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 first aggregated and, using unsupervised data mining methods, typical system states are computed. This is achieved by use of one of two available clustering methods. After the states are computed, they are aggregated using an agglomerative clustering algorithm.

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), 12 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 aggregation 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 and transi-

[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.

tions, and aggregating them into a hierarchy of Markovian processes. 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.

We implement our methodology in a four step process, which includes:

(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.

[TODO The remainder of this paper is structured as follows]

2. THEORY AND IMPLEMENTATION

This section presents the detailed implementation of the StreamStory system, by presenting in details each step of our four step process. We begin the section by first discussing the preprocessing step proposed in step 1 of our four step process defined in section 1.

2.1. Data Stream Aggregation and Resampling

In recent years much attention in the research community has been focused on a new class of applications: applications in which the data is modeled best not as persistent relations but rather as transient data stream [Babcock et al. 2002].

Data streams differ from conventional batch data in several ways:

- the data elements arrive online, sequentially and the system has no control over their order and
- 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 represents the timestamp if the i-th measurement and x_i represents its value. In our work we assume the streams follow the following model: $x_{i+1} = x_i + X$, where X is a normally distributed random variable $X \sim N(\mu, \sigma)$. [TODO really???]

To build a model in offline mode, StreamStory consumes batches of multiple such data streams and merges them, to that all the stream are sampled at the same timestamps. This results in a single joined data stream in the following form:

$$\left\{\left(t_1, x_1^{(1)}, x_1^{(2)}, ..., x_1^{(d)}\right), \left(t_1 + \Delta t, x_2^{(1)}, x_2^{(2)}, ..., x_2^{(d)}\right), ..., \left(t_0 + (k-1)\Delta t, x_k^{(1)}, x_k^{(2)}, ..., x_k^{(d)}\right)\right\}$$

Since in general not all the input streams are equally sampled, missing values need to be interpolated. For this purpose StreamStory supports two interpolation methods: linear and previous point interpolation. The choice of the interpolation method influences the online behavior of the system. Although in theory linear interpolation produces better results [TODO ref], linear interpolation cannot interpolate a value until it observes at least one value in the [TODO future]. Which results in a certain lag (non-real-time behavior) of the system.

2.2. State Identification

Once the data streams are preprocessed, StreamStory identifies their typical lowest-level states. This is achieved by clustering the joined data stream.

Clustering is an unsupervised machine learning procedure, which organizes a collection of patterns, usually represented as a vectors of measurements or points in a multidimensional space, into clusters based on a chosen measure of similarity or distance [Jain et al. 1999]. There is no universally agreed upon definition of a cluster. Most researchers describe it by considering the internal homogeneity and external separation [Xu and Wunsch 2005].

Intuitively patterns in the same cluster should be more similar to each other than to patterns in other clusters. There are many clustering algorithms that solve different problems. These include partitioning, [TODO].

We support two partitioning methods: K-Means and DPMeans. The first is a well studied method which, given k - the desired number of clusters, partitions the dataset into k clusters by minimizing the distance between data points and the nearest centroid [Coates and Ng 2012]. It works by, initially selecting k random centroids, then in each iteration it first assigns all the feature vectors to their nearest centroid constructing a Voronoi diagram regions are associated with centroids $C = \{c_1, c_2, ..., c_k\}$. More formally a region of the diagram is defined as $R(c_i) = \left\{x \in \mathbb{R}^d | d(c_i, x) \leq d(c_j, x)\, \forall j\right\}$. The method then recomputes each centroid c_i as the mean value of all the data points that lie in region $R(c_i)$:

$$c_i = \frac{1}{\|R_i\|} \sum_{p_j \in R(c_i)} p_j .$$

The procedure terminates when the partitions are the same in two consecutive iterations.

The second method, DP-means, proposed by [Kulis and Jordan 2011] behaves similarly to K-Means with the exception that a new cluster is formed whenever a feature vector is further than λ away from every existing centroid. Thus, the number of clusters on the output is not known in advance, but is controlled by λ.

When identifying states, we first remove timestamps from the joined data stream, resulting in feature vectors of the form

$$x_i = \left(x_i^{(1)}, x_i^{(2)}, ..., x_i^{(d)}\right) .$$

These are then clustered using one of the above methods, resulting in a set of k partitions which are used as lowest level states. When StreamStory [TODO sees] a new data point it will assign it to the state (partition) with the nearest centroid.

$$S(p_i) = \underset{p_j \in R(c_j)}{\operatorname{argmin}} \, d(p_i, c_j)$$

Where d represents the Euclidean distance.

