A-mediAS: An Adaptive Event Notification System

Annika Hinze
Institute of Computer Science
Freie Universität Berlin
hinze@inf.fu-berlin.de

ABSTRACT

Event Notification Services inform about the occurrences of events that are of special interest for their users. Upcoming systems for event notification cover multiple applications and integrate event data from different sources. A main challenge is the evaluation of user profiles under various and changing application requirements. Of particular interest are (1) the profile evaluation using differing semantics and (2) high filter performance under changing system load. This paper introduces the design and implementation of A-mediAS – an adaptive event notification system. We present a case study of a remote monitoring application, which shows the effective adaptation of the evaluation semantics in A-mediAS. We describe different event filtering strategies and show their adaptation to varying event and profile distributions. Finally, we analyze the filter performance for different workload scenarios.

Categories and Subject Descriptors

H.3.4 [**Information Storage and Retrieval**]: Systems and Software

Keywords

adaptation, integration, publish/subscribe

1. INTRODUCTION

An Event Notification Service (ENS) informs its users about new events that occurred on providers' sites. An *event* could be, for example, a new temperature value or the occurrence of a traffic jam. Each provider may employ several event sources, e.g., temperature sensors. Users define their interest in events by means of *profiles*. A user profile defines a periodically-evaluated query (similar to a search query).

Recently, new applications have emerged that require the integration of event information from different sources under various or changing application requirements. Examples of such integrating or multi-purpose applications are traveller information systems and facility management for commercial buildings. For example, in commercial building management systems, such diverse objectives as security, surveillance, visitor guidance and maintenance are followed within a single system. These different objectives require different event evaluation strategies, for instance, to enable various profile semantics and to achieve maximal performance under changing load, i.e., changing event and profile distributions.

Consequently, the new applications call for an ENS that flexibly supports (1) various and changing applications and sources, as well as (2) efficient filter algorithms that keep high performance under changing system load. We call these two requirements *qualitative and quantitative adaptation*, respectively. We illustrate the situation and its challenges in a brief example.

EXAMPLE 1. *Consider the following profile $P1$ that may be defined for a facility management system:*

> *P1: Notify the technician when the air conditioning system has a malfunction.*

The challenge of quantitative adaptivity is evident: when the ENS additionally covers a second air conditioning system, the system load rises and the event distributions change.

Events and profiles also undergo application-specific variations: the security service observes the building at night, while for the operator protection services (for employee security) low-activity periods at nights alternate with periods of high event frequency during the days. Thus, the overall distribution of events and profiles changes in the ENS.

For qualitative adaptation, depending on the available sensor types and the application context, the profile has to be evaluated differently: Notifications may be sent on every failure message or only once. Also, the profile may have a different filter priority. Sensors in a new building with different characteristics may be added to the system. The applications for laboratory safety and employee security may require different profile evaluation on the same data.

If the system does not *flexibly adapt to event sources and applications, the users are forced to redefine their profiles for every new event source (which could easily be hundreds of sources) and for every possible usage of the building.*

In this paper, we present an integrating event notification service that adapts to different application requirements in a number of ways: (1) The handling of composite events can be adapted to the needs of the service applications and clients. (2) The service flexibly reacts to application-dependent message delays. (3) In order to gain maximal performance,

the filter process for primitive events adapts to changing value distributions in events and user interests. (4) Several methods for the filtering of composite events are supported – which of them is employed depends on the profile distribution. The advantages of our system have been proven by a prototypical implementation.

This paper is structured as follows: Section 2 briefly introduces necessary background information. Section 3 focuses on semantical and correctness issues (qualitative adaptation), while Section 4 addresses performance of filter algorithms (quantitative adaptation). The A-mediAS architecture is described briefly in Section 5. We present case studies and selected results of the system's performance analysis (Section 6). The paper concludes with a brief discussion of related work and directions for future work in this domain.

2. BACKGROUND

This section first extends our example from the field of computer-aided facility management (CAFM). Then, the basic event-related concepts are introduced.

2.1 Scenario

Remote monitoring and control in commercial buildings are an important domain for adaptive ENS. A surveillance system for several buildings monitors lighting, heating, air condition, sun protection, and visitor movements. Various sensors are located in the monitored buildings: Some send status information on a regular basis to the system. Other sensors send only critical events, i.e., if the status values cross a predefined threshold. A third group passively collects data and is to be observed by the system (polled). Sensors may also have different reaction times and granularities.

Several different applications may use the data from the event notification service: access management, security, maintenance, energy management, laboratory safety, and budget management (for an extensive requirements analysis see [12]). Each of these applications may use the data in different ways. In multi-purpose buildings, the applications may change frequently. For example, depending on the actual usage of multi-purpose buildings and rooms, the surveillance system of a building covers certain profiles and events: for festive arrangements, the guests' security has to be ensured while for cultural exhibitions, strict environmental conditions have to be maintained for the presented pieces of art.

