Optimal Transmission Scheduling in Data-Intensive Audio Sensor Networks

Alvin C. Valera*, Niels Clayton†, Winston K.G. Seah*, and Tao Zheng‡
*{fistname.lastname}@ecs.vuw.ac.nz, †claytoniel@myvuw.ac.nz
School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
‡zhengtao@bjtu.edu.cn
School of Electronic and Information Engineering, Beijing Jiaotong University, China

***Abstract*—We consider the problem of scheduling audio data transmissions in data-intensive audio sensor networks for animal tracking where the sensor nodes must use WiFi duty cycling to reduce power consumption. WiFi duty cycling entails a startup cost due to probing, authentication, association, and host configuration which are significant and can reach more than 10 seconds. As such, transmission scheduling is not trivial because a naïve approach of switching on WiFi whenever there is a data to send would result in excessive energy consumption overhead. We model duty cycling after an *M/G/1 queue with removable server*, formulate the optimization problem considering both energy and latency overheads, and obtain the optimal N^* by which the interface should be switched on to schedule data transmission. We propose Optimal Threshold-based Transmission Scheduling (OTTS), a low-complexity algorithm for determining the optimal threshold and commencing transmission. Experiments and trace-based simulations show that OTTS can yield substantial reduction in power consumption that is controllable through an energy-latency trade-off parameter. Compared with interval-based scheduling, OTTS provides lower delay which is more significant at tighter power consumption constraints.**

I. Introduction

Animal tracking and monitoring enables biologists to gain insights to the movement of animals in their natural habitats which is important in many applications such as conservation, health, and food [1]. Advances in computing, communications and electronics have made it possible to remotely track a wide variety of animals and over long periods of time. In this research, we consider an *audio sensor network* for monitoring the presence of certain animal species in a region of interest using their acoustic signature. The network consists of sensor nodes equipped with microphone array for capturing audio and direction of arrival (DOA), and a gateway for performing identification and localization of the audio source. As shown in Fig. 1, sensor nodes that detect acoustic energy record the sound for a duration of T seconds and transmits them as audio files to a gateway which performs the necessary algorithms.

Audio sensor networks are data-intensive as they require the regular transfer of large volumes of data from sensor nodes to gateway. Compared to low data rate sensor networks which are vastly studied in the literature and supported by low-power wireless networking such as LoRa [2] and IEEE 802.15.4 [3], data-intensive sensor networks require the use of higher data rate technologies such as IEEE 802.11 (WiFi) [4]. As WiFi power consumption is significantly higher, its operation needs to be duty cycled to conserve energy and prolong the lifetime of the sensor nodes [5].

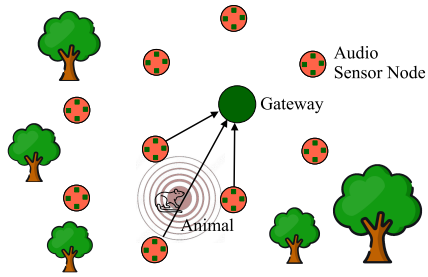


Fig. 1: Audio sensor network for tracking animals of interest in their habitat. Sensor nodes that detect acoustic energy record the sound for a duration of T seconds and transmits them to a gateway which performs identification and localization.

Duty cycling necessitates transmission scheduling as the WiFi interface is not always available for communication with the gateway. When a sensor node has an audio file to send, the most straightforward approach is to immediately schedule the transmission, i.e., switch on the interface and send the file to the gateway. Unfortunately, switching the interface on entails a *startup cost* as it needs to perform probing, authentication, association, and host configuration which has been known to be significant and can reach more than 10 seconds [5]–[7]. It therefore makes more sense to commence transmission when there are several files in the queue as this will result in "sharing" of the startup cost leading to better energy efficiency. However, such an approach introduces a *latency cost* as file delay increases due to the need to wait for several more files.

