Int. J. Communications, Network and System Sciences, 2009, 3, 169-247
doi:10.4236/ijcns.2009.23023 Published Online June 2009 (http://www.SciRP.org/journal/ijcns/).

Mobility Trigger Management: Implementation and Evaluation

Jukka MÄKELÄ, Kostas PENTIKOUSIS, Vesa KYLLÖNEN
VTT Technical Research Centre of Finland, Oulu, Finland
Email: {*jukka,makela, kostas.pentikousis, vesa.kyllonen*}*@vtt.fi*
Received December 11, 2008; *revised April* 21, 2009; *accepted April* 23, 2009

ABSTRACT

Modern mobile devices have several network interfaces and can run various network applications. In order to remain always best connected, events need to be communicated through the entire protocol stack in an efficient manner. Current implementations can handle only a handful of low level events that may trigger actions for mobility management, such as signal strength indicators and cell load. In this paper, we present a framework for managing mobility triggers that can deal with a greater variety of triggering events, which may originate from any component of the node's protocol stack as well as mobility management entities within the network. We explain the main concepts that govern our trigger management framework and discuss its architecture which aims at operating in a richer mobility management framework, enabling the deployment of new applications and services. We address several implementation issues, such as, event collection and processing, storage, and trigger dissemination, and introduce a real implementation for commodity mobile devices. We review our testbed environment and provide experimental results showcasing a lossless streaming video session handover between a laptop and a PDA using mobility and sensor-driven orientation triggers. Moreover, we empirically evaluate and analyze the performance of our prototype. We position our work and implementation within the Ambient Networks architecture and common prototype, centring in particular on the use of policies to steer operation. Finally, we outline current and future work items.

Keywords: Triggering, Mobility Management, Mobile Networks, Handover, Cross-Layer Information Management

1. Introduction

Modern mobile devices, such as smartphones, Internet tablets and PDAs, have several network interfaces and can run various network applications, like web browsers, email clients, and media players. Indeed, it is becoming common that said devices can take advantage of wireless LAN, PAN and cellular connectivity, and we expect that in the coming years mobile WiMAX will be supported as well. In such a multiaccess environment, mobility management support for both horizontal and vertical handovers should be one of the basic functionalities in future devices. Moreover, in order to allow a mobile device to remain always best connected, several events need to be communicated through the entire protocol stack, as we explain in the following section. Nevertheless, current implementations of state-of-the-art mobility management protocols, such as Mobile IP [1] or Host Identity Protocol [2]), can only handle a small set of event notifications that may lead to mobility management actions, including handover execution.

In this paper, we argue for a novel mobility trigger management framework that can handle a much larger set of notifications related to events originating not only from the lower layers of the protocol stack (physical, data link, and network), but also from the upper layers enabling the efficient use of cross-layer information for mobility management. This framework needs to be open, flexible, with low overhead, and incrementally deployable. After describing the main parts of the architecture, we present the implementation of such a framework, which allows mobile devices to manage, on the one hand, conventional mobility events, such as the availability of a new network access, received signal strength indications

(RSSI), network capacity load and, on the other hand, higher level events, such as security alerts, policy violations, end-to-end quality of service deterioration, and network access cost changes. In our framework, event sources can deliver notifications to interested applications and other system entities used in a standardized manner. We will refer to these standardized notifications as triggers in the remainder.

The main elements of our trigger management framework are detailed in [3,4], and include the entities which generate the events (producers) and entities that use the trigger information (consumers). Our trigger management framework is capable of collecting event information from various producers through a specific collection interface. The collected events are then processed and converted into a unified trigger format, described in Section 5, and distributed to interested consumer entities. A trigger consumer can be any entity implementing the collection interface and can be located in the same or in different node in the network. It should be noted also that a same entity can act both as a producer and a consumer.

In this paper we concentrate on the evaluation of the implementation of our framework in the VTT Converging Networks Laboratory. Indeed this paper demonstrates the feasibility of our designed framework over a real testbed network. The concept and architecture behind our framework with some analysis to the similar existing concepts are also summarized below.

The rest of this paper is organized as follows. Section 2 introduces the fundamental elements of our framework for managing triggers, reviews the related work in this area and motivates our evaluation. Section 3 presents our implementation of the triggering framework and Section 4 discusses the role of policies and rules in the system design. Results from our experimental lab evaluation are presented in Section 5. Related work is discussed in Section 6, and Section 7 concludes the paper.

2. A Framework for Managing Mobility Triggers

After surveying the relevant literature (see, for example, [5,6−10]), and based on our own expertise, we identified more than one hundred different types of network events related to mobility management. We cluster triggers, regardless of the underlying communication technology, based on groups of events related to changes in network topology and routing, available access media, radio link conditions, user actions and preferences, context information, operator policies, quality of service (QoS) parameters, network composition [11], and security alerts.