We now extend our example profile from Section 1:

> *P2: Notify the technician if in a certain room the temperature rises above* $35°C$ *within a time interval of 1 week length after a failure in the air conditioning system.*

Depending on the sensor type and the application, the technician has to be notified, e.g., about every occurrence of that event or just the first one. Similarly, also for each of the contributing events (high temperature and system failure), different evaluation methods may have to be applied. Additionally, the notification of the technician depends on the usage of that room: whether it is used as a chemical laboratory or for a conference. In the former case, a quick reaction is required and all critical occurrences have to be logged. In the latter case, the air conditioning has to be adjusted.

As pointed out in [2], applications for cooperative buildings have to offer scalability in quality as well as in quantity. The A-mediAS system fulfils these requirements by providing adaptive profile support plus efficient event filtering under different and changing application characteristics.

2.2 Basic Concepts

The central concepts for ENS are *events* and *profiles*.

DEFINITION 1 (EVENT). *An event is the occurrence of a state transition of an object at a certain point in time.*

The stream of incoming events at an ENS is called history, or *trace* of events. We distinguish *event instances* from *event classes*. An event instance relates to the actual occurrence of an event while event class is a set of event instances that share certain properties. We simply use the term *event* whenever the distinction is clear from the context. Even though instances of the same event class share some properties (e.g., temperature of a sensor), they may differ in others (e.g., location). Events (instances) are denoted by lower Latin e with indices, i.e., $e_1, e_2, \ldots$, while event classes are denoted by an upper Latin E with indices, i.e., $E_1, E_2, \ldots$. The fact that an event e_i is an instance of an event class E_j is denoted *membership*, i.e., $e_i \in E_j$.

We consider *primitive events* and *composite events*. Primitive events refer to single occurrences of state transitions. Composite events are formed by temporal combinations of event instances. For example, the sequence of two events (high temperature after system failure) is a composite event.

The definition of composite events requires temporal operators, e.g., sequence $(E_1; E_2)_t$ and conjunction $(E_1, E_2)_t$. For example, a *sequence* $(E_1; E_2)_t$ occurs when first $e_1 \in E_1$ and then $e_2 \in E_2$ occurs. The parameter t defines in the profile the maximal temporal distance between the events. The occurrence time $t(e_3)$ of event instance $e_3 := (e_1; e_2)$ is equal to the time of e_2, i.e., $t(e_3) := t(e_2)$.

DEFINITION 2 (USER PROFILE). *A profile is a query that is periodically evaluated against incoming events.*

Note that user profiles describe event classes. Duplicates of events are event instances that belong to the same event class. We distinguish two aspects of duplicate handling:

1. *Event instance selection* defines for each event, whether the user is interested in all, the first, last, or n^{th} event instance in a sequence of duplicate events.

2. *Event instance consumption* defines for each composite event, whether the event instances participate in only one pair of event instances or in several pairs.

EXAMPLE 2. *Consider the profile P2 defined by the technician. Let us assume that within one week after a system failure the temperature reaches three times a temperature peak. Then, several options are feasible, e.g., only one notification needs to be sent, because the technician is then aware of the problem. Or, three notifications have to be sent, because in each case, direct action is required, e.g., to protect the presented pieces of art.*

Consequently, these parameters have to be considered for the profile definitions.

3. ADAPTIVE EVENT COMPOSITION

We discuss strategies for adapting event composition to changing application requirements (*qualitative adaptation*).

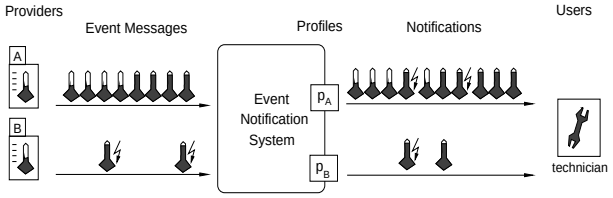


Figure 1: Illustration of Example 3

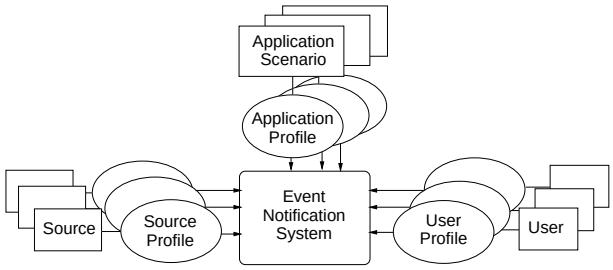


Figure 2: System players and their profile types

3.1 Event Composition Problem

To support a flexible filtering of composite events in A-mediAS, we implemented a parameterized algebra [10]. Selected aspects of a profile query can easily be changed without having to change the complete query definition. The algebra consists of temporal operators enriched by the parameters introduced above. An additional parameter is the *event evaluation time*: final vs continuous. Final evaluation waits for the end of a composite profile's maximal temporal distance t. Continuous evaluation is carried out continuously until t is reached.

