Lumped Markovian Estimation for Wi-Fi Channel Utilization Prediction

Sepehr Kazemian
Department of Computing Science
University of Alberta
Edmonton, Canada
skazemia@ualberta.ca

Ioanis Nikolaidis
Department of Computing Science
University of Alberta
Edmonton, Canada
nikolaidis@ualberta.ca

***Abstract*—We present a model to predict the short-term utilization of an IEEE 802.11 channel. We approximate the time-varying utilization process via a Markovian state transition model and subsequently create a lumped representation of the transition matrix. Each lumped state can then be treated as a class. The lumped matrix provides a simpler to understand description of the channel utilization behavior and naturally includes the persistence in one lumped state which resembles the characteristic behavior of naïve predictors (where predicted state equals the current state). We demonstrate that treating the lumped states as classes allows good prediction models to be built using Logistic Regression and Neural Network models. Our results are based on IEEE 802.11 wireless utilization data collected as reported in the channel utilization (CU) field of the QBSS Load Element in Beacon frames. The presented approach can be implemented as an edge computing task, whereby edge nodes calculate the lumped states and train models, informing nearby client devices of the model parameters, allowing them to produce predictions on their own.**

***Index Terms*—channel utilization, IEEE 802.11, Markovian models, lumping, prediction**

I. Introduction

Consider the following, simplified, motivating example: a node turns on and can only perform a single measurement, i.e., receive a nearby access point (AP) – ostensibly, the AP with which it intends to communicate – Beacon transmission informing it of medium utilization (as per 802.11e). Then, based on this measurement it has to predict the channel utilization (CU) for the next time interval. Also, anecdotal evidence (that are confirmed later in this study) point that a naïve predictor, simply predicting the utilization in the next δ seconds to be the same as the measured CU in the most recent δ seconds, works very well in many cases.

A feature of the current study is that we analyze a real Wi-Fi data set collected over a one month period in an academic campus environment. Our main contribution is that of relating the observed behavior to a simplified Markov chain by way of state transition *lumping*. Using the lumped matrix we can perform predictions. The act of lumping states, creates "cliques" of states that, seen collectively as a meta-state, result in the process remaining in the meta-state for prolonged periods of time. Each of our lumped states also represents a range of utilization values. This lumped view is what makes naïve prediction work so well (the process remains in the same state in the next step). With the lumping, we may construct more than one lumped state; hence, a rich set of lumped states ("cliques"), each representing a different utilization range, are produced, while "transient'' states also emerge linking those cliques. The resulting prediction based on the lumped matrix is found to outperform the naïve predictor, especially when it matters the most – during busy-hour traffic.

In our study, we use a set of standard Machine Learning (ML) algorithms to perform class predictions. Note also that no modification of the protocol stack is performed that would render the approach incompatible with IEEE 802.11. Emphasis is placed on building a model from locally-collected utilization data. As such, either the APs directly assist the collection of CU data by e.g., sending CU values to an edge node, or additional "sniffer" collecting CU data and passing them to the edge node.

The rest of the paper is structured as follows. In Section II we described related work and the standards that enabled our work. In Section III we lay out the methodology on both the data collection side, as well as on the generation of the lumped transition matrices. In Section IV we describe the evaluation setup and the alternatives that were evaluated. In Section V we outline the results produced and discuss by comparing and contrasting various schemes. Finally, Section VI provides concluding remarks and directions for future work.

II. Related Work

In recent years, due to the widespread use of APs in buildings, occupancy detection [1], [2], [3] methods based on Wi-Fi traffic have been proposed. For example, Balaji et al. [4] exploits authentication, authorization, and accounting logs of APs in buildings to detect the occupancy but is not concerned about CU statistics. We also assume accessibility to such logs, is not possible, or restricted for privacy reasons, while Beacons reveal no information of client devices. Moreover, in most of environments, it is common for only a subset of APs to be centrally managed (if at all).

Efforts have been made to use APs signals or Wi-Fi management packets (Probe packets) for prediction. Wei Wang, et al. [1] uses a Markov based feedback recurrent neural network for predicting occupancy of a building. They include the Markov

978-3-903176-24-9

model due to stochastic and chronological interdependent features of Wi-Fi data packets. The work bears some resemblance to our work due to the Markovian facet; however, they are not concerned about time-series prediction. More importantly, they include ad-hoc pre-processing steps to filter certain MAC addresses since their model needs to avoid counting MAC addresses belong to a person as several occupants.