Figure 1 illustrates six different trigger groups as boxes. The "offshoots" on top point to example triggers belonging to each group. The rightmost group includes representative link layer "up/down" triggers (irrespective of the radio access technology). The leftmost group includes triggers originating from the application layer. In this example, certain triggers originate from the node ("System Resources") while others originate from the network ("Macro Mobility"). The "origin" corresponds to the entity that produces the trigger, for example, the radio access component. An advantage of our grouping approach is that it allows us to detect relations between otherwise disparate triggers. This prevents the generation of excessive transient triggers based on, for example, the same root-cause event, such as a link outage, and reduces the number of events that need to be processed.

Event sources need to be able to deliver notifications to interested applications and other system entities in a uniform, concise, and standardized manner. This approach simplifies notification handling considerably, while guaranteeing sufficient diversity for event separation and classification. In order to manage and efficiently propagate triggers originating from a variety

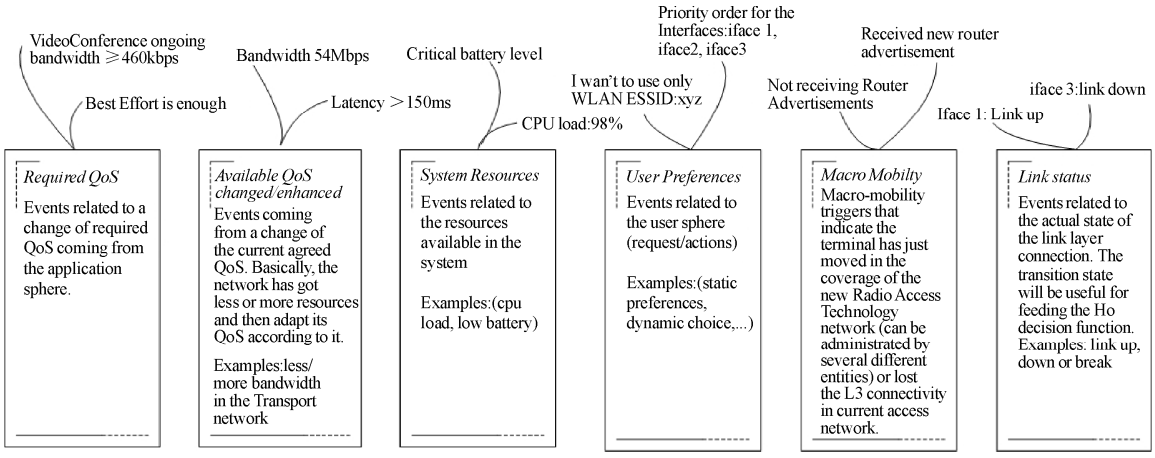

Figure 1. An example of trigger groups.

of sources we developed a trigger management framework, which we call TRG. TRG lays the foundation upon which sophisticated handover (HO) operations can be performed. We aim at establishing an extensible frame- work where new sources of triggers can be defined and included in a system implementation as necessary. Note that this is quite different from other, in our opinion, more closed and specific approaches, such as the one followed in the IEEE 802.21 [12] working group. On the surface, both TRG and the IEEE 802.21 Media Independent Handover Services standard seem quite similar, aiming to improve mobility management performance. However as we argue in [13] the mechanisms and services introduced by the IEEE standard do not include dynamic information elements and any extensions will have to be introduced with lengthy standardization procedures in the future. Moreover, triggers cannot originate from the higher layers of the protocol stack, and system level events are simply out of scope of IEEE 802.21. Finally, 802.21 provides services to command and use the lower layer information to enable seamless handovers and multiaccess, which is not in the domain of TRG, but of the mobility management protocol. Last but not least, TRG is designed to handle much more event sources than MIHF. It is important to highlight that TRG provides the means to disseminate and filter mobility-related information between one or more event sources and several trigger consumers but that HO decisions are still the responsibility of the mobility management protocol, say, Mobile IP [1] or HIP [2]. TRG can also provide hints about moving the communication endpoint from one device to another, as explained in Section 5.

A central part of the design is designating different system entities as producers and consumers of triggers. Policies, described in Section 4, are handled by the Policy Manager. For communicating with different entities, TRG exposes three service access points (SAP). Event sources use the Producer SAP, to register events and emit notifications to TRG when changes occur. Consumers use another SAP, to subscribe with TRG and receive triggers in a single format when they become available. Finally, the Policy Manager uses another SAP to inform TRG about policies. Internally, TRG implements a local trigger repository and functional blocks for processing triggers.