For each primitive profile within a composite profile, the event instance selection parameter is set. For each composite profile, the instance consumption parameter is set. For example, the composite profile as defined before could be defined using the parameters EIS for event instance selection, EIC for event instance consumption and EET for the event evaluation time (the context of the evaluation):

$$profile(((E_{fail}^{EIS=all}; E_{temp}^{EIS=all})_{1\text{ week}})_{EET=final}^{EIC=remove})$$

In other systems, a rigid filter behavior is defined implicitly either by some (often inconsistent) language constructs or by the filter implementation. Our parameterized algebra flexibly allows for changes of profiles depending on the context. Thus, as we shall see, our system can support integrating event-based applications and changing event sources without the need to redefine user profiles.

A direct approach to event filtering evaluates the user profiles as they have been defined by the clients. Using this approach, clients are notified about events that match the same profile query. However, these events may carry different semantics, which may lead to false notifications.

EXAMPLE 3. *Consider a client that is interested in the event, that the temperature in a room rises above* $35°C$. *Events are provided by two different sources A and B covering the buildings A an B, respectively. Source A employs a surveillance sensor that sends the current temperature values every 30 seconds, source B employs a warning system that alerts the system in case of critical temperature changes.*

In order to be notified only about critical changes, the client would have to define separate profiles for each source: In the profile for source A, only the first *event in a series of duplicate events is necessary. In the profile for source B,* all *events are of interest. Defining only the first profile leads to missed events from building B, defining only the second profile leads to false alarm in building A.*

A different handling of profiles for each source and application context is needed. The solution that comes first into mind are wrappers for events sources. For our context, wrappers may be used for ENS that have a fixed profile evaluation strategy. For systems with changing application characteristics as considered here, new wrappers would have to be provided for each source and application.

We propose the employment of three profile types: user profiles are introduced before, source profiles to describe the available event types, and application profiles to capture the applications' characteristics, see Figure 2. Now, for each system context (available sources and applications), a specific profile evaluation strategy may be applied.

3.2 Adaptive Event Composition

In this subsection, we analyze the system players that may influence the event composition: For a given composite profile, the appropriate parameter values for EIS, EIC, and EET depend on the characteristics of the *event sources*, the *user preferences* defined in the profiles, and the ENS *application scenario*.

Information about the nature of the observed events that are provided by a given event source (e.g., a temperature sensor) may be defined in a *source profile*, see Figure 2.

DEFINITION 3 (SOURCE PROFILE). *A source profile describes the event types available at a particular source. Additional information may refer to the characteristics of the event observation, such as observation method and frequency.*

For each event source, a source profile has to exist at the ENS. These profiles prevent user profiles regarding events that are not covered by the service. To prevent that a user has to define multiple user profiles that have to be updated whenever a new source joins the service (to cover each source), a translation between client profiles and source profiles is needed. This translation could be controlled by an application expert, e.g., in defining translation rules. This is especially useful for the inclusion of new sources with different characteristics. Additionally, the translation may cover application-specific aspects, e.g., tolerances, accuracy bounds, and typical composition parameters.

We introduce the concept of *application profiles* to store information about the filter semantics required by applications. We use concepts known from query transformation in database systems (see, e.g., [11]). For example, an application profile contains the transformation rules to translate profiles that have been defined for a specific source characteristic into profiles for other source characteristics.

DEFINITION 4 (APPLICATION PROFILE). *An application profile is a set of query transformation rules for translating queries according to source profiles into new queries.*

This transformation of user profile queries changes the event streams directed to the users. The queries are translated

such that for different source profiles, a query with equivalent semantics is created for each of the available sources. Using our parameterized algebra, most transformation rules only require the application of new parameter values.

One approach is to define each user profile with an explicit reference to a certain source type. Then an application profile defines $N_s \times (N_s - 1)$ transformation rules for translating the queries for each source accordingly (N_s refers to the number of distinct source profiles).

We follow another approach that is more appropriate for realistic application contexts: For each application, a typical (default) source type is defined by the application expert. It is assumed that all user profiles are defined relative to that default source type. Based on this default source type for the profiles, $(N_s - 1)$ transformation rules are defined in the application profile. In both cases, the transformation rules depend on the application context, such that an automatic creation of the rules based on the source profiles is only feasible for selected and simple characteristics.

The application of the event instance selection parameter EIS depends on user preferences, available source types, and the application. The application of the parameters for event instance consumption and evaluation time depend on user preferences and system restrictions only. Due to limited system resources, the buffer of event instances that await matching composite partners may have to be restricted or their evaluation may be postponed.

