The advantage of dropping older packets at the source is saving communication resources. One possible disadvantage is that dropped packets might contain useful information. For example, in a position tracking system, older packets can be used to predict future movement.

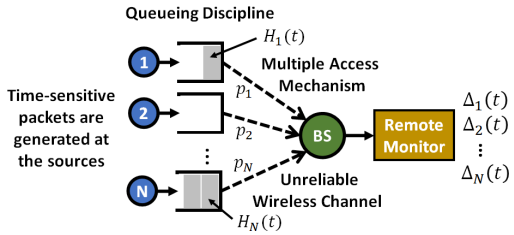


Fig. 5: Illustration of the wireless network.

B. Multiple Access Mechanism

Consider the network in Fig. 5 with N sources sending time-sensitive information to the remote monitor via the wireless BS. Packets are generated at the sources and enqueued in separate queues. The multiple access mechanism controls the method utilized by each of the N sources for sharing the common wireless channel. In this section, we compare two types of multiple access mechanism, Random Access and Polling, in terms of information freshness.

To capture the freshness of the information *in the network*, we define the *expected network AoI (NAoI)* as

$$\lim_{T\to\infty} \frac{1}{TN} \int_{t=0}^{T} \sum_{i=1}^{N} \mathbb{E}\left[\Delta_i(t)\right] dt \,, \tag{1}$$

where $T > 0$ is the time-horizon. To minimize NAoI, the multiple access mechanism should attempt to: i) provide high communication efficiency by preventing packet collisions, minimizing the effects of external interference, and reducing control overhead; and ii) prioritize transmissions from sources with high current age $\Delta_i(t)$ and favorable channel conditions.

Random Access is a widely deployed class of multiple access mechanisms, e.g. WiFi, ZigBee, Wireless Body Area networks [39], and traditional cellular systems such as GSM. The fundamental idea is that, when a source has a packet to transmit, it uses a randomized algorithm to contend for channel access. Randomization is employed to reduce the probability of two or more sources transmitting packets simultaneously, which would result in a packet collision. Some advantages of Random Access are simplicity, decentralization and low control signaling overhead. Some disadvantages are the probability of packet collision that increases with the number of sources N, the susceptibility to external interference, and the distributed operation that makes it challenging to implement a

dynamic transmission prioritization based on parameters such as age $\Delta_i(t)$ and/or current channel conditions.

Polling mechanism is a well-known alternative to Random Access [40]. The BS coordinates the communication in the network by sending *poll packets* to the sources selected for transmission. The BS selects the next source to poll based on the *scheduling policy*, which may be a function of dynamic parameters such as age $\Delta_i(t)$ and/or current channel conditions. The polling mechanism attempts to leverage all the available communication resources, and it does not back-off when transmission errors occur due to the unreliability of the wireless channel or due to external interference. It is the role of the scheduling policy to estimate the channel conditions and adapt future scheduling decisions accordingly. The polling mechanism attempts to maximize communication efficiency and enables dynamic prioritization, making it suitable for large-scale time-critical applications.

Two important challenges associated with polling mechanisms are the control overhead and the choice of scheduling policy. *Control overhead:* the BS transmits a poll packet before receiving each data packet. In contrast, Random Access may require that the BS transmit an acknowledgment packet following the reception of each data packet. Hence, the control overhead of both mechanisms is comparable. *Scheduling policy:* the BS dynamically chooses the next source to poll. Evidently, a naive policy can degrade the performance. Next, we discuss scheduling policies that are designed to optimize the information freshness in the network.

C. Scheduling Policy

The problem of obtaining an optimal scheduling policy for single-hop wireless networks in terms of information freshness was shown in [23] to be NP-hard. Numerous heuristic policies based on Approximate Dynamic Programming [25], Restless Multi-Armed Bandits [17], [18] and Lyapunov Optimization [15]–[18], [24] have been proposed in the literature. *This paper is the first to implement an AoI-based scheduling policy in a real network.* The Max-Weight policy is chosen for WiFresh because it is intuitive, low-complexity and has superior performance [16].

Max-Weight (MW) policy. Consider the network in Fig. 5 employing LCFS queues and a Polling mechanism. Assume that t is the current decision time of the next poll packet. Let $p_i \in (0, 1]$ be the channel reliability associated with source i, namely the probability of a successful reception of a data packet following the transmission of a poll packet to source i. Let $\tau^{HoL}(t)$ be the time-stamp of the current Head-of-Line packet from source i at time t and let $H_i(t) := t - \tau^{HoL}(t)$ be the *current system time of this HoL packet*. Notice that if this HoL packet were delivered to the BS at time t, then $H_i(t)$ would be the associated packet delay and the age would be reduced from $\Delta_i(t)$ to $H_i(t)$, as illustrated in Fig. 3. Hence, the difference $\Delta_i(t) - H_i(t)$ represents the *potential age reduction* of polling source i at time t. Assume that the scheduling policy knows $\Delta_i(t)$, $H_i(t)$ and p_i, and denote $\mathcal{I}(i,t) := p_i(\Delta_i(t) - H_i(t))^2$ as the index of source i at time t. Then, the MW policy selects, at every decision time t, the source $i^*(t)$ with highest value of $\mathcal{I}(i,t)$, with ties being broken arbitrarily. Intuitively, the MW policy is polling the source with highest *weighted potential age reduction*. The MW policy was developed in [16] where we also obtained performance guarantees in terms of AoI. To implement the MW policy in a real network, we augment WiFresh with algorithms that estimate $\Delta_i(t)$, $H_i(t)$ and p_i over time, as described in Sec. IV-B.

IV. Design and Implementation

In this section, we discuss the design and implementation of WiFresh Real-Time and WiFresh App. Prior to delving into the details, we describe the main challenges.