In this paper, we study the problem of determining the optimal number of files N^* that need to be queued before scheduling transmission to the gateway. It turns out that this problem can be formulated using an *M/G/1 queue with removable server* [8], [9]. The key contributions of this paper are as follows:

- Using the M/G/1 queue with removable server model, we formulate the problem of determining the optimal threshold N^* as a minimization of total expected cost which considers both startup and latency costs.
- A key advantage of the queueing theoretic formulation is that it is possible to obtain the optimal solution in

closed form. Our solution is in agreement with the more general result from the literature [9]. The optimal solution contains an *energy-latency trade-off factor* $\mathcal{E}$ which can be adjusted to satisfy application requirements.
- We propose a low-complexity algorithm called Optimal Threshold-based Transmission Scheduling (OTTS) for determining the optimal threshold and commencing transmission. OTTS uses exponentially-weighted moving average (EWMA) to estimate file generation and transmission rates. Experiments and trace-based simulations show that OTTS can provide reduction in power consumption that is controllable through $\mathcal{E}$. Compared with interval-based scheduling, OTTS provides better delay especially at tighter power consumption constraints.

The rest of the paper is organized as follows: In Section II, we discuss the related work. In Section III, we show how transmission scheduling in conjunction with WiFi duty cycling can be modeled after an M/G/1 queue with removable server and formulate the optimization problem accordingly. In Section IV, we solve the optimization problem and introduce OTTS, whereas in Section V, we present the results from experiments and trace-based simulations. Finally, we conclude the paper in Section VI.

II. Related Work

Audio sensor networks belong to a class of wireless sensor networks (WSNs) known as multimedia sensor networks. Unlike traditional WSNs, multimedia sensor networks are designed to carry voluminous data such as digital images, video and audio, from sensors that capture the phenomenon being monitored [10]. Big data pose various challenges to traditional WSNs [11] [12] as their transmission require significant network bandwidth that exceeds the capacity of typical WSN wireless technology such as IEEE 802.15.4, requiring higher bandwidth technology like IEEE 802.11 which consumes more energy. But while the latter's power consumption is higher, recent results point out that it is more energy efficient than other low power low data rate technologies in high bitrate applications [5], [13].

Energy-efficient scheduling schemes for periodic data collection have been proposed but computationally complex to produce optimal schedules, hence, sub-optimal solutions based on heuristics are adopted [14]. Moreover, existing duty cycling mechanisms for WSNs [15], [16] are not suitable for audio sensor networks as they target low bitrate sensing applications and mostly designed to operate at the MAC layer.

III. Motivation and Problem Formulation

Our audio sensor network is data-intensive as large audio files need to be regularly transferred from the sensor node to the gateway. To illustrate, a mono audio recording for T seconds sampled at 48,000 Hz using 16 bits per channel would require $48000 \times 16 \times T = 768000T$ bits. Even with lossless compression which can provide an average compression ratio of 3:1 [17], around $256T$ kilobits of data still need to be transmitted to the gateway. Sending such a payload over LoRa and IEEE 802.15.4 is simply infeasible.

The high bitrate requirement of the application necessitates the use of high data rate wireless technologies such as WiFi. Unfortunately, WiFi power consumption is significantly higher compared to IEEE 802.15.4 and LoRa and continuously operating it can lower the sensor node lifetime. *Hence, to converse as much energy as possible, there is a need to duty cycle the WiFi interface*: switch it on only when there are audio files to be sent to the gateway, and switch it off at other times.

A. WiFi Duty Cycling Startup Cost

Duty cycling the WiFi interface on and off is not straightforward as it entails overheads. When the interface is powered up, it needs to perform several steps to establish a link to the target access point, namely, probing, authentication, association, and host configuration [5]–[7]. For brevity, we shall refer to these steps as the *startup process*. Note that the time taken by this process is not insignificant as it could take more than 10 seconds [6]. We performed experiments to measure the WiFi startup process time of ESP32, Raspberry Pi 3 B+, and Raspberry Pi Zero W and found them to be quite significant as previously reported.

B. Problem Formulation

Consider a sensor node v that generates files following a Poisson distribution with an average rate of λ. Instead of immediately sending every generated file to the gateway node w, they are inserted to a queue. The storage capacity of the node is much larger than the individual file sizes, hence we can model the queue to have infinite capacity.

Node v transitions between `wait` and `send` states. In the former, v does not perform transmission even if its queue is not empty while in the latter, v switches on its IEEE 802.11 interface, transmits all files from its queue, and switches off the interface. Each file is sent to w at an average rate μ following a general distribution. The operation of v can therefore be modelled after an *M/G/1 queue with removable server*, i.e. the server is switched on when there are at least N items waiting in the queue, and switched off when there are none present [8], [9]. In this model, N is referred to as the *threshold*. To clarify is mode of operation, we show an example file generation and transmission schedule in Fig. 2 where $N = 3$.