The principle of profile transformation can also be used to cooperate with other ENS in a network of services. Our integrating service may use other ENS as intelligent event sources by forwarding user profiles to these services. The event operators supported by these services may differ from our service. Therefore, the profiles have to be translated. Similarly, the notifications sent by these services have to be processed, e.g., the notifications announcing two conjunctive events have to be filtered such that only valid sequences remain. We implemented this version in a separate prototype.

3.3 Influencing Composition Accuracy

Accuracy of event timestamps depends on observation methods and the time system of the event sources [6]. Therefore, sources have to provide information about the used observation and timestamping methods within the source profile. Based on the accuracy requirements defined in user profiles, the ENS has to calculate the accuracy of composite event information. For different applications, different accuracy restrictions may hold. Therefore, the handling of composition accuracy has be be controlled by the application profile. Notification about composite events that exceed the tolerance range defined in the user profile may have to be discarded or they have to be accompanied by a warning.

4. ADAPTIVE EVENT FILTERING

In this section, we discuss strategies for adapting the event filter process to the value distributions of events and profiles to gain maximal performance (*quantitative adaptation*).

4.1 The Filter Algorithms

Tree-based algorithms provide best performance for primitive event filtering in ENS. All profiles are combined into a profile tree, each level of the tree corresponds to one attribute, each tree-branch to an attribute-value. Depending on the distributions of events and profiles, the tree may be reordered to achieve even more efficient filtering. We introduced selectivity measures for the tree-reordering. Value-dependent measures control the reordering of the tree-branches, i.e., the order in which the different attribute-values in the tree are tested. Attribute dependent measures control the reordering of the tree-levels. The distribution-based approach improves the average case performance of tree-based algorithms, for details see [8].

For the filtering of composite events, current systems require a second filtering step after the identification of the primitive components. These two-step algorithms perform unnecessary filter operations. We propose a single-step method for the filtering of composite events. Our algorithm takes advantage of the idea of partial evaluation: only those profiles are evaluated that may directly contribute to a composite profile. Using our method, the filter response times for composite events are significantly reduced. Additionally, the overall performance of the event filtering is improved. Depending on the distribution of composite profiles in the set of all profiles, the appropriate filtering method for composite events is selected by the ENS, for details see [7].

4.2 Measuring the Distributions

If the system load does not change over time, i.e., the application's profile and event distributions remain relatively stable, these application characteristics can be used to directly adjust the system for best performance.

If the system load changes over time, the distributions of events and profiles have to be observed. For the profiles, this is a relatively simple task, since the required information is directly available at the system side. In a distributed system, for each of the broker nodes, the distribution of the profile values within each attribute domain have to be analyzed. For a detailed profile analysis, a profile history may have to be maintained. For the events, the distribution of event values within the attribute domains have to be observed by collecting and evaluating the event information in a persistent history of events.

4.3 Adapting the Algorithms

Different filtering variants should be used when (a) the application changes, (b) a different use case is temporarily applied, or (c) the system load (events and profiles) changes. While the first two cases are relatively easy to manage, the third one requires detailed consideration: Performance costs are influenced by methods for event persistency, distribution measurement and handling of the algorithm adaptation.

How shall we determine the current event distribution at runtime? Events and profiles can either be stored as a history and analyzed later, or they are directly evaluated. The first case provides more flexibility in the evaluation but requires access to persistent storage devices. An option is the storing of a number of events in main memory with regular access to persistent storage during times of low system load. In the second case, the distribution parameters are computed directly, causing minor performance overhead.

For both cases, the distributions are not to be computed over the complete system runtime but for selected intervals. The length of the interval determines the topicality of the computed values but carries the danger of over- or undervaluation of load changes. The length of the interval is defined by the application administrator or based on a application

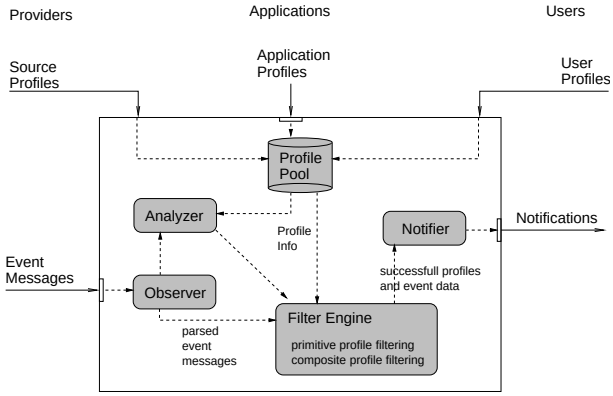


Figure 3: Architecture of an A-mediAS server

profile for the change-frequency of events and profiles.

In order to prevent a jitter in an auto-adaptive system, the evaluation interval has do be defined long enough. Additionally, a hysteresis can be defined: changes in the algorithms are only realized after the new conditions hold for at least the length of the hysteresis value or the changes are larger than a predefined hysteresis value.