A. Challenges

Complexity of implementation. To achieve high performance, the LCFS queues, the Polling mechanism, and the MW policy were *fully implemented in FPGAs* with 10 MHz clocks, enabling WiFresh Real-Time to make the scheduling decision and trigger the transmission of the next poll packet in approximately 20 microseconds. Keeping this time-interval short and limiting the length of the poll packet are important factors in reducing the control overhead and achieving high performance. The main challenge of implementing WiFresh Real-Time at the *MAC layer* is the complexity associated with implementing numerous real-time functions using hardware-level programming.

Barrier to adoption. Targeting an alternative implementation of WiFresh that could be easily integrated into applications that already run over WiFi such as [2]–[10], we propose WiFresh App which is implemented in Python 3 and runs at the *Application layer*, without modifications to lower layers of the networking protocol stack. The main challenge of WiFresh App is in the design of a Python application that is capable of driving a standard WiFi network (with FCFS queues and Random Access) to behave as WiFresh (with LCFS queues and Polling mechanism with MW policy). This design is discussed in Sec. IV-D.

Bridging theory and practice. Theoretical works on AoI often assume that: 1) nodes are synchronized; 2) nodes generate packets on-demand or according to known stochastic processes; 3) each node is associated with a single type of information such as position, inertial measurements or images; 4) each data packet contains a complete information update; 5) channel reliabilities $\{p_i\}_{i=1}^N$ are fixed and known; and/or 6) system times of HoL packets $\{H_i(t)\}_{i=1}^N$ are known. To leverage the theory and implement an AoI-based network architecture, we augment WiFresh with algorithms that synchronize clocks, dynamically learn $\{p_i\}_{i=1}^N$ and $\{H_i(t)\}_{i=1}^N$, manage sources with multiple information types, and manage packet fragmentation.

Fragmentation of information updates. The age is reduced when fresh information is received at the destination. The evolution of $\Delta_i(t)$, as described in Sec. III, assumes that each data packet contains a complete information update.

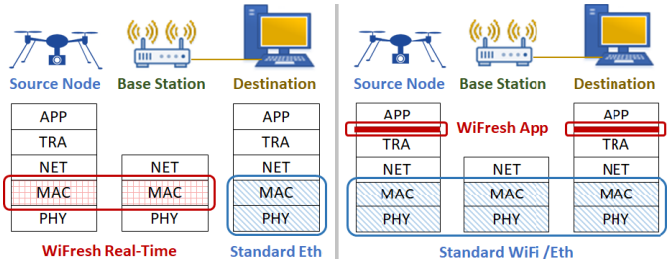


Fig. 6: Layers of the WiFresh RT system (LHS) and WiFresh App system (RHS).

To accommodate large information updates, such as images, WiFresh has to manage packet fragmentation. Two issues are discussed below.

The first issue is when to reduce the age $\Delta_i(t)$. In general, the age $\Delta_i(t)$ can be reduced (or partially reduced) upon reception of a subset of fragments. In this work, fragmentation is used for transmitting images and, in this case, it makes sense to reduce age only when all fragments are received. The second issue is whether the LCFS queue should replace the HoL packet as soon as a new information update arrives, or if the LCFS queue should wait until all fragments from the previous information update are delivered before replacing the HoL packet. Notice that if information updates are generated with a high rate, then replacing the HoL packet as soon as a new information update arrives may hinder the *complete transmission* of information updates. For this reason, in this work, we choose to transmit all fragments before replacing the information update at the LCFS queue.

WiFresh Real-Time runs at the MAC layer. Hence, it is blind to the concept of *information* and can only see individual *data packets*. This makes the process discussed above challenging to implement. To overcome this problem, WiFresh Real-Time could gather information regarding fragmentation from other layers of the communication system. In this work, we implement fragmentation only in WiFresh App, which runs at the Application layer and is aware of information updates. Recall that information updates are generated and received by the Application layer.

B. Design of WiFresh Real-Time

In this section, we describe WiFresh Real-Time (WiFresh RT) in detail. WiFresh RT is implemented at the *MAC layer*, as illustrated in Fig. 6. Next, we describe the main functions associated with the sources and the base station.

WiFresh RT source. The source generates information updates in the Application layer and forwards them to lower layers of the networking protocol stack. When a data packet arrives at the MAC layer, WiFresh RT appends a time-stamp to the packet and then stores it in a head-drop FCFS queue of size 1 packet. Recall that this queue keeps only the freshest packet and discards older packets. The source can be in one of two states: 1) waiting for a poll packet from the BS; or 2) transmitting the freshest data packet to the BS. Upon receiving a poll packet, if the queue is empty, the source transmits an *empty packet* to the BS. The empty packet is used by the BS to differentiate between not receiving data due to a transmission error or due to an empty queue at the source, which impacts the estimation of p_i and $H_i(t)$ at the BS. After transmitting either the data packet or the empty packet, the source goes back to waiting for the next poll packet.

WiFresh RT Base Station. The BS does not generate data packets. Its main responsibility is to coordinate the communication in the network. The BS can be in one of two states: 1) waiting for a data packet; or 2) transmitting a poll packet. While waiting for a data packet, the BS keeps track of the waiting period. If the waiting period exceeds 100 microseconds or a data packet is received, the BS updates its estimate of the network state $(\hat{\Delta}_i(t), \hat{H}_i(t), \hat{p}_i(t))_{i=1}^N$, where $\hat{\Delta}_i(t), \hat{H}_i(t)$ and $\hat{p}_i(t)$ are the estimates of $\Delta_i(t), H_i(t)$ and p_i, respectively, at time t. These estimates are used by the MW policy to calculate $i^*(t) = \arg\max\{\hat{p}_i(t)(\hat{\Delta}_i(t) - \hat{H}_i(t))^2\}$. After transmitting a poll packet to source $i^*(t)$, the BS goes back to waiting for the next data packet. The algorithms used to estimate $\Delta_i(t)$, p_i and $H_i(t)$ are discussed next.

