A Multi-Level Methodology for Explaining Data Streams

LUKA STOPAR, Jozef Stefan Institute
PRIMOZ SKRABA, Jozef Stefan Institute
DUNJA MLADENIC, Jozef Stefan Institute
MARKO GROBELNIK, Jozef Stefan Institute

This paper presents a novel multi-scale methodology for modeling a collection of continuously time-varying data streams. The data streams are aggregated using unsupervised data mining methods. Typical system states are then computed on the aggregated data. This is used as input to construct a Markovian transition model capturing the dynamics of the monitored system. The hierarchical organization of the states enables a visual representation of the dynamics on multiple aggregation levels.

CCS Concepts: •**Computer systems organization** → **Embedded systems;** *Redundancy;* Robotics; •**Networks** → Network reliability;

Additional Key Words and Phrases: [TODO]

ACM Reference Format:
Luka Stopar, Primoz Skraba, Dunja Mladenic, Marko Grobelnik, 2015. A Multi-Level Methodology for Explaining Data Streams. *ACM Trans. Embedd. Comput. Syst.* 9, 4, Article 39 (March 2010), 9 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Sensory systems typically operate in cycles with a continuously time-varying. These systems include, for example, the solar system, manufacturing systems or weather systems. Such systems can be characterized by a set of states, along with associated state transitions. States, on a high level, may include a ”day” state and a ”night” state or maybe states with high and low productivity. For example, when a pilot wishes to change an aircrafts heading, they will put the aircraft into state ”banking turn” by lowering one aileron and raising the other, causing the aircraft to perform a circular arc. After some time, the wings of the aircraft will be brought level by an opposing motion of the ailerons and the aircraft will go back into state ”level”.

Such high-level states can be decomposed into lower-level states, giving us a multi-level view of the system and allowing us to observe the system on multiple detail levels. For example, a ”banking turn” state can be decomposed by the aircrafts roll and angular velocity, resulting in perhaps three states: ”initiate turn”, ”full turn” and ”end turn”.

We present a methodology for modeling such systems and demonstrate its implementation called StreamStory. StreamStory models such systems as a hierarchical Markovian process by automatically learning the typical low-level states, transitions

[TODO] This work is supported by the National Science Foundation, under grant CNS-0435060, grant CCR-0325197 and grant EN-CS-0329609.
Author's addresses: G. Zhou, Computer Science Department, College of William and Mary; Y. Wu and J. A. Stankovic, Computer Science Department, University of Virginia; T. Yan, Eaton Innovation Center; T. He, Computer Science Department, University of Minnesota; C. Huang, Google; T. F. Abdelzaher, (Current address) NASA Ames Research Center, Moffett Field, California 94035.

DOI: 0000001.0000001

and aggregating them into a hierarchy, obtaining a unique Markovian process on multiple detail levels. As such it gives users a unique view into the monitored system.

Furthermore, we divide the inputs streams into two sets: observation and control set. Attributes in the observation set are the attributes that tell us the state of the system and, we assume, cannot directly influence its dynamics. These are parameters that users cannot directly manipulate, like aircraft tilt from the previous example, which must be indirectly manipulated through the angles of ailerons. We use observation attributes to identify, and aggregate, low-level states, detect outliers (anomalies) and determine the current state of the system.

In contrast, users can directly manipulate attributes in the control set. These are attributes like angles of ailerons that may directly influence the behavior (observation attributes) and performance of the system. For example, when an operator in a steel factory sets the cooling temperature to a high value, the product will take longer to go from state ”hot” to state ”cool”. As such, we assume, control attributes may influence the occurrence, and expected time, of undesired states, associated with undesired events.

Our system uses control attributes to model state transitions, allowing us to observe the dynamics with respect to the current configuration and gives us insight into the expected dynamics with respect to some alternate attribute configuration.

The main research contributions presented in this paper can be summarized as follows:

- We present a novel methodology for modeling multivariate continuously time-varying data streams in a hierarchical manner providing a unique model of the several detail levels.
- We present a novel approach for modeling state transitions in a Markov chain allowing to observe the dynamics of the chain under alternate configurations.
- We propose a recursive algorithm for partitioning recurrent continuous time Markov chains.

The remainder of this paper is structured as follows. In Section 2 we present our methodology in detail. Section 3 we present user interaction. Section **??** we present our experimental results and finally in Section **??** we conclude this paper and give some ideas for further work.

