Summarizing Distributed Data Streams for Storage in Data Warehouses

Raja Chiky and Georges Hébrail

TELECOM ParisTech
LTCI-UMR 5141 CNRS
Paris, France
firstname.lastname@telecom-paristech.fr

Abstract. Data warehouses are increasingly supplied with data produced by a large number of distributed sensors in many applications: medicine, military, road traffic, weather forecast, utilities like electric power suppliers etc. Such data is widely distributed and produced continuously as data streams. The rate at which data is collected at each sensor node affects the communication resources, the bandwidth and/or the computational load at the central server. In this paper, we propose a generic tool for summarizing distributed data streams where the amount of data being collected from each sensor adapts to data characteristics. Experiments done on electric power consumption real data are reported and show the efficiency of the proposed approach.

1 Introduction

A growing number of applications in areas like utilities, retail industry and sensor networks deal with a challenging type of data: data elements arrive in multiple, continuous, rapid and time-varying data streams. A key requirement of such applications is to continuously monitor and react to interesting phenomena occurring in the input streams. For instance, electric power utilities may want to alert their customers when their consumption is going to exceed a threshold during a period with expensive electric power.

Many solutions have been developed recently to process data streams that arrive from a single or from multiple locations. The concept of Data Stream Management System (DSMS) has been introduced and several systems (including commercial ones) have been developed [4,7,17,18,21]. These systems enable to process data streams on-the-fly and issue alarms or compute aggregates in real-time without storing all data on disk. But in many applications there is a need to keep track of such data in a data warehouse for further analyses. For instance electric power utilities (or telecommunication companies) need to keep track of the way their customers use electric power (telecommunication services) upon time: applications concern power (telecommunication) consumption forecast, definition of new price policies, ... It is inconceivable to load all such data in a data warehouse due to its volume, arrival rate and possible spatial distribution.

This paper proposes an approach to summarize the contents of a large number of distributed data streams describing uni-dimensional numerical data, where each data source corresponds to a sensor which periodically records measures in a specific field (electric power consumption, use of telecommunication services, ...). The approach is based on

I.-Y. Song, J. Eder, and T.M. Nguyen (Eds.): DaWaK 2008, LNCS 5182, pp. 65–74, 2008.

optimized temporal sampling of data sources in order to produce summaries of good quality with respect to computing, storage and network constraints. The computation of the summaries is decentralized and some sensor computing capacities are used.

The basic scheme is to choose the best policy of summarizing data over each temporal window t taking into account data of the previous window $t-1$. This assumes that data is somewhat periodic and that the window length is chosen to be compatible with this period. This assumption is realistic in many applications and in particular in the one studied in this paper concerning sensors which are communicating electric power meters.

The paper is organized as follows. Section 2 describes related work in this area. In Section 3, we present the proposed summarizing system. Section 4 evaluates the approach on real data and Section 5 gives an outlook upon our ongoing and future research in this area.

2 Related Work

Summarizing online data streams has been largely studied with several techniques like sketching ([8,9]), sampling and building histograms ([2,10]). Authors in [15] experimentally compare existing summarizing techniques and use the best one in terms of reconstruction accuracy; they also introduce a user-defined model to reduce data quality when data gets older. However, most work concentrates on summarizing a single stream and without exploiting possible bi-directional communications between a central server and its sensors nor computing facilities within sensors.

In the context of DSMS's, load shedding techniques ([1,19]) drop randomly data stream records when the load of the system increases beyond what it can handle. But these approaches do not scale to a large number of streams issued by remote sensors.

In the context of sensor networks, several adaptive sampling techniques have been developed to manage limited resources. The aim is to preserve network bandwidth (or the energy used for wireless communications) and storage memory by filtering out data that may not be relevant in the current context. Data collection rates thus become dynamic and adapted to the environment.

An adaptive sampling scheme which adjusts data collection rates in response to the contents of the stream was proposed in [11]. A Kalman filter is used at each sensor to make predictions of future values based on those already seen. The sampling interval SI is adjusted based on the prediction error. If the needed sampling interval for a sensor exceeds that is allowed by a specified Sampling Interval range, a new SI is requested to the server. The central server delivers new SIs according to available bandwidth, network congestion and streaming source priority. Again a probabilistic model is used in [5]: a replicated dynamic probabilistic model minimizes communications between sensor nodes and the central server. The model is constructed by exploiting intelligently historical data and spatial correlations across sensors. Correlations between sensors can be computed incrementally as described in [16].