Closer to the Cognitive Radio (CR) literature, we find [5] where the authors try to predict the idleness or busyness of a channel, and hence do not attempt a finer-grain view of the CU as we do. Nevertheless, they adopt a non-stationary Hidden Markov Chain to deal with the perceived non-stationarity of the channel but, crucially, the presented validation is only using simulations of primary user following exponential busy and idle distributions. Yet another work that proposed a Markov model to fit real measurements is [6] which used real-time measurements made in the 928-948 MHz pager band, but does not indicate measures to deal with non-stationarity (the period of observations is unclear as well) and, similarly to [5], the objective is the busy/idle distinction of the primary user.

Armed with the observations made regarding Markovian models, we follow an explicit Markov chain construction whose state transition probabilities are time-dependent, further describe in Section III-B. Contrary to busy/idle indicators, we are concerned with the prediction of the degree of utilization.

A. The QBSS Load Element

Based on 802.11e amendment, QoS Basic Service Set (QBSS) Load Elements advertises in every Beacon and Probe Response frames. Virtually all APs in use today (2019) support the 802.11e extensions. One component of the Load Element is channel utilization (CU), whose values from 0 to 255 map linearly to the range from 0% to 100%, and express the utilization of the wireless medium seen by the AP (including the traffic with its own clients). Non-802.11 transmissions are also, indirectly, accounted for, as long as their transmissions are such that the physical carrier-sensing of the AP assesses the medium as being busy.

Note that we do not consider channel selection. Channel selection makes sense for the case where APs can be controlled, and then schemes such as DCA [7] and similar works, e.g., [8], apply. We assume we have no form of control over the APs' operating channels. A per-channel application of what we propose here is possible but channel switching admits a wider set of options, given that one can attempt to minimize the number of tests before picking the right channel – something outside the scope of the current work.

III. Methodology

A. Data Collection

We captured several days of Wi-Fi AP Beacon transmissions from a plethora of APs in a campus building at the University of Alberta. The majority of APs are centrally managed to provide ubiquitous Wi-Fi service, each one of which supports QBSS and transmits approximately 10 beacons per second. The capture of the Beacon frames was performed using inexpensive Wi-Fi sniffers in the 2.4 GHz band. The sniffers would round-robin, staying two seconds in each channel, across the three non-overlapping channels (1, 6, and 11) to capture beacon frames on each one. Not capturing some beacons (for approximately four of every six seconds) was examined and found to have no significant impact. Specifically, to reduce the impact of missing Beacon CU measurements, we used the maximum CU reported in the two seconds of observation interval and use it as the maximum CU over the six seconds interval, as it was found close to the maximum had we continuously listened to the same channel.

B. Discrete-Time Markov Model

Consider time progressing in time steps of length δ (were as noted, $\delta = 6$sec. due to the data collection setup), then the evolution of the CU of an AP on a specific channel from time t to $t + \delta$ will be approximated by a first-order Markov chain. We can trivially produce estimates for the transition probabilities of the transition matrix, $\mathbf{P}$, by counting the corresponding transitions between successive CU values from the collected data, and normalizing accordingly. Technically, $\mathbf{P}$ is a 256×256 matrix. We handle the non-stationarity by considering that each 30-minute interval is characterized by a separate transition matrix, $\mathbf{P}^{(i)}$, where i identifies the 30-minute interval within a day. When determining the transition probabilities of $\mathbf{P}^{(i)}$, we use the transitions in the CU values for the corresponding 30-minute interval only. In the interest of brevity we drop the exponent from $\mathbf{P}^{(i)}$ with the understanding that we are talking about models for 30-minute intervals, and noting places where this is not the case.

Our objective is to, starting from a 256×256 matrix, for each 30 minute interval, to reduce the model to a small (no more than 5×5) matrix which we will subsequently use for predicting the next state. Towards this end, we perform a first grouping in an ad-hoc manner, simply to reduce the data handled in subsequent steps, therefore reducing the computation demands – and can be skipped if computational resources are abundant. Namely, we translate the 256×256 matrix to 26×26 by grouping states $0 - 9$ together, $10 - 19$ together, ..., $240 - 249$ together, and $250 - 255$ together by accumulating the transition probabilities and re-normalizing accordingly. The rationale is that we consider changes in the utilization of $100 \times (10/255) \approx 4$ % as insignificant for our purposes.

