♦ Monitoring High Performance Data Streams in Vertical Markets: Theory and Applications in Public Safety and Healthcare

Rafal Maison, Daryl J. Steen, Maciej Zakrzewicz, and Zenon Biniek

Over the last several years, monitoring high performance data stream sources has become very important in various vertical markets. For example, in the public safety sector, monitoring and automatically identifying individuals suspected of terrorist or criminal activity without physically interacting with them has become a crucial security function. In the healthcare industry, noninvasive mechanical home ventilation monitoring has allowed patients with chronic respiratory failure to be moved from the hospital to a home setting without jeopardizing quality of life. In order to improve the efficiency of large data stream processing in such applications, we contend that data stream management systems (DSMS) should be introduced into the monitoring infrastructure. We also argue that monitoring tasks should be performed by executing data stream queries defined in Continuous Query Language (CQL), which we have extended with: 1) new operators that allow creation of a sophisticated event-based alerting system through the definition of threshold schemes and threshold activity scheduling, and 2) multimedia support, which allows manipulation of continuous multimedia data streams using a similarity-based join operator which permits correlation of data arriving in multimedia streams with static content stored in a conventional multimedia database. We developed a prototype in order to assess these proposed concepts and verified the effectiveness of our framework in a lab environment. © 2011 Alcatel-Lucent.

Introduction

Increasingly, the ability to process information generated by high performance data stream sources has become very important in various vertical markets including public safety and healthcare. Consider, for example, the use of noninvasive mechanical home ventilation (MHV) [14, 28] in the care of patients with chronic respiratory failure. This kind of treatment raises ventilatory effectiveness, improves physical

Bell Labs Technical Journal 16(3), 163–180 (2011) © 2011 Alcatel-Lucent. Published by Wiley Periodicals, Inc.
Published online in Wiley Online Library (wileyonlinelibrary.com) • DOI: 10.1002/bltj.20528

Panel 1. Abbreviations, Acronyms, and Terms

C2G—Second generation
avg—Average
CDL—Constant data length
CPU—Central processing unit
CQL—Continuous Query Language
DARPA—Defense Advanced Research Projects Agency
DB—Database
DNA—Deoxyribonucleic acid
DSMS—Data stream management systems
FBG—Face bunch graph
FERET—Facial Recognition Technology
GSM—Global System for Mobile Communications
HJ—Hash join
HMM—Hidden Markov model
I/O—Input/output
LDA—Linear discriminate analysis
max—Maximum
MHV—Mechanical home ventilation
min—Minute
MMDSMS—Multimedia data stream management system
MPEG—Moving Picture Experts Group
NFA—Nondeterministic finite automaton
NLJ—Nested loop join
OOD—Object oriented design
PCA—Principal component analysis
PDBNN—Probabilistic decision-based neural networks
RAM—Random access memory
RAN—Radio access network
SASE+—Stream-based and Shared Event
SQL—Structured Query Language
TS—Tableset

comfort, and allows, in many cases, the care of chronic patients to be moved from a hospital to a home environment. Patients treated with MHV at home communicate increased feelings of security and an improved quality of life [18, 21].

When a person is being treated for chronic respiratory distress through MHV, the medical equipment can raise various alarms, including low and high-pressure alarms, due to changes in the ventilation process. For example, low-pressure alarms can be caused by patient disconnection from the ventilator; circuit, airway, and chest tube leaks; patient coughing; or accumulation of water in the circuit. These alarms can be categorized into discrete events, which must be continuously monitored to detect life-threatening exceptions in breathing. Immediate detection of these changes in ventilation, automatic prioritization of the resulting events, and sending immediate notifications to distant medical personnel are the most important monitoring challenges. Existing solutions do not address these problems sufficiently.

We argue that in order to improve the quality of the event processing described above, a data stream management system (DSMS) [1, 4, 8] should become part of the monitoring infrastructure and should perform analysis tasks through the execution of data stream queries. We propose the following data model for describing tuples arriving in the feed from an MHV ventilator:

- *ID*. Unique stream identifier.
- *PRESSURE*. The airflow volume per unit surface area (expressed in mbar).
- *FLOWRATE*. The rate at which air is distributed to the patient (expressed in l/min).
- *TIMESTAMP*. The time when the tuple arrives.

Given these elements, the schema of a ventilator data stream can be formally defined in Continuous Query Language (CQL) [5] as follows:

CREATE STREAM ventilation(string ID, float PRESSURE, float FLOWRATE, timestamp TIMESTAMP);

To track patients treated with MHV and identify occurrences of low-pressure alarms (pressure $<$ 10 mbar) we define the following CQL query using our new constructs:

EVENT ventilation_major_alarm ON ventilation

WHEN

SCHEDULE (0:00, 24:00) THEN THRESHOLD PRESSURE < 10; (1)

Another application of data streams processing lies in the public safety sector. Since September 11, 2001, efforts to identify potential terrorist or criminal activities have greatly intensified. This has sparked an increased interest in systems able to detect suspects in public places such as airports, railway stations, stadiums, and subways. Automated recognition of these individuals' faces is the most natural way of determining identity. In fact, visual recognition is more attractive than using biometric sensors and detectors, because surveillance can be performed without the target being aware of the intrusion. Biometric evidence such as a retinal pattern, fingerprint, or DNA sample requires physical interaction with the monitored subject for collection purposes. Facial recognition involves the following steps: detection, tracking, and classification [3, 22, 26]. In addition to uncovering individuals who may pose security threats, effective implementation of these steps will reduce the number of false positives for people who are not real risks. We observe that a very important feature of these data applications is the processing of information from transient data streams, such as Moving Picture Experts Group 7 (MPEG 7) formatted video feeds, rather than from sets of scalar data stored in databases. We argue that by executing monitoring queries over multimedia data streams captured from devices located in selected public places, security agencies can continuously search for individuals suspected of nefarious activities in more flexible and efficient way.

We propose the following data model for describing video stream tuples arriving from multimedia devices:

- *ID*. Unique stream identifier.
- *NAME*. A label for the processed stream.
- *CAM*. The video frame captured from the high-resolution camera.
- *TYPE*. The multimedia data type.
- *TIMESTAMP*. The time when the video frame arrives.

Facial reference data is stored in the form of multimedia pictures in conventional multimedia databases. A typical tuple consists of multiple fields including:

- *ID*. Unique image identifier.
- *NAME*. A label for the image.
- *FACEIMG*. A face picture.
- *TYPE*. The multimedia data type.

Given these elements, the schema of a video stream can be formally defined in CQL as:

CREATE STREAM camvideo (string ID, string NAME, image CAM,

string TYPE, timestamp TIMESTAMP); (2)

and the schema of the image reference relation as:

CREATE TABLE facialdb (string ID, string NAME, image FACEIMG, string TYPE); (3)

For example, to track particular individuals appearing in a video stream and identify those who potentially pose a threat based on reference data stored in multimedia data types, the following CQL query can be defined using our new similarity-based operator *SIMILAR WITH*:

SELECT mb.name FROM camvideo ms, facialdb mb WHERE mb.faceimg SIMILAR ms.cam WITH 0.7; (4)

where the *SIMILAR WITH* operator is defined as:

image1 SIMILAR image2 WITH probability, where

image1. First image used for similarity comparison.

image2. Second image used for similarity comparison.

probability. Denotes the probability of similarity between the two compared images.

In this paper, we propose extension of the DSMS architecture with: 1) new operators that allow creation of a sophisticated event-based alerting system through the definition of threshold schemes and threshold activity scheduling, and 2) multimedia support, which allows manipulation of continuous multimedia data streams using similarity-based join operators which permit correlation of data arriving in multimedia streams with static content stored in a conventional multimedia database. We present

applications of the proposals in vertical markets to demonstrate potential of significant practical impact.

Related Work

The concept of a data stream management system was first introduced in [1, 4, 8], and a number of DSMSs have since been developed. The DSMS architecture has been able to support filtering and correlation of tuples arriving in continuous data streams, which is important for the healthcare and public safety applications mentioned above. Within DSMSs, the following types of architectures have been proposed:

- *Active databases* [15, 19, 32], which are capable of tracking simple events and do not support sliding windows or correlation between events over high performance data streams.
- *Publisher/subscriber model databases* [2, 7, 12], which provide the capability of filtering individual events using simple predicates.
- *Sequence databases* [23, 25], which allow multi-event pattern matching expressed in Structured Query Language (SQL)-like languages.