5. A-MEDIAS IMPLEMENTATION

A-mediAS is implemented as distributed service consisting of a number of event servers. Here, we omit the description and evaluation details for the service's distribution. Each of the event servers follows the A-mediAS architecture as presented in Figure 3. We enhanced the basic ENS architecture [9] by an adaptation component. The components of the architecture include *interfaces* for providers, users, and application experts to define their profiles, which are stored in the the *profile pool*.

The event processing is driven by the observer: Whenever an event is observed, a time stamp is created and a message is forwarded to the filter engine. The information about the event is additionally logged by the analyzer. The core of the system is the interplay of analyzer component and filter engine. The algorithms used in the filter engine are controlled by the analyzer output. The analyzer has access to event and profile information: It logs the event history and has access to the profile tree. Based on this information, the analyzer computes the distributions of events and profiles. Currently, this information is only made available for the system administrator. The administrator then triggers any changes of the filter strategy at the analyzer component. The analyzer controls necessary reordering of the filter tree for primitive event filtering as well as changed strategies for composite event evaluation. Once the matching profiles have been identified, a notification is sent to the profile's user.

6. A-MEDIAS EVALUATION

In this section, we discuss the performance of different filter algorithms in an A-mediAS server. In case studies, we exemplarily show the influence of adaptive system behavior.

6.1 Experimental Environment

We implemented the algorithms in A-mediAS using Java. The distributed servers are connected by an acyclic overlay network. Here, we concentrate on experiments regarding a single server. The experiments were performed on a PC (Win2000) with 512MB main memory. The events and profiles where created using our event and profile generator for automatic creation of test cases. The tool randomly generates events and profiles based on predefined distributions of attribute values. In the experiments, each user profile contains one predicate on each of the attributes defined for the respective event type. We created different workloads by changing the workload parameters of the event and profile generator.

Parameter	Range	Description
N_p	5,000 - 55,000	Number of profiles
N_e	5,000 - 55,000	Number of events
N_a	1-5	Number of attributes
P_p	PD0-PD4	Profile distributions
P_e	ED0-ED5	Event distributions

Table 1: Workload Parameters

We measured the performance of the server as the *filter time* to find all matching profiles for each of the incoming events. For the filter time per event, the mean value of the filter time for a number of events N_e is used. The costs of creating the events and profiles as well as the profile representation are not included in the filter time.

A list of workload parameters is given in Table 1. If not explicitly stated differently, a uniform distribution of attribute values is applied ($ED0$ and $PD0$). For the profile distributions, the profile attributes cover the given percentages (dense uniform distribution, starting at the lower end of the attribute domain) of each of the five attribute domains as shown in Table 2, top. For the event distributions, in each attribute the values cover the given percentages of attribute domains. All attributes have the same distribution. Each attribute value distribution holds the characteristics (dense uniform distribution, starting at the lower end of the attribute domain) as shown in Table 2, bottom.

P_p	Profile Distribution Description
$PD1$	50/40/30/20/10, (small steps)
$PD2$	50/10/40/20/30, (small steps)
$PD3$	10/20/30/40/50, (small steps)
$PD4$	100/80/60/40/20, (wide steps)
P_e	Event Distribution Description
$ED1$	20% uniformly covered / 80% uncovered
$ED2$	50% uniformly covered / 50% uncovered
$ED3$	100% uniformly covered / 0% uncovered
$ED4$	50% uncovered / 50% uniformly covered
$ED5$	20% uncovered / 80% uniformly covered

Table 2: Distributions of events and profiles

6.2 Application-dependent Event Composition

A simple case study shows how the A-mediAS system adapts to different application contexts. We consider two technicians with a composite profile $P2$. Each of the technicians is responsible for one air conditioning system. Six different sources are covered: four temperature sensors and two air conditioning systems are monitored. The events sent by these sources are shown in the event histories Figure 4, top. We consider three application scenarios:

In application 1, the four temperature sensors are in four separate rooms. Due to an insurance contract, after a system failure each technician has to inspect the affected rooms for at least one week. In application 2, the four temperature sensors are in four separate rooms. The technician has to check the affected rooms for at least a week. The rooms have only to be checked by one technician. In application 3, the rooms in the buildings have been changed (movable walls) so that the four sensors refer to the same room. Similar to application 2, only one technician is responsible at a time.

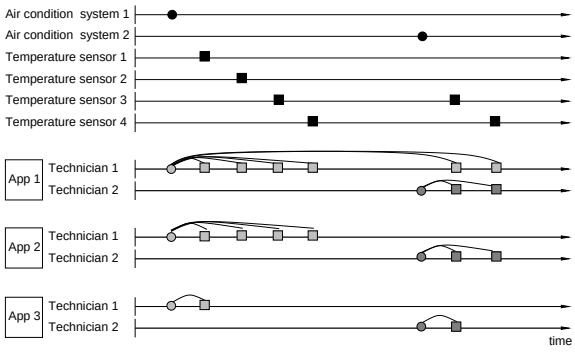


Figure 4: Case Study: Adaptation of event composition to application scenarios

Figure 4 shows the composite events each technician is notified about in each of the three application scenarios. The arcs refer to the events pairs contributing in each composite event. From the number of matched events, we see that qualitative adaptation additionally influences the system load, which, in turn, calls for the quantitative adaptation of the system.