C. Quasi-Lumpability

State lumping of a Markov chain reduces a large state space into a smaller one [9]. A discrete time Markov chain is lumpable with respect to a given state space partition $S = \bigcup_i S_i$ with $S_i \bigcap S_j = \varnothing \ \forall i \neq j$ if its transition probability matrix $\mathbf{P}$ satisfies the lumpability condition:

$$\forall S_i, S_j \subset S \quad \forall s \in S_i : \sum_{s' \in S_j} p_{s,s'} = k_{i,j} \quad \forall i, j, \qquad (1)$$

where $p_{s,s'}$ is the one-step transition probability from state s to state s', and $k_{i,j}$ is a constant depending only on i and

j. The $k_{i,j}$ are the elements of the lumped matrix, $\mathbf{K}$. That is, a Markov chain is lumpable if the transition probability from each state in a given partition to another partition is the same. The probability of transitioning from a given state to a partition is equivalent to the sum of the transition probabilities from this given state to each state in the partition.

However, Markov chains are seldom (exactly) lumpable, in that we cannot generally satisfy the equals relation in Equation 1 regardless how we partition the state space. In those cases, we adopt the relaxed definition known as quasi-lumpability [10]. Intuitively, quasi-lumpability allows for the sum of rows of the resulting lumped matrices to not sum up to 1. Formally, a Markov chain is ϵ quasi-lumpable with respect to a given state space partition $S = \bigcup_i S_i$ with $S_i \bigcap S_j = \varnothing \ \forall i \neq j$ if its transition probability matrix $\mathbf{P}$ can be written as $\mathbf{P} = \mathbf{P}^{-} + \mathbf{P}^{\epsilon}$, where the elements of $\mathbf{P}^{-}$ are lower or equal to the corresponding elements of $\mathbf{P}$ and satisfies the lumpability condition:

$$\forall S_i, S_j \subset S \quad \forall s \in S_i : \sum_{s' \in S_j} p^{-}_{s,s'} = k_{i,j} \quad \forall i \neq j \qquad (2)$$

where now, $p^{-}_{s,s'}$ is the one-step transition probability from state s to state s' in the matrix $\mathbf{P}^{-}$ and no element in $\mathbf{P}^{\epsilon}$ is greater than ϵ. A more helpful formulation is to permute the rows and columns of the transition matrix to bring it in the form $\mathbf{P} = diag(\mathbf{P}_{1,1}, \mathbf{P}_{2,2}, ..., \mathbf{P}_{N,N}) + \mathbf{E}$ where $\mathbf{E}$ is an "error" matrix, and the $\mathbf{P}_{i,i}$ represent sub-matrices that correspond to the i–th lumped state (for N lumps) – hence the derived $\mathbf{K}$ matrix would be of $N \times N$. The closeness of the approximation is judged by the norm $\|\mathbf{E}\|_{\infty}$.

The permutations implied by the $\mathbf{P}_{i,i}$-based formulation of lumping allow for lumping together any collection of states. This is meaningless in our setting since via lumping we are trying to produce ranges of utilization values we can collectively consider as a single state. To this end, we add the constraint of lumping only neighboring states together. In addition, because of the limitation in computational resources we set the maximum number of lumped states to be equal to five. We then find a lumped representation with no more than five states that minimizes $\|\mathbf{E}\|_{\infty}$. Finally, after lumping we need to adjust the lumped matrix to include states that were not present (no corresponding measurements) in the collected data. Our strategy is to lump states that were not present in the training set to the closest lumped state, hence the number of lumped states is retained.

IV. Evaluation

Given the set of training data collected over multiple days, comprising of the CU measurements reported by an AP on a channel, we construct the lumped matrix for each 30 minute period of a day. The lumps correspond to utilization ranges, and each range can be seen as a *class*. Thus, the problem is re-framed as predicting the correct class. Part of the data is used for testing the models. The test data are used in the following way: a CU value is read from the captured data and based on it (and depending on the model listed next), the class/range of the next CU is predicted. If the next CU value indeed belongs to this class, the outcome is positive, else negative. In other words, the correct class has to be determined. The next CU value is then revealed and the process continues in this manner. The following are the predictors used.