In general, many of the DSMS implementations offer comprehensive event analysis, but do not support Kleene closure [13]. However, this capability is provided as a SQL extension in SQL-tableset (TS) [23]. Cayuga [10, 11] is a modern publish/subscribe pattern-matching system designed to model finite state automata (nondeterministic finite automaton (NFA)). It offers a sophisticated language based on event algebra to deal with parameterization and aggregations. It is capable of handling simultaneous events and supports efficient multi-query optimization. SEQUIN [25] introduces the concept of sequence databases to support sequential data. It provides a declarative language based on operator algebra, which supports nested expressions to handle data sequences. However, SEQUIN's capabilities for manipulating data are rather simple in contrast to those of pub/sub architectures. SQL-TS [23] introduces a query language for dealing with sophisticated time series pattern matching. The language employs analyses of logical interdependencies between pattern elements to avoid multiple passes over the input data. This system supports Kleene closure; however, pattern matching is limited to adjacent tuples. SASE+ [13] extends the prior SASE framework with Kleene closure support based on the NFA model. It provides complex capabilities for event matching over streams and a language for defining patterns.

The techniques described above have focused on pattern matching over streams and have been used to address various business problems existing in numerous vertical markets. To our knowledge, none of these existing approaches provides an alerting mechanism within their declarative query languages through the definition of a threshold scheme and threshold activity schedules over sliding windows. We present such an approach in this paper.

The organization of the rest of the paper is as follows. First, we describe the essential components of our event system such as query plans, query plan execution, and novel operators. Next, we provide analysis of our multimedia support within DSMS and our new similarity-based CQL extension operator. Finally, we present applications of this technology in the healthcare and public safety markets and discuss the results of our experiments.

Event Based Alerting in CQL

We have designed our framework to run on top of STREAM [4, 20] and have added new functionalities—*event*, *threshold* and *schedule*—to its architecture to support advanced threshold and scheduling capabilities. In addition, we have extended the declarative CQL and added new syntax to specify criteria for data filtering and event construction. STREAM [4] follows an object-oriented design (OOD), and the changes introduced during the prototype development phase have not led to performance degradation. The extensions above have been implemented as separate coding units, which have been designed in an OOD fashion to avoid unpredictable side effects in the existing components. The functional description of STREAM legacy components is presented in [4] and is not duplicated here. Our event-based query processing additions consist of the following:

- *Event*. A relation-to-relation operator responsible for event filtering based on predicates.
- *Threshold*. A condition operator for specifying event generation criteria based on a particular attribute or aggregation function value.

- *Schedule*. An operator for specifying periods during which defined thresholds are active.

We propose the following syntax of an event definition:

CREATE EVENT event_name ON stream1[, stream2]

[WHERE stream1.attr1 = stream2.attr2]

WHEN SCHEDULE time_interval THEN
THRESHOLD agg(attr) < thresh_value

[SCHEDULE time_interval THEN
THRESHOLD agg(attr) < thresh_value]

. . .

ELSE THRESHOLD agg(attr) < thresh_value,
where:

event_name = The name of an event to create.

stream1,2 = The name of a existing stream on which the event is created.

attr1 . . . n = The name of attribute for which the threshold is created.

time_interval = The value specifies period during which defined threshold is active.

agg = The name of the aggregation function which computes a single result value from a set of input values.

thresh_value = The value over which the event is generated.

A sample *CREATE EVENT* command is given below:

CREATE EVENT ventilation_major_alarm ON ventilation

WHEN

SCHEDULE (0:00, 8:00) THEN THRESHOLD
max(PRESSURE) > 40

SCHEDULE (9:00, 24:00) THEN THRESHOLD
max(PRESSURE) > 50

ELSE max(PRESSURE) > 45;

The statement above generates a stream of events which can be analyzed by another application. The events in the result stream can be prioritized based on threshold criteria and used to produce minor, normal, major, or critical severity alerts.

Event Query Plan Execution

When an event statement is issued by an end user and registered in our DSMS, the query processing engine generates a query plan [5] which describes the manner in which the query will be evaluated. This query plan can be represented as a tree of operators or other objects such a queue or a synopsis [5] to describe the order in which information will be processed when the continuous query is evaluated over an arriving data stream. An example plan for an event query is illustrated in **Figure 1**. As shown in this drawing, operators performing data processing activities are connected to each other via queues, which either hold tuples from a data stream or intermediate results from processing. The state of an individual operator can be stored in a synopsis and shared between other objects in the query processing plan for better resource management.

In this example, seven operators of six types—join, aggregate, sliding-window, scheduling, event, and select/project—are placed into the query plan. The plan includes stream carrying queues q_1 and q_2 and other relation-carrying queues q_3–q_8. The stream-to-relation operators O_1 and O_2 consume arriving tuples from q_1 and q_2, respectively, and update window synopses s_1 and s_2 correspondingly. The join operator O_3 consumes tuples from the relation-carrying queues q_3 and q_4, which represent sliding windows over streams S_1 and S_2. Operator O_3 maintains two synopses s_3 and s_4 which store materialized matched tuples from q_3 and q_4. These synopses are later used by O_3 to insert joined tuples into relation queue q_5. The aggregate operator O_4 calculates aggregated values (e.g., average (avg) or maximum (max)) of attribute(s) originating from the stream specified in the registered query statement. It is necessary to maintain materialized results of join matching from q_5 in O_4's synopsis store s_5 for later processing. When the sliding window moves, O_4 performs aggregation over the materialized values in s_5 and inserts the results in relation queue q_6. The scheduling operator O_5 keeps track of the query engine local time and decides which