2. PROPOSED MULTI-LEVEL METHODOLOGY

We begin this section with an overview of our multi-level methodology show in figure 1. As shown in the figure our methodology encompasses four main steps. These include:

(1) aggregating and resampling the input streams, interpolating wherever needed, producing feature vectors,
(2) clustering the feature vectors to obtain lowest-level states and computing statistics on each state,
(3) aggregating lowest-level states into a state hierarchy and computing the level on which each aggregated state lives and
(4) modeling state transitions with respect to control attributes and computing statistics on transitions.

Each of these steps will be presented in the next subsections in detail.

2.1. Merging and Processing Data Streams

In recent years much attention of the research community has been focused on a class of applications where the data are modeled best not as persistent relations, but rather as transient data streams [Babcock et al. 2002]. Data streams differ from conventional

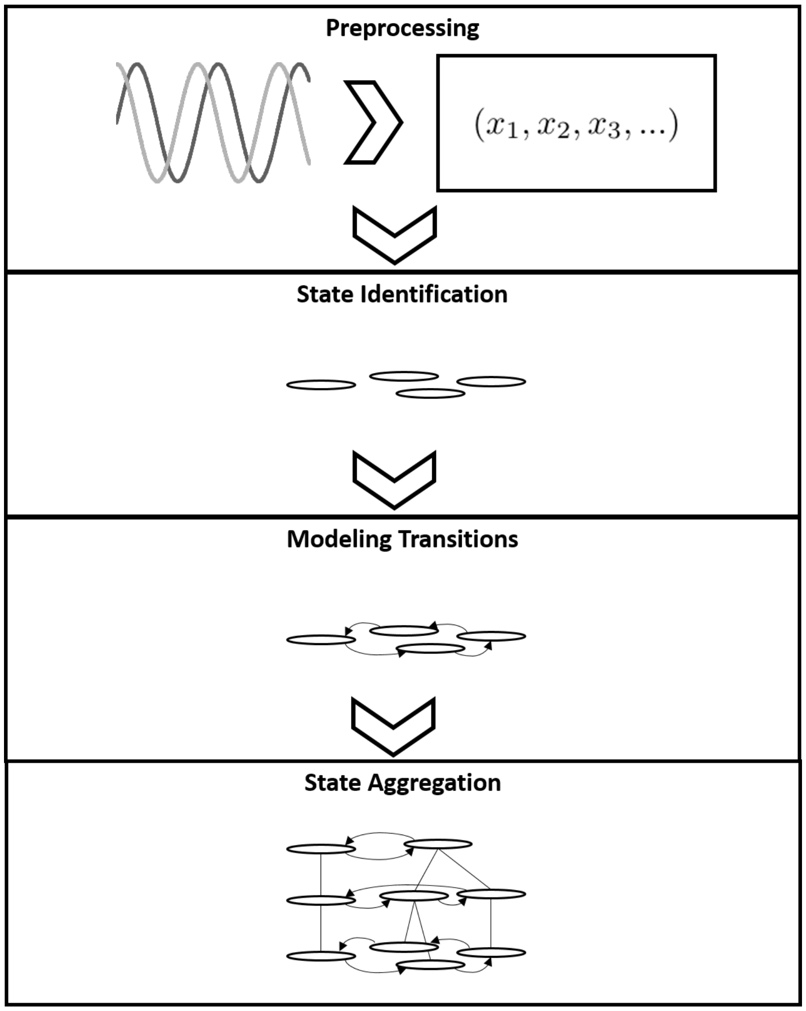


Fig. 1. The proposed multi-level methodology.

batch data in two main aspects: (1) their elements arrive online, sequentially and the system has no control over their order and (2) data streams are potentially unbounded.

As such a data stream can be defined as a sequence of pairs $(t_i, x_i)_{i\geq 0}$ where t_i is the timestamp of the i-th example and x_i is its value. In the first step our methodology consumes a batch of data streams and preprocesses them to obtain feature vectors which can be used to model the data in further steps. The preprocessing is done by first merging the data streams to produce feature vectors with a common time stamp and later resampling them using a domain-specific appropriate sampling rate. After the resampling step, the methodology foresees the calculation of domain specific features which the user would like to use when modeling the data. The last step can include the addition of generic features like current derivative or some domain specific calculation like perhaps the number of times an attribute was above a threshold in a domain specific time window.