2.3. State Aggregation

Once the lowest level states are computed, StreamStory aggregates them into a state hierarchy. Before grouping states, we need to measure their relative distance to each other. Since StreamStory was developed to work with dense dataset, we use the Euclidean distance. Many hierarchical clustering techniques have been proposed in the literature. They can be characterized as greedy in the algorithmic sense [Murtagh and Contreras 2011]. Assuming that a pair of states is merged or agglomerated at each step, the techniques construct a binary tree commonly known as a dendrogram. This produces a set of states at each level - or each threshold value which produces a new partition. Hierarchical clustering methods generally fall into two groups [Maimon and Rokach 2005]:

- **Agglomerative** - each object initially represents a cluster of its own. Clusters are then iteratively merged, until the desired structure is obtained.
- **Divisive** - when the procedure starts, all object belong to the same cluster. Then clusters are recursively divided into sub-clusters until the desired structure is obtained.

These groups can be further subdivided according to the manner that the distance measure is calculated:

- **Single-link** - methods that consider the distance between two clusters equal to the minimum distance of any member of one cluster to any member of the other.
- **Complete-link** - methods that consider the distance between two clusters equal to the maximum distance of any member of one cluster to any member of the other.
- **Complete-line** - methods that consider the distance between two clusters equal to the average distance of any member of one cluster to any member of the other.

StreamStory uses an agglomerative clustering technique and supports all three of the above mentioned linkage strategies.

2.4. Modeling Transitions

We model transitions using a Markovian model. The main characteristic of Markovian models is that it retains no memory of where it has 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, referred to as Markov chains. What makes these processes useful is that not only do they model many phenomena of interest, but also the lack of memory property makes it possible to predict how a Markov chain may behave, and to compute probabilities and expected values which quantify that behavior [Norris 1998]. A Markov chain is a memoryless stochastic process $(X_k)_{k \geq 0}$ that can assume only a finite or countable set of states $i \in I$, where I is called the state space. Markov chains generally fall into two categories: discrete time and continuous time.

Discrete time Markov chains move in discrete time steps. They are defined using a stochastic matrix P where $(P)_{ij} = p_{ij}$ is the probability of jumping from state $i \in I$

to state $j \in I$ in a single time step. More formally we say that a stochastic process $(X_n)_{n\geq 0}$ is a discrete time Markov chain with initial distribution λ if

(1) X_0 has distribution λ and
(2) $P(X_{n+1} = i_{n+1} | X_0 = i_0, X_1 = i_1, ..., X_n = i_n) = p_{i_n i_{n+1}}$

Once of the interesting properties of Markov chains are their stationary distributions. A stationary distribution describes the proportion of time the process spends in each state. In our visualization it represents the size of states. The stationary distribution can be computed as the left eigenvector of P with the corresponding eigenvalue 1.

When drawing the hierarchy we need some way of aggregating states in the Markov chain. Suppose we have two non-intersecting sets of states $A = \{j_1, j_2, ..., j_k\}$ and $B = \{i_1, i_2, ..., i_h\}$. Then we use the following formula to compute the transition probability from set A to set B:

$$p_{AB} = P(X_{n+1} \in A | X_n \in B) = \sum_{j \in A} P(X_{n+1} = j | X_n \in B) =$$

$$\sum_{j \in A} \frac{P(X_{n+1} = j) P(X_n \in B | X_{n+1} = j)}{P(X_n \in B)} = \frac{\sum_{j \in A} \pi_j \sum_{i \in B} p_{ij} \frac{\pi_i}{\pi_j}}{\sum_{j \in B} \pi_i} = \tag{1}$$