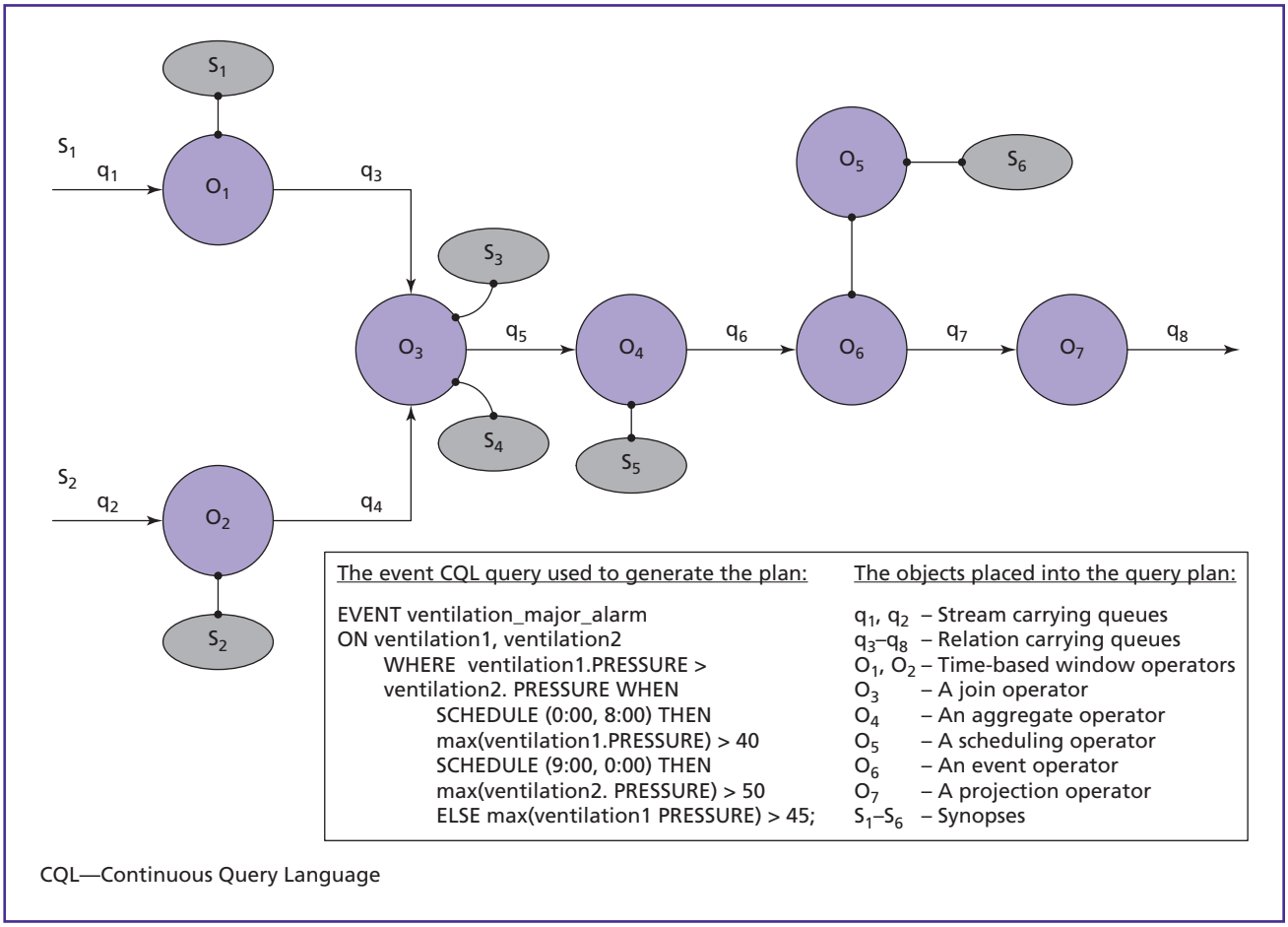

Figure 1.
The query plan generated for an event CQL statement.

threshold should be active based on constraints specified in the *schedule* clause of the query. It materializes the running list of active thresholds in synopsis s_6. Thus, over time, the query engine may change which data is written to outcome queue q_7 based on the *threshold* condition associated with the current time. Event operator O_6 is the relation-to-relation condition operator which evaluates data from q_6 against appropriate thresholds from s_6 and inserts the results into relation queue q_7. O_7 dequeues tuples from q_7 and projects attributes from these tuples as outcomes of the issued query in q_8. In contrast to standard stateless select operators, the event operator is stateful, because it maintains the state of the evaluated thresholds based on scheduling operator O_5.

Multimedia Support in CQL

In standard data stream processing, a typical tuple consists of several simple attribute types such as text, number, and date, which can easily be evaluated and compared through traditional means. On the other hand, multimedia data types such as image, audio, and video require more specialized and complex processing. There are several commercial implementations of multimedia support in conventional DBMSs; however, these DBMSs are not adequate for data stream queries, because operations are evaluated on static snapshots of data stored in relations. In order to effectively process dynamic multimedia data streams, continuous queries and multimedia data stream management systems (MMDSMS) have been recently

introduced in a computation-oriented multimedia data stream model [9].

We have extended STREAM to support multimedia data and query processing including use of a novel similarity-based join. Moreover, our solution employs load shedding techniques to increase DSMS processing capability. Load shedding techniques, which drop excess data, can have a significant impact on query processing. A system which does not handle an overload scenario may be unstable and produce uncontrolled delays when evaluating the results. We have implemented random dropping of tuples when overloading occurs and discarding excess tuples regardless of the data content. Hence our approach performs considerably better than one without such a mechanism. An analysis of the load shedding algorithm does not fall within the scope of this paper; however, the efficiency of this approach has been discussed in our previous work [17]. The associated performance improvements have been verified in the lab.

Query Engine

The query engine described in previous sections of this paper has also been enhanced to operate on various transient media data types and to support numerous file formats and audio and video codecs. Aforementioned concepts such as query plan execution and threshold scheduling have been reused; however, several changes have been introduced to deal with multimedia data. STREAM follows an object-oriented design, and the changes introduced during the prototype development phase have not led to performance degradation. The most important module we implemented in the DSMS is the adaptation layer responsible for adjusting the multimedia data types for query engine processing. We added a new similarity-based CQL operator which has been designed to perform multimedia stream joins of complex data types such as image, voice, and video. This new operator requires merging of voluminous data, which may consume unbounded input/output (I/O) resources. Thus, as in our standard DSMS, we employed window semantics when performing joins to limit the amount of data being matched. In our DSMS, the query engine makes decisions about algorithms used for joining two multimedia streams based on cost estimation. These estimation methods are described below.

Similarity-Based Join

Although this section defines algorithms for joining multimedia video/image streams, the same mechanisms can be applied to audio feeds. In contrast to traditional DSMSs where simple operators can be used to compare data, our DSMS uses video search and pattern matching algorithms to find matches. **Figure 2** illustrates how two video streams are jointed together.

A full join operation between two video streams, A and B, can be thought of as two separate tasks: a single join operation of A frames to B frames and a single join operation of B frames to A frames. In the first task, the join operator consumes each A stream frame $\{a_1, a_2, \ldots, a_r\}$ as it comes into the system and searches the B stream sliding window tuples for matching images. Let us express this cost as C_{scanB}. At the same time this search is being performed, B stream frames are also entering the system and the B stream sliding window needs to be updated to include the new frames and to remove any tuples that have fallen outside the window. Let us express the cost of modifying the window for each incoming B frame as $C_{win\_updateB}$. This means that for all A stream frames r_A and B stream frames r_B arriving during a given period of time, the A to B single join cost C_{jA} during this time can be written as:

$$C_{jA} = r_A \times C_{scanB} + r_B \times C_{win\_updateB}$$

Likewise, the B to A single join cost for the same period of time can be expressed as:

$$C_{jB} = r_B \times C_{scanA} + r_A \times C_{win\_updateA}$$

Putting it all together, the cost for full join over this time period is:

$$C_j = (r_A \times C_{scanB} + r_B \times C_{win\_updateB}) + (r_B \times C_{scanA} + r_A \times C_{win\_updateA})$$

Within our DSMS, we have implemented two join techniques that can be used when combining multimedia data streams: nested loop join (NLJ) and hash join (HJ). The NLJ approach is very simplistic and, for single A-to-B joins, essentially involves comparing