Consumers must state their need to receive triggers and can choose to stop receiving them anytime. For example, the Mobile IP daemon can receive all triggers related to link layer events, but opt to receive only the upper-layer triggers associated with security or policy violations. In the former case, such a consumer takes advantage of the trigger grouping functionality; in the latter, it additionally requests trigger filtering. Consumers can use these triggers to generate their own and, thus, serve as an event producer for other entities. We expect that TRG will be used to guide HO decision making and execution. In particular, consumers can use triggers to derive whether the mobile device is moving within a single network or it is crossing different access technology boundaries, and whether the addressing scheme, trust and provider domains should be changed accordingly.

3. Architecture and Implementation

The core implementation of TRG has three major components: triggering event collection, trigger processing, and the trigger repository [3,4]. Triggering events collection receives events from various sources in the network system via the trigger collection interface. New triggers can be introduced in a straightforward manner by implementing the trigger event collection functionality and supporting the trigger collection interface. The latter allows sources to register their triggers and to make them available to consumers. A specific TRG implementation may contain several event collectors, which may be distributed, and are responsible for collecting different types of events. The trigger repository is designed to meet the stringent requirements placed by mobility management, but can be used to store non-mobility triggers as well. The basic primitives include adding, removing, updating, and disseminating triggers in a standardized format. Each stored trigger has an associated lifetime and is removed automatically once its time-to-live (TTL) expires.

The need for different event collectors arises from the fact that the origin of an event source can be a hardware device, a system component implemented in kernel space, or an application implemented in user space. For example, each device driver could implement its own event collection functionality, which would be capable of handling triggering events produced by the specific device only. Moreover, sources can also be located in the network such as at active network elements or at the user's home network. Finally, a particular TRG implementation can act as a consumer to another TRG located in a different node. Thus, orchestrating the collaboration of, perhaps, several collection entities is needed in order to efficiently gather a larger amount of events.

Having dedicated collectors for different event sources enables the use of TRG in different operating systems as well. The collector can format the events to the format that TRG understands and there is no need to modify the core of TRG functionality; instead the collector can be modified as necessary. This is also one of the key points in the architectural design of TRG that enables it to handle cross-layer information by having a collector at different layers as needed. For example TRG can get simi-

lar information considering the connectivity in FreeBSD through a collector that uses Route Socket and in Linux through a similar collector using RTnetlink socket but obviously these collectors need to have their own implementation. The core of TRG could be implemented in kernel space for performance reasons and allowing for direct access to lower layer information. On the other hand, TRG can be implemented in application space allowing for greater flexibility and easing implementation and code evolution. The prototype described in this paper follows the latter approach. Of course, certain event collectors will have to be implemented closer to the lower layers in the future.

The event sources are connected with TRG via producer SAP, as described also in section 2. The performance of the event collectors is obviously very important. They need to be fast enough to react to all different events, but the collector implementation itself is not part of the TRG framework architecture. TRG provides the interfaces to connect different event producers with the possible consumers by defining the SAP's between TRG and them. TRG core functionality per se provides the mechanisms for distributing, filtering and handling the policies for the whole system of the mobility event handling, but the collectors are out of scope of this paper. Figure 2 illustrates the TRG framework with the different event producers and consumers.

After events are collected from the producers, they are handed over to the trigger processing engine which is responsible for time-stamping and reformatting triggers (if necessary), and assigning them to the appropriate group. Consumers can subscribe by specifying a set of triggers (and, optionally, filtering rules) and are expected to unsubscribe when they do not wish to receive them any longer. For each consumer subscription, TRG makes sure that filters are grammatically and syntactically correct, and accepts or rejects the subscription. Basic rules can also be used as building blocks for crafting more sophisticated rules.

4. Policies and Rules in TRG

TRG supports the application of different triggering policies, defined as a set of classification, filtering, trust, and authorization criteria/rules. This allows our implementation to enforce a different policy at different times or wh en the node operates in different contexts. The availability of a system-wide policy and consumer- supplied filters lies at the centre of our TRG design. These two are orthogonal, providing flexibility and adaptability.

System policies ensure that only designated consumers can receive certain groups or types of triggers. For example, a node may operate under different policies regarding network attachment depending on whether the user is on a business or a leisure trip. Policies can also establish different trigger classification and groupings in different contexts and are typically stored in a separate repository, accessible to the TRG implementation. Filters allow a consumer to focus a trigger subscription. For example, a monitoring application may be interested in receiving all network utilization measurements, while a VoIP application may be interested in receiving a trigger only when utilization exceeds a certain threshold and the user is in a call. In fact, a VoIP application can even opt to be an intermittent trigger consumer, subscribing and unsubscribing to receive certain triggers solely when needed.

Our TRG implementation uses access control policies to define:

- Which producers are allowed to register and send triggers to TRG. Producers are identified by the trigger IDs they register, and can be chosen on a system basis. For example, a policy allows only specific producers to register with TRG.