Clock synchronization is needed to accurately compute $\Delta_i(t) := t - \tau_i(t)$, where t is the current time *measured by the BS* and $\tau_i(t)$ is a time-stamp *created by source* i. If clocks are not synchronized, the values of $\Delta_i(t)$ for different sources may have different biases, which may lead to poor scheduling decisions by the MW policy. To estimate the time-stamp offset between each source and the BS, and obtain the estimates $\{\hat{\Delta}_i(t)\}_{i=1}^N$, some possible approaches are: adding GPS antennas to every source in the system and then using GPS time; synchronizing the Operating System (OS) of every source using the Network Time Protocol (NTP) [41] via the Internet and then using the OS time; or implementing a synchronization algorithm as part of WiFresh. In WiFresh RT we use the OS time. In WiFresh App we implement a built-in synchronization algorithm based on NTP.

Learning channel reliability. To estimate the value of $p_i \in (0, 1]$ associated with each source i, we implement a low-complexity estimator. Let $\mathcal{P}_i(t)$ be the number of poll packets transmitted to source i in the last $\mathcal{W}$ seconds and let $\mathcal{D}_i(t)$ be the number of *data packets* and *empty packets* successfully received from source i in the same period. Then, the estimate of p_i at time t is given by $\hat{p}_i(t) = (\mathcal{D}_i(t) + 1)/(\mathcal{P}_i(t) + 1)$. We choose a time window of $\mathcal{W} = 0.5$ seconds. Notice that when the number of poll packets $\mathcal{P}_i(t)$ is low, the estimate $\hat{p}_i(t)$ tends to be optimistic, i.e. higher. In particular, when $\mathcal{P}_i(t) = \mathcal{D}_i(t) = 0$, we have $\hat{p}_i(t) = 1$. This high value of $\hat{p}_i(t)$ when the number of poll packets is low creates an incentive for the MW policy to select sources that have not been polled recently.

To determine the changes in $\mathcal{D}_i(t)$ and $\mathcal{P}_i(t)$ for each source i over time, we log the transmission and reception events within the window $\mathcal{W}$ using arrays. This log is created at the on-board processor of the SDR (as opposed to the FPGA) in order to spare the limited FPGA resources. The disadvantage of keeping the log at the processor is the added round-trip communication delay between on-board processor and FPGA which is of approximately 500 microseconds.

Since $\hat{p}_i(t)$ represents the average channel reliability in the last $\mathcal{W} = 0.5$ seconds, it follows that this relatively small round-trip communication delay has a negligible impact on the performance of the MW policy. The estimate of p_i is the only portion of WiFresh RT which is not fully implemented at the FPGA.

Learning the system times. Recall that the difference $\Delta_i(t) - H_i(t)$ represents the *potential age reduction* of polling source i at time t and that the MW policy wishes to use this difference for selecting the appropriate source to poll. The problem is that the MW policy does not know the system times of the HoL packets $\{H_i(t)\}_{i=1}^N$, which are only known by the respective sources, as illustrated in Fig. 5. One approach for estimating $H_i(t)$ could be to develop an algorithm that generates estimates $\hat{H}_i(t)$ based on the entire history of transmission and reception events, especially the sequence of previously received time-stamps. The main drawback of this approach is its computational complexity. A less accurate but much simpler approach is to estimate $H_i(t)$ based on the latest received packet only. In particular, we know that when the freshest data packet from source i is received at time t, the potential age reduction of polling source i again at time t is (most likely[2]) zero, which is represented by $H_i(t) = \Delta_i(t)$. Similarly, when an empty packet is received at time t, the potential age reduction of polling source i again at time t is (most likely) zero. Hence, we can estimate $H_i(t)$ using the following mechanism:

- $\hat{H}_i(t) \leftarrow \hat{\Delta}_i(t)$ following the successful reception of a data packet or an empty packet from source i at time t; and
- $\hat{H}_i(t)$ remains constant over time while no packet is received.

This low-complexity mechanism for obtaining $\hat{H}_i(t)$ prevents the MW policy to repeatedly schedule the same source i with a high age $\hat{\Delta}_i(t)$ and an *empty queue*. In Sec. V-C, we compare the MW policy with the Maximum Age First (MAF) policy which schedules the source i with highest current age $\hat{\Delta}_i(t)$, disregarding the estimates $\hat{p}_i(t)$ and $\hat{H}_i(t)$. We show that, as expected, MW outperforms MAF in every experiment.

Estimation errors in $\hat{\Delta}_i(t)$, $\hat{p}_i(t)$ and/or $\hat{H}_i(t)$ affect the performance of WiFresh RT only when they result in poor scheduling decisions. In Sec. V, we evaluate the performance of WiFresh RT using the low-complexity mechanisms described in this section and show that WiFresh RT achieves low NAoI in a variety of network settings. Next, we discuss the implementation of WiFresh RT.

C. Implementation of WiFresh Real-Time

We implement WiFresh RT in the SDR testbed in Fig. 1 composed of one NI USRP 2974 operating as the wireless base station, and ten sources: seven NI USRP 2974 and three NI USRP 2953R. The code is developed using a modifiable WiFi reference design [42] with Transport layer based on UDP, MAC layer based on the Distributed Coordination Function (DCF), PHY layer based on the IEEE 802.11n standard with center frequency 2.437 GHz, 20 MHz bandwidth and a fixed MCS index of 5. We use this WiFi reference design as a starting point to implement WiFresh RT.