2.2. State Identification

Once the data is represented as feature vectors it is ready to be modeled. Our next step includes identifying typical low-level system states from the feature vectors created in step 1. We do this by partitioning the feature vectors and associating each state with a partition. There are many partitioning algorithms available in the literature including algorithms that produce hierarchical partitions. In general however the computational complexity of these algorithms is $O(n^2)$ which renders them unsuitable for big data scenarios. We therefore propose a flat partition using k-means resulting in a Voronoi diagram on the data space and enabling us to construct states by assigning feature vectors to the partition with the nearest centroid and constructing a hierarchical representation later in the process. We then collect statistics in the states including

- Distribution of attributes - we store the distribution of all the attributes in a state as a histogram. The number of bins in a histogram is a configuration parameter and should be based on domain knowledge.
- Mean value and median of each attribute inside a state.
- Percentiles for each attribute.

These statistics can be used when the model is applied in real-time and provide a clustering-based anomaly detection technique.

and later aggregate it do obtain a hierarchical partition.

2.3. Modeling Transitions

We model transitions using a Markovian model. The main characteristics of Markovian models is that they retain no memory of where they have been in the past. This means that only the current state can influence where the process will go next. In this work we are interested only in processes that can assume a finite set of states, called Markov chains [Norris 1998]. More formally, we are interested in Markov chains $(X_t)_{t\geq 0}$ which can assume values in a finite state space S. We define $p_{ij}(t)$ to be the probability of the process being in state j at time t when starting from state i at time 0. We define a stochastic matrix $P(t)$, where $(P(t))_{ij} = p_{ij}(t)$. Thus each row of $P(t)$ is a probability distribution over the state space. Such a chain can be represented by a transition rate matrix Q, where Q satisfies the following system of equations

$$\frac{d}{dt}P(t) = P(t)Q, \qquad P(0) = I. \tag{1}$$

and has the following properties:

(1) $-\infty < q_{ii} \leq 0$
(2) $q_{ij} \geq 0$ for $j \neq i$
(3) $\sum_{j\in S} q_{ij} = 0$

Each non-diagonal entry of Q, q_{ij} thus represents the rate of going from state i to state j while the diagonal entries $-q_{ii}$ represent the rate of leaving state i.

We assume that $P(t)$ is recurrent for all t and define $\pi_j = \lim_{t\to\infty} p_{ij}(t)$. It can be shown that for recurrent chains π_j is independent of the starting state and vector $\pi = (\pi_1, \pi_2, ..., \pi_k)$ is called the stationary distribution of $(X_t)_{t\geq 0}$ and represents the proportion of time the process spends in each state after funning for an infinite amount of time.

Using this representation, the stationary distribution π can be computed as the left eigenvector of Q corresponding to eigenvalue 0.

2.3.1. Transition Rate Estimation. We estimate the transition rates of $(X_t)_{t\geq 0}$ by first discretizing the the continuous parameter space $t \in [0, \infty)$ into a discrete sequence $(0, h, 2h, ...)$ and estimating the transition probabilities $\tilde{p}_{ij} = p_{ij}(h)$. Once $\tilde{p}_{ij}$ are estimated, the transition rates can be calculated as

$$q_{ij} = \frac{h}{\tilde{p}_{ij}}. \tag{2}$$

As proposed in section 1 we allow the users to simulate the dynamics based on an alternate configurations of the predefined attributes which we denote A_S. Suppose the process is in state i at time 0 and define a random variable $J_i = j \Leftrightarrow X_h = j$. Thus J_i has a multinomial distribution with parameters $(p_{i1}, p_{i2}, ..., p_{in})$ which can be modeled using a generalized linear model (GLM) [Dobson and Barnett 2008]. Because there is no natural ordering in the response category, we use nominal logistic regression to estimate p_{ij} based on values $x_k \in A_S$. We select reference category as p_{ii} and model the relationship between x_k and p_{ij} as follows:

$$\text{logit}(p_{ij}) = \log\left(\frac{p_{ij}}{p_{ii}}\right) = \beta_{ij}x_k \tag{3}$$

thus, for each state we get a set of $n-1$ equations which are used simultaneously to estimate the parameters β_{ij}. Once these have been obtained, the linear predictors can be calculated as:

$$\tilde{p}_{ij}(x_k) = \frac{e^{\beta_{ij}x_k}}{1 + \sum_{l\neq i} e^{\beta_{il}x_k}}. \tag{4}$$