Because of the association process, we associate a per unit time startup cost C_s which is incurred every time v switches to `send` state. Moreover, each file in the system needs to wait for transmission, thereby incurring a per unit time latency cost C_l. Let $E[S]$ and $E[L]$ denote the expected number of `send` cycles and expected number of files waiting to be transmitted, respectively. Then the total expected cost per unit time $E[C_T]$ is given by

$$E[C_T] = C_s E[S] + C_l E[L]. \tag{1}$$

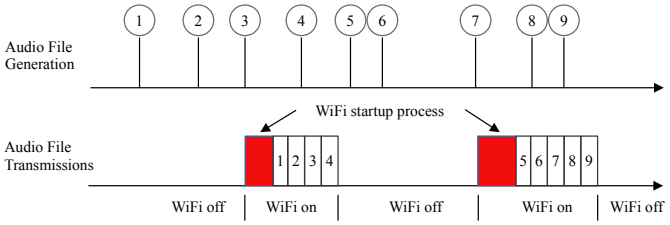


Fig. 2: Audio file generation and transmission example with WiFi duty cycling: Horizontal lines denote time axes. Audio files 1–9 are generated at times indicated by the respective vertical lines. Transmission is initiated when there are at least 3 files. WiFi is off when the node is not transmitting. When WiFi is switched on, it must first complete the startup process before it can send the files in its queue. After sending all the files, it switches WiFi off again.

For an M/G/1 queue with removable server, Wang and Ke [9] obtained expressions for $E[S]$ and $E[L]$ as functions of the threshold N, namely,

$$E[S] = \frac{\lambda(1-\rho)}{N} \tag{2}$$

and

$$E[L] = \frac{N-1}{2} + \rho + \frac{\lambda^2 E[\mathcal{S}^2]}{2(1-\rho)} \tag{3}$$

where $\rho = \lambda/\mu$ and $E[\mathcal{S}^2]$ is the second moment of the service time. Substituting these expressions to Eq. (1) yields

$$E[C_T] = C_s \frac{\lambda(1-\rho)}{N} + C_l \left[\frac{N-1}{2} + \rho + \frac{\lambda^2 E[\mathcal{S}^2]}{2(1-\rho)} \right]. \tag{4}$$

We want to minimize $E[C_T]$ with respect to N, hence, we remove terms that do not contain N. Denoting this cost function as $F(N)$, we now have

$$F(N) = C_s \frac{\lambda(1-\rho)}{N} + C_l \frac{N}{2}. \tag{5}$$

Finally, our problem is to find the optimal threshold N^* which minimizes $F(N)$, that is

$$N^* = \arg\min_N \left\{ C_s \frac{\lambda(1-\rho)}{N} + C_l \frac{N}{2} \right\}. \tag{6}$$

IV. Optimal Transmission Scheduling

In this section, we present the optimal solution to the minimization problem in Eq. (6) and devise an algorithm that performs optimal transmission scheduling using the optimal solution.

A. Optimal Solution

To solve the minimization problem in Eq. (6), we need to solve for N when $F'(N) = 0$ and show that $F''(N^*) \geq 0$. Differentiating $F(N)$ and equating to 0, we have

$$F'(N) = \frac{C_l}{2} - C_s \frac{\lambda(1-\rho)}{N^2} = 0.$$

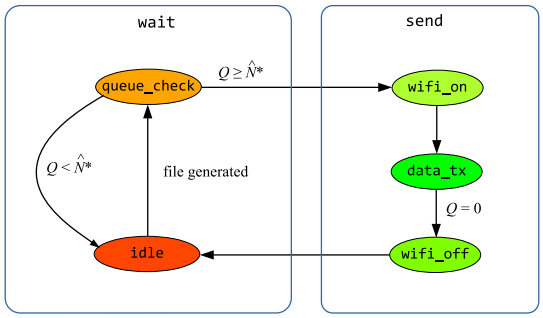


Fig. 3: OTTS state transition diagram.