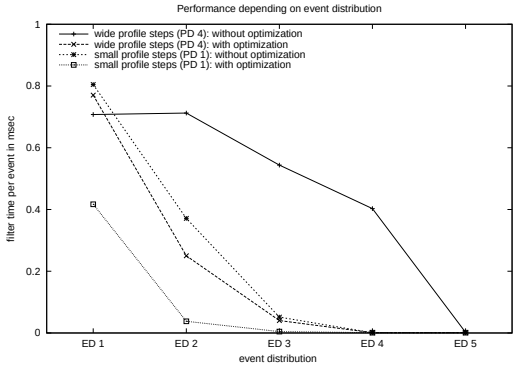
6.3 Primitive Event Filtering

We describe the results of two experiments and a case study to evaluate the performance of primitive event filter algorithms. We show the results for selected profile distributions P_p and event distributions P_e.

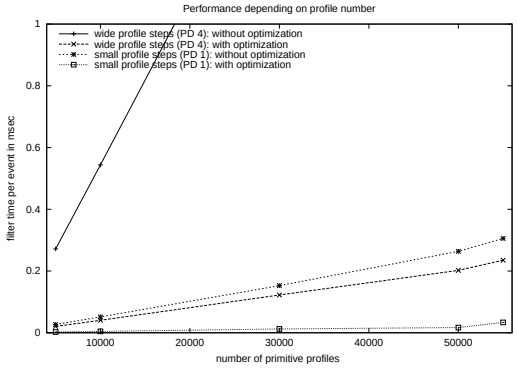
Figure 5(a) shows the filter time under varying event distributions. Most events are matched in $ED1$, least in $ED5$. The data support our finding that the optimized filtering method (one-step algorithm) outperforms the two-step algorithm if unmatched events occur. For $ED1$, the un-optimized algorithm is faster than the optimized one because all events are matched by profiles. For this case, a different optimization strategy would have to be used (for details see [8]).

Figure 5(b) displays the response time under varying profile numbers. The optimized tree shows a higher performance than filtering using the arbitrary order of the attributes in the event type. The reordering is about 3 to 4 times faster than the original algorithm. This ratio increases with the event and profile number, respectively.

Figure 6 shows the response time for the following case study: A system starts with 10,000 profiles; 50,000 events are filtered. Then, 10,000 more user profiles are defined; another 50,000 events are filtered. At last, 10,000 additional profiles are defined, followed by 50,000 events to be filtered. The profile sets are defined in the following order: $P_p = PD1$, $P_p = PD2$, and $P_p = PD3$. Every possible optimization decision is shown: after each added profile set the profile tree may be reordered or not.



(a) Filter time based on event distribution, $N_p = 10,000$, $N_e = 50,000$, $N_a = 5$, $P_p = PD0$, $P_e = ED0$



(b) Filter time based on profile number, $N_e = 50,000$, $N_a = 5$, $P_p = PD1$, $P_e = ED0$

Figure 5: Performance of primitive profile matching

The optimal filter behavior is shown as thick line. Two modes may be used: continuous analysis or scheduled analysis. For the first, the required tree-reordering may be applied to an offline filter pool which is then exchanged with the running system. In the scheduled analysis, the filter pool is optimized based on a scheduled analysis of the filter pool.

We see that under unfavorable conditions, an optimized tree with additional profiles has lower performance than a non-optimized tree. This results from profile distributions that require opposite optimizations: the optimized tree for one profile set is the worst-case constellation for another one. A regular tree-optimization shows best performance results. The results of the case study emphasize the importance of tree-optimization after appropriate intervals.

6.4 Composite Event Filtering

In this subsection, we present the results of three experiments and a case study to evaluate the performance of the composite event matching algorithms in A-mediAS. We studied the influence of the profile number and the construction of the profile set (i.e., the influence of the proportion of

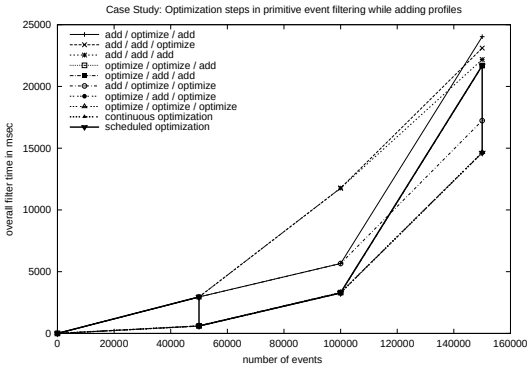


Figure 6: Case Study: Adaptation of primitive event filtering to variable profile sets $[PD1 + PD2 + PD3]$

composite and primitive profiles in the set).