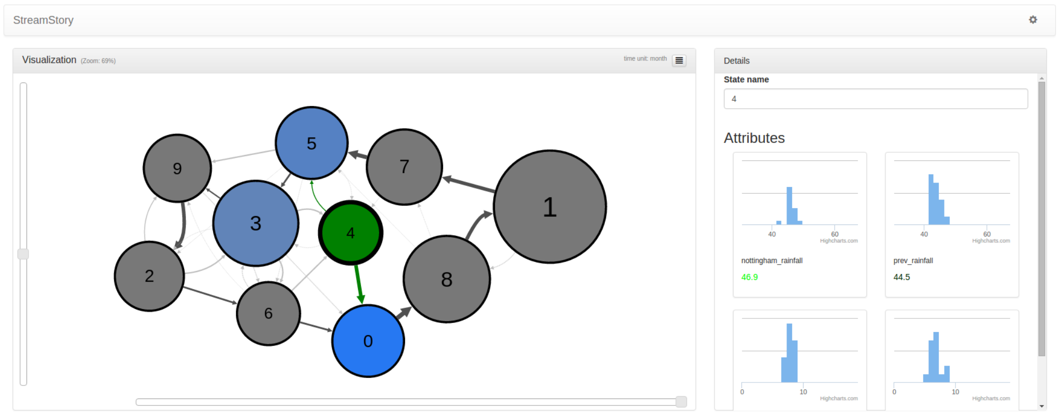
$$\frac{\sum_{i \in B} \pi_i \sum_{j \in A} p_{ij}}{\sum_{j \in B} \pi_i}$$

2.5. Visualization

[TODO]

3. VISUALIZATION AND USER INTERACTION

When interacting with the StreamStory system, the user is presented with a two panel user interface shown in figure 1.



State name

4

nottingham_rainfall 46.9

prev_rainfall 44.5

Fig. 1. 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 get automatically expanded.

By clicking on a state, the state becomes selected and its details are shown in the ”Details” panel on the right side of figure 1. 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.

We propose a suboptimal distribution to be used by each node, which is easy to compute and does not depend on the number of competing nodes. A natural candidate is an increasing geometric sequence, in which

$$P(t) = \frac{b^{\frac{t+1}{T+1}} - b^{\frac{t}{T+1}}}{b-1}, \tag{2}$$

where $t = 0, \ldots, T$, and b is a number greater than 1.

In our algorithm, we use the suboptimal approach for simplicity and generality. We need to make the distribution of the selected back-off time slice at each node conform to what is shown in Equation (2). It is implemented as follows: First, a random variable α with a uniform distribution within the interval $(0, 1)$ is generated on each node, then time slice i is selected according to the following equation:

$$i = \lfloor (T+1)\log_b[\alpha(b-1)+1] \rfloor.$$

It can be easily proven that the distribution of i conforms to Equation (2).

So protocols [Bahl 2002,Culler 2001,Zhou 2006,Adya 2001,Culler 2001; Tzamaloukas-01; Akyildiz-01] that use RTS/CTS controls[1] for frequency negotiation and reservation are not suitable for WSN applications, even though they exhibit good performance in general wireless ad hoc networks.

3.0.1. Exclusive Frequency Assignment. In exclusive frequency assignment, nodes first exchange their IDs among two communication hops so that each node knows its two-hop neighbors' IDs. In the second broadcast, each node beacons all neighbors' IDs it has collected during the first broadcast period.

Eavesdropping. Even though the even selection scheme leads to even sharing of available frequencies among any two-hop neighborhood, it involves a number of two-hop broadcasts. To reduce the communication cost, we propose a lightweight eavesdropping scheme.

3.1. Basic Notations

As Algorithm 1 states, for each frequency number, each node calculates a random number (*Rnd*$_\alpha$) for itself and a random number (*Rnd*$_\beta$) for each of its two-hop neighbors with the same pseudorandom number generator.

Bus masters are divided into two disjoint sets, $\mathcal{M}_{RT}$ and $\mathcal{M}_{NRT}$.

RT Masters. $\mathcal{M}_{RT} = \{\vec{m}_1, \ldots, \vec{m}_n\}$ denotes the n RT masters issuing real-time constrained requests. To model the current request issued by an $\vec{m}_i$ in $\mathcal{M}_{RT}$, three parameters—the recurrence time (r_i), the service cycle (c_i), and the relative deadline (d_i)—are used, with their relationships.

NRT Masters. $\mathcal{M}_{NRT} = \{\vec{m}_{n+1}, \ldots, \vec{m}_{n+m}\}$ is a set of m masters issuing nonreal-time constrained requests. In our model, each $\vec{m}_j$ in $\mathcal{M}_{NRT}$ needs only one parameter, the service cycle, to model the current request it issues.

Here, a question may arise, since each node has a global ID. Why don't we just map nodes' IDs within two hops into a group of frequency numbers and assign those numbers to all nodes within two hops?