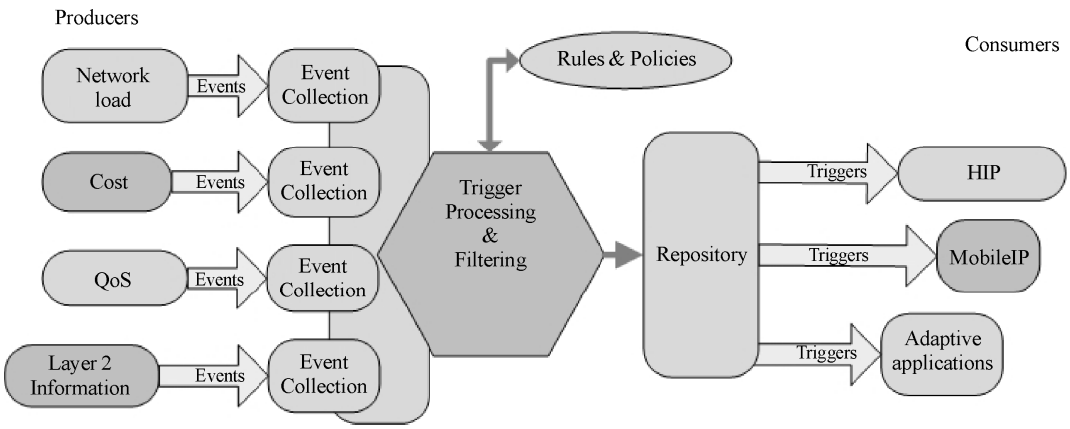

Figure 2. TRG architecture.

- Which consumers are allowed to subscribe to triggers. Policies can be very specific, prescribing which consumers can receive certain triggers and from which producers. Consumers are identified by their locator (typically a host address). For example, in our proof of concept implementation described in Section 5, we can enforce a policy that dictates that triggers from producer with ID=50 are allowed to be subscribed only from "localhost" entities.

The Policy Manager applies access control using policies described in XACML (OASIS eXtensible Access Control Markup Language) [14]. Figure 3 illustrates which Policy Manager functions are called when a producer registers or a consumer subscribes. Typically the decision on whether to allow producer registrations and consumer subscriptions is made immediately based on the system policies and the result is returned to the initiating entity. In the case where a consumer attempts to subscribe to all triggers, the decision may be deferred for when triggers become available. That is, the subscription for "all triggers" effectively becomes a subscription for "all triggers allowed", when system policies dictate so. In our current prototype implementation, policies are described using access control lists read from a configuration file. Policies also define which consumers are allowed to subscribe and for which trigger.

5. Results

We tested our user-space C++ implementation of TRG on laptops running FreeBSD release 6.1, Linux Fedora Core 3 with kernel 2.6.12 and Windows XP, and on a PDA running Linux Familiar v.0.8.4 with kernel 2.4.19. Architecture design with the possibility to use separate event collectors in different environment, as discussed in Section 3, makes our TRG implementation portable and is currently being integrated in several prototypes, including the Ambient Networks [15] prototype [16–18]. For communication between producer, TRG and consumer a Web Service XML-based communication on top of HTTP was used. In this integrated prototype, TRG takes care of the delivery of all mobility-related events. Events were formatted according to the unified trigger format shown in Table 1.

In previous work we presented a proof-of-concept testbed and demonstrated the feasibility of the concepts governing our TRG implementation. These preliminary validation results are summarized briefly in Subsection 5.1; further details are available in [4]. Subsection 5.2 presents the first detailed results of our stress-test empirical evaluation of TRG in the lab.

5.1. Proof of Concept Validation

In [4,19], TRG was employed to enable streaming video session handovers between different mobile devices. In the scenario, the user starts watching a video streamed to his laptop. His GNU/Linux PDA is nearby and the user decides to move to another room but would like to keep watching the video on the way. The commercial, off-the- shelf (COTS) PDA is augmented with a multi-sensor device (detailed in [20]), which was extended to provide "device orientation" triggers. For example, when the user picks up the PDA, a "vertical orientation" trigger is produced, initiating a session HO from the laptop to the PDA. The two devices have to coordinate and arrange for the transfer of the video streaming session. A successful session handover allows the user to receive the streaming video on the PDA over the WLAN seamlessly. The user can also explicitly initiate a session HO by pressing a PDA button. In this example, TRG handles triggers associated with mobility,