A backcasting approach to reduce communication and energy consumption while maintaining high accuracy in a wireless sensor network was proposed in [20]. Backcasting operates by first activating only a small subset of sensor nodes which

communicate their information to a fusion center. This provides an estimate of the environment being sensed, indicating some sensors may not need to be activated to achieve a desired level of accuracy. The fusion center then backcasts information based on the estimate to the network and selectively activates additional sensors to obtain a target error level. In this approach, adaptive sampling can save energy by only activating a fraction of the available sensors.

In [14], authors present a feedback control mechanism which makes dynamic and adaptable the frequency of measurements in each sensor. Sampled data are compared against a model representing the environment. An error value is calculated on the basis of the comparison. If the error value is more than a predefined threshold, then a sensor node collects data at a higher sampling rate; otherwise, the sampling rate is decreased. Sensor nodes are completely autonomous in adapting their sampling rate.

In [13], authors present a method to prevent sensor nodes to send redundant information; this is predicted by a sink node using an ARIMA prediction model. Energy efficiency is achieved by suppressing the transmission of some samples, whose ARIMA based prediction values are within a predefined tolerance value with respect to their actual values. A similar approach is proposed by Cormode and Garofalakis [6]. Their results show that reduced communication between sensors and the central server can be sufficient by using an appropriate prediction model. A wide range of queries (including heavy hitters, wavelets and multi-dimensional histograms) can be answered by the central server using approximate sketches.

3 Architecture of the Summarizing System

We consider a distributed computing environment, describing a collection of n remote *sensors* and a designated *central server*. Sensors communicate with the central server which is responsible for loading into the data warehouse measurements done by sensors. We assume that communications between the central server and sensors are bidirectional and that there is no communication between sensors. Such architecture is representative of a large class of applications. In the particular case of electric power metering, several million meters feed a unique data warehouse but using intermediate network nodes which play the role of the central server of our model. There are typically several hundreds of meters directly connected to an intermediate node.

We assume that sensor measures are generated regularly at the same rate for every sensor. We cut out all streams into periods (also called temporal windows) of the same duration. A typical period corresponds to a duration of one day. A temporal window is composed of p elements for each sensor (the number of elements is the same for all sensors). Elements issued by a sensor during a period is also referred as a *curve* in the following. For each sensor, the number of elements gathered by the central server within a temporal window will vary between m (fixed parameter) and p. Let us define s as the maximum number of elements which can be received by the central server from the n sensors during a temporal window ($s < n * p$).

From the observation of what is happening during a period $t - 1$, we determine a data collection model to apply to the n sensors for period t. The problem consists of finding the best "policy of summarizing" each sensor, respecting the constraints of

the maximum number of data collected by the central server (parameter s) and the minimum number of data taken from each sensor (parameter m).

The data collection model should preserve as much as possible the detailed information by reducing summarizing errors. Several measures can be used to measure errors. We use in this paper the Sum of Square Errors SSE, which is the square of the L_2 distance between the original curve $C = \{c_1, c_2, ..., c_p\}$ and an estimation from the summarized curve $\hat{C} = \{\hat{c}_1, \hat{c}_2, ..., \hat{c}_p\}$. SSE is computed as follows: $SSE(C, \hat{C}) = \sum_{i=1}^{p}(c_i - \hat{c}_i)^2$ where p is the number of elements in C. Other error measures can be defined depending on the application. SSE is well adapted to electric power consumption. Note that error measures cannot be defined arbitrarily and some properties (not studied in this paper) are necessary for the optimization problem to be solved in reasonable time.

During a period t, sensors send data to the server at the scheduled sampling rate along with SSE's values for different assumptions of sampling rates corresponding to sending between m and p values. Then the server solves an optimization problem to find the best sampling rate for each sensor during period $t + 1$. Actually, we assume that the number of sensors connected to a server is large enough to distribute uniformly with time all data exchanges so that there is no temporary overload of the server.

3.1 Sensor Side Module

Several sampling schemes are considered by each sensor to summarize its curve during a period. Almost all methods of curve compression can be used depending on the handled data. For the needs of our application, we concentrated on three methods: regular sampling, curve segmentation and wavelet compression.

Regular sampling: Curves are sampled using a step j chosen between 1 and $m' = \lfloor \frac{p}{m} \rfloor$ ($\lfloor x \rfloor$ is the floor function). $j = 1$ means that all elements are selected, $j = 2$ means that every two elements are selected and so on. This technique is described as *regular* because selected points are temporally equidistant, a jump j is made between two sampled elements. As for the estimation of the original curve from the sampled elements, two interpolation methods are used: "Linear Interpolation" and "Constant Interpolation". Given one interpolation method, SSE's for the different values of j can be computed by the sensor and sent to the server.