[1]RTS/CTS controls are required to be implemented by 802.11-compliant devices. They can be used as an optional mechanism to avoid Hidden Terminal Problems in the 802.11 standard and protocols based on those similar to [Akyildiz 2001] and [Adya 2001].

ALGORITHM 1: Frequency Number Computation

```
Input: Node $\alpha$'s ID ($ID_\alpha$), and node $\alpha$'s neighbors' IDs within two communication hops.
Output: The frequency number ($FreNum_\alpha$) node $\alpha$ gets assigned.
$index$ = 0; $FreNum_\alpha$ = -1;
repeat
    $Rnd_\alpha$ = Random($ID_\alpha$, $index$);
    $Found$ = $TRUE$;
    for each node $\beta$ in $\alpha$'s two communication hops do
        $Rnd_\beta$ = Random($ID_\beta$, $index$);
        if ($Rnd_\alpha < Rnd_\beta$) or ($Rnd_\alpha == Rnd_\beta$ and $ID_\alpha < ID_\beta$);
        then
            $Found$ = $FALSE$; break;
        end
    end
    if $Found$ then
        $FreNum_\alpha$ = $index$;
    else
        $index$ ++;
    end
until $FreNum_\alpha > -1$;
```

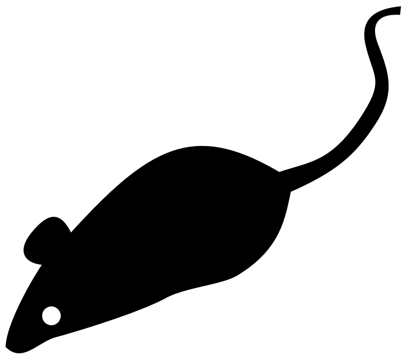


Fig. 2. Code before preprocessing.

4. SIMULATOR

If the model checker requests successors of a state which are not created yet, the state space uses the simulator to create the successors on-the-fly. To create successor states the simulator conducts the following steps.

(1) Load state into microcontroller model.
(2) Determine assignments needed for resolving nondeterminism.
(3) For each assignment.
    (a) either call interrupt handler or simulate effect of next instruction, or
    (b) evaluate truth values of atomic propositions.
(4) Return resulting states.

Figure 2 shows a typical microcontroller C program that controls an automotive power window lift. The program is one of the programs used in the case study described in Section 4. At first sight, the programs looks like an ANSI C program. It contains function calls, assignments, if clauses, and while loops.

Table I. Simulation Configuration

TERRAIN[a]	(200m×200m) Square
Node Number	289
Node Placement	Uniform
Application	Many-to-Many/Gossip CBR Streams
Payload Size	32 bytes
Routing Layer	GF
MAC Layer	CSMA/MMSN
Radio Layer	RADIO-ACCNOISE
Radio Bandwidth	250Kbps
Radio Range	20m–45m

Source: This is a table sourcenote. This is a table sourcenote. This is a table sourcenote.

Note: This is a table footnote.

[a]This is a table footnote. This is a table footnote. This is a table footnote.

4.1. Problem Formulation

The objective of variable coalescence-based offset assignment is to find both the coalescence scheme and the MWPC on the coalesced graph. We start with a few definitions and lemmas for variable coalescence.

Definition 4.1 (*Coalesced Node (C-Node)*). A C-node is a set of live ranges (webs) in the AG or IG that are coalesced. Nodes within the same C-node cannot interfere with each other on the IG. Before any coalescing is done, each live range is a C-node by itself.

Definition 4.2 (*C-AG (Coalesced Access Graph)*). The C-AG is the access graph after node coalescence, which is composed of all C-nodes and C-edges.

LEMMA 4.3. *The C-MWPC problem is NP-complete.*

PROOF. C-MWPC can be easily reduced to the MWPC problem assuming a coalescence graph without any edge or a fully connected interference graph. Therefore, each C-node is an uncoalesced live range after value separation and C-PC is equivalent to PC. A fully connected interference graph is made possible when all live ranges interfere with each other. Thus, the C-MWPC problem is NP-complete. □

LEMMA 4.4 (LEMMA SUBHEAD). *The solution to the C-MWPC problem is no worse than the solution to the MWPC.*

PROOF. Simply, any solution to the MWPC is also a solution to the C-MWPC. But some solutions to C-MWPC may not apply to the MWPC (if any coalescing were made). □

5. PERFORMANCE EVALUATION

During all the experiments, the Geographic Forwarding (GF) [Akyildiz 2001] routing protocol is used. GF exploits geographic information of nodes and conducts local data-forwarding to achieve end-to-end routing. Our simulation is configured according to the settings in Table I. Each run lasts for 2 minutes and repeated 100 times. For each data value we present in the results, we also give its 90% confidence interval.