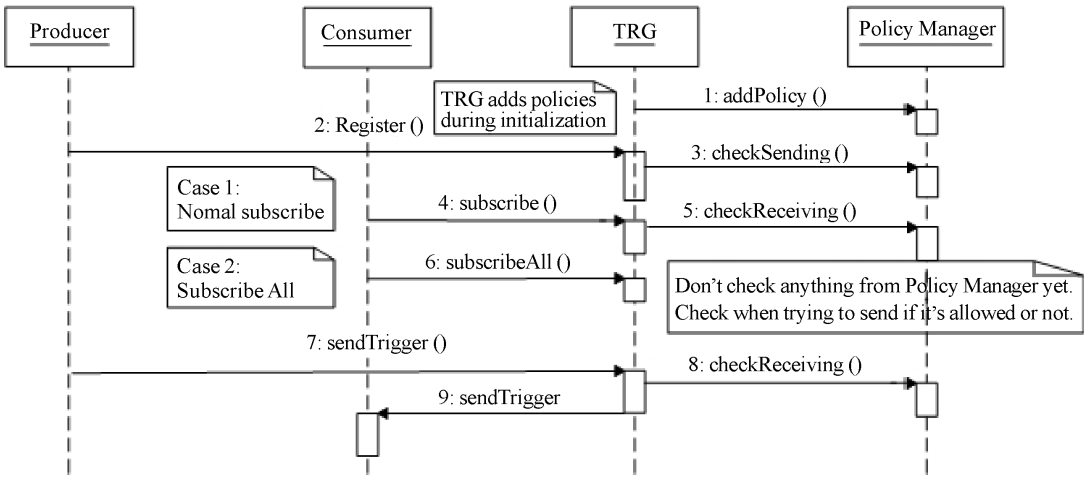

Figure 3. TRG-policy manager message sequence diagram.

Table 1. Trigger format.

Trigger data member	Type	Description
id	integer	Trigger identifier, same as producer identifier. Maps producer name to identifier.
type	integer	Specific to the trigger identifier. Mapping producer information to type.
value	std:string	Specific to trigger type.
timestamp	time_t	Time that a trigger enters the TRG repository.

orientation, and user preferences, keeping the video flowing smoothly while changing the communication end-point. Two logical topologies were evaluated in our lab. First, all devices are connected using IEEE 802.11 in ad-hoc mode, as if the user streamed a video from his digital collection at home. Second, the video streaming server is located in a different network, as would be the case when watching a video from a service provider over the Internet. For both setups in our lab proof-of-concept validation, we stream a 10-minute video encoded at 576 kb/s over UDP. At t = 3 min, a session HO from the laptop to the PDA is triggered, and at t = 7 min, the session is "moved" back to the laptop.

We captured all traffic traces during the experiments using tcpdump and cross-checked all packet IDs sent by the video server with the packet IDs received at the laptop and PDA video clients to confirm that no packet losses occurred. Moreover, the effect of TRG signalling and the actual session handover on packet delay is negligible, compared to packet delays before and after the session handover. Figure 4 illustrates the packet inter- transmission times as recorded by tcpdump at the streaming server and the packet inter-arrival times at (a) the receiving laptop and (b) the receiving PDA. On the left-hand side, the packet inter-arrival time measured at the streaming server, laptop and the PDA during the delivery of the 10-min video stream are shown. The band around 50 ms indicates that the packets are sent and received in an orderly manner. We note a small number of inter-arrival times outside this band. The vast majority of inter-arrival times do not exceed 150 ms; only a handful of packets out of more than 12000 exceed this threshold. On the right-hand side of Figure 4, we zoom in at around t=3 min when the first session HO is triggered from the laptop to the PDA. As the figure illustrates, only a few packets had >0.1 s inter-arrival time. These results are very promising, despite the fact that this is a prototype implementation, especially when taking into consideration that the PDA was running the video client and captured packets using tcpdump throughout the experiment leaving few spare system resources available.

This paper focuses on the empirical validation and evaluation of TRG. The theoretical aspects (scalability, security, reliability) have been partly addressed elsewhere [21] and further analysis is also part of our future work agenda. It is important to note that these set of experiments go beyond showcasing the concept of TRG-assisted session HOs. This is simply a particular application of triggers leading to a HO. Instead, we emphasize that these experiments aim at assessing the feasibility of introducing a TRG implementation in small COTS handheld devices, a result which was not warranted when we embarked in developing TRG.

5.2. Experimental Evaluation

Since we conducted the experiments presented in the previous subsection, we continued the development and evaluation of TRG and used an updated implementation of TRG enhanced with web service interfaces and ran tests where we submitted 100000 triggers from several sources to TRG and delivered those to different consumers. We consider two test cases, with the aim of quan-