Curve segmentation: Segmentation approximates a curve by using a *piecewise representation*. The curve is split into a given number of segments and each segment is associated with a constant value. Segmentation has been considered in many areas and there exists an algorithm from Bellman [3] which computes exactly the segment bounds which minimize the SSE between the curve and its piecewise representation given the number of segments. Other error measures can be used and lead to other optimization algorithms (see [12]). The number of segments applicable to a sensor varies between $\lfloor \frac{m}{2} \rfloor$ and p. Indeed, the number of elements of each segment is also needed to construct the piecewise representation. If a k-segmentation is applied to a curve C, the central server needs $\{(\hat{c}_1, n_1), (\hat{c}_2, n_2), ..., (\hat{c}_k, n_k)\}$ to construct the piecewise representation $\hat{C}$, where $\hat{c}_l$ is the average value of segment l composed of n_l elements. Thus, we can assign an integer value j to a curve from 1 (all elements are collected) to m' (optimally

the curve is segmented in $\lfloor \frac{p}{2*m'} \rfloor$ segments). Here again each sensor performs segmentation locally with different numbers of segments, computes the corresponding SSE's and can send SSE's corresponding to different summarizing levels indexed by a value j varying between 1 and m'.

Curve compression: Wavelet coefficients are projections of the curve onto an orthogonal set of basis vectors. The choice of basis vectors determines the type of wavelets. Haar wavelets are often used for their ease of computation and are adapted to electric power consumption curves. The estimation of a curve from its summarized version is a linear combination of a number of basis vectors chosen from $\{1, 2^1, 2^2, ...2^J\}$, ($j = 1$ means the whole curve is collected). J is the biggest integer such that $2^J \leq m'$. The number of data points in a curve must be a power of 2 in the case of the Haar wavelet. If it is not the case zeros are padded at the right end of the curve. Again, each sensor computes the wavelet decomposition with different levels indexed by j varying from 1 to m' and send the corresponding SSE's to the server.

At the end of every temporal window, sensors compute the SSE's corresponding to each summarizing procedure (regular sampling, curve segmentation and curve compression) and send for each value of index j the minimum of errors between these three methods to the central server.

3.2 Central Server Module

At the end of each period, the central server receives from each sensor different values of SSE's corresponding to different levels of summarization, indexed by j varying between 1 and m'. These SSE's are stored in a matrix $W_{n*m'}$ of n lines and m' columns (n is the number of sensors connected to the server). Element w_{ij} of the matrix corresponds to the SSE obtained when collecting data from sensor i with a summarizing level j.

Instead of distributing equally the summarizing levels between all sensors for period $t + 1$, an optimization is performed to assign different sampling rates to sensors from SSE's computed during period t. The central server solves the following optimization problem minimizing the sum of SSE's on sensors:

$$\text{Minimize } \sum_{i=1}^{n} \sum_{j=1}^{m'} (W_{ij} \times X_{ij})$$

subject to:

$$\begin{cases} X_{ij} = 0 \text{ or } 1 & \\ \sum_{j=1}^{m'} X_{ij} = 1 & \text{i from 1 to n} \\ \sum_{i=1}^{n} \sum_{j=1}^{m'} (\left\lfloor \frac{p}{j} \right\rfloor \times X_{ij}) \leq s & \text{i from 1 to n} \end{cases}$$

This is a problem of assignment of summarizing levels to the curves respecting the constraints above. A variable $X_{ij} = 1$ means that the central server assigns summarizing level j to the curve of index i. The second constraint $\sum_{j=1}^{m'} X_{ij} = 1$ imposes only one value of j per curve. Lastly, the third constraint means that the number of data to be communicated to the central server should not exceed the threshold s. Once the optimization problem is solved, the j_i are sent to each sensor. The sensor uses the

summarizing method corresponding to the smallest SSE value. At the end of the period, sensors send the SSEs to update the matrix of errors as explained above. This process continues as long as data streams arrive on line. This optimization problem can be solved using standard integer linear programming techniques. We used a solver based on the simplex method and Branch-and-bound methods. This approach solves in reasonable time(less than one minute using a standard PC 2GHZ with 1.5 GB) problems with less than 30000 variables, which is quite enough in our context (up to 1000 sensors connected to each intermediate node).