The WiFi reference design is composed of two parts: the Host code (running at the on-board Intel i7 6822EQ 2 GHz Quad Core processor) and the FPGA code (running at the Xilinx Kintex-7 XC7K410T FPGA). The Host code is responsible for the generation of data packets, radio configuration, and displaying measurements and plots. The FPGA code is responsible for processing data packets, generating control packets (e.g. Clear-to-Send, Request-to-Send and Acknowledgments), accessing the wireless channel using DCF, time management (e.g. Interframe spaces and timeouts), etc. The FPGA code allows us to implement real-time functions at the hardware level. The FPGA clock is of 10 MHz, meaning that these functions run at the microsecond time-scale. For implementing WiFresh RT, we developed several new real-time functions at the FPGA, including: 1) Polling mechanism; 2) Max-Weight policy; 3) head-drop FCFS queue with size 1 packet; 4) time-stamp processing; 5) learning algorithms; and 6) measurement logs.

D. Design of WiFresh App

WiFresh App is an implementation of WiFresh that aims to be easily integrated into time-sensitive applications that already run over WiFi such as [2]–[10]. WiFresh App is implemented in Python and runs at the Application layer, without modifications to lower layers of the networking protocol stack, as illustrated in Fig. 6. It is designed to drive a standard WiFi network (with FCFS queues and Random Access) to behave as WiFresh (with LCFS queues and Polling mechanism with MW policy). WiFresh App contains all elements of WiFresh RT and some additional features, namely fragmentation of large information updates, a simple built-in synchronization algorithm, and support for sources that generate multiple types of information. Next, we describe WiFresh App.

WiFresh App source. The source generates information updates at the Application layer. WiFresh App time-stamps the information updates and stores them in a LCFS queue, which is implemented using a *Python LIFO stack*. The LCFS queue releases a single information update only when the source receives a poll packet from the destination. If the released information update fits into a single data packet, this information update is encapsulated into a data packet and forwarded to lower layers of the networking protocol stack. Otherwise, the information update is fragmented, stored and the first data packet is forwarded. Fragments are stored in a FCFS queue which is separate from the LCFS queue containing information updates. Upon receiving the next poll packet acknowledging the first fragment, the source forwards the second fragment, and so on, until all fragments are successfully delivered to the destination. When the poll packet acknowledging the final fragment is received, the LCFS queue releases the next

[2]The potential age reduction of polling source i again at time t might be greater than zero if source i generates a new data packet while the previous data packet was being transmitted. We assume that this is an unlikely event and neglect its effect.

information update, which is then fragmented, stored and transmitted following the same procedure.

When a fragment reaches the source's MAC layer, WiFi stores it in a FCFS queue and transmits it to the destination using Random Access. Ideally, since the destination only generates a new poll packet after the previous fragment is received, there should be *at most one source* attempting transmission using Random Access at any given time. This means that, even when all sources are generating information updates with a high rate, the underlying WiFi network is handling one data packet at a time, leading to low packet delay and low probability of packet collision. When transmission errors occur, WiFi may attempt to retransmit the data packet. Notice that by implementing LCFS queues and Polling mechanisms with MW policy at the Application layer, *WiFresh App is driving the underlying WiFi network to behave as a WiFresh network.*

WiFresh App destination. Similarly to the WiFresh RT Base Station, the destination in WiFresh App generates poll packets, implements a timeout of 300 milliseconds, updates its estimate of the network state $(\hat{\Delta}_i(t), \hat{H}_i(t), \hat{p}_i(t))_{i=1}^N$, and uses the MW policy to decide which source to poll next. The main differences are that: 1) the scheduling decisions are made at the Application layer at the millisecond time-scale, as opposed to the MAC layer at the microsecond time-scale; and 2) the destination manages the fragmentation procedure described above, uses a built-in clock synchronization algorithm based on the *on-wire protocol* that is part of NTP [41, Sec. 8] to estimate the current age $\hat{\Delta}_i(t)$, and supports sources that generate multiple types of information.

Multiple information types per source. The age is associated with a single type of information such as position, inertial measurements or images. In a network with sources that generate multiple types of information, we create for each tuple (source, information type) a separate instance of WiFresh App with independent LCFS queue, age $\hat{\Delta}_i(t)$, channel reliability $\hat{p}_i(t)$, and system time $\hat{H}_i(t)$. The destination treats each instance of WiFresh App at the sources as an independent entity, and sends individual poll packets to each of them.

E. Implementation of WiFresh App

We implement WiFresh App in a Raspberry Pi (Raspi) testbed composed of one desktop computer operating as the destination, one Raspberry Pi 3B+ with a WiFi USB adapter operating as the wireless BS, and twenty four sources: ten Raspberry Pi 3B+ fetching data from sensors and fourteen Raspberry Pi Zero W generating synthetic data that emulates the sensors. For the measurements in Sec. V, sources are static and placed indoors. The distance between sources and destination is between 2 and 3 meters. In Fig. 1, we display some of the sources[3] and the three sensors described below:

- cameras (Arducam 5 Megapixels 1080p) generating jpg images with resolution 256x144 pixels and size of approximately 19 kbytes at a rate of 2 Hz.
- Inertial Measurement Units (Pololu MinIMU-9 v5 Gyro, Accelerometer, and Compass) generating information updates of size 20 bytes at a rate of 100 Hz; and
- GPS units (Stratux GPYes 2.0 u-blox 8) generating information updates of size 50 bytes at a rate of 1 Hz;

To create synthetic GPS data for indoor environments, we use a NMEA sentence generator [43] that emulates the GPS unit.