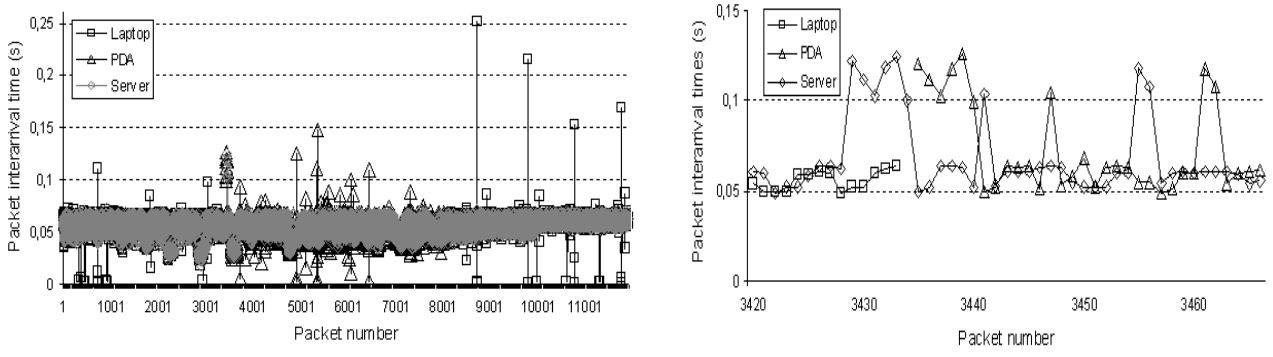

Figure 4. Experimental results when triggering a session HO.

tifying TRG performance under stress (and perhaps clearly unrealistic) conditions. Test Case 1 employs n producers connected with m consumers via TRG. During the test, each producer sends 100000 back-to-back triggers and all triggers are distributed to all m consumers. This means that TRG needs to process $n \times 10^5$ triggers and deliver $n \times m \times 10^5$ triggers. On the left-hand side of Figure 5, we illustrate an example case where n=3 producers A, B and C each send 100000 triggers, with trigger IDs 51, 52 and 53, respectively, to m=4 consumers (labelled I, II, III, IV). That is, in this particular scenario, each of the four consumers will receive 300000 triggers from TRG.

Table 2 shows the number of delivered triggers with average processing times in milliseconds for each trigger received by TRG from the producers in Test Case 1. In this case, only the number of consumers has a significant effect on the processing time of each trigger. This indicates that TRG can cope with several registered producers even when there is no subscribed consumer from certain producers. Moreover, the average trigger processing time is only few milliseconds per subscribed consumer in this stress test of the prototype implementation.

Since there are several possible scenarios about how triggers are distributed between producers and consumers we made also a Test Case 2 setup, illustrated in the right-hand side of Fig. 5, where each consumer has only one dedicated producer. This means that TRG needs to process $n \times 10^5$ triggers and deliver $m \times 10^5$ triggers. If there are more producers than consumers, triggers will be distributed evenly between the available consumers. As mentioned above, all tests were made using a C++ implementation of TRG with a web service interface towards producers and consumers. We used a laptop with an Intel Pentium M 1.70 GHz PC with 1 GB RAM, running FreeBSD release 6.1 in the tests reported in this subsection.

Figure 6 shows the total processing time of Test Case 1, with and without employing the TRG filtering mechanism. It can be seen that when the number of the consumers and producers increases, so does the total processing time. This is expected since the number of processed triggers is increasing when adding more consumers and producers. The costs of adding consumers and producers are both linear. But the cost of adding consumers is greater than the cost of adding producers. For example when comparing the calculated slope $k = \Delta y/\Delta x$ of the curves of total processing time, with and without filtering, we see that the processing time increases faster the more consumers are introduced (slope of the curve with one consumer k = 177,6 and with 5 consumer k = 454,9 in Test Case 1 without filtering), this can be explained as a cost of the duplication of triggers because the number of triggers that have to be duplicated and delivered to consumers increases when adding more consumers. Anyhow this does not increase the average processing time of one trigger. The number of producers has also effect to the total processing time, but not as much as the number of consumers.

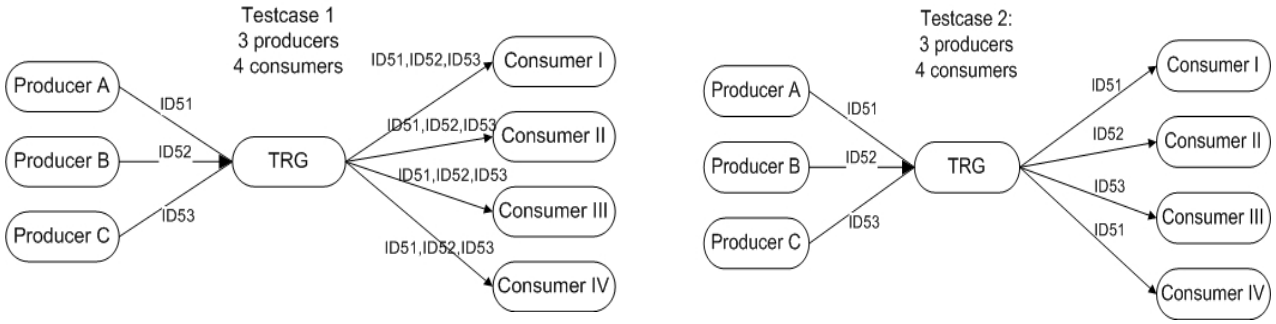

Figure 5. Triggers in Test Cases 1 (left) & 2 (right).