We present our tests of composite profiles referring to simple event sequences. Two extreme types of sequences are used: in a set of n composite profiles, either (1) all second primitive profiles refer to the same event (N to 1), or (2) all first profiles refer to the same event (1 to N).

For each test, we evaluated four variants: two-step and single-step algorithm used for single or multiple matches. In single matches, each profile is matched only once and then removed. This simulates the removal of profiles after a certain lifespan. In multiple matches the profile persists.

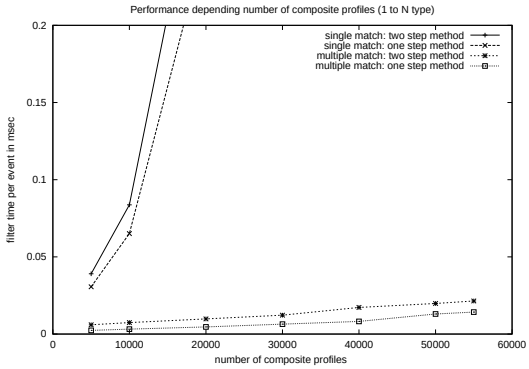
The filter time per event for varying number of profiles (1 to N) is displayed in Figure 7(a). In both cases of single and multiple matches, our one-step method shows higher performance than the performance of the two-step method.

The Figures 7(b) and 7(c) display the filter times per event under changing profile sets. Starting with 30,000 composite profiles, we successively added primitive profiles. In a second phase we successively removed composite profiles until 30,000 primitive profiles are reached. Note that the profile number is changing in the profile sets. We show the results for both, (1 to N) and (N to 1) types of profiles.

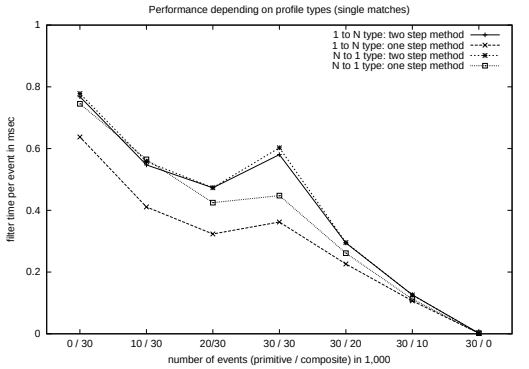
For single matches, the single-step method outperforms the two-step method in all tests. For multiple matches on to (1 to N)-type composite profiles, the single-step method is slower than the two-step method if the number of composite profiles exceeds the number of primitive profiles. The reason for the behavior is that under this conditions the profiles do not overlap. The more profiles that have to be newly inserted into the tree the slower the matching time. For profile sets with high overlap, the single step method shows better performance – for profile sets with only (1 to N) type profiles without overlap, the two-step method has to be used.

Figure 8 shows the results of our case study regarding composite events. We measured performance under changing user profiles using a blend of different profile types: 10,000 profiles with (1 to N) type are inserted, then 10,000 profiles with (N to 1) type, and finally, 10,000 primitive profiles. After each update, 50,000 events are submitted to the system. All possible optimization steps are shown. In contrast to the tree reordering, the optimization of composite profile filtering applied at one time does not influence the performance at a later time. The reason is that not the complete profile pool structure is changed but only the parts used in the particular filter methods.

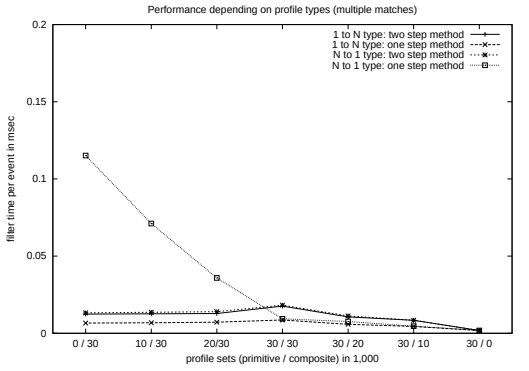
The performance of our adaptive A-mediAS server is shown



(a) Filter time based on profile number, $N_e = 50,000$, $N_a = 1$, $P_p = PD0$, $P_e = ED0$



(b) Filter time based on profile set (single matches), $N_e = 50,000$, $N_a = 1$, $P_p = PD0$, $P_e = ED0$



(c) Filter time based on profile set (multiple matches), $N_e = 50,000$, $N_a = 1$, $P_p = PD0$, $P_e = ED0$