4 Experimental Validation

The approach described above has been assessed empirically on real data against a standard non adaptive approach called the *fixed* approach. In this approach, the three summarizing methods have been considered but the number of selected elements is equally distributed between sensors depending on parameter s.

Experiments were carried out on two real data sets issued from electric meters. Each meter produces a curve representing the electric power consumption of one customer. The first data set describes 1000 meters each producing one measure every 10 minutes during one day (144 measurements per meter per day). This data set is used to assess the efficiency of the approach in the case of a *stationary* consumption for each customer: the optimized sampling rates are assessed on the same curve which is used to choose them. This experiment is referred as "stationary window" below.

The second data set describes 140 meters each producing one measure every 30 minutes but during one year (365 days). It is used to assess the efficiency on a non-stationary consumption for each customer: the optimized sampling rates are assessed on the curve of a different day than the one used for choosing them. This second experiment will be referred as "jumping window" below.

These data sets are those available today. They do not show very high rates of elements. In the next future with the generalization of automatic communicating meters, each meter will possibly generate a measure every one second.

4.1 Stationary Window Results

In these experiments, we used $m = 7$ ($m' = \lfloor \frac{144}{7} \rfloor = 20$). This means that sampled curves consist of at least 7 values if they are built regularly, of 3 values if built with segmentation and of 9 with wavelet compression. We point out that a curve in our example consists of 144 points. Several values of threshold s have been tested: the results are presented in Table 1. Each column of the table corresponds to a threshold value s. $s = 144000 = 144 * 1000$ means that all elements are collected for all curves. Experimented thresholds are equal to 144000 divided by one of $\{1, 2, 3, 5, 7, 10, 15\}$. For instance, $s = 48000$ is the third of total number of data ($\frac{144000}{3}$).

For each method the Sum of Square Errors have been computed. The standard deviation is also presented to illustrate the dispersion of number of data computed by optimization around the average which is equal to $\frac{s}{1000}$ ($n = 1000$).

Table 1. Experimental Results. SSE: Sum of Square Error. opt: sampling steps (#segments or #wavelet coefficients) obtained by optimization. avg: Average and std: Standard Deviation. Samp CI: Regular Sampling with Constant Interpolation, Samp LI: Regular Sampling with Linear Interpolation, seg: Segmentation and wav: Wavelet Compression.

Threshold s	144000	72000	48000	28800	20571	14400	9600
SSE opt Samp CI	0	696.36	1571.79	3037.02	4265.46	5839.95	8767.05
SSE opt Samp LI	0	434.35	996.34	1969.67	2781.54	3883.9	5676.6
SSE opt seg	0	161.62	385.13	874.1	1346.58	2027.17	3432.16
SSE opt wav	0	381.15	916.22	1881.66	2724.08	3888.6	6118.8
SSE opt all	0	139.29	346.05	801.91	1240.62	1859.6	3026.24
SSE fixed CI	0	2770.8	4329	8040	9313	13157	16048
SSE fixed LI	0	1956	2821	4245	5501	7024	9087
SSE fixed seg	0	505.8	860.8	1535	2042.6	2761.5	4677.1
SSE fixed wav	0	1385.32	2725.35	4766.09	4766.09	7179.5	7179.5
avg data/meter	144	72	48	28.8	20.57	14.4	9.6
std data CI	0	50.3	44.8	32.05	24.48	15.62	4.4
std data LI	0	51	45.45	33.6	24.7	16.5	4.9
std data seg	0	43	35.73	25.32	17.64	10.6	3.9
std data wav	0	48.9	42	30	21.8	13.5	3

As can be seen in Table 1, the optimization performs well for all summarizing methods, it considerably reduces the Sum of Square Errors compared to the "fixed" approach. For instance, in the case of regular sampling with Linear Interpolation, the optimization reduces the SSE by about 77% for threshold $s = 72000$ compared to a"fixed regular sampling". The transmission cost of SSE values is not included in the reported experiments. This impacts only on the comparison between the 'fixed' and 'optimized' approaches, since SSE's transmission is not needed in the 'fixed' approach. However, the number of SSE values to transmit is bounded and can be considered as negligible compared to the number of elements p in the window.

In our experiments, segmentation always gives better results than the two other summarizing methods. We also experimented with the case where meters choose themselves the summarizing method to adopt according to the affected j. We notice that the Sum of Square Errors is again reduced thanks to the automatic selection of summarizing method (line 'SSE opt all' of table 1).