A. Lumped Markov Chain

Sequentially, a measured CU value is revealed and hence its class as well. The corresponding 30 minute interval lumped matrix is used to predict, by means of a transition chosen with the given probabilities of the lumped matrix, what will be the next class/range.

B. Naïve

This predictor returns as prediction the same class as the class of the measured CU. The classes are as defined by the lumped model corresponding to the particular 30 minute interval.

C. Logistic Regression

Another model used is that of standard Multinominal Logistic Regression [11], where the generated probability value is converted to a prediction for the class. The classes used are the same derived by the lumped Markov chain for each corresponding 30 minute period. Because of the 30 minute intervals used in the definitions of utilization classes, the derivation of the lumped matrices is already "aware" of the temporal aspect of the process. We therefore endow the training data for the Logistic Regression with engineered features capturing time. Namely, we introduce 48 features (one for half hour of a day), plus five features for the day of the week[1], and an additional 300 (one for each 6 second measurement interval within the 30 minute period) to capture the time offset from the beginning of the 30 minute "slot".

D. Neural Network

Two variations of Neural Network [11] are used, (i), with one hidden layer and, (ii), with five hidden layers, always using 100 nodes per layer. The same feature engineering is used as for Logistic Regression.

V. Results

Our dataset contains 38 days worth of data. For the sake of simplicity, and without hurting the generality and applicability of the presented techniques, we narrow our attention to 28 working days (10 days were weekends). The 28 days were split into four groups of 7 days, based on which we performed 4-fold cross validation, training on 21 days and testing on seven for each fold. To save space, only the average across folds is reported.

[1]For the sake of brevity, at this and all following discussion we restrict our attention to working days, i.e., weekends are excluded.

Table I
PREDICTION RESULTS – UNIVERSAL LUMPED CHAIN

	Naïve	Markov	LR	NN-1	NN-5
A	0.92	0.93	0.94	0.94	0.94
F	0.92	0.90	0.94	0.94	0.94

Table II
PREDICTION RESULTS – 30-MINUTE LUMPED CHAINS

	Naïve	Markov	LR	NN-1	NN-5
A	0.96	0.96	0.96	0.97	0.97
F	0.96	0.95	0.96	0.97	0.97

Table III
PREDICTION RESULTS – BUSY HOUR (11AM-4PM)

	Naïve	Markov	LR	NN-1	NN-5
A	0.88	0.86	0.95	0.95	0.95
F	0.88	0.83	0.95	0.95	0.95

Table IV
PREDICTION RESULTS – TWELVE BUSY HOURS (8AM-8PM)

	Naïve	Markov	LR	NN-1	NN-5
A	0.9	0.88	0.97	0.98	0.98
F	0.9	0.86	0.97	0.98	0.98

A. Single ("Universal") Lumped Matrix

First, we establish a comparison baseline by assuming that the process is stationary and hence, a single transition matrix (and corresponding lumped matrix) can describe the entire process, i.e. we do not consider 30-minute intervals separately. Based on the entire 28 days of data, when we lumped the transition matrix, we ended up with a five-state lumped matrix corresponding to the following intervals of CU values (and utilization figures in parentheses): [0-100] (0-39%), [100-120] (39-47%), [120-190] (47-74%), and [190-255] (74-100%), and producing an error $\|\mathbf{E}\|_\infty = 0.007$.

The results, assuming a "universal" lumped matrix are shown in Table II where 'A' is the accuracy, 'F' is the F_1-score, LR indicates Logistic Regression, NN indicates Neural Network (followed by the number of hidden layers), and Markov is the lumped Markov chain based prediction.

We now focus on lumped descriptios that change every 30-minutes, to approximate the non-stationarity of the CU process. The difference is that now, with different number of lumped states for each 30-minute interval, the number of classes potentially changes, and hence the performance metrics described are averaged across all 30-minute intervals of the test data. Additionally, $\|\mathbf{E}\|_\infty$ now varies from one 30-minute interval to the next. In the reported results the value of $\|\mathbf{E}\|_\infty$ ranged from a minimum of 0.0008 to a maximum of 0.055.