The Raspberry Pis run the Raspbian Stretch OS and communicate via WiFi, in particular the IEEE 802.11g standard at 2.4 GHz. WiFresh App is implemented using Python 3. The main functionalities we developed are: 1) Polling mechanism; 2) Max-Weight policy; 3) LCFS queue; 4) fragmentation management; 5) time-stamp processing; 6) learning algorithms; 7) interface with sensors; 8) synthetic generation of data packets emulating each type of sensor; 9) graphical user interfaces; and 10) measurement logs. The Transport, Network, MAC and PHY layers were kept unchanged, as illustrated in Fig. 6. *WiFresh App is built over standard UDP, IP and WiFi protocols.*

V. Experimental Results

In this section, we evaluate the performance of WiFresh in a dynamic indoor office space with *multiple external sources of interference* such as mobile phones, laptops and campus WiFi base stations. We evaluate WiFresh RT and WiFresh App, and compare them with other communication systems. In particular, using the SDR testbed described in Sec. IV-C, we compare:

- **WiFresh RT:** as described in Sec. IV-B;
- **WiFresh RT FCFS:** identical to WiFresh RT but with sources employing FCFS queues;
- **WiFi UDP FCFS**: UDP over standard WiFi; and
- **WiFi UDP LCFS**: UDP over standard WiFi but with sources employing LCFS queues instead of FCFS queues.

In addition, using the Raspi testbed described in Sec. IV-E, we compare:

- **WiFresh App:** as described in Sec. IV-D;
- **WiFresh Max. Age First (MAF)**: identical to WiFresh App but with a scheduling policy that, at every decision time t, selects the source $i^*(t)$ with highest current age $\Delta_i(t)$. The MAF policy was proposed in [17];
- **WiFi UDP FCFS**: UDP over standard WiFi;
- **WiFi TCP FCFS**: TCP over standard WiFi; and
- **WiFi ACP FCFS**: Age Control Protocol (ACP) over standard WiFi. ACP is a Transport layer protocol recently proposed in [11] that adapts the packet generation rate of each source i in order to minimize the NAoI in Eq.(1). Recall that in our testbed, the packet generation rate is fixed and determined by the associated sensor. Hence, in our implementation of ACP, we approximate the target packet generation rate by regularly discarding some of the packets before they reach the FCFS queue.

Next, we present experimental evaluations of WiFresh. Each experiment runs for 10 minutes.

[3]The remote control cars and battery packs are used for running tests outdoors. The measurements discussed in this paper were performed indoors.

TABLE II: Single source measurements with the SDR testbed.

SDR	WiFresh RT		WiFi UDP FCFS	
	AoI (sec)	Thr. (Mbps)	AoI (sec)	Thr. (Mbps)
$\lambda = 5$k	0.003	4.866	0.306	2.406
$\lambda = 6$k	0.003	4.905	0.304	2.433
$\lambda = 7$k	0.004	4.412	0.320	2.328

TABLE III: Single source measurements with the Raspi testbed.

Raspi	WiFresh App		WiFi UDP FCFS	
	AoI (sec)	Thr. (Mbps)	AoI (sec)	Thr. (Mbps)
$\lambda = 5$k	0.040	0.229	224.8	1.242
$\lambda = 6$k	0.046	0.197	248.2	1.183
$\lambda = 7$k	0.042	0.208	242.3	1.281

A. Single Source with High Load

In this section, we consider a network with a destination, a wireless BS and a *single source* generating packets of 150 bytes with rate $\lambda \in \{5, 6, 7\}$ kHz. These short packets of 150 bytes represent status updates, and different values of λ represent different levels of congestion. In Tables II and III, we measure the time-average AoI (in seconds) and the effective throughput (in Mbps). The effective throughput is measured at the Application layer of the destination and, thus, it refers to the number of *useful bits* received per second. In Table II, we consider WiFresh RT and WiFi UDP FCFS in the SDR testbed, and in Table III, we consider WiFresh App and WiFi UDP FCFS in the Raspi testbed.

External interference. The results in Tables II and III show that when the packet generation rate increases from 5 kHz to 7 kHz, the effective throughput does not change significantly, indicating that sources with $\lambda \geq 5$ kHz are saturated, i.e. always have data to transmit. Table II shows that the throughput of WiFresh RT is higher than the throughput of WiFi UDP FCFS. This is because *WiFresh RT does not back-off* when a transmission error occurs due to the unreliability of the wireless channel or due to collisions with external wireless networks, making WiFresh RT less susceptible to external interference than WiFi UDP FCFS. Recall that WiFresh RT is designed to support large-scale time-critical applications and, to that end, it attempts to maximize its channel utilization. In contrast, WiFresh App runs over standard WiFi, making it as susceptible to external interference as WiFi UDP FCFS. Table III shows that the throughput of WiFresh App is lower than the throughput of WiFi UDP FCFS. The main reason for the lower throughput is the *control overhead* associated with running a Polling mechanism over standard WiFi. Notice that acknowledgement packets follow the successful transmission of every poll and data packets, thus increasing the control overhead. Despite the lower throughput, WiFresh App significantly outperforms WiFi UDP FCFS in terms of AoI, as we see next.

Queueing discipline. The results in Tables II and III show that WiFresh RT and WiFresh App can improve age by two orders of magnitude when compared to WiFi UDP FCFS. In this single source scenario, the performance gain comes from using LCFS instead of FCFS. In Fig. 7, we compare the age $\Delta_i(t)$ evolution over time for systems employing FCFS and LCFS. Notice that a high packet generation rate λ degrades the age $\Delta_i(t)$ performance of the FCFS queue, and improves the age performance of the LCFS queue.

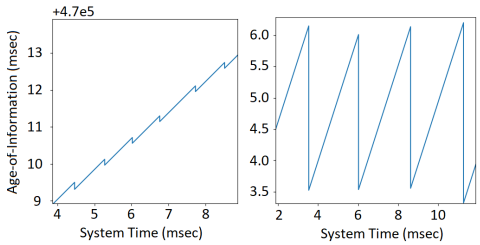


Fig. 7: Age $\Delta_i(t)$ evolution over time in the Raspi testbed with $\lambda = 6$ kHz. On the LHS we have WiFi UDP FCFS and on the RHS we have WiFresh App, which uses LCFS.