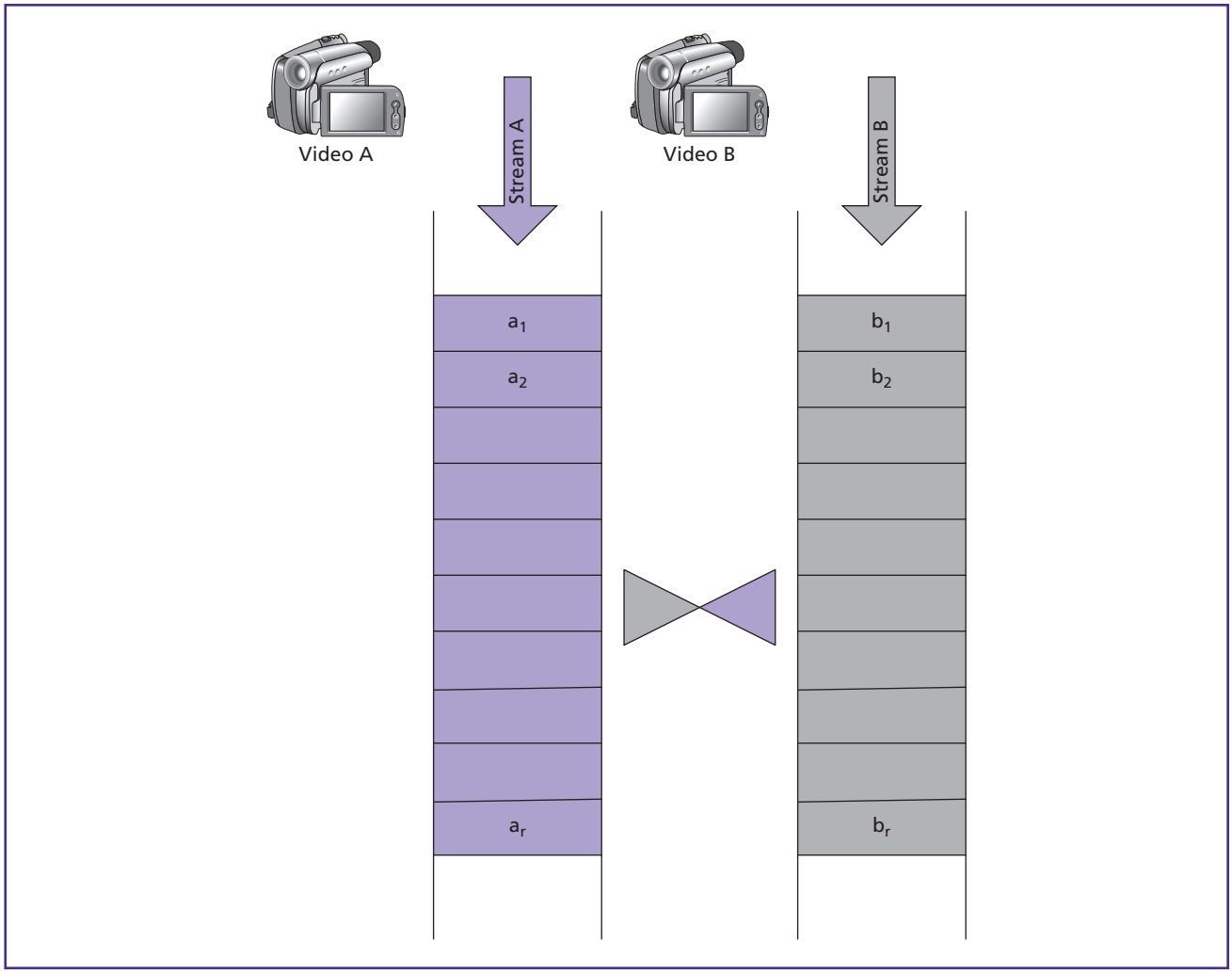

Figure 2.
Joining of two video streams.

each incoming A stream frame against each frame in stream B's sliding window. This C_{scanB} cost can be expressed as the number of frames in the sliding window (henceforth denoted as N_B) multiplied by cost of comparing two frames (henceforth called C_f), or:

$$C_{scanB} = N_B \times C_f$$

The cost of comparing two frames C_f can be expressed as:

$$C_f = C_a + C_m$$

where C_a is the cost of a sliding window access and C_m is the cost of search and pattern matching algorithms used to find matches between two frames. Also, since there is nothing special involved in updating B's sliding window other than adding frames to and removing frames from a queue, the $C_{win\_updateB}$ cost can be represented as two times C_a. Given this information, the cost of a single NLJ join from stream A to stream B is as follows:

$$C_{jA} = r_A \times N_B \times C_f + r_B \times 2 \times C_a$$

If reduced scanning time is desired, hash joins of two multimedia streams can be employed. In this approach, the sliding window is implemented as a dynamic hash table. For single A-to-B joins, each incoming B frame is inserted into a specific hash

bucket in the B window based on its calculated hash value. Accordingly, each arriving A frame is only compared against B window tuples from the corresponding hash bucket. The associated average C_{scanB} cost works out to be:

$$C_{scanB} = \frac{N_B}{|N_B|} \times C_{fh}$$

where $|N_B|$ denotes the number of hash buckets used by B's window and C_{fh} expresses the cost of comparing two frames in the hash join. In turn, C_{fh} can be written as:

$$C_{fh} = C_h + C_m$$

where C_h represents the cost of a hash bucket operation.

While the hash join scanning cost is smaller than that of the NLJ, the HJ window update cost is actually more expensive. The window insertion side is relatively simple—just involving the calculation of a B frame hash value and an insertion into the appropriate bucket—so it can be written as C_h. The window deletion side is more complex and requires removing stale entries from each hash bucket at a cost of $|N_B| \times C_h$. Taking this into account, the value of $C_{win\_updateB}$ is:

$$C_{win\_updateB} = (|N_B| + 1) \times C_h$$

Given this information, the cost of a single HJ join from stream A to stream B is as follows:

$$C_{jA} = \left(r_A \times \frac{N_B}{|N_B|} \times C_{fh} + r_B \times (|N_B| + 1) \right) \times C_h$$

Finally, the cost of a full join between two multimedia data streams can be calculated as either the sum of two single NLJ join costs, the sum of two single HJ join costs, or the sum of a single NLJ join cost and a single HJ join cost.

SIMILAR WITH Operator Application

Another challenging problem arising in multimedia queries is a lack of explicit comparison operators that can be applied to image, voice, and video streams. For example, going back to our aforementioned terrorist/criminal identification application, we first need to be able to glean facial information from the video frames and render it into some sort of normalized format before comparing it against another stream of faces. This normalization takes place before joining occurs, so its cost is not included in our calculations above. This automated face recognition process consists of the following phases:

- Video tuple frame segmentation, which leads to the separation of "face-like" regions within a captured image.
- Facial feature extraction from these tuple regions.
- Face identification based on extracted features.

Facial feature extraction is a challenging problem, because a "face" can change depending on angle of view, scene lighting, and a person's mood (e.g., gloomy, cheerful). From a practical point of view there are numerous additional peripheral factors to be considered, such as glasses, hairstyle, make-up, or head coverings. Face detection is problematic as well, because systems in real-life situations are not able to capture pictures on pure or solid-color backgrounds for ease of face-like region recognition. In order to generalize problems related to face detection/recognition, numerous methods have been proposed such as principal component analysis (PCA) with eigenfaces [26], linear discriminate analysis (LDA) with Fisherface [31], face bunch graph (FBG) [30], hidden Markov model (HMM) [24], and probabilistic decision-based neural networks (PDBNN) [16]. We have chosen the PCA with eigenfaces algorithm to implement a video tuple matching similarity-based operator for multimedia hash and nested loop joins. We also use the Viola-Jones method based on Haar cascade face detection and appropriate face cascade classifiers for facial feature determination [29].

Using reference information from publicly available face databases, we created a PCA eigenfaces training data set to store dissimilarities between the pictures. First, the average image of all faces is obtained and eigenfaces are then produced to capture each face's differences from the mean. The first eigenface shows the most dominant facial dissimilarities. During video stream processing, each input video tuple is converted into its eigenvalues representation. Afterward, each converted input image is compared against the training set of eigenfaces to search for a

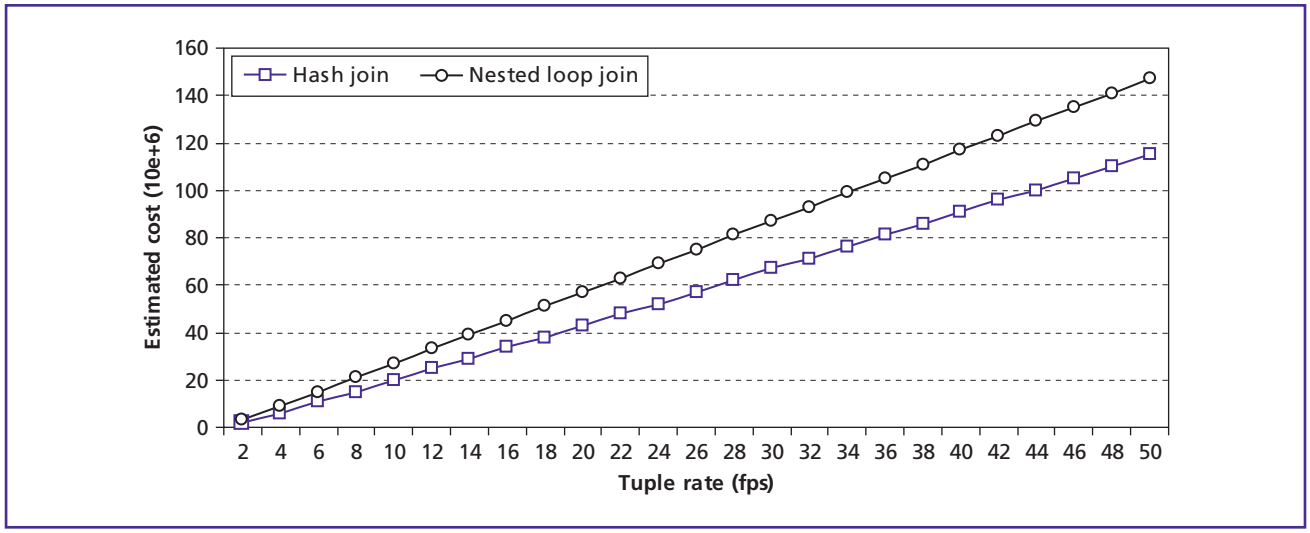

Figure 3.
The cost of joining a video stream in relation to the number of face images.

match. It is important to note here that the multimedia join does not explicitly depend on the similarity-based operator implementation; thus the system can handle multimedia streams in a general manner. The single join cost C_{jA} of merging video streams using the similarity-based operator described above is illustrated in **Figure 3**. From **Figure 4**, it may be concluded that the greater the frame rate in the arriving data streams, the higher the cost of joining the data due to the increased resources required to store the data structures in fixed-window memory.