Solving for N yields the optimal threshold

$$N^* = \sqrt{\frac{2C_s\lambda(1-\rho)}{C_l}}. \tag{7}$$

This result is consistent with the general result obtained by Wang and Ke [9]. Differentiating $F'(N)$ further results in

$$F''(N) = 2C_s \frac{\lambda(1-\rho)}{N^3}$$

which is obviously positive at $N = N^* > 0$. This second derivative test confirms that $F(N^*)$ is a minimum.

B. Energy-Latency Trade-Off Factor

We now introduce the quantity

$$\mathcal{E} := C_s/C_l \tag{8}$$

which we call the *energy-latency trade-off factor*. This simplifies Eq. (7) to

$$N^* = \sqrt{2\lambda(1-\rho)\mathcal{E}}. \tag{9}$$

This quantity can be used as a tuning parameter for trading off between energy-efficiency and latency: a low $\mathcal{E}$ means that latency costs are higher whereas a high $\mathcal{E}$ means that the energy costs are higher. We can select a suitable value for $\mathcal{E}$ based on our priority: (*i*) for better energy-efficiency, choose high $\mathcal{E}$; or (*ii*) for better latency, choose low $\mathcal{E}$.

C. Optimal Threshold-based Transmission Scheduling (OTTS)

Using the results obtained in the preceding subsection, we now present the algorithm for achieving optimal transmission scheduling.

1) File Generation Rate and Transmission Time Estimation: Note that the computation of N^* requires the average file generation rate λ and queue utilization rate $\rho = \lambda/\mu$ where μ is the average file transmission rate. This makes the performance of the algorithm to be heavily dependent on the estimation of λ and μ. To estimate the former, we employ a sliding-window, exponentially-weighted moving average (EWMA) estimator

$$\hat{\lambda}(k+1) = \alpha \frac{n(k)}{T_{\text{win}}} + (1-\alpha)\hat{\lambda}(k), \tag{10}$$

where α is a constant, $\hat{\lambda}(k)$ is the estimate in the current window, $n(k)$ is the number of files generated in the current

window, and T_{win} is the sliding window duration. To estimate the latter, we use an EWMA estimator

$$\hat{\mu}(k+1) = \beta\frac{1}{t_{\text{tx}}} + (1-\beta)\hat{\mu}(k), \tag{11}$$

where β is a constant, $\hat{\mu}(k)$ is the last estimate, and t_{tx} is the transmission time of the last transmitted file. Note that unlike $\hat{\lambda}(k+1)$ which is computed at every window, $\hat{\mu}(k+1)$ is computed after every successful file transmission. The low computational overhead makes OTTS a low-complexity algorithm which can be executed at every node in real time.

2) State Transition Diagram: Fig. 3 shows the state transition diagram of OTTS. Q denotes the queue length at node v. Whenever a file is generated, v transitions to `queue_check` where it will check the number of packets in the transmit queue. If $Q < \hat{N}^*$, the node goes back to `idle` state. $\hat{N}^*$ is the optimal threshold using the estimates $\hat{\lambda}$ and $\hat{\mu}$ and rounded up to an integer, that is,

$$\hat{N}^* = \left\lceil \sqrt{2\hat{\lambda}(1-\hat{\lambda}/\hat{\mu})\mathcal{E}} \right\rceil.$$

When $Q \geq \hat{N}^*$, v transitions to `wifi_on` where it will switch on the WiFi interface. After that, v starts transmitting files in its queue to t. Once all packets have been transmitted, it will transition to `wifi_off` state where it will switch off the WiFi interface. After this, it will transition back to the `idle` state.

V. Evaluation

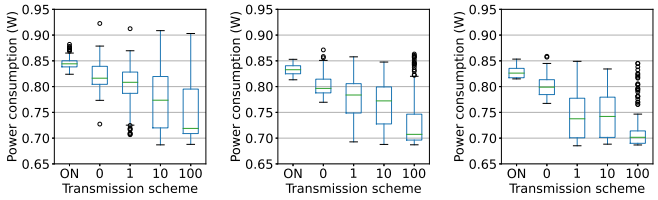
To evaluate OTTS, we performed both experimentation and simulation. The aim of experimentation was to measure energy savings, if any, due to the use of OTTS on a single sensor node as it generates and transmits files to a gateway. The simulation, in the meantime, was aimed at examining and comparing the performance of the scheme under a wider range of parameters using an empirical power consumption model.