2.4. State Aggregation

Once the process is modeled on the lowest level, we aggregate states to obtain a hierarchy of such processes. We define state aggregation as the problem of partitioning a Markov chain and adapt the algorithm proposed in [Deng et al. 2009] to work in a continuous time setting. Let a Markov chain on a state space X be defined by a transition rate matrix Q. We define a surjective partition function $\phi : X \to Y$ and a corresponding transition rate matrix transformation function $\Phi : \mathbb{R}^{n\times n} \to \mathbb{R}^{m\times m}$ using the following formula:

$$\Phi(Q) = (P'\Pi P)^{-1}P'\Pi QP \tag{5}$$

where $\Pi = diag(\pi)$ and P is a $|X| \times |Y|$ partition matrix with elements

$$(P)_{ij} = \begin{cases} 1 & \text{if } \phi(i) = j \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Thus once knowing the partition function on a specific detail level, the associated Markov chain can be computed efficiently. We note however that information is lost when aggregating the state space of a Markov chain and the original chain can never be computed from only the aggregated one.

Now that we can represent a Markov chain knowing a partition function, we present an algorithm for partitioning such chains using spectral graph theory. We first propose a bi-partitioning algorithm which is used recursively to split the aggregated states.

2.4.1. Bi-Partitioning Markov Chains. The spectral analysis of undirected graphs has been studied extensively in the literature and many algorithms for bi-partitioning such graphs proposed. These papers use the Cheeger inequality, one of the main tools for bounding the mixing times for random walks on undirected graphs, on the graph

Laplacian. The transition rate matrix of a Markov chain is exactly the negative Laplacian of a directed weighted graph $Q = -L(G)$ [Agaev and Chebotarev 2005]. The Laplacian is however not symmetric and in general does not have real eigenvalues. Therefore we transform the Laplacian using the following formula:

$$Q_s = \frac{1}{2}(\Pi Q + Q'\Pi) \tag{7}$$

The transformation above has the effect of symmetrizing the transition rate matrix and furthermore the matrix $\Pi^{-1}Q_s$ preserves the ergodic properties of the original Markov chain. To obtain a bi-partition of the original Markov chain we then solve the following generalized eigenvalue problem:

$$Q_s v = \lambda_2 \Pi v \tag{8}$$

The bi-partition function ϕ is then constructed using the following rule:

$$\phi(i) = \begin{cases} 1 & \text{if } v_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

2.4.2. Partitioning Markov Chains. Since the bi-partitioning problem can be solved using the second eigenvector of the transformed Laplacian we can now present a recursive algorithm for the partitioning of continuous time Markov chains. We first define the entropy rate function

$$D(Q||\tilde{Q}) = \sum_{A\in \operatorname{Im}\phi} \sum_{B\in \operatorname{Im}\phi} \sum_{i\in\phi^{-1}(A)} \pi_i \sum_{j\in\phi^{-1}(B)} q_{ij} \log \frac{\sum_{j\in\phi^{-1}(B)} q_{ij}}{\tilde{q}_{AB}} \tag{10}$$

[TODO entropy rate not yet developed]

There are a couple of options to calculate the distance between Markov chains:

- Wasserstein
- Total Variation
- Entropy rat

The algorithm starts with an initial partition containing all the states of the original Markov chain. On the n-th iteration the algorithm assumes that n partitions (aggregated states) are given. The objective then is to select the best aggregated state to split, obtaining $n + 1$ partitions. For $i = 1, 2, ..., n$ we denote $Q^{(i)}$ to be the transition rate matrix of a Markov chain within the i-th partition. A bi-partition of $Q^{(i)}$ provides $n+1$ partitions, which we denote $Q^{(i)}_{n+1}$. The objective then is to select the best partition $Q^{(i)}$ to split. This is achieved by minimizing the following criteria:

$$\underset{i\in\{1,2,\ldots,n\}}{\operatorname{argmin}} \; D(Q||Q^{(i)}_{n+1}) \tag{11}$$