Queue size. In all three WiFi UDP FCFS experiments in Table III, the age $\Delta_i(t)$ grows as in Fig. 7 throughout the entire experiment, i.e. for 600 seconds, giving a time-average age of at least 220 seconds. This result suggests that the FCFS queue of the Raspberry Pi did not overflow, which would have helped stabilizing the age. In contrast, in all three WiFi UDP FCFS experiments in Table II, the FCFS queue[4] overflows in the first few seconds, limiting the age $\Delta_i(t)$ growth and resulting in a time-average age of around 0.3 seconds. This suggests that smaller FCFS queues result in better age performance.

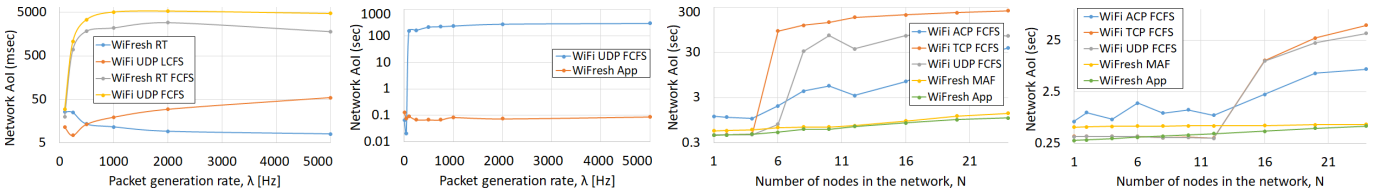
B. Network with Increasing Load

In this section, we consider a network with a destination, a wireless BS and *ten sources* generating packets of 150 bytes with rate λ. In Fig. 8(a), we display the NAoI measurements (in milliseconds on a log scale) for the SDR testbed employing the following communication systems: 1) WiFresh RT; 2) WiFi UDP LCFS; 3) WiFresh RT FCFS; and 4) WiFi UDP FCFS.

By comparing the results of WiFresh RT and WiFi UDP FCFS for $\lambda \geq 500$ Hz, we can see that *WiFresh RT improves information freshness by (at least) a factor of* 200 *when compared to an equivalent standard WiFi network*. To understand how much of this improvement is due to the queueing discipline and how much is due to the multiple access mechanism, we draw additional comparisons. By comparing WiFresh RT and WiFi UDP LCFS, both of which use LCFS queues, we can assess the impact of the multiple access mechanism on NAoI. As expected, the improvement of Polling over Random Access increases with the network congestion. In particular, for $\lambda = 5$ kHz, WiFresh RT improves age by a factor of 7 when compared to WiFi UDP LCFS. To assess the impact of queueing, we compare WiFresh RT and WiFresh RT FCFS, both of which use Polling with MW policy. For $\lambda \geq 500$ Hz, the LCFS queue improves information freshness by (at least) a factor of 100 when compared to the FCFS queue. *Both the queueing discipline and the multiple access mechanism improve NAoI significantly*, but the effect of queueing is clearly dominant.

In Fig. 8(b), we display the expected NAoI measurements (in seconds on a log scale) for the Raspi testbed employing

[4]The transmission queue of the SDR can store one megabyte of data. Notice that for $\lambda = 5$kHz we are generating 0.75 megabyte per second.



(a) SDR testbed with $N = 10$ sources generating packets of 150 bytes with increasing rate λ.

(b) Raspi testbed with $N = 10$ sources generating packets of 150 bytes with increasing rate λ.

(c) Raspi testbed with increasing number of sources N generating position information and images.

(d) Raspi testbed with increasing number of sources N generating position information and inertial measurements.

Fig. 8: Time-average NAoI measurements for the SDR testbed and Raspi testbed shown in Fig. 1.

the following communication systems: 1) WiFresh App; and 2) WiFi UDP FCFS. The results in Fig. 8(b) show that for $\lambda \geq 100$ Hz, *WiFresh App improves information freshness by three orders of magnitude when compared to an equivalent standard WiFi network*. We note that WiFi UDP FCFS performs differently in the Raspi and SDR testbeds due to differences in the platforms, and in particular due to *differences in the FCFS queue sizes*. The large FCFS queues at the Raspberry Pis have a negative effect on WiFi UDP FCFS, which amplifies the performance gain of WiFresh App at high packet generation rates λ.

No need for congestion control. The results in Figs. 8(a) and 8(b) show that the combination of LCFS queues and Polling mechanism with MW policy is the *only architecture in which a higher rate λ leads to a lower NAoI*, meaning that the WiFresh architecture eliminates the need for controlling the packet generation rate at the sources. Notice that any of the other three architectures, which employ either FCFS queues or Random Access, need to control λ in order to minimize NAoI.

C. Network with Increasing Size

In this section, we consider a network with a destination, a wireless BS and N sources, each source generates up to three types of information updates: positions information of 50 bytes at 1 Hz, inertial measurements of 20 bytes at 100 Hz, and images of 19 kbytes at 2 Hz. Notice that a network with N physical sources can have up to $3N$ sources of information, each source of information with its own independent instance of WiFresh App.

In Figs. 8(c) and 8(d), we display the NAoI (in seconds on a log scale) for the Raspi testbed employing the following communication systems: 1) WiFresh App; 2) WiFresh MAF; 3) WiFi UDP FCFS; 4) WiFi TCP FCFS; and 5) WiFi ACP FCFS. In Fig. 8(c), we consider sources generating both position information and images, and in Fig. 8(d), we consider sources generating both position information and inertial measurements.