A. Experiments

1) Setup: We assembled a sensor node using Raspberry Pi Zero W and *minidsp uma-8* (a microphone array with 7 MEMS microphones) and implemented OTTS in Python (both α and β were set to 0.5). The sensor node communicated with a gateway, based on Raspberry Pi, which also acted as the WiFi access point. The command `rfkill` was used to duty cycle WiFi, whereas `ftp` was used for transferring files. Raspberry Pi Zero W average power consumption was measured using Agilent Technologies U3402A and logged every 10 seconds. The sensor node recorded audio using the parameter listed on Table I following an exponentially distributed interval. An experiment ran for 1800 seconds.

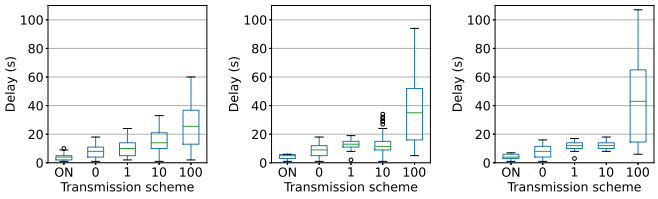
TABLE I: Experiment audio capture parameters

Parameter	Value
Resolution	16 bits
Sampling rate	48,000 Hz
Encoding	Linear PCM
File format	WAV
Recording duration	5 seconds



(a) $1/\lambda = 10$ s (b) $1/\lambda = 20$ s (c) $1/\lambda = 30$ s

Fig. 4: Power consumption of the different transmission schemes at different generation intervals $1/\lambda$.



(a) $1/\lambda = 10$ s (b) $1/\lambda = 20$ s (c) $1/\lambda = 30$ s

Fig. 5: Delay of the different transmission schemes at different generation intervals $1/\lambda$.

2) Power Consumption: Figs. 4 and 5 show the power consumption and delay of OTTS and ON under different file generation intervals ($1/\lambda$). ON refers to the scheme where the WiFi interface is always switched on, whereas 0, 1, 10, and 100 refer to OTTS where $\mathcal{E}$ is set to 0, 1, 10, and 100, respectively. $\mathcal{E} = 0$ is equivalent to forcing the WiFi interface to switch on and transmit at every file generation time. Regardless of the generation interval, duty cycling had noticeable effect on the power consumption: the larger $\mathcal{E}$ is, the lower the power consumption. Comparing the schemes at opposite ends, i.e. ON and $\mathcal{E} = 100$, we observed that the median power consumption of the latter was 13 mW or 15% lower than the former for all generation intervals. Note however that even with the use of duty cycling ($\mathcal{E} = 0$ to $\mathcal{E} = 100$), we could see numerous data points with power consumption comparable to ON. This was due to the fact that the power consumption measurements were logged every 10 seconds. The high power consumption data points were due the activation of WiFi for transmission in the last 10 seconds.

3) Delay: Fig. 5 shows the effect of duty cycling on the delay. Without duty cycling, median delay was well below 4 seconds for all file generation intervals whereas with duty cycling, we saw delays increased with increasing $\mathcal{E}$. We also observed that as generation interval increased, $\mathcal{E} = 100$ showed wider delay variation. We expected this behaviour because at higher intervals, OTTS had to wait longer to reach optimal number of files. It was much more noticeable for $\mathcal{E} = 100$ because it had to wait for significantly more files than $\mathcal{E} = 0$, $\mathcal{E} = 1$ and $\mathcal{E} = 10$. Once again, comparing delays of the schemes at opposite ends, i.e. ON and $\mathcal{E} = 100$, we observed that the median delay of the latter was higher than the former by 25, 32, and 43 s for file generation intervals 10, 20, and 30 s, respectively. These results, combined with the results in Fig. 4 clearly demonstrate the effectiveness of using $\mathcal{E}$ for trading off between energy efficiency and latency.

B. Simulations

We performed *trace-based simulations* to investigate the performance of OTTS under a wider variety of conditions. We collected logs of 1,000 WiFi startup processes and 10,000 file transfers from the sensor node to the gateway from 10 different positions. Files of random size between 1-2 MB were generated. In each WiFi startup process, duration and system load were logged whereas in each file transfer, file size, file transfer duration, and system load were logged. We then used these logs to drive the simulations.