Prototype Applications in Vertical Markets

The proposed similarity-based operator, threshold calculation operators, and scheduling operators have been verified against the aforementioned real-world problems in healthcare and public safety markets. Our DSMS has been installed on hosts

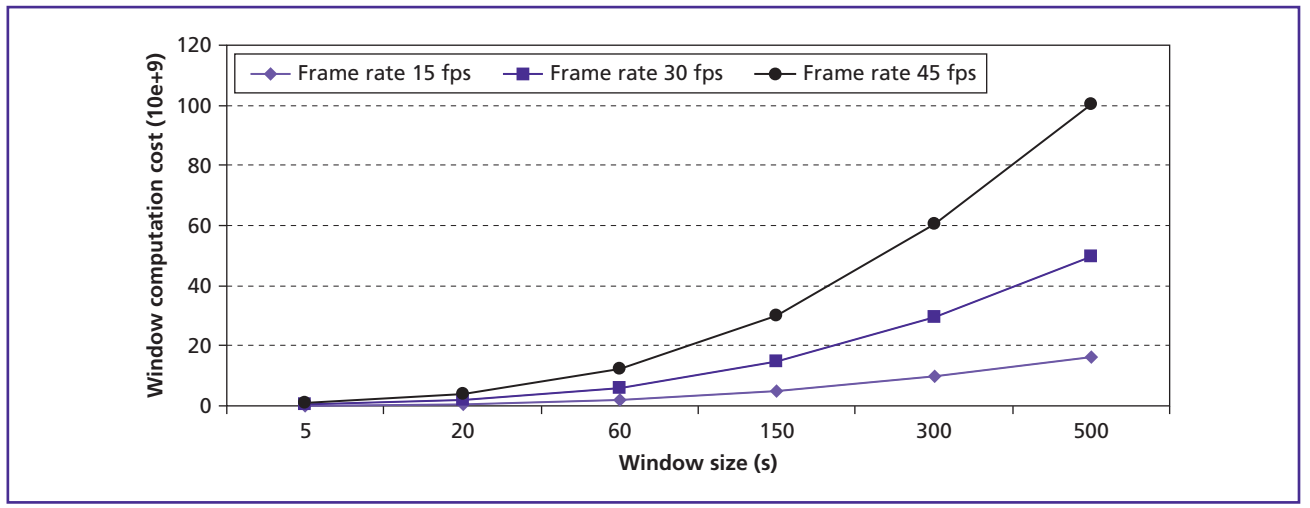

Figure 4.
The cost of joining a video stream in relation to the number of face images within the window.

configured with two 2.2 GHz Intel Core*2 Duo central processing units (CPUs) and 4.0 GB of random access memory (RAM). Our tests have been conducted against messages from a Puritan Bennett Legendair* portable ventilator and video from a high-density camera commonly used in the public safety sector.

We deployed a Legendair ventilator and Global System for Mobile Communications (GSM) modem in our MHV lab to transmit pressure and flow data over a 2G GSM radio access network (RAN) to our DSMS. This network could scale very easily by simply adding new ventilators (and modems, should the need arise). The wireless transmission medium supports maximum flexibility for the geographical location of patients. This would allow respiratory failure patients who live far away from healthcare facilities to be monitored at home; thereby making such services available to a larger segment of the public while reducing the number of medical practitioners needed. The architecture of a potential real-world network is illustrated in **Figure 5**. Typical pressure and flow rate measurements are depicted in **Figure 6**.

We argue that, when monitoring queries registered with the DSMS are continuously processed, immediate detection of adverse changes in a patient's condition could be made when query thresholds are

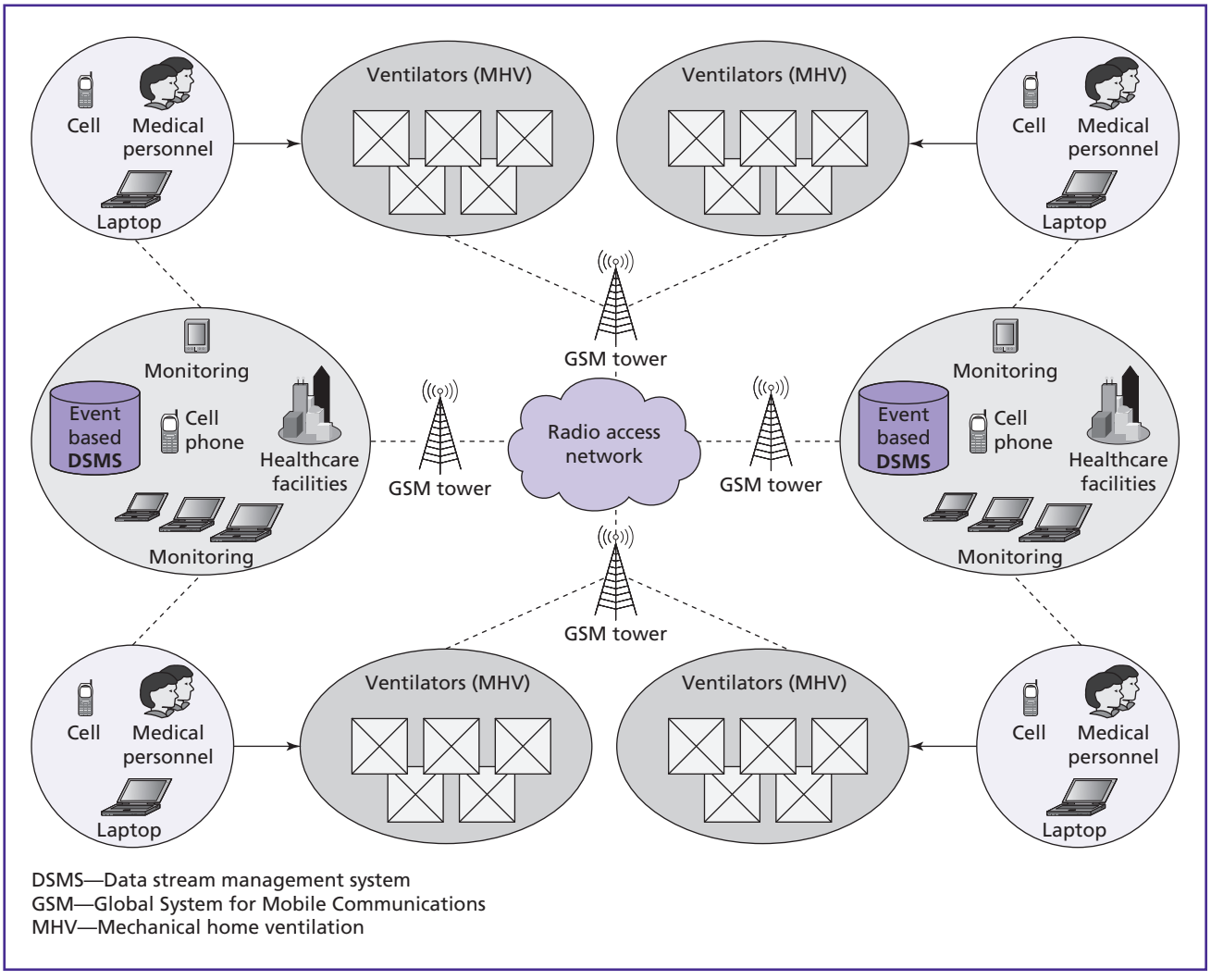

Figure 5.
Remote health monitoring application of our event based DSMS.