Figure 7: Performance of comp. profile matching

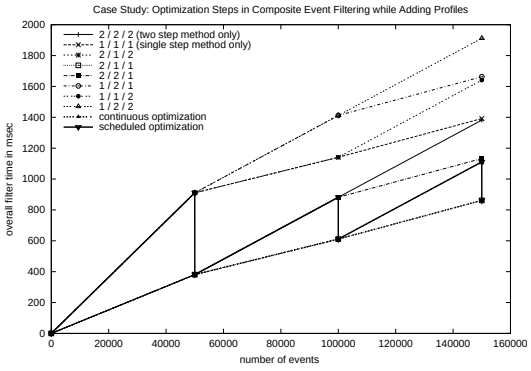


Figure 8: Case Study: Adaptation of composite event filtering to profile set [N to 1/ 1 to N / primitive]

as thick line. Again, two modes may be used: the continuous analysis of the profile set with immediate adaptation or analysis and adaptation after n profiles have been changed. Both versions are shown in the figure. The results of the case study emphasize the importance of adaptive profile filtering for primitive as well as for composite events.

7. RELATED WORK

Siena [4] and Elvin [13] use source profiles to reject events no user is subscribed to and to prevent profiles without matching event types. The HiFi [1] is based on a reusable semi-complete framework that may be easily reorganized for different applications. Different to our approach, the HiFi system focuses on a flexible software development process.

Adaptation is a well known concept in software development for interface conversion [5]. Two design patters are used: `adapter` and `proxy`. An adapter converts between different types and leaves the semantics unchanged, while a proxy modifies the semantics of a call without changing its type. Automatic selection of adaptors gains increasing attention in mobile computation and ad-hoc networks [14].

User adaptivity methods build user profiles based on the user action (see, e.g., [3]). Such methods could be included in an ENS, but are not in the focus of our work.

8. SUMMARY

Event Notification Services play a major role in many recent applications – upcoming systems cover multiple applications and integrate event data from different sources. These systems call for ENS that qualitatively and quantitatively adapt to changing sources and different application requirements. Otherwise, the users would have to define profiles for changing application and source anew; the filter performance would not achieve the best possible results.

In this paper, we presented the design of our adaptive event notification system A-mediAS. We discussed the implementation concepts and set special focus on adaptivity. The A-mediAS system adapts to changing application context and new sources by translating the user queries according to an application profile. We have demonstrated this adaptive system behavior in a case study.

The system adapts to changing system loads of events and profiles by selecting the optimal filter algorithms for both primitive and composite events. We extensively evaluated the system performance under changing workload. Case studies for primitive and composite events have shown the adaptive filter behavior.

A-mediAS is designed to be used for mixed applications such as the management of commercial buildings as introduced here. Currently, we are analyzing the A-mediAS system in the context of Commercial Big Buidings (CBB), in cooperation with Lichtvision GmbH, a Berlin-based company for facility management. Adaptive systems, such as A-mediAS, can also be used for mobile systems, where the available data sources change over time and the client can only process a limited amount of data.

9. REFERENCES

[1] E. Al-Shaer, M. Fayed, and H. Abdel-Wahab. Adaptive object-oriented filtering framework for event management applications. *ACM Computing Surveys*, 32(1):37–37, 2000.

[2] B. Bruegge, R. Pfleghar, and T. Reicher. Internet framework for cooperative buildings. EIB Event, Munich, Germany, October 2000.

[3] P. Brusilovsky and M. Maybury. From adaptive hypermedia to the adaptive web. *CACM*, 45(5):30–33, 2002.

[4] A. Carzaniga, D. S. Rosenblum, and A. L Wolf. Design and evaluation of a wide-area event notification service. *ACM TOCS*, 19(3):332–383, August 2001.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns, Elements of Reusable Object-Oriented Softeare*. Addison Wesley, 1995.

[6] A. Hinze. How does the observation strategy influence the correctness of alerting services? In *Proc. of the German National Database Conference (BTW)*, March 2001.

[7] A. Hinze. Efficient filtering of composite events. In *Proc. of the British National Conf. on Datbases (BNCOD)*, 2003.

[8] A. Hinze and S. Bittner. Efficient distribution-based event filtering. In *Proc of the DEBS workshop at the ICDCS*, 2002.

[9] A. Hinze and D. Faensen. A Unified Model of Internet Scale Alerting Services. In *Proc. of ICSC*, volume 1749 of *LNCS*, December 1999.

[10] A. Hinze and A. Voisard. A parameterized algebra for event notification services. In *Proc. of the TIME*, 2002.

[11] L. Raschid, Y. Chang, and B. Dorr. Query transformation techniques for interoperable query processing in cooperative information systems. In *Proc. of the Coopis*, 1994.

[12] T. Schürle, A. Boy, and D. Fritsch. *Geographic Information Systems and Facility Management, Report of WG IV/III.1*. 1998.

[13] B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content Based Routing with Elvin4. In *Proc. of the AUUG2K*, Canberra, Australia, 2000.

[14] J. Vayssiere. Transparent Dissemination of Adapters in Jini. In *Proc. of the DOA*, 2001.