Near optimal update-broadcast of data sets

Sachin Agarwal
Deutsche Telekom A.G., Laboratories
Berlin, Germany-10587
sachin.agarwal@telekom.de Andrew Hagedorn Ari Trachtenberg
Boston University
Boston, MA, USA - 02215
{achag, trachten}@bu.edu

A version of this paper appeared as:

- Sachin Agarwal, Andrew Hagedorn, and Ari Trachtenberg, "Near optimal update-broadcast of data sets", International Workshop on Data Intensive Sensor Networks 2007 (DISN'07), May 2007.

***Abstract*— We consider the problem of efficiently broadcasting incremental updates to multiple terminals that contain outdated (and possibly different) initial copies of the data. This situation occurs, for example, with the broadcast of Short Messaging Service [SMS] or Multimedia Messaging Service [MMS] cellphone messages to various clients whose phones are sometimes unavailable. We propose an efficient protocol for effecting such broadcast based on a novel combination of recent work on rateless coding and set reconciliation. Our approach is non-interactive, in that terminal nodes need not send any messages to the source, and stateless, in that the source need not know (or store) any information about the terminals. It also minimizes communication complexity and energy expenditure at the terminal nodes, at the expense of added computation. In support of our work, we provide several energy usage measurements on MICA2 sensor motes that clearly highlight the advantages of random linear decoding over wholesale data transfer.**

I. Introduction

The problem of efficient broadcast is intrinsic to a wide variety of applications, especially those operating over wireless media. Sample applications include Short Messaging Service [SMS] or Multimedia Messaging Service [MMS] cellphone messages, routing table or (key, value) data broadcasts in wireless sensor networks, and maintaining consistency of directory contents, email messages in a folder, or address book entries, etc.

Our setup consists of one source and multiple terminals, each holding a common data set that is incrementally updated (addition/deletion of set elements) on the source. In this scenario, terminals should periodically update their local data sets when a version is available on the source. The number of differences between the source's newer data set and

older copies on the terminals are often small, but the intermittent terminal connectivity can result in different copies on each terminal, making a simple "difference" broadcast of new updates impractical. In this work, we focus on an efficient broadcast protocol that updates each terminal's data set to the source's data set with one way communication and no interaction (*i.e.,* no multiple communication rounds between the source and the terminals).

Formally we define the data set broadcast problem as follows. The source s holds a data D_s represented as a set of b-bit elements chosen at random from universal set U. This data set is to be propagated to n terminals $t_1 \dots t_n$ using only a forward broadcast channel from the source to terminals; each terminal t_i holds a (potentially old) version D_i of the data set. Updates to the data set are assumed to be incremental (*i.e.,* $|D_s - D_i| \ll |D_s|$ and $|D_i - D_s| \ll |D_i|$ for all $1 \leq i \leq n$).

Our proposed solutions are based on utilizing a recent set reconciliation algorithm, CPISync, for *rateless encoding*. In the typical mode, rateless encoding involves cleverly encoding a source data set into a large number of packets, any $\epsilon\Delta$ of which can be used to decode Δ new source items (which are considered *erasures*), for some protocol-dependent multiplicative constant $\epsilon > 0$. In our situation, we are interested in both insertions and deletions to the source data (*i.e.,* not necessarily just erasure errors), and we are also concerned with how the value Δ is determined, since the typical rateless code must expend significant computational resources (and sometimes additional communication resources) in determining this value. We thus propose a two-channel protocol in which Δ is determined over one channel with CPISync and utilized for rateless decoding on a secondary channel. We provide a demonstration of the potential reduction in communication and energy expenditure afforded by our approaches in application to SMS/MMS services and sensor network data.

A. Organization

In Section II we briefly discuss other approaches to similar problems in the literature. In Section III-A we provide communication lower bounds on the data set broadcast problem. We briefly describe CPISync, a key component of the proposed solution, in Section III-B. We present our solution, which is based on CPISync and random linear coding, in Section IV, and we also provide bounds on a terminal's communication complexity (*i.e.,* number of bits received). We provide simulation results in Section V and measurements of energy usage on MICA2 sensor motes in Section VI. Conclusions and future work are presented in Section VII.