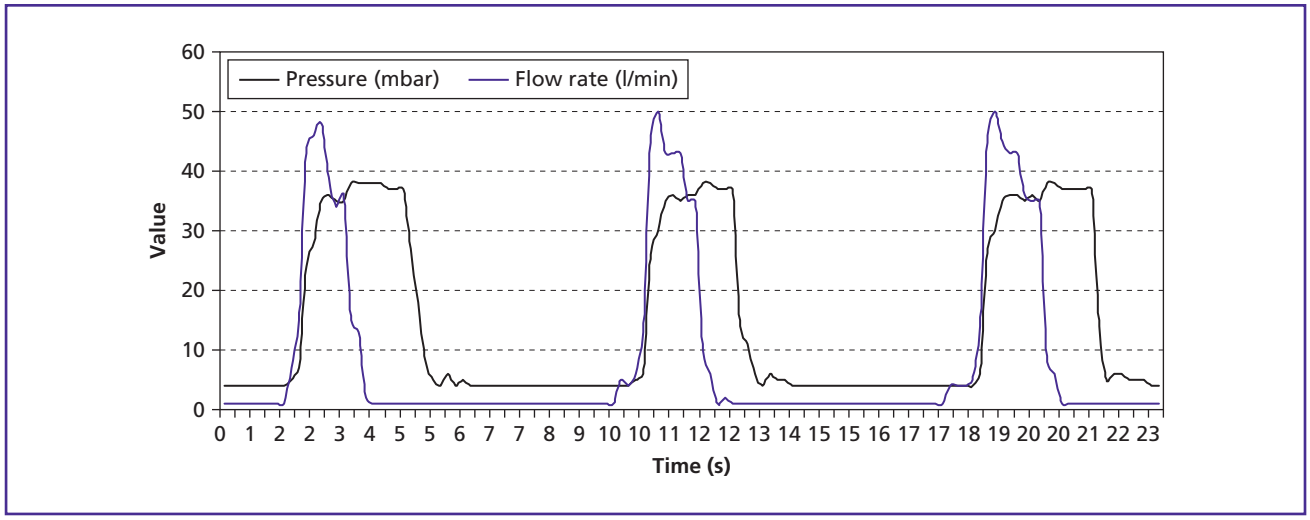

Figure 6.
Pressure and flow rate measurements from a ventilator over time.

violated. Because these thresholds are time-driven, they could also be coordinated with patient lifestyles. For example, pressure volume thresholds could be set lower during nighttime hours so that false alarms would not be raised while a patient was asleep. See **Figure 7** for illustration of such a situation. In addition, the availability of threshold scheduling allows alerting to be turned off during planned "down time" when the patient is not using the ventilator, such as during mealtimes.

Through judicious use of this "intelligent" threshold capability, medical personnel could determine when it is appropriate to offer immediate medical help or consultancy without panicking patients unnecessarily. This, in turn, would strengthen patient confidence in the monitoring system, increase patient feelings of security, and improve family quality of life. The solution could be further enhanced through the use of cameras to monitor patient facial expressions and transfer them to our DSMS. Correlation of these

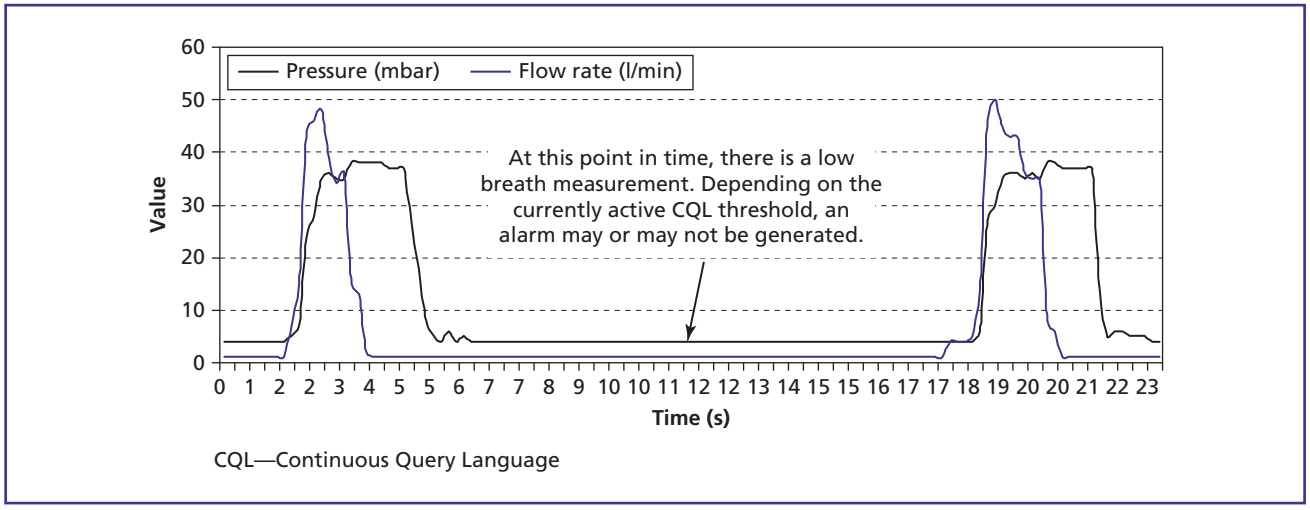

Figure 7.
Missing pressure and flow rate measurements from a ventilator.

additional video streams with pressure and flow rate data could present even better assessments of patient state to doctors; however, analysis of such refinements is not within the scope of this paper.

An architecture in which our DSMS application could serve as the core of a real-time system for detecting terrorist or criminal suspects is shown in **Figure 8**. This architecture is based on placing several cameras in a configuration that limits the number of captured faces per image. For example, locating a camera at an airport screening checkpoint or above a subway escalator would significantly reduce the

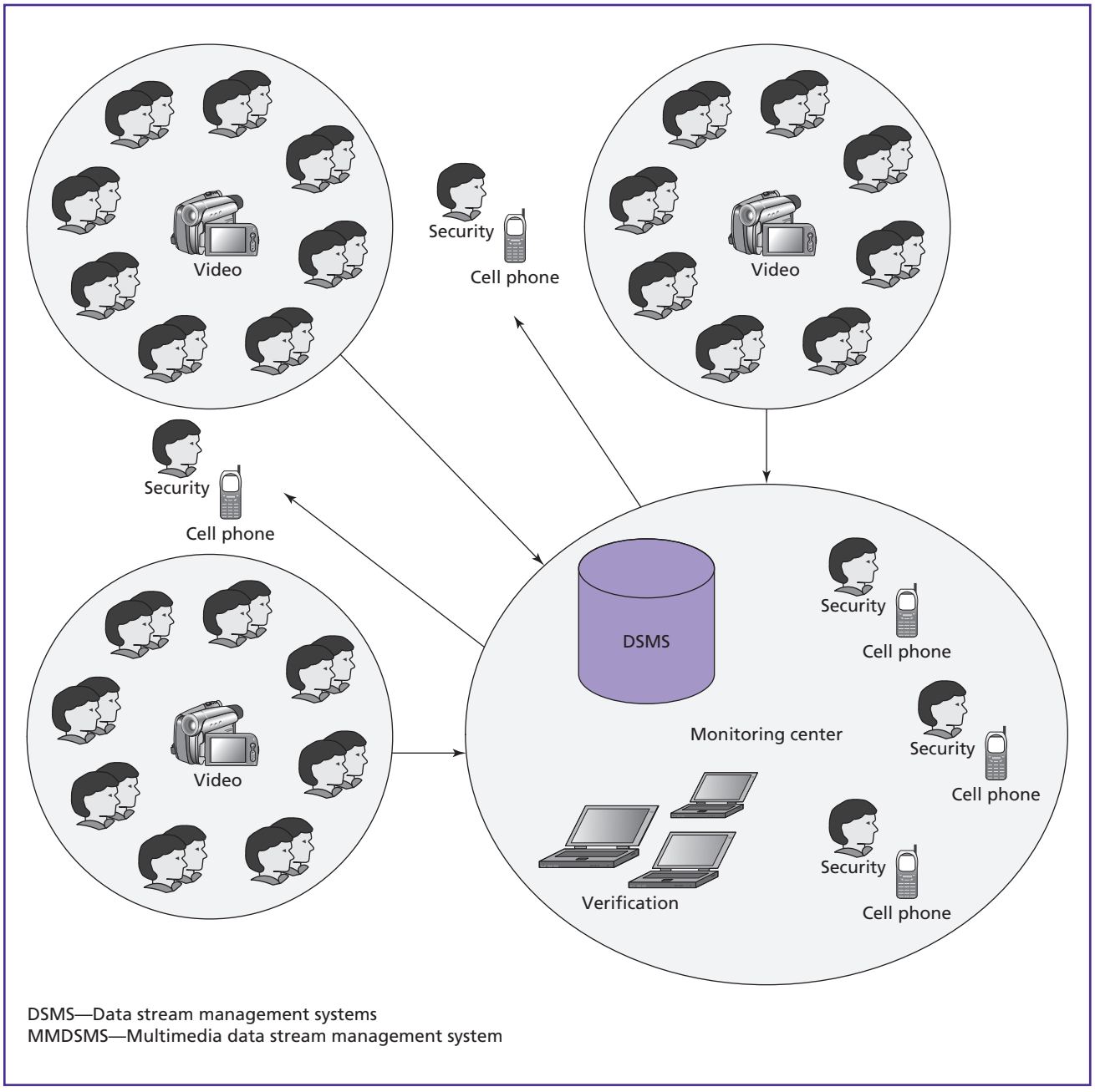

Figure 8.
Public safety application of our MMDSMS.

number of individuals being photographed at any one time. Installing equipment in this way would also minimize the camera motion required to track a particular face.