B. Busy vs. Non-Busy Hour Predictions

The complete appreciation of the advantage of the 30-minute lumped states requires that we narrow our focus to the two extreme utilization conditions, light load (after hours) and heavy load (peak of working hours). We found that all approaches were almost equally good at light (essentially zero!) load as one would expect – even naïve shines. The same is not the case during the busy hour, where accurate prediction is crucial to avoid overloading the system by deciding to transmit when the transmission cannot be accommodated.

In Table III, a five hour period that is characteristic of high CU values across the entire data set is used to narrow the comparison when it most matters, i.e., in a busy system. For the sake of completeness, we also show results (Table IV) where the entire day is split in a rather ad-hoc manner into two 12hour intervals (8am to 8pm and 8pm to 8am) where one contains the five busiest hours. The lumped model gives better results by a considerably margin during busy hours, when used to define the classes for the Logistic Regression and Neural Network model (the size of the network apparently has not much impact) compared to Naïve. This can be understood by the fact that during busy hours, the circumstances in which the utilization of the channel remains the same (the key to Naïve's good performance) is happening less frequently. One has to also note that predicting the busy hour behavior is always strictly more challenging than the non-busy hour – as we noticed that both Naïve and lumped Markov performance drops when moving from non-busy to busy. Finally, the differences are less pronounced if we partition the day into two rather arbitrary periods of 12 hours as the impact of the mis-predictions is diluted across a longer time interval. Yet it is possible to spot that the busy predictions by the lumped states come close to being excellent (0.98) during the busy hours (Table IV).

VI. CONCLUSION

We presented alternatives to predicting the short-term utilization of a Wi-Fi channel, based on exploiting channel utilization measurements reported in Beacon transmissions from APs. A significant motivation was our observation that a naïve predictor performs remarkably well in many cases. By introducing a lumped Markovian model, we captured the essence of the naïve predictor (long periods of staying within a lumped meta-state) but were also able to construct classes (lumps) of utilization that subsequently aided us to train ML predictors to produce excellent results. In followup work we are producing lumping schemes that penalize the creation of large lumps representing large utilization ranges. Finally, we plan to create a model that captures both the temporal, as well as the spatial facets of the utilization process, interpolating the channel utilization across many points in space from where the measurements are collected.

VII. ACKNOWLEDGEMENT

This work was partially supported by a Discovery Grant and a Research Tools and Instruments Grant from the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] Wei Wang, Jiayu Chen, Tianzhen Hong, and Na Zhu. Occupancy prediction through markov based feedback recurrent neural network (m-frnn) algorithm with wifi probe technology. *Building and Environment*, 138:160–170, 2018.

[2] Y Wang and L Shao. Understanding occupancy pattern and improving building energy efficiency through wi-fi based indoor positioning. *Building and Environment*, 114:106–117, 2017.

[3] Wei Wang, Jiayu Chen, and Xinyi Song. Modeling and predicting occupancy profile in office space with a wi-fi probe-based dynamic markov time-window inference approach. *Building and Environment*, 124:130–142, 2017.

[4] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. Sentinel: occupancy based hvac actuation using existing wifi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, page 17. ACM, 2013.

[5] Xiaoshuang Xing, Tao Jing, Yan Huo, Hongjuan Li, and Xiuzhen Cheng. Channel quality prediction based on bayesian inference in cognitive radio networks. In *2013 Proceedings IEEE INFOCOM*, pages 1465–1473. IEEE, 2013.

[6] Chittabrata Ghosh, Carlos Cordeiro, Dharma P Agrawal, and M Bhaskara Rao. Markov chain existence and hidden markov models in spectrum sensing. In *2009 IEEE International Conference on Pervasive Computing and Communications*, pages 1–6. IEEE, 2009.

[7] The dynamic channel assignment white paper. White Paper, 2019.

[8] Robert Akl and Anurag Arepally. Dynamic channel assignment in ieee 802.11 networks. In *2007 IEEE international conference on portable information devices*, pages 1–5. IEEE, 2007.

[9] Peter Buchholz. Exact and ordinary lumpability in finite markov chains. *Journal of applied probability*, 31(1):59–75, 1994.

[10] Tugrul Dayar and William J Stewart. Quasi lumpability, lower-bounding coupling matrices, and nearly completely decomposable markov chains. *SIAM Journal on Matrix Analysis and Applications*, 18(2):482–498, 1997.

[11] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.