TCP over WiFi. The results in Figs. 8(c) and 8(d) show that WiFi TCP FCFS has the worst performance in terms of AoI. TCP provides reliable and in-order packet delivery by requesting retransmissions and rearranging out-of-order packets before forwarding them to the Application layer. Both of these features can degrade information freshness, especially when sources are generating packets at high rates.

ACP over WiFi. ACP dynamically adapts the packet generation rates at the sources (by regularly discarding some of the packets) in order to drive the underlying WiFi network to the point of minimum AoI. The results in Figs. 8(c) and 8(d) show that, for $N = 20$, WiFi ACP FCFS improves NAoI by a factor of four when compared to WiFi UDP FCFS; in turn WiFresh App improves NAoI by (at least) a factor of forty when compared to WiFi UDP FCFS.

Impact of scheduling policy. The only difference between WiFresh App and WiFresh MAF is the scheduling policy. MAF schedules the source with highest value of $\hat{\Delta}_i(t)$, and neglects information about channel conditions $\hat{p}_i(t)$ and about the HoL packet at the source's queue $\hat{H}_i(t)$. For this reason, MAF can often poll sources with poor channel condition or an empty queue, what degrades its NAoI performance. This is a main reason for the performance gap between WiFresh MAF and WiFresh App in Figs. 8(c) and 8(d).

Impact of traffic load. The results in Fig. 8(c) show that for $N \geq 16$, WiFresh App improves NAoI by a factor of 65 when compared to WiFi UDP FCFS, and by a factor of 230 when compared to WiFi TCP FCFS. The results in Fig. 8(d) show that for $N \geq 16$, WiFresh App improves information freshness by a factor of 20 when compared to either WiFi UDP FCFS or WiFi TCP FCFS. The improvement is more evident in Fig. 8(c) since cameras generate more traffic than IMUs. In particular, the camera generates approximately 304 kbits per second per source while the IMU generates approximately 16 kbits per second per source.

Summary. From the measurements in this section, we can see that: 1) the more congested the network, the more prominent is the superiority of WiFresh when compared with WiFi in terms of NAoI; 2) the average NAoI in a WiFresh network scales gracefully with the packet generation rate λ, as seen in Sec. V-B, and with the number of sources N, as seen in Sec. V-C; and 3) WiFresh RT achieves the highest performance in terms of throughput and average NAoI, while WiFresh App achieves high performance and can be easily integrated into time-sensitive applications that already run over WiFi, as discussed in Sec. IV-D.

VI. Final Remarks

In this paper, we propose WiFresh: an unconventional network architecture that scales gracefully, achieving near optimal information freshness in wireless networks of any size N, regardless of the level of congestion λ, even when the

network is overloaded. The superior performance of WiFresh is due to the combination of three elements: Last-Come First-Served queues, Polling Multiple Access mechanism, and Max-Weight scheduling policy. The choice of each of these elements is underpinned by theoretical research. We propose and realize two strategies for implementing WiFresh: WiFresh Real-Time, which is designed to maximize performance, and is implemented at the MAC layer in a network of FPGA-enabled SDRs using hardware-level programming; and WiFresh App which is designed to lower the barriers to adoption, and is implemented at the Application layer, without modifications to lower layers of the networking protocol stack, in a network of Raspberry Pis using Python 3. WiFresh App runs over UDP and standard WiFi, making it easy to integrate into time-sensitive applications that are implemented using WiFi such as [2]–[10]. Our experimental results show that WiFresh can improve the expected network AoI by two orders of magnitude when compared to an equivalent standard WiFi network. Moreover, our results show that the more congested the network, the more prominent is the superiority of WiFresh when compared with WiFi in terms of information freshness.

ACKNOWLEDGMENTS

We thank Mohammad Alizadeh and Ping-Chun Hsieh for their helpful suggestions and feedback. This work was supported by NSF Grant CNS-1713725 and by Army Research Office (ARO) grant number W911NF-17-1-0508.

REFERENCES

[1] I. Kadota, M. S. Rahman, and E. Modiano, "Poster: Age of information in wireless networks: from theory to implementation," in *Proc. of ACM MobiCom*, 2020.

[2] P. Valerio, "Amazon robotics: IoT in the warehouse," online: https://www.informationweek.com/strategic-cio/amazon-robotics-iot-in-the-warehouse/d/d-id/1322366, 2015.

[3] J. Gozalvez, M. Sepulcre, and R. Bauza, "IEEE 802.11p vehicle to infrastructure communications in urban environments," *IEEE Communications Magazine*, 2012.

[4] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, 2019.

[5] P. Urcola, M. T. Lázaro, J. A. Castellanos, and L. Montano, "Cooperative minimum expected length planning for robot formations in stochastic maps," *Robotics and Autonomous Systems*, 2017.

[6] F. Mascarich, H. Nguyen, T. Dang, S. Khattak, C. Papachristos, and K. Alexis, "A self-deployed multi-channel wireless communications system for subterranean robots," in *Proc. of IEEE AeroConf*, 2020.

[7] N. Mahdoui, V. Frémont, and E. Natalizio, "Communicating multi-UAV system for cooperative SLAM-based exploration," *Journal of Intelligent & Robotic Systems*, 2020.

[8] M. T. Lázaro, L. M. Paz, P. Pinies, J. A. Castellanos, and G. Grisetti, "Multi-robot SLAM using condensed measurements," in *Proc. of IEEE/RSJ IROS*, 2013.

[9] L. Matignon and O. Simonin, "Multi-robot simultaneous coverage and mapping of complex scene - comparison of different strategies," in *Proc. of ACM AAMAS*, 2018.

[10] D. Vasisht, Z. Kapetanovic, J. Won, X. Jin, R. Chandra, S. Sinha, A. Kapoor, M. Sudarshan, and S. Stratman, "FarmBeats: An IoT platform for data-driven agriculture," in *Proc. of NSDI*, 2017.