Table 2. Total number of delivered triggers and average processing time (in ms) per trigger in Test Case 1.

	Number of Producers				
Number of Consumers	1	2	3	4	5
1	100k, 1.7 ms	200k, 1.7 ms	300k ,1.8 ms	400k, 1.7 ms	500k, 1.8 ms
2	200k, 2.3 ms	400k, 2.5 ms	600k, 2.4 ms	800k, 2.5 ms	1000k, 2.4 ms
3	300k, 3.2 ms	600k, 3.2 ms	900k, 3.2 ms	1 200k, 3.1 ms	1500k, 3.3 ms
4	400k, 3.7 ms	800k, 3.8 ms	1200k, 3.8 ms	1600k, 3.8 ms	2000k, 3.8 ms
5	500k, 4.5 ms	1000k,4.6 ms	1500k, 4.7 ms	2000k, 4.7 ms	2500k, 4.5 ms

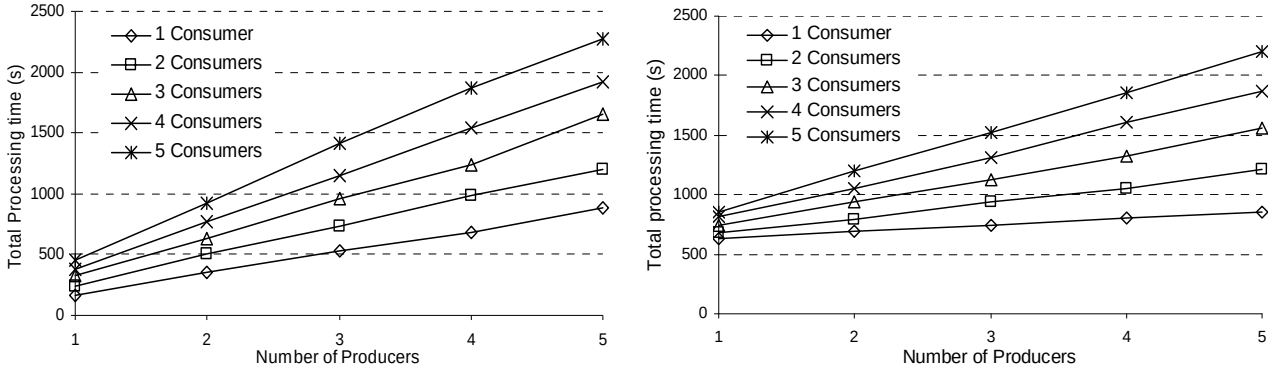

Figure 6. Total processing time in Test Case 1 without (left) and with (right) filter processing.

We also evaluated the cost of using the filtering function of TRG. With Test Case 1 we had all five producers registered and each one was sending 100000 triggers. It follows that TRG was receiving total of 500000 triggers during the test. The right-hand of Figure 6 shows the total processing times when the filtering mechanism was used. When there is one producer, the triggers from the other four producers are filtered away, and the triggers from the sole producer are duplicated and delivered to all four consumers. In the case with two producers the triggers from three producers are filtered away, and so on. The results show that it takes more time to process all triggers but this is not caused by the filtering mechanism itself. When comparing the total processing times, in the case where triggers from 1 producer are delivered to consumers in Figure 6, the total processing time is increased when the filtering mechanism is used, but this is because now there are five times more triggers received by TRG than in the case without the filtering mechanism, since all five producers are sending 100000 triggers all the time during the test. When the filtering mechanism is not used, the number of producers is controlled by making a new registration per producer.

To further quantify system behaviour when filtering is employed, we consider Test Case 2. When evaluating the filtering function in Test Case 2, each consumer had a filtering rule that was true for all triggers, allowing the distribution of all triggers to the subscribed consumers. By having this "receive all triggers" rule we were able to test the effect of the filtering mechanism, since every time a trigger is produced, TRG needs to run the filtering code before disseminating the trigger to consumers even though none of the triggers are in practice going to be filtered away. The purpose was to test the effect and cost of running the filtering function. The TRG filtering mechanism per se does not have a significant effect on the overall processing time, especially when compared to the effect of increasing the number of consumers. When comparing the processing times in Test Cases 1 (Table 2) and Test Case 2 (Figure 7) we see that the duplication of each trigger to every consumer, needed in Test Case 1, increases processing times. In Test Case 2, when the number of producers and consumers are equal, the difference of the processing times can be measured in microseconds, since now there is no need to duplicate triggers.