II. Related Work

The analysis of communication complexity of two way protocols was proposed by Yao's seminal paper on communication complexity [1]. We shall use some of these results in Section III-A.

Schemes based on rateless coding for the erasure channel, such as digital fountain [2], have been proposed for large scale content distribution (in a client-server type model). Random linear coding [3] has been used for achieving network coded [4] min-cut bound on multicast in networks. Tornado

codes [5], LT codes [6], and Raptor codes [7] are other rateless codes which offer lower decoding complexity than random linear codes, at the expense of some inefficiency for low block lengths.

Our protocol seeks to minimize the terminal's communication complexity (*i.e.,* the number of bits it downloads in order to reconcile its data set with the source data set). We assume only one-way broadcast communication, similar to the model in [8, 9].

III. Preliminaries

We first provide an information-theoretic view of the amount of communication needed for one way broadcast-set dissemination. Then we briefly describe CPISync, a tool used in the protocols proposed in Section IV.

A. Lower bounds

Yao's well known results on zero-error interactive, deterministic communication [1] lower bound the set equivalence problem to a linear multiple of the set size. Set equivalence, a problem of determining if two sets are equal with minimum communication, is a special case of the set reconciliation problem, and the lower bound thus applies also to the latter.

It is a simple matter to generalize this result to reconciling multiple targets with a source through a broadcast medium, with the result that this too requires communication linear in the sizes of the various sets. All this applies to deterministic zero-error set reconciliation. Probabilistic methods, on the other hand, allow for small probabilities of error to achieve better communication bounds.

The straightforward method of transmitting the updated source set D_s to all the terminals is by broadcasting the entire set at one time so that the terminals can simply store this new version of the set. The communication complexity is linear in the set-size $|D_s|$, making this method expensive for large data sets.

1) Combinatorial omniscient lower bound: We derive the lower bound on the communication bound for the case when terminals sets D_i are known at the source; our derivation is based on the two-party reconciliation bound in [10].

Assume that $m_s = |D_s - D_i|$ and $m_i = |D_i - D_s|$. Therefore m_s corresponds to the number of insertions and m_i corresponds to the number of deletions from D_i needed to derive D_s from D_i at terminal t_i. The communication complexity for a deterministic protocol, when D_i is known at the broadcast source s, is lower bounded by the logarithm of the possible number of distinct protocol messages.

$$COMM \geq log_2 \left[\binom{2^b - |D_i|}{m_s} \cdot \binom{|D_i|}{m_i} \right] bits \quad (1)$$

Then using a well known lower bound on binomial coefficients [11], we obtain

$$\begin{aligned} COMM &\geq m_s log(2^b - |D_i|) - m_s log m_s \quad (2) \\ &+ m_i log(|D_i|) - m_i log m_i \quad (3) \end{aligned}$$

Assuming sparse sets ($|D_i| \ll 2^b$) and a small number of insertions and deletions ($m_s, m_i \ll |D_i|$) we get a lower bound on the communication as linear in the *number of insertion and deletions*, rather than the overall set sizes.

B. CPISync

The Characteristic Polynomial Interpolation Synchronization [CPISync] approach was first proposed in [10] and has been successfully employed in ad hoc two-way set reconciliation in practical settings [12–14]. The algorithm is based on an algebraic solution to the problem of reconciling two remote sets of information, and it has been shown to be efficient [15] in a very common data synchronization scenario - where the number of differences to be reconciled is far smaller than the sizes of the reconciling data sets.

One feature of the CPISync algorithm is that its communication complexity is linear in the number of differences between data sets, and is for all intents and purposes independent of the overall sizes of the data sets being reconciled. CPISync does not need to maintain any state information about any other devices on a network.

a) Protocol Overview: The CPISync algorithm represents data set elements as integers in a finite field, with each integer representing a set element. A set $\{x_1, x_2, x_3, \ldots, x_n\}$ is represented by a *characteristic polynomial*

$$\chi_S(Z) = (Z - x_1)(Z - x_2)(Z - x_3)\ldots(Z - x_n).$$

The key to CPISync is the observation that for two sets S_A and S_B,

$$\frac{\chi_{S_A}(Z)}{\chi_{S_B}(Z)} = \frac{\chi_{\Delta_A}(Z)}{\chi_{\Delta_B}(Z)}, \tag{4}$$

where Δ_A represents the set difference $S_A - S_B$, and similarly $\Delta_B = S_B - S_A$. This is because terms common to both sets cancel out in the numerator and denominator. Thus, the rational function in 4 can be uniquely interpolated from $\bar{m}$ samples, if there are at most $\bar{m}$ differences between synchronizing sets.