We observe high values of standard deviations mainly for high values of threshold s. The sampling rates significantly deviate from the average. This confirms the importance of selecting sampling steps by optimization. The same observation is true for all summarizing methods.

4.2 Jumping Window Results

Previous experiments evaluate the error made over a period from an optimization carried out over the same period. The global approach consists of summarizing curves over a period based on optimization results computed over a previous period. We present here the method adopted for a streaming environment as well as first experiments completed on 140 load curves available over one year period.

Optimization Using d-1. We used the second dataset of 140 curves measured every 30 minutes over one year to validate the efficiency of the proposed data collection scheme. Table 2 gives the percentage of periods (temporal windows) for which the SSE using optimized summarizing is lower than the SSE using "fixed" summarizing. The length of the temporal window used for these experiments is one day. The optimization stage uses data of the previous day and resulting steps j are applied to data of the current day. We experimented thresholds equal to the total number 6720 divided by each of $\{1, 2, 3, 5, 7, 10, 15\}$.

Table 2. Percentage of days whose SSE with optimization is better than "fixed" methods. opt-1: Optimization at day $d-1$ and sampling at d. %d $< 10\%$ SSE corresponds to the percentage of days whose SSE opt-1 is lower by 10 % than the SSE obtained by "fixed" methods.

Threshold s	6720	3360	2240	1344	960	672	448
%d$< 10\%$ SSE LI opt-1	-	84%	84%	84%	85%	85%	85%
%d $< 10\%$ SSE CI opt-1	-	84%	84%	85%	86%	97%	100%
%d $< 10\%$ SSE seg opt-1	-	83%	83%	84%	84%	84%	84%
%d $< 10\%$ SSE wav opt-1	-	84%	85%	94%	89%	100%	100%

Results in Table 2 show that the approach is very accurate. In fact, the optimized summarizing methods decrease SSE by at least 10% compared to the fixed methods on more than 83% of the days and for all tested threshold values. Note that the more threshold decreases the more performances of optimized summarizing methods increase.

Optimization Using d-7. Previous experiments were done on data of day $d-1$ to summarize data of day d. Fig. 1 shows the evolution of SSEs during one month (30 days) for threshold $s = 1344$. We observe that errors are not distributed regularly with time. This is due to the weekly cycle of electric power consumption. In fact the consumption remains generally stable during 5 working days and the consumption level changes on

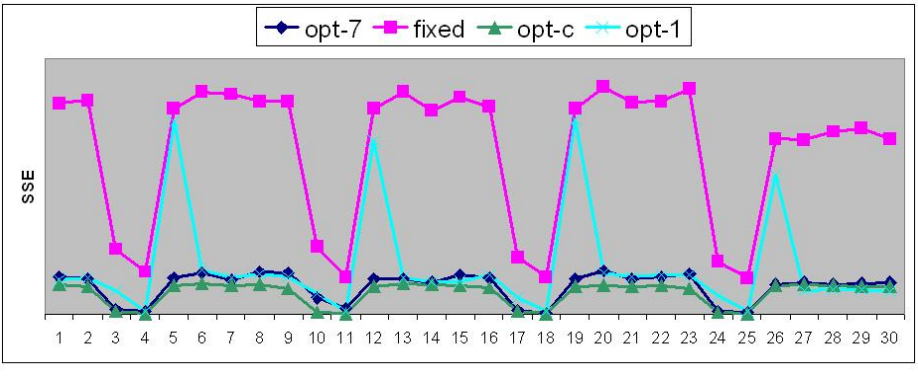

Fig. 1. Evolution of SSEs during 30 days. opt-7: Optimization at day d-7 and sampling at d. opt-1: Optimization at day d-1 and sampling at d. opt-c: Optimization and sampling at day d (current day). fixed: Fixed segmentation.

weekends. Here sampling rates for mondays are determined with data of sundays and those of saturdays based on data of fridays. Additional experiments were carried out with a sampling scheme for a day d based on sampling rates computed from day $d-7$. Results are illustrated in curve opt-7 of Fig. 1 (curve opt-7). We note that this *smoothes* the SSE evolution and thus decreases errors due to the periodicity of electric power consumption.

5 Conclusion and Future Work

We have proposed a new generic framework for summarizing distributed data streams to be loaded into a data warehouse. The proposed approach limits the load of the central server hosting the data warehouse and optimizes the sampling rates assigned to each input stream by minimizing the sum of square errors. This optimization is dynamic and adapts continuously with the contents of the streams. Moreover, the compression technique used for each stream adapts continuously to its contents and is selected among several approaches.