6. CONCLUSIONS

In this article, we develop the first multifrequency MAC protocol for WSN applications in which each device adopts a single radio transceiver. The different MAC design requirements for WSNs and general wireless ad-hoc networks are compared, and a complete WSN multifrequency MAC design (MMSN) is put forth. During the MMSN design, we analyze and evaluate different choices for frequency assignments and also discuss the nonuniform back-off algorithms for the slotted media access design.

7. TYPICAL REFERENCES IN NEW ACM REFERENCE FORMAT

A paginated journal article [Abril and Plant 2007], an enumerated journal article [Cohen et al. 2007], a reference to an entire issue [Cohen 1996], a monograph (whole book) [Kosiur 2001], a monograph/whole book in a series (see 2a in spec. document) [Harel 1979], a divisible-book such as an anthology or compilation [Editor 2007] followed by the same example, however we only output the series if the volume number is given [Editor 2008] (so Editor00a's series should NOT be present since it has no vol. no.), a chapter in a divisible book [Spector 1990], a chapter in a divisible book in a series [Douglass et al. 1998], a multi-volume work as book [Knuth 1997], an article in a proceedings (of a conference, symposium, workshop for example) (paginated proceedings article) [Andler 1979], a proceedings article with all possible elements [Smith 2010], an example of an enumerated proceedings article [Gundy et al. 2007], an informally published work [Harel 1978], a doctoral dissertation [Clarkson 1985], a master's thesis: [Anisi 2003], an online document / world wide web resource [Thornburg 2001], [Ablamowicz and Fauser 2007], [Poker-Edge.Com 2006], a video game (Case 1) [Obama 2008] and (Case 2) [Novak 2003] and [Lee 2005] and (Case 3) a patent [Scientist 2009], work accepted for publication [Rous 2008], 'YYYYb'-test for prolific author [Saeedi et al. 2010a] and [Saeedi et al. 2010b]. Other cites might contain 'duplicate' DOI and URLs (some SIAM articles) [Kirschmer and Voight 2010]. Boris / Barbara Beeton: multi-volume works as books [Hörmander 1985b] and [Hörmander 1985a].

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

Rafal Ablamowicz and Bertfried Fauser. 2007. CLIFFORD: a Maple 11 Package for Clifford Algebra Computations, version 11. (2007). Retrieved February 28, 2008 from http://math.tntech.edu/rafal/cliff11/index.html

Patricia S. Abril and Robert Plant. 2007. The patent holder's dilemma: Buy, sell, or troll? *Commun. ACM* 50, 1 (Jan. 2007), 36–44. DOI:http://dx.doi.org/10.1145/1188913.1188915

Sten Andler. 1979. Predicate Path expressions. In *Proceedings of the 6th. ACM SIGACT-SIGPLAN symposium on Principles of Programming Languages (POPL '79)*. ACM Press, New York, NY, 226–236. DOI:http://dx.doi.org/10.1145/567752.567774

David A. Anisi. 2003. *Optimal Motion Control of a Ground Vehicle*. Master's thesis. Royal Institute of Technology (KTH), Stockholm, Sweden.

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

Kenneth L. Clarkson. 1985. *Algorithms for Closest-Point Problems (Computational Geometry)*. Ph.D. Dissertation. Stanford University, Palo Alto, CA. UMI Order Number: AAT 8506171.

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

Jacques Cohen (Ed.). 1996. Special Issue: Digital Libraries. *Commun. ACM* 39, 11 (Nov. 1996).