The CPISync algorithm can be generally described as follows:

1) Hosts A and B evaluate their characteristic polynomials on $\bar{m}$ sample points (over a chosen finite field). Host A sends its evaluation values to host B.
2) The evaluation values are combined to compute $\bar{m}$ sample points of the rational function $\frac{\chi_{S_A}(Z)}{\chi_{S_B}(Z)}$, which are interpolated to determine $\frac{\chi_{\Delta_A}(Z)}{\chi_{\Delta_B}(Z)}$.
3) The numerator and denominator of the interpolated function are factored to determine the differences between S_A and S_B.

CPISync then, can be thought of as a rateless protocol over an error channel that may both add and delete data from a data set (unlike the erasure channel that only "erases" some data).

C. Random Linear Coding

Random linear coding provides a simple method of data dissemination, and has the advantages of easy implementation. In this model, source s represents each set element $x_i \in D_s$ in an appropriate finite field $\mathbb{F}$. These elements are then encoded into $n > k$ packets $\{y_1, y_2, \ldots y_n\}$ as the following random linear combinations $y_i = \sum_{j=1}^{k} \beta_{i,j} x_j$, where $\beta_{i,j}$ are randomly chosen elements in the finite field $\mathbb{F}$. The parameters of the encoding can be easily adjusted so that the rows $[\beta_{i,1}, \beta_{i,2}, \ldots \beta_{i,k}]$ are linearly independent with high probability. Thus, any terminal that receives k of the encoded packets (y_i's) can solve the corresponding system of linear

equations to determine each unknown set element x_i.

The $[\beta_{i,1}, \beta_{i,2}, \ldots \beta_{i,k}]$ rows can either be known a priori at the terminal through a common random generator seed, or can be efficiently transmitted with the encoded packets [3].

IV. Protocols

Our first algorithm is a simple broadcast generalization of the peer-to-peer synchronization algorithms in [15, 16].

A. One Channel Protocol

1) Algorithm on the source: The data set source s constructs the characteristic polynomial of set D_s as explained in Section III-B. The characteristic polynomial $\chi_{D_s}(z)$ is sampled on $|D_s| = N$ sample points to obtain (sample point, evaluation value) tuples of the form $\{(s_1, \chi_{D_s}(s_1)), (s_2, \chi_{D_s}(s_2)), \ldots, (s_N, \chi_{D_s}(s_N))\}$.

The sample points should be different from the data points, otherwise the sample points would be one of the zeros of the characteristic polynomial. If each element of the data set D_s is represented by a b-bit number, then by choosing sample points in the range of $(2^b + 1, 2^{b+1} - 1)$ we avoid any collisions between the data points and the sample points. This makes each (sample point, evaluation value) tuple $2b + 2$ bits long. A pseudo-random number can be used to generate the sample points at the source and terminals, thus reducing the size of each transmitted tuple by half.

The data set source then starts transmitting the N evaluation values on the broadcast channel. The transmission is in the form of a "data-carousel" [2] *i.e.,* transmission is "looped-around" repeatedly for the time period when the source wants to transmit updates to the terminals. Terminals "tune-in" to this broadcast and download a requisite number of evaluation values to decode the additions and deletions to the data set.

2) Algorithm on the data terminals: A terminal t_i with data set D_i starts downloading the evaluation values being broadcast by the source. The number of tuples needed to correctly interpolate the rational function of Equation 4 is lower bounded by the sizes of the difference sets

$$m = |D_s - D_i| + |D_i - D_s|,$$

as explained in III-B.

The terminal however, does not have this set difference size m. It can guess a small value m_1, download $m_1 + k$ evaluation values, and then try to interpolate the rational function of Equation 4. A small number of (reusable) k evaluation values are used to check if the interpolated rational function is the correct interpolation (Selecting k is explained in [13, 16]). If this check indicates incorrect interpolation, then further evaluation points are downloaded and the process is iterated until the correct rational function is obtained.