Experiments have been done on real data describing measures of electric power consumption generated by customer meters. These experiments show the efficiency of the approach. Other experiments are currently being done on larger data sets describing more meters on a longer period.

This work suggests several developments which are under study:

- Testing the approach on more chaotic data with other summarizing techniques.
- Study how error measures different from SSE can be used, for instance the L_1 distance or L_∞ to minimize the maximum of errors instead of the mean. This impacts on the optimization algorithm.
- Extending the approach to sensors producing multidimensional numerical data.

References

1. Babcock, B., Datar, M., Motwani, R.: Load shedding for aggregation queries over data streams. In: 20th International Conference on Data Engineering (2004)
2. Babcock, B., Datar, M., Motwani, R.: Sampling From a Moving Window Over Streaming Data. In: 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2002), pp. 633–634 (2002)
3. Bellman, R.: On the approximation of curves by line segments using dynamic programming. Communications of the ACM 4(6), 284 (1961)
4. Carney, D., Cetinternel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N., Zdonik, S.: Monitoring streams - A New Class of Data Management Applications. In: Proc. Int. Conf. on Very Large Data Bases, pp. 215–226 (2002)
5. Chu, D., Deshpande, A., Hellerstein, J.M., Hong, W.: Approximate Data Collection in Sensor Networks using Probabilistic Models. In: Proceedings of the 22nd international Conference on Data Engineering. ICDE 2006 (2006)
6. Cormode, G., Garofalakis, M.: Approximate Continuous Querying over Distributed Streams. ACM Transactions on Database Systems 33(2) (June 2008)
7. Demers, A., Gehrke, J., Panda, B., Riedewald, M., Sharma, V., White, W.M.: Cayuga: A general purpose event monitoring system. In: CIDR, pp. 412–422 (2007)

8. Dobra, A., Garofalakis, M., Gehrke, J., Rastogi, R.: Processing complex aggregate queries over data streams. In: Proceedings of the 2002 SIGMOD Conference, pp. 61–72 (2002)
9. Gehrke, J., Korn, F., Srivastava, D.: On computing correlated aggregates over continual data streams. In: Proceedings of the 2001 SIGMOD Conference, pp. 13–24 (2001)
10. Guha, S., Koudas, N., Shim, K.: Data-streams and histograms. In: Proceedings of the 2001 STOC Conference, pp. 471–475 (2001)
11. Jain, A., Chang, E.Y.: Adaptive sampling for sensor networks. In: Proceeedings of the 1st international Workshop on Data Management For Sensor Networks: in Conjunction with VLDB 2004, Toronto, Canada (2004)
12. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An Online Algorithm for Segmenting Time Series. In: Proceedings of IEEE International Conference on Data Mining, pp. 289–296 (2001)
13. Liu, C., Wu, K., Tsao, M.: Energy Efficient Information Collection with the ARIMA model in Wireless Sensor Networks. In: Proceedings of IEEE GlobeCom 2005, St. Louis, Missouri (November 2005)
14. Marbini, A.D., Sacks, L.E.: Adaptive sampling mechanisms in sensor networks. In: London Communications Symposium, London, UK (2003)
15. Palpanas, T., Vlachos, M., Keogh, E., Gunopulos, D.: Streaming Time Series Summarization Using User-Defined Amnesic Functions. IEEE Transactions on Knowledge and Data Engineering (2008)
16. Sakurai, Y., Papadimitriou, S., Faloutsos, C.: Automatic Discovery of Lag Correlations in Stream Data. In: ICDE 2005, Tokyo, Japan (2005)
17. Stanford Stream Data Management (STREAM) Project, `http://www-db.stanford.edu/stream`
18. StreamBase Systems, Inc., `http://www.streambase.com/`
19. Tatbul, N., Cetintemel, U., Zdonik, S., Cherniack, M., Stonebraker, M.: Load shedding in a data stream manager. In: Proc. of the 29th Intl. Conf. on Very Large Databases (VLDB 2003) (2003)
20. Willett, R., Martin, A., Nowak, R.: Backcasting: adaptive sampling for sensor networks. In: Proceedings of the Third international Symposium on information Processing in Sensor Networks, Berkeley, California (2004)
21. Zdonik, S., Stonebraker, M., Cherniack, M., Cetintemel, U., Balazinska, M., Balakrishnan, H.: The Aurora and Medusa Projects. In: Bulletin of the Technical Committee on Data Engineering, March 2003, pp. 3–10. IEEE Computer Society, Los Alamitos (2003)