We simulated WIFI-ON (no duty cycling), OTTS, and interval-based transmission scheduling. The latter represents the other class of scheduling schemes for aggregating multiple files into a single transmission session whereby a node checks the queue at fixed intervals. If there is at least one file in the queue, WiFi is activated and all files in the queue (including those generated during the transmission) are transmitted; otherwise, the node will check again after the fixed interval. Table II lists the key parameters used in the simulations.

1) Power Consumption Model: We applied the empirical model proposed by Kaup et al. [18] for Raspberry Pi to obtain the power consumption with and without WiFi duty cycling. Let τ, $\tau_{\text{up},k}$ and K denote the simulation time, time used for uploading file k, and number of files uploaded, respectively. Without duty cycling, the average power consumption P_{on} of a sensor node can be obtained using Eq. (12) from Kaup et al. [18],

$$P_{\text{on}} = P_{\text{idle}} + P_{\text{WiFi,idle}} + \frac{1}{\tau}\sum_{k=1}^{K}\tau_{\text{up},k}[P_{\text{CPU}}(u_k) + P_{\text{WiFi,up}}(r_k)] \quad (12)$$

where P_{idle} is the power consumption of the system excluding the interface at load 0, $P_{\text{CPU}}(u_k)$ is the *additional* power consumption due to load of $u_k > 0$, $P_{\text{WiFi,idle}}$ is the *additional* power consumption when the interface is on but not performing any upload, and $P_{\text{WiFi,up}}(r_k)$ is the *additional* power consumption when the interface is uploading file k at throughput r_k. From experiments, Kaup et al. [18] obtained the following results: $P_{\text{idle}} = 1.5778$ W, $P_{\text{CPU}}(u) = 0.181 \cdot u$ W, $P_{\text{WiFi,idle}} = 0.942$ W, and

$$P_{\text{WiFi,up}}(r) = 0.02 + 24.387 \times 10^{-3} r - 1128 \times 10^{-6} r^2.$$

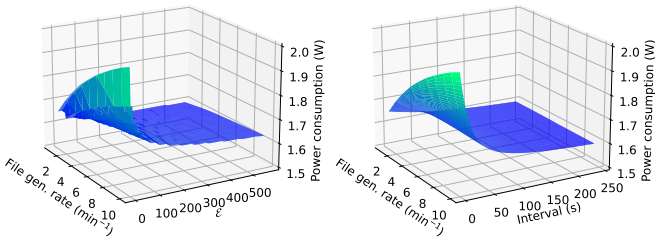
The latter is the second-order WiFi power consumption model for uploads. Now, to obtain the power consumption with duty cycling P_{dc}, we note that since $\tau_{\text{up},k}$ denotes the duration that the interface is uploading file k, it is also denotes the partial time the interface switched on. The other part is due to the startup process time, which we denote $\tau_{\text{start},j}$. Then

$$P_{\text{dc}} = P_{\text{idle}} + \frac{1}{\tau}\sum_{j=1}^{J}\tau_{\text{start,j}}[P_{\text{CPU}}(u_j) + P_{\text{WiFi,idle}}] + \frac{1}{\tau}\sum_{k=1}^{K}\tau_{\text{up},k}[P_{\text{CPU}}(u_k) + P_{\text{WiFi,idle}} + P_{\text{WiFi,up}}(r_k)]. \quad (13)$$

Note that J is the number of times the startup process was initiated, which should be at most K. Note further that for startups, there is no consumption associated to uploading.

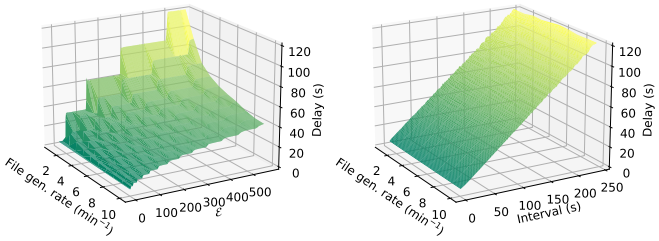
TABLE II: Simulation parameters

Parameter	Value
Simulated duration	3,600 seconds
Seeds per experiment	100
File generation process	Poisson
File generation rate (λ)	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ files/minute
File size	1-2 megabytes
$\mathcal{E}$ (for OTTS)	θ^2 where $\theta \in \{0, 0.1, 0.2, ..., 23.9\}$ (240 parameter values)
Interval (for interval-based)	$\{1, 2, 3, ..., 240\}$ seconds (240 parameter values)