3) Analysis: We now analyze the proposed protocol based on the analysis provided in [16]. Each evaluation value of χ_{D_s} requires $b+2$ bits to download. If the terminal downloads only one evaluation at a time before trying CPISync, then the terminal communication complexity is limited to $(b+2)(m+k)$ bits.

For faster operation, the terminal can increase the number of downloaded evaluation values by a factor of c in every trial. This would make the communication complexity

$$(b+1)c(m+k) + \lceil log_c(m+k) \rceil. \quad (5)$$

Further, this approach guarantees that the terminal will need to "tune" into the carousal stream to download extra evaluation values no more than $\lceil log_c(m+k) \rceil$ times.

B. Two Channel Protocol

The algorithm of Section IV-A works very well for data sets containing small set elements. In case each data element of the set is very large, the finite field used in CPISync to represent set elements becomes correspondingly large. This has two substantial effects on the terminal: (i) the larger finite field slows down factorization and, thus, decoding; (ii) the penalty (*i.e.*, wasted bandwidth) for downloading too many large evaluation values (higher value of c in Equation 5) increases.

For large data elements, we thus split the broadcast channel into two components. A smaller channel carries the CPISync evaluation values tuples computed from a the set of *hash-values* of the real data set (say, 128-bit MD5 [17] hashes of the large set elements), and another larger channel that carries random linear encodings of the real data set elements, constructed along the lines of [3].

Both channels are "streamed" in a data-carousel fashion over the two independent channels. Terminals first use CPISync to compute the set of hash values comprising D_s and then download an appropriate number of random-linear encoded packets. In effect, we are using CPISync to determine $|D_s - D_i|$ and $|D_i - D_s|$ before solving the system of equations for random linear decoding of the different data set elements. As such, only $m_s = |D_s - D_i|$ encoded y's need to be downloaded to successfully decode the m_s unknowns. Note that CPISync also provides complete information of deletions by reporting the hash values of deleted entries; this can be matched to the corresponding elements in D_i that need to be removed.

In addition, by ascertaining the exact number of the relatively large random-linear encoded packets to be downloaded a priori using CPISync, the receiving terminal further reduces its energy usage by avoiding excess downloads of large random linear encoded packets.

1) Analysis: The two channel protocol exploits the relative sparsity of the data set D_s to represent each B-bit set element by a much smaller b bit hash *i.e.*, usually,

$$|D_s| \ll 2^B, \ and$$
$$b \ll B.$$

For example, each set element of D_s may be a set of several thousands of high resolution images of several megabytes each, and a MD5 [17] hash could represent each set element by a distinct $b = 128$ bit number. The high penalty of choosing a larger value of c in Equation 5 can be avoided by first reconciling hash-sets (b-bit elements).

V. Example: Broadcast of bulk SMS and MMS

We consider the scenario of a cellular network wireless-operator that broadcasts SMS and MMS

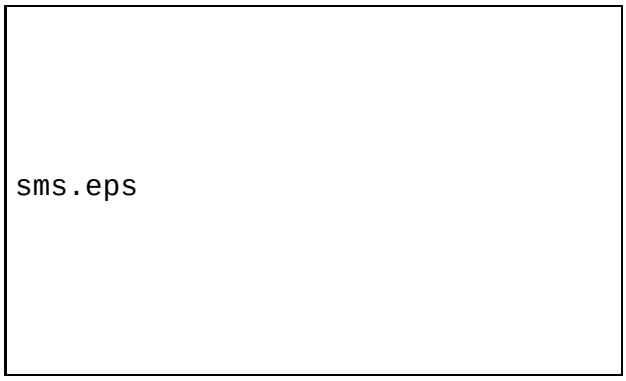

Fig. 1. Comparison of one-channel protocol to unicast/simple broadcast

promotional advertisements and coupons to subscribers' cellphones. Our specific example will be based on T-Mobile's United Kingdom cellular network, which has about 20 million subscribers.