[11] T. Shreedhar, S. Kaul, and R. D. Yates, "An age control transport protocol for delivering fresh updates in the internet-of-things," in *Proc. of IEEE WoWMoM*, 2019.

[12] M. Costa, M. Codreanu, and A. Ephremides, "On the age of information in status update systems with packet management," *IEEE Trans. IT*, 2016.

[13] S. Kaul, R. D. Yates, and M. Gruteser, "Status updates through queues," in *Proc. of IEEE CISS*, 2012.

[14] A. M. Bedewy, Y. Sun, and N. B. Shroff, "The age of information in multihop networks," *IEEE/ACM ToN*, 2019.

[15] C. Joo and A. Eryilmaz, "Wireless scheduling for information freshness and synchrony: Drift-based design and heavy-traffic analysis," in *Proc. of IEEE WiOpt*, 2017.

[16] I. Kadota and E. Modiano, "Minimizing the age of information in wireless networks with stochastic arrivals," in *Proc. of ACM MobiHoc*, 2019.

[17] I. Kadota, A. Sinha, E. Uysal-Biyikoglu, R. Singh, and E. Modiano, "Scheduling policies for minimizing age of information in broadcast wireless networks," *IEEE/ACM ToN*, 2018.

[18] Y. Sun, I. Kadota, R. Talak, and E. Modiano, *Age of Information: A New Metric for Information Freshness*. Morgan & Claypool, 2019.

[19] S. Kaul, R. D. Yates, and M. Gruteser, "Real-time status: How often should one update?" in *Proc. of IEEE INFOCOM*.

[20] L. Huang and E. Modiano, "Optimizing age-of-information in a multi-class queueing system," in *Proc. of IEEE ISIT*, 2015.

[21] R. D. Yates, E. Najm, E. Soljanin, and J. Zhong, "Timely updates over an erasure channel," in *Proc. of IEEE ISIT*, 2017.

[22] P. Mayekar, P. Parag, and H. Tyagi, "Optimal lossless source codes for timely updates," in *Proc. of IEEE ISIT*, 2018.

[23] Q. He, D. Yuan, and A. Ephremides, "Optimizing freshness of information: On minimum age link scheduling in wireless systems," in *Proc. of IEEE WiOpt*, 2016.

[24] I. Kadota, A. Sinha, and E. Modiano, "Scheduling algorithms for optimizing age of information in wireless networks with throughput constraints," *IEEE/ACM ToN*, 2018.

[25] Y.-P. Hsu, E. Modiano, and L. Duan, "Age of information: Design and analysis of optimal scheduling algorithms," in *Proc. of IEEE ISIT*, 2017.

[26] R. Talak, S. Karaman, and E. Modiano, "Optimizing information freshness in wireless networks under general interference constraints," in *Proc. of ACM MobiHoc*, 2018.

[27] S. Kaul and R. D. Yates, "Status updates over unreliable multiaccess channels," in *Proc. of IEEE ISIT*, 2017.

[28] N. Pappas, J. Gunnarsson, L. Kratz, M. Kountouris, and V. Angelakis, "Age of information of multiple sources with queue management," in *Proc. of IEEE ICC*, 2015.

[29] Y. Sun, E. Uysal-Biyikoglu, and S. Kompella, "Age-optimal updates of multiple information flows," in *IEEE INFOCOM workshop*, 2018.

[30] C. Kam, S. Kompella, and A. Ephremides, "Experimental evaluation of the age of information via emulation," in *Proc. of IEEE MILCOM*, 2015.

[31] E. Altman, R. El-Azouzi, D. S. Menasche, and Y. Xu, "Forever young: Aging control for hybrid networks," in *Proc. of ACM MobiHoc*, 2019.

[32] A. Franco, E. Fitzgerald, B. Landfeldt, N. Pappas, and V. Angelakis, "LUPMAC: a cross-layer MAC technique to improve the age of information over dense WLANs," in *Proc. of IEEE ICT*, 2016.

[33] A. Baiocchi and I. Turcanu, "A model for the optimization of beacon message age-of-information in a VANET," in *Proc. of ITC*, 2017.

[34] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: A new concept, metric, and tool," *Foundations and Trends in Networking*, 2017.

[35] C. Sönmez, S. Baghaee, A. Ergişi, and E. Uysal-Biyikoglu, "Age-of-information in practice: Status age measured over TCP/IP connections through WiFi, ethernet and LTE," in *Proc. of IEEE BlackSeaCom*, 2018.

[36] S. Kaul, M. Gruteser, V. Rai, and J. Kenney, "Minimizing age of information in vehicular networks," in *Proc. of IEEE SECON*, 2011.

[37] A. M. Bedewy, Y. Sun, and N. B. Shroff, "Minimizing the age of information through queues," *IEEE Trans. IT*, 2019.

[38] M. Harchol-Balter, *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press, 2013.

[39] K. S. Kwak, S. Ullah, and N. Ullah, "An overview of IEEE 802.15.6 standard," in *Proc. of the IEEE ISABEL*, 2010.

[40] E. Biton, D. Sade, D. Shklarsky, M. Zussman, and G. Zussman, "Challenge: CeTV and Ca-Fi - cellular and Wi-Fi over CATV," in *Proc. of ACM MobiCom*, 2005.

[41] D. Mills, J. Martin, J. Burbank, and W. Kasch, "Network time protocol version 4: Protocol and algorithms specification," Internet Requests for Comments, RFC 5905, 2010.

[42] N. Instruments, "LabVIEW communications 802.11 application framework 3.0," online: http://www.ni.com/manuals/, 2019.

[43] D. Assencio, "Nmea generator," online: https://nmeagen.org/, 2016.