In order to prepare the test system for evaluation, we have populated our reference face database using samples from the AT&T Laboratories Cambridge Database of Faces [6] and DARPA's Facial Recognition Technology (FERET) database [27]. See equation 3 in the introduction of this paper for a sample CQL statement used to generate this database. Video captures from our camera have been duplicated and used to transmit multiple streams to our DSMS. These streams have been analyzed via queries expressed in CQL (see equation 2 in the introduction for an example). As new face images are processed and inserted into the multimedia database, associated metadata are computed for each of the images. Such an approach permits the system to achieve better performance when processing CQL query join operations, because the search and matching algorithm manipulates the reference image once, rather than performing calculations whenever the image is accessed. For example, when PCA is used to evaluate frame similarity between multimedia data stream images and static pictures from reference data, the metadata for the static images are represented as eigenvalues which are stored by the query engine. The metadata might be different for each search and matching algorithm used to evaluate similarity.

Face recognition accuracy in our DSMS depends on the number of eigenvectors [26] used in detection. An 83 percent accuracy rate has been achieved when 65 eigenvectors are employed. However, certain issues arise when multiple multimedia streams are processed. The time required for face identification increases and the video frame analysis rate is reduced. In our experiments, test runs of two or six hash joins between multimedia data stream frames and static reference pictures were processed by the query engine simultaneously. Each test run was executed two times, once with load shedding, and once without. **Figure 9** shows comparative graphs of these tests. In the non-load-shedding cases, overload started after

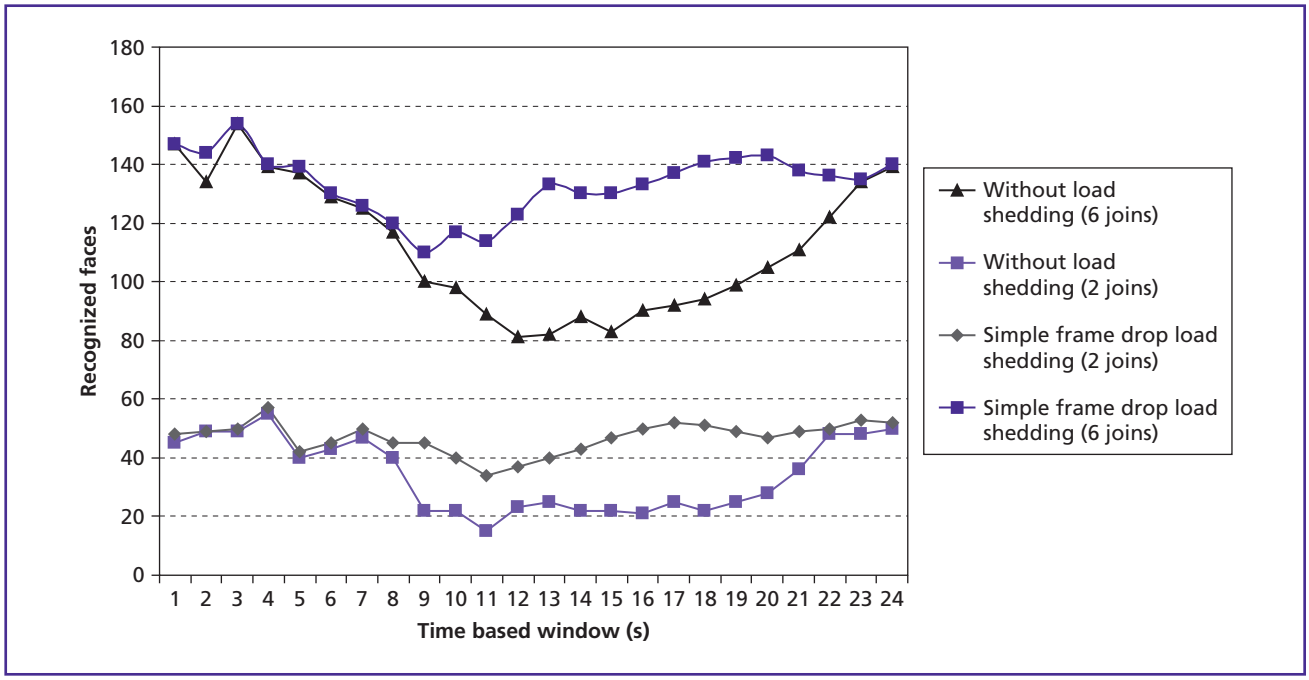

Figure 9.
Average number of recognized faces for the registered joins.

the 8th second and ended around the 23rd second. The average number of recognized faces decreased as resource contention grew, because the query engine performed join operations for each arriving frame even though the arrival rate was higher than processing capabilities would allow.

However, in the load-shedding scenarios, the query engine shedding mechanism continuously monitored all arriving tuples and simply removed those exceeding processing capabilities [17]. Hence, our similarity-based operator performed considerably better when random drop load shedding was employed, and the average number of recognized faces was higher.

We are currently working on the implementation of advanced load shedding techniques [17] to reduce the cost of processing incoming data and avoid system overload while minimizing the effect on recognition accuracy. These algorithms will operate by determining which video frames are most important to the integrity of the feed and avoiding the discard of tuples that affect the overall image quality.

Conclusion

In this paper, we have proposed novel continuous query language event, threshold, and schedule operator extensions to data stream management systems to improve the amount of useful information that could be gleaned from continuous data streams. We have also put forth an event-driven architecture, which would allow the creation of a sophisticated alerting system through the definition of threshold schemes and threshold activity schedules. Furthermore, we have described a similarity-based CQL operator which permits correlation of data arriving in multiple multimedia streams against static data stored in conventional multimedia databases. Finally, we have demonstrated applications of these concepts in the public safety and health-care arenas. We are currently designing an advanced load-shedding mechanism based on prediction methods [17]; however, more extensive study is required to provide a comprehensive analysis.

References

[1] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik, "Aurora: A New Model and Architecture for Data Stream Management," Internat. J. Very Large Data Bases (VLDB J.), 12:2 (2003), 120–139.

[2] M. K. Aguilera, R. E. Strom, D. C. Sturman, M. Astley, and T. D. Chandra, "Matching Events in a Content-Based Subscription System," Proc. 18th ACM Symp. on Principles of Distrib. Comput. (PODC '99) (Atlanta, GA, 1999), pp. 53–61.

[3] N. Apostoloff and A. Zisserman, "Who Are You? – Real-Time Person Identification," Proc. British Machine Vision Conf. (BMVC '07) (Coventry, UK, 2007), pp. 509–518.

[4] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom, "STREAM: The Stanford Data Stream Management System," Stanford InfoLab, 2004, <http://dbpubs.stanford.edu:8090/pub/2004-20>.

[5] A. Arasu, S. Babu, and J. Widom, "The CQL Continuous Query Language: Semantic Foundations and Query Execution," Internat. J. Very Large Data Bases (VLDB J.), 15:2 (2006), 121–142.

[6] AT&T Laboratories Cambridge, "The Database of Faces," Apr. 1994, <http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>.