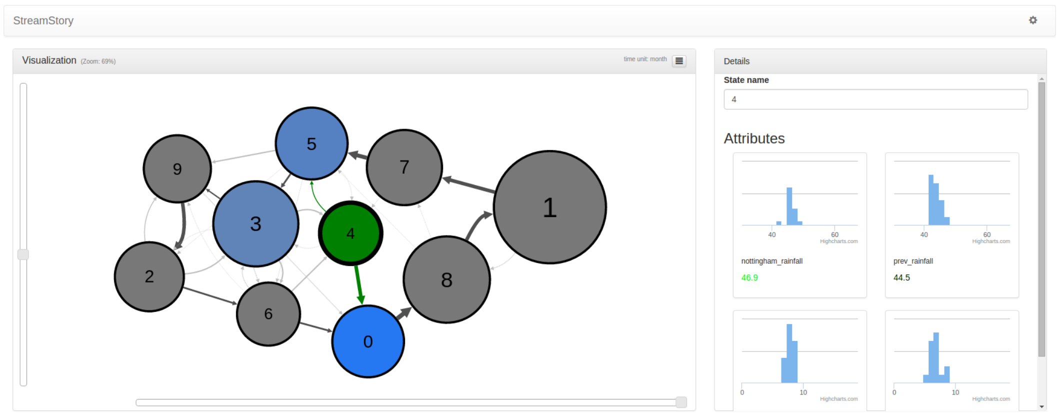
2.5. Visualization

In the user interface, the model is presented on only one level of the hierarchy. States are presented as circles while transitions are presented as arrows between states. The area of each state is proportional to the mean time the process spends in the state which is extracted from the stationary distribution of the Markov chain on the current detail level. The user can zoom into and out of more detailed levels by using the zoom function. We offer several services that allow the users to identify the meaning of states, which are presented in Section 3.

To visualize states, their planar coordinates are precomputed by the state identification component. This is achieved by computing the initial or seed positions using principal component analysis (PCA) on the state centroids. To avoid state overlap, we then refine state positions using a repulsive-simulation based technique across all the detail levels simultaneously.

3. VISUALIZATION AND USER INTERACTION

When interacting with the StreamStory system, the user is presented with a two panel user interface shown in figure 2.



State name
4

nottingham_rainfall
46.9

prev_rainfall
44.5

Fig. 2. User interface of the StreamStory system.

The visualization panel on the left visualizes the hierarchical Markovian model. States are represented by circles, while transitions are represented by arrows. The size of a state is proportional to the fraction of time the monitored process spends in the state. When StreamStory is used in online mode, the current state is colored green, the most likely future states blue and the previous state has a red border.

The thickness of an arrow is proportional to the probability of the corresponding transition, and the most likely transitions have a darker color.

When first opening user interface the user is presented with a top-level chain with only two states. They can then use the scroll function to zoom into the hierarchy and the states are expanded automatically.

Users can view a state by using the click function. By doing this, the state becomes selected and the user is presented with several services that allow them to identify the meaning of the state. These will be presented shortly in the following subsections.

3.1. State Details and Attribute Highlighting

3.2. Decision Trees

3.3. Rule Extraction

By clicking on a state, the state becomes selected and its details are shown in the ”Details” panel on the right side of figure 2. The details panel allows the user to name the state which, we believe, increases the models interpretability. It also shows the distribution of all the parameters inside the state as well as the mean value of each parameter in the state. The mean value is highlighted red or green depending on how specific the value is for that state. If the value is gray it means that the value of the attribute is normal compared to the value of the same attribute in other states. However, if the value is green or red, it indicates the value is the value is specific for this state compared to other states.

This is achieved by classifying the instances of the selected state against instances off all the other states on the same level using a logistic regression model [TODO ref] and extracting weights. Values highlighted green indicate a positive weight, while values highlighted red indicate a negative weight.

APPENDIX

In this appendix, we measure the channel switching time of Micaz [CROSSBOW] sensor devices. In our experiments, one mote alternatingly switches between Channels 11 and 12. Every time after the node switches to a channel, it sends out a packet immediately and then changes to a new channel as soon as the transmission is finished. We measure the number of packets the test mote can send in 10 seconds, denoted as N_1. In contrast, we also measure the same value of the test mote without switching channels, denoted as N_2. We calculate the channel-switching time s as

$$s = \frac{10}{N_1} - \frac{10}{N_2}.$$

By repeating the experiments 100 times, we get the average channel-switching time of Micaz motes: 24.3μs.

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Maura Turolla of Telecom Italia for providing specifications about the application scenario.

REFERENCES

Rafig Agaev and Pavel Chebotarev. 2005. On the spectra of nonsymmetric Laplacian matrices. *Linear Algebra Appl.* 399 (2005), 157 – 168. DOI:http://dx.doi.org/10.1016/j.laa.2004.09.003 Special Issue devoted to papers presented at the International Meeting on Matrix Analysis and Applications, Ft. Lauderdale, FL, 14-16 December 2003.

Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. 2002. Models and Issues in Data Stream Systems. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '02)*. ACM, New York, NY, USA, 1–16. DOI:http://dx.doi.org/10.1145/543613.543615

Adam Coates and AndrewY. Ng. 2012. Learning Feature Representations with K-Means. In *Neural Networks: Tricks of the Trade*, Grgoire Montavon, GeneviveB. Orr, and Klaus-Robert Mller (Eds.). Lecture Notes in Computer Science, Vol. 7700. Springer Berlin Heidelberg, 561–580. DOI:http://dx.doi.org/10.1007/978-3-642-35289-8_30

M. Crosskey and M. Maggioni. 2014. ATLAS: A geometric approach to learning high-dimensional stochastic systems near manifolds. *ArXiv e-prints* (April 2014).

Kun Deng, Yu Sun, P.G. Mehta, and S.P. Meyn. 2009. An information-theoretic framework to aggregate a Markov chain. In *American Control Conference, 2009. ACC '09*. 731–736. DOI: http://dx.doi.org/10.1109/ACC.2009.5160607

Annette J. Dobson and Adrian G. Barnett. 2008. *An Introduction to Generalized Linear Models, Third Edition*. Chapman & Hall/CRC Press, Boca Raton, FL. http://eprints.qut.edu.au/15448/ For more information about this book please refer to the publisher's website (see link) or contact the author.

A. K. Jain, M. N. Murty, and P. J. Flynn. 1999. Data Clustering: A Review. *ACM Comput. Surv.* 31, 3 (Sept. 1999), 264–323. DOI: http://dx.doi.org/10.1145/331499.331504

Brian Kulis and Michael I. Jordan. 2011. Revisiting k-means: New Algorithms via Bayesian Nonparametrics. *CoRR* abs/1111.0352 (2011). http://arxiv.org/abs/1111.0352

Oded Maimon and Lior Rokach. 2005. *Data Mining and Knowledge Discovery Handbook*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Fionn Murtagh and Pedro Contreras. 2011. Methods of Hierarchical Clustering. *CoRR* abs/1105.0121 (2011). http://arxiv.org/abs/1105.0121

J.R. Norris. 1998. *Markov Chains*. Number št. 2008 in Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press. https://books.google.si/books?id=qM65VRmOJZAC

Vijay S. Pande, Kyle Beauchamp, and Gregory R. Bowman. 2010. Everything you wanted to know about Markov State Models but were afraid to ask. *Methods* 52 (2010), 99–105.

R. Sibson. 1973. SLINK: An optimally efficient algorithm for the single-link cluster method. *Comput. J.* 16, 1 (1973), 30–34. DOI: http://dx.doi.org/10.1093/comjnl/16.1.30

Rui Xu and II Wunsch, D. 2005. Survey of clustering algorithms. *Neural Networks, IEEE Transactions on* 16, 3 (May 2005), 645–678. DOI: http://dx.doi.org/10.1109/TNN.2005.845141

Received February 2007; revised March 2009; accepted June 2009

Online Appendix to: A Multi-Level Methodology for Explaining Data Streams

LUKA STOPAR, Jozef Stefan Institute
PRIMOZ SKRABA, Jozef Stefan Institute
DUNJA MLADENIC, Jozef Stefan Institute
MARKO GROBELNIK, Jozef Stefan Institute

A. THIS IS AN EXAMPLE OF APPENDIX SECTION HEAD

Channel-switching time is measured as the time length it takes for motes to successfully switch from one channel to another. This parameter impacts the maximum network throughput, because motes cannot receive or send any packet during this period of time, and it also affects the efficiency of toggle snooping in MMSN, where motes need to sense through channels rapidly.

By repeating experiments 100 times, we get the average channel-switching time of Micaz motes: 24.3 μs. We then conduct the same experiments with different Micaz motes, as well as experiments with the transmitter switching from Channel 11 to other channels. In both scenarios, the channel-switching time does not have obvious changes. (In our experiments, all values are in the range of 23.6 μs to 24.9 μs.)

B. APPENDIX SECTION HEAD

The primary consumer of energy in WSNs is idle listening. The key to reduce idle listening is executing low duty-cycle on nodes. Two primary approaches are considered in controlling duty-cycles in the MAC layer.

© 2010 ACM. 1539-9087/2010/03-ART39 $15.00
DOI: 0000001.0000001