We assume that at any time the current SMS or MMS promotion data set D_s contains 1000 messages and the set is updated once a day, with 5 percent of the set elements being added or removed randomly per day. In a realistic scenario, subscribers' cellphones may be turned on or off during the broadcast of updates. Different cellphones (terminals) may therefore hold different versions of the SMS or MMS data set at any given time.

A. Short Messaging Service (SMS)

The subscribers would not wish to receive full copies of (plaintext) D_s because receiving 1000 SMS messages (each at most 140 bytes) over the wireless channel significantly drains the battery of their cellphones. Moreover, the broadcasting source may not want to maintain state about each cellphone in the network and its corresponding version of the data set. There will also be network feasibility issues when broadcasting 20 billion SMS messages (20 million cellphones $\times$ 1000 SMS messages per cellphone) per day. This problem is even further aggravated by the fact that SMS messages are passed over control channels and, thus, not only use up network bandwidth but also potentially interfere with setting up other calls.

If the source broadcaster continuously broadcasts a carousal of D_s messages (looping around when it has broadcast the entire set), each cellphone will still have to receive 1000 SMS messages in the worst case in order to compare and update its local data set to D_s. This approach does mitigate the issue of cellphones being turned off, but it is not helpful in reducing energy usage on the cellphones.

Using the protocol described in Section IV-A would be much more efficient and feasible for the broadcasting source and each cellphone. A cellphone updating from a previous day would download about 50 messages only (5% differences). More significantly, cellphones holding older and different versions can all update to D_s without any two-way communication with the source broadcaster and without the latter having to store per-cellphone state.

Figure 1 shows the terminal communication complexity in terms of the number of SMS messages to be received on a cell-phone in order to update to the latest promotion SMS data set D_s. We note that for incremental changes to the data set (less than 50% of the data set size), the proposed algorithm significantly outperforms broadcast.

B. Multimedia Messaging Service (MMS)

MMS messages use a combination of SMS and WAP messaging [18] to transmit multimedia binary data to cellphones. An SMS message from the server to the cellphone is first used to signal the URL of the actual multimedia content to the cellphone and the cellphone then initiates a WAP session to download this data from the MMS server via HTTP.

Our one channel protocol is obviously not scalable to the type of bulk transfer suggested by the above example. Instead, the protocol of in IV-B can be used to disseminate data set information using CPISync, and the large multimedia messages can then be efficiently disseminated using random linear codes.

VI. Sensor Energy Usage Measurements

In this section we provide some preliminary energy usage measurements for downloading and decoding random linear encoded data, and we compare these with the energy cost of downloading the entire data set from the source. The computational complexity of encoding is $O(mn)$ for n elements of length m, but this has a negligible effect on the receiver. Our experiments were performed on 900 MHz MICA2 [19] motes, and energy was measured as the root mean square voltage across a 10 Ohm resistor in series with the mote; each data message (set element) was set at 29 bytes. We compare the energy used while downloading 5 percent of the dataset and then decoding the random linear encoded data to the energy used for downloading the entire data set.

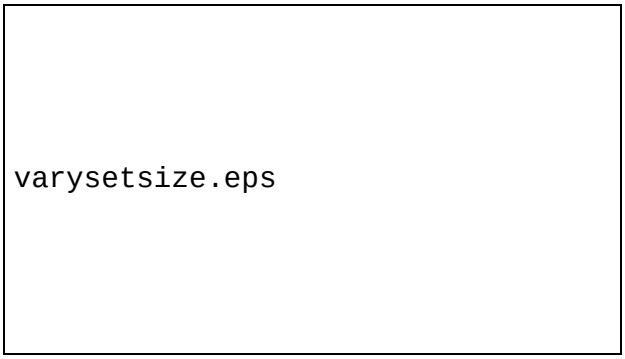

Fig. 2. Comparison of full download and RLC energy

Figure 2 shows that the energy required to download the entire data set is significantly more than the energy cost of downloading 5 percent (corresponding to 5 percent differences) and then decoding. The computational complexity of decoding is non-linear ($\omega(n^2)$) in the number of decoded messages n due to Gaussian elimination step in the decoding algorithm, and this explains the non-linear rise in the energy usage in Figure 2.