(a) OTTS (b) interval-based

Fig. 6: Power consumption of OTTS and interval-based



(a) OTTS (b) interval-based

Fig. 7: Delay of OTTS and interval-based

2) Delay vs. Power Consumption Trade-off: Figs. 6 and 7 shows the power consumption and delay, respectively, of OTTS and interval-based transmission scheduling. The plots provide insights as to how the schemes trade-off delay with power consumption using $\mathcal{E}$ (for OTTS) and interval (for interval-based). WIFI-ON is not shown as did not provide any trade-off: its delay and power consumption was 0.55 s and 2.51 W on the average, respectively, regardless of λ. Meanwhile, interval-based and OTTS enabled trade-off between delay and power consumption. For interval-based, the trade-off was controlled by the interval length: as the interval length increased, power consumption decreased while delay increased. For OTTS, the trade-off was controlled by $\mathcal{E}$: as $\mathcal{E}$ increased, power consumption decreased while delay increased. Both schemes showed lower power consumption at lower λ because fewer transmissions were scheduled. The shape of the power consumption plots are somewhat similar because the effect of increased $\mathcal{E}$ and interval was to reduce the number of WiFi startup processes J in Eq. (13).

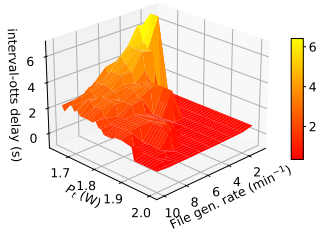


Fig. 8: Delay difference between interval-based and OTTS

While the power consumption plots were somewhat similar, the shape of the delay plots were substantially different. The delay of interval-based was purely determined by the interval whereas both $\mathcal{E}$ and file generation rate affected the delay of OTTS. As expected, the delay for interval-based was half of the interval because of the Poisson arrivals. For OTTS, the delay decreased with increasing λ because at higher λ, the waiting time to attain the optimal N^* was shorter.

3) Performance Comparison: To compare the performance of OTTS and interval-based, we studied their delay performance when the power consumption was constrained by an upper bound P_t. That is, we wanted to

$$\begin{array}{ll} \text{minimize} & \text{Delay} \\ \text{subject to} & P_{\text{dc}} \leq P_t \end{array}$$

where P_t was varied from 1.65 to 2 W. For clarity, we plotted the difference between interval-based and OTTS delay as a function of P_t and λ in Fig. 8. All the differences were positive, indicating interval-based had higher delay than OTTS in all cases. Furthermore, the difference was more significant at tighter power consumption constraints. The main drawback of interval-based is that in general, there is always a delay between the last generated file and the closest transmission schedule. This is not the case with OTTS as its transmission schedule is immediately triggered by the arrival of a file. In the most aggressive case where both schemes schedule a transmission when there is at least one file in the queue, the delay of interval-based consists of the time from file generation to the scheduled transmission time, WiFi startup time, and file transmission time. Whereas for for OTTS, the delay only consists of the latter two. So in order for interval-based to match the delay of OTTS, it must use lower interval lengths which results in more transmission schedules and higher power consumption. We would like to highlight that our results are in line with theoretical expectations that in general, schemes that use interval perform worse compared to schemes that use queue occupancy [19].

VI. Conclusion

We considered the problem of scheduling file transmissions in a data-intensive sensor network for animal tracking wherein the WiFi interface was duty cycled to conserve energy. We formulated the optimization problem using an M/G/1 queue with removable server model and obtained a closed form solution for the optimal number files that must be in the queue before a transmission could be scheduled. We proposed OTTS, a low-complexity algorithm for determining the optimal threshold and commencing transmission. Experiments and trace-based simulations show that OTTS can provide reduction in power consumption that is controllable through $\mathcal{E}$. Compared with interval-based scheduling, OTTS provides better delay especially at tighter power consumption constraints.

References

[1] R. Kays, M. C. Crofoot, W. Jetz, and M. Wikelski, "Terrestrial animal tracking as an eye on life and planet," *Science*, vol. 348, no. 6240, p. aaa2478, 2015.