Sarah Cohen, Werner Nutt, and Yehoshua Sagic. 2007. Deciding equivalances among conjunctive aggregate queries. *J. ACM* 54, 2, Article 5 (April 2007), 50 pages. DOI:http://dx.doi.org/10.1145/1219092.1219093

Bruce P. Douglass, David Harel, and Mark B. Trakhtenbrot. 1998. Statecarts in use: structured analysis and object-orientation. In *Lectures on Embedded Systems*, Grzegorz Rozenberg and Frits W. Vaandrager (Eds.). Lecture Notes in Computer Science, Vol. 1494. Springer-Verlag, London, 368–394. DOI:http://dx.doi.org/10.1007/3-540-65193-4_29

Ian Editor (Ed.). 2007. *The title of book one* (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:http://dx.doi.org/10.1007/3-540-09237-4

Ian Editor (Ed.). 2008. *The title of book two* (2nd. ed.). University of Chicago Press, Chicago, Chapter 100. DOI:http://dx.doi.org/10.1007/3-540-09237-4

Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. 2007. Catch me, if you can: Evading network signatures with web-based polymorphic worms. In *Proceedings of the first USENIX workshop on Offensive Technologies (WOOT '07)*. USENIX Association, Berkley, CA, Article 7, 9 pages.

David Harel. 1978. *LOGICS of Programs: AXIOMATICS and DESCRIPTIVE POWER*. MIT Research Lab Technical Report TR-200. Massachusetts Institute of Technology, Cambridge, MA.

David Harel. 1979. *First-Order Dynamic Logic*. Lecture Notes in Computer Science, Vol. 68. Springer-Verlag, New York, NY. DOI:http://dx.doi.org/10.1007/3-540-09237-4

Lars Hörmander. 1985a. *The analysis of linear partial differential operators. III*. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. viii+525 pages. Pseudodifferential operators.

Lars Hörmander. 1985b. *The analysis of linear partial differential operators. IV*. Grundlehren der Mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences], Vol. 275. Springer-Verlag, Berlin, Germany. vii+352 pages. Fourier integral operators.

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

Markus Kirschmer and John Voight. 2010. Algorithmic Enumeration of Ideal Classes for Quaternion Orders. *SIAM J. Comput.* 39, 5 (Jan. 2010), 1714–1747. DOI:http://dx.doi.org/10.1137/080734467

Donald E. Knuth. 1997. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd. ed.)*. Addison Wesley Longman Publishing Co., Inc.

David Kosiur. 2001. *Understanding Policy-Based Networking* (2nd. ed.). Wiley, New York, NY.

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

Newton Lee. 2005. Interview with Bill Kinder: January 13, 2005. Video, *Comput. Entertain.* 3, 1, Article 4 (Jan.-March 2005). DOI:http://dx.doi.org/10.1145/1057270.1057278

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

Dave Novak. 2003. Solder man. Video. In *ACM SIGGRAPH 2003 Video Review on Animation theater Program: Part I - Vol. 145 (July 27–27, 2003)*. ACM Press, New York, NY, 4. DOI:http://dx.doi.org/99.9999/woot07-S422

Barack Obama. 2008. A more perfect union. Video. (5 March 2008). Retrieved March 21, 2008 from http://video.google.com/videoplay?docid=6528042696351994555

Poker-Edge.Com. 2006. Stats and Analysis. (March 2006). Retrieved June 7, 2006 from http://www.poker-edge.com/stats.php

Bernard Rous. 2008. The Enabling of Digital Libraries. *Digital Libraries* 12, 3, Article 5 (July 2008). To appear.

Mehdi Saeedi, Morteza Saheb Zamani, and Mehdi Sedighi. 2010a. A library-based synthesis methodology for reversible logic. *Microelectron. J.* 41, 4 (April 2010), 185–194.

Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi, and Zahra Sasanian. 2010b. Synthesis of Reversible Circuit Using Cycle-Based Approach. *J. Emerg. Technol. Comput. Syst.* 6, 4 (Dec. 2010).

Joseph Scientist. 2009. The fountain of youth. (Aug. 2009). Patent No. 12345, Filed July 1st., 2008, Issued Aug. 9th., 2009.

Stan W. Smith. 2010. An experiment in bibliographic mark-up: Parsing metadata for XML export. In *Proceedings of the 3rd. annual workshop on Librarians and Computers (LAC '10)*, Reginald N. Smythe and Alexander Noble (Eds.), Vol. 3. Paparazzi Press, Milan Italy, 422–431. DOI:http://dx.doi.org/99.9999/woot07-S422

Asad Z. Spector. 1990. Achieving application requirements. In *Distributed Systems* (2nd. ed.), Sape Mullender (Ed.). ACM Press, New York, NY, 19–33. DOI:http://dx.doi.org/10.1145/90417.90738

Harry Thornburg. 2001. Introduction to Bayesian Statistics. (March 2001). Retrieved March 2, 2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html

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.

DOI: 0000001.0000001