Figure 3 shows the percentage of the data set at which the energy cost of decoding is equivalent to downloading the entire file. When the data set is 100 messages, 35.4 percent of the file can be downloaded and decoded before this happens, but as the data set size increases to 800 messages the percentage of the data set drops to 11 percent. It is important to note that current memory constraints on the wireless motes require a piecemeal decoding of large data sets, so that a divide-and-conquer approach for limiting n can be applied in practice.

VII. Conclusions

We have proposed two non-interactive protocols for efficient one-way updates of data sets in a broad-

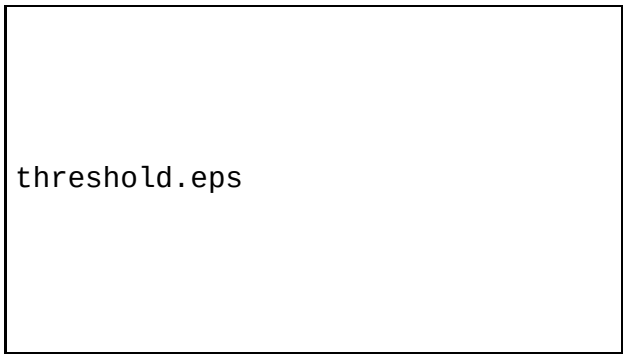

Fig. 3. Percentage when of full download and RLC energy are equal

cast setting. Our protocols are designed to minimize received bytes on the terminals, a very important practical parameter for power conservation in wireless sensors and mobile devices.

We have provided analytical lower bounds on the communication complexity of probabilistic algorithms for one-way broadcast reconciliation of data sets, supported by practical examples. Measurements on sensor motes further support the case that the proposed protocols can achieve significant power savings in broadcast sensor networks.

As future work, we plan to extend the protocols for large data sets using a divide-and-conquer approach to limit set size. Understanding the tradeoffs between various rateless codes in our data set broadcast applications is also an important area of future work.

REFERENCES

[1] A. C. Yao, "Some complexity questions related to distributive computing," in *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, 1979, pp. 209–213.

[2] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," *Proceedings of ACM SIGCOMM '98*, pp. 56–67, September 1998.

[3] P. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Proceedings of the 51st Allerton Conf. Communication, Control and Computing*, 2003.

[4] R. Ahlswede, Ning Cai, S. Y. R. Li, and R. W. Yeung, "Network information flow," *Information Theory, IEEE Transactions on*, vol. 46, no. 4, pp. 1204–1216, 2000.

[5] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann, "Practical loss-resilient codes," *Proceedings of the 29th ACM Symposium on Theory of Computation*, 1997.

[6] Michael Luby, "Lt codes," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 271–282.

[7] A. Shokrollahi, "Raptor codes," 2003.

[8] Utku Irmak, Svilen Mihaylov, and Torsten Suel, "Improved single-round protocols for remote file synchronization.," in *INFOCOM*, 2005, pp. 1665–1676.

[9] Julian Chesterfield and Pablo Rodriguez, "Deltacast: efficient file reconciliation in wireless broadcast systems," in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*, New York, NY, USA, 2005, pp. 93–106, ACM Press.

[10] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," in *International Symposium on Information Theory*, June 2001, p. 232.

[11] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.

[12] Yaron Minsky, "The SKS OpenPGP Key Server," http://www.nognu.org/sks/.

[13] Ari Trachtenberg, David Starobinski, and Sachin Agarwal, "Fast PDA Synchronization Using Characteristic Polynomial Interpolation," in *Proceedings of the IEEE Infocom*, 2002.

[14] Y. Minsky and A. Trachtenberg, "Scalable set reconciliation," in *Proc. 40-th Allerton Conference on Comm., Control, and Computing*, Monticello, IL., October 2002.

[15] D. Starobinski, A. Trachtenberg, and S. Agarwal, "Efficient pda synchronization," *IEEE Trans. on Mobile Computing*, vol. 2, no. 1, January-March 2003.

[16] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set recon-

ciliation with nearly optimal communication complexity," *IEEE Transactions on Information Theory*, vol. 49, no. 9, pp. 2213 – 2218, 2004.

[17] R.L. Rivest, "The md5 message digest algorithm," April 1992.

[18] "Wireless application protocol," http://www.wapforum.org/.

[19] "Mica2 sensor motes," http://xbow.com.