[7] R. S. Barga, J. Goldstein, M. Ali, and M. Hong, "Consistent Streaming Through Time: A Vision for Event Stream Processing," Proc. 3rd Biennial Conf. on Innovative Data Syst. Res. (CIDR '07) (Asilomar, CA, 2007), pp. 363–373.

[8] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah, "TelegraphCQ: Continuous Dataflow Processing for an Uncertain World," Proc. 1st Biennial Conf. on Innovative Data Syst. Res. (CIDR '03) (Asilomar, CA, 2003).

[9] S.-K. Chang, L. Zhao, S. Guirguis, and R. Kulkarni, "A Computation-Oriented Multimedia Data Streams Model for Content-Based Information Retrieval," Multimedia Tools and Applications, 46:2–3 (2010), 399–423.

[10] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White, "Towards

Expressive Publish/Subscribe Systems," Proc. 10th Internat. Conf. on Extending Database Technol. (EDBT '06) (Munich, Ger., 2006), pp. 627–644.

[11] A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White, "Cayuga: A General Purpose Event Monitoring System," Proc. 3rd Biennial Conf. on Innovative Data Syst. Res. (CIDR '07) (Asilomar, CA, 2007), pp. 412–422.

[12] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, "Filtering Algorithms and Implementation for Very Fast Publish/Subscribe Systems," Proc. ACM SIGMOD Internat. Conf. on Management of Data (SIGMOD '01) (Santa Barbara, CA, 2001), pp. 115–126.

[13] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman, "On Supporting Kleene Closure over Event Streams," Proc. 24th IEEE Internat. Conf. on Data Eng. (ICDE '08) (Cancun, Mex., 2008), pp. 1391–1393.

[14] C. Hörmann, M. Baum, C. Putensen, N. J. Mutz, and H. Benzer, "Biphasic Positive Airway Pressure (BIPAP) – A New Mode of Ventilatory Support," Eur. J. Anaesthesiol., 11:1 (1994), 37–42.

[15] D. F. Lieuwen, N. Gehani, and R. Arlein, "The Ode Active Database: Trigger Semantics and Implementation," Proc. 12th IEEE Internat. Conf. on Data Eng. (ICDE '96) (New Orleans, LA, 1996), pp. 412–420.

[16] S.-H. Lin, S.-Y. Kung, and L.-J. Lin, "Face Recognition/Detection by Probabilistic Decision-Based Neural Network," IEEE Trans. Neural Networks, 8:1 (1997), 114–132.

[17] R. Maison and M. Zakrzewicz, "Prediction-Based Load Shedding for Burst Data Streams," Bell Labs Tech. J., 16:1 (2011), 121–132.

[18] A. Markström, K. Sundell, M. Lysdahl, G. Andersson, U. Schedin, and B. Klang, "Quality-of-Life Evaluation of Patients with Neuromuscular and Skeletal Diseases Treated with Noninvasive and Invasive Home Mechanical Ventilation," Chest, 122:5 (2002), 1695–1700.

[19] R. Meo, G. Psaila, and S. Ceri, "Composite Events in Chimera," Proc. 5th Internat. Conf. on Extending Database Technol. (EDBT '96) (Avignon, Fra., 1996), pp. 56–76.

[20] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query Processing, Approximation, and Resource Management in a Data Stream Management System," Proc. 1st Biennial Conf. on Innovative Data Syst. Research (CIDR '03) (Asilomar, CA, 2003).

[21] K. Pehrsson, J. Olofson, S. Larsson, and M. Sullivan, "Quality of Life of Patients Treated by Home Mechanical Ventilation Due to Restrictive Ventilatory Disorders," Respiratory Med., 88:1 (1994), 21–26.

[22] A. Pentland, B. Moghaddam, and T. Starner, "View-Based and Modular Eigenspaces for Face Recognition," Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR '94) (Seattle, WA, 1994), pp. 84–91.

[23] R. Sadri, C. Zaniolo, A. Zarkesh, and J. Adibi, "Expressing and Optimizing Sequence Queries in Database Systems," ACM Trans. Database Syst., 29:2 (2004), 282–318.

[24] F. Sameria and S. Young, "HMM-Based Architecture for Face Identification," Image and Vision Comput., 12:8 (1994), 537–543.

[25] P. Seshadri, M. Livny, and R. Ramakrishnan, "The Design and Implementation of a Sequence Database System," Proc. 22nd Internat. Conf. on Very Large Data Bases (VLDB '96) (Mumbai, Ind., 1996), pp. 99–110.

[26] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," Proc. IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR '91) (Maui, HI, 1991), pp. 586–591.

[27] United States, Department of Defense, Defense Advanced Research Projects Agency (DARPA), "The Facial Recognition Technology (FERET) Database," 1997, <http://www.itl.nist.gov/iad/humanid/feret/feret_master.html>.

[28] D. Vanpee, D. Clause, L. Delaunois, and S. Nava, "Non-Invasive Mechanical Ventilation," Thorax, 56:8 (2001), 666.

[29] P. Viola and M. Jones, "Robust Real-Time Object Detection," Proc. 2nd Internat. Workshop on Statistical and Computational Theories of Vision (SCTV '01) (Vancouver, BC, Can., 2001).

[30] L. Wiskott, J.-M. Fellous, N. Krüger, and C. von der Malsburg, "Face Recognition by Elastic Bunch Graph Matching," IEEE Trans. Pattern Analysis and Machine Intell., 19:7 (1997), 775–779.

[31] W. Zhao, A. Krishnaswamy, R. Chellappa, D. L. Swets, and J. Weng, "Discriminant Analysis of Principal Components for Face Recognition," Face Recognition: From Theory to Applications (H. Wechsler, P. J. Phillips,

V. Bruce, F. Fogelman Soulié, and T. S. Huang, eds.), Springer-Verlag, Berlin, Heidelberg, New York, 1998, pp. 73–85.

[32] D. Zimmer and R. Unland, "On the Semantics of Complex Events in Active Database Management Systems," Proc. 15th IEEE Internat. Conf. on Data Eng. (ICDE '99) (Sydney, Aus., 1999), pp. 392–399.

(Manuscript approved May 2011)

RAFAL MAISON is a member of technical staff in the

Network Operations Software Department at Alcatel-Lucent in Bydgoszcz, Poland. He holds an M.S. degree in telecommunications from the University of Technology and Life Sciences in Bydgoszcz, Poland and a B.S. in computer science from the Poznań University of Technology in Poland. He is a Ph.D. student in the Institute of Computing Science of the Poznań University of Technology. His work at Alcatel-Lucent has spanned the areas of network traffic patterning and network traffic management. His research interests are focused primarily on database stream management systems, with a special emphasis on load shedding, continuous query processing, and approximate query processing and multimedia support.

DARYL J. STEEN is a distinguished member of technical

staff in Alcatel-Lucent's Services Group in Columbus, Ohio. Over his 25 year career, he has applied network management data modeling and database management techniques to a variety of telecommunications technologies, including Internet Protocol television (IPTV), IP Multimedia Subsystem (IMS), Universal Mobile Telecommunications System (UMTS), optical, Internet Protocol (IP), Asynchronous Transfer Mode (ATM), Digital Subscriber Line (DSL), and Time Division Multiplexing (TDM) networks. He has served in system engineering, database and data warehouse design, architecture, software development, customer technical advocate, and technical sales support roles. His publications include white papers on event correlation and network management architecture, requirements, and design. He holds a B.S. degree in mathematics as well as a B.S. in computer science from Denison University, Granville, Ohio, and an M.S. in computer science from Purdue University, West Lafayette, Indiana.

MACIEJ ZAKRZEWICZ works for the Institute of

Computing Science at Poznań University of Technology in Poland. His research interests include data mining, database/data warehouse systems, and Internet technologies. He teaches courses on information systems design and implementation for companies and universities in Poland, Germany, the United Kingdom, and the United States, and manages and consults on Internet-related projects including web enabling databases and e-commerce. Dr. Zakrzewicz received his Ph.D. in computer science from Poznań University of Technology.

ZENON BINIEK is a professor at the Warsaw University

of Finance and Management in Poland. His research interests include decision support, computer modeling and simulation systems, system analysis and database design. He manages research projects in the area of databases. Dr. Biniek received his Ph.D. in economics (economic cybernetics) from Dresden University of Technology, Germany. ◆