[2] A. M. Yousuf, E. M. Rochester, B. Ousat, and M. Ghaderi, "Throughput, Coverage and Scalability of LoRa LPWAN for Internet of Things," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–10.

[3] M. Petrova, J. Riihijarvi, P. Mahonen, and S. Labella, "Performance study of IEEE 802.15.4 using measurements and simulations," in *IEEE Wireless Communications and Networking Conference, 2006. WCNC 2006.*, vol. 1, 2006, pp. 487–492.

[4] G. Klimiashvili, C. Tapparello, and W. Heinzelman, "LoRa vs. WiFi Ad Hoc: A Performance Analysis and Comparison," in *2020 International Conference on Computing, Networking and Communications (ICNC)*, 2020, pp. 654–660.

[5] A. Abedi, O. Abari, and T. Brecht, "Wi-le: Can wifi replace bluetooth?" in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, 2019, pp. 117–124.

[6] C. Pei, Z. Wang, Y. Zhao, Z. Wang, Y. Meng, D. Pei, Y. Peng, W. Tang, and X. Qu, "Why it takes so long to connect to a WiFi access point," in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*. IEEE, 2017, pp. 1–9.

[7] V. K. Ramanna, J. Sheth, S. Liu, and B. Dezfouli, "Towards Understanding and Enhancing Association and Long Sleep in Low-Power WiFi IoT Systems," *IEEE Transactions on Green Communications and Networking*, vol. 5, no. 4, pp. 1833–1845, 2021.

[8] M. Yadin and P. Naor, "Queueing systems with a removable service station," *Journal of the Operational Research Society*, vol. 14, no. 4, pp. 393–405, 1963.

[9] K.-H. Wang and J.-C. Ke, "A recursive method to the optimal control of an M/G/1 queueing system with finite capacity and infinite capacity," *Applied Mathematical Modelling*, vol. 24, no. 12, pp. 899–914, 2000.

[10] A. Nauman, Y. A. Qadri, M. Amjad, Y. B. Zikria, M. K. Afzal, and S. W. Kim, "Multimedia Internet of Things: A Comprehensive Survey," *IEEE Access*, vol. 8, pp. 8202–8250, 2020.

[11] Q. Jawhar, K. Thakur, and K. J. Singh, "Recent Advances in Handling Big Data for Wireless Sensor Networks," *IEEE Potentials*, vol. 39, no. 6, pp. 22–27, 2020.

[12] L.-M. Ang and K. P. Seng, "Big Sensor Data Applications in Urban Environments," *Big Data Research*, vol. 4, pp. 1–12, 2016.

[13] B. Snajder, V. Jelicic, and V. Bilas, "Performance Evaluation of IEEE 802.15.4 and 802.11 Protocols for Image Transmission in WSNs," in *European Wireless 2014; 20th European Wireless Conference*, Barcelona, Spain, 14-16 May 2014, pp. 1–6.

[14] S. Kumar and H. Kim, "Energy efficient scheduling in wireless sensor networks for periodic data gathering," *IEEE Access*, vol. 7, pp. 11 410–11 426, 2019.

[15] R. C. Carrano, D. Passos, L. C. Magalhaes, and C. V. Albuquerque, "Survey and taxonomy of duty cycling mechanisms in wireless sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 181–194, 2013.

[16] A. C. Valera, W.-S. Soh, and H.-P. Tan, "Survey on wakeup scheduling for environmentally-powered wireless sensor networks," *Computer Communications*, vol. 52, pp. 21–36, 2014.

[17] M. Hans and R. W. Schafer, "Lossless compression of digital audio," *IEEE Signal processing magazine*, vol. 18, no. 4, pp. 21–32, 2001.

[18] F. Kaup, P. Gottschling, and D. Hausheer, "PowerPi: Measuring and modeling the power consumption of the Raspberry Pi," in *39th Annual IEEE Conference on Local Computer Networks*, 2014, pp. 236–243.

[19] K.-H. Wang, D.-Y. Yang, and W. L. Pearn, "Comparison of two randomized policy M/G/1 queues with second optional service, server breakdown and startup," *Journal of Computational and Applied Mathematics*, vol. 234, no. 3, pp. 812–824, 2010.