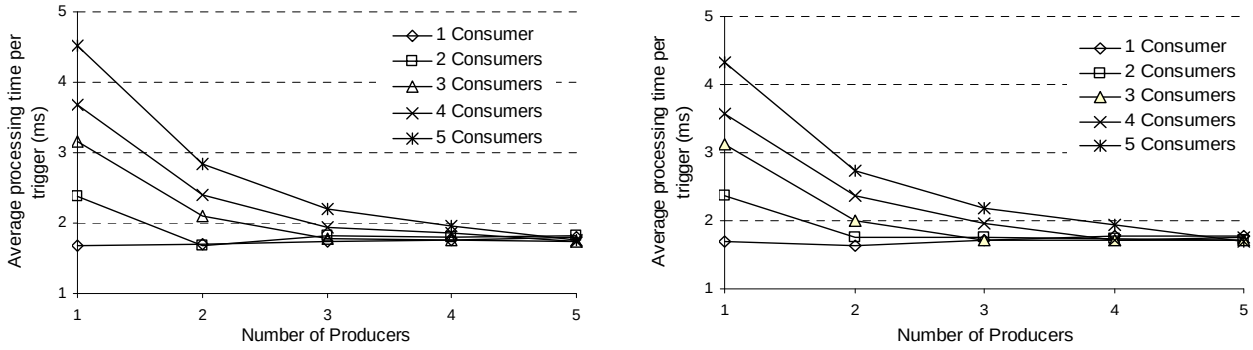

Figure 7. Average processing time in Test Case2 per trigger without (left) and with (right) filtering.

The test and evaluation cases presented in this Section showed that it is in fact the duplication of triggers and number of messages that have a biggest effect on the processing time of triggers. It can be seen in Figure 7, for n = 5 producers and m = 5 consumers, that the processing time does not depend on the number of consumers. There is no duplication of triggers in this case either.

The feasibility of using TRG to process, filter and disseminate a very large amount of triggering events was shown in practice. Each case showed that the processing time of one trigger does not increase, even when processing a huge amount of triggers. Although the stress-test cases are clearly unrealistic, they demonstrate that using TRG does not cause any major delays to handover times. On the contrary, TRG enables handover decision making mechanisms to react more rapidly and to larger set of events. It is also important to note that the TRG filtering mechanism does not have a major effect to processing times and this allows the handover decision making mechanisms to react faster to relevant events. Although the filtering mechanism can be used for the pre-decision about which events are to be collected, the handover decision per se is left to separate mechanisms with the decision algorithm. It was also shown that the cost of adding more consumers and producers increase processing times linearly and the cost of using filtering has only a marginal effect on the processing times. Of course the more triggers there are, the more total processing time is needed for processing and disseminating all triggers. However, by implementing grouping and classification of triggers [4] and having mechanism, e.g. in the TRG source for prioritizing trigger delivery which allows critical triggers to be processed and distributed faster, TRG is ready to process the triggering events.

6. Related work and Discussion

Previously published work [7−9] shows the benefits of using event information, for example, to proactively perform a handover in order to maintain QoS levels. Our goal is to define a framework that supports the event collection and processing, and trigger distribution possibly from hundreds of different sources. We concur with Vidales *et al*. [7] that in heterogeneous network environments several sources of events and context information should be consulted in order to achieve seamless connectivity and develop swift mobility management mechanisms. Furthermore, earlier work in other event/notification systems [22,23], which introduces mechanisms on how to implement such systems, along with the evaluated event generation cases is very encouraging and complimentary to our effort in defining TRG as a specialized notification system for mobility-related events which originate from the entire protocol stack.

The IEEE 802.21 Media Independent Handover (MIH) Services [12] working group is standardizing an information service that will facilitate media independent handovers. The scope of the IEEE 802.21 standard is to provide a mechanism that provides link layer intelligence and other related network information to upper layers to optimize handovers between heterogeneous IEEE 802 systems and facilitates HOs between IEEE 802 and cellular systems. IEEE 802.21 assists in HO Initiation, Network Selection and Interface Activation. The purpose is to enhance the experience of mobile device users. The standard supports HOs for both stationary and mobile users. For mobile users, HOs are usually needed when the wireless link conditions change. For stationary users, HOs are needed when the surrounding environment changes. Both mobile node and network may make decisions about connectivity. The HO may be conditioned by measurements and triggers supplied by the link layers on the mobile node. The IEEE 802.21 standard defines services that enhance HO between heterogeneous access links. Event service, Command service and Information service can be used to determine, manage and control the state of the underlying multiple interfaces. By using the services provided by MIH Function users, like Mobile IP, are able to better maintain service continuity, service adaptation, battery life conservation, networks discovery and link discovery. MIH Function also facilitates seamless handovers between heterogeneous networks.

The IEEE 802.21 Event Service has common characteristics with our TRG design but does not prescribe a particular implementation and stops short of allowing upper-layer entities to provide events that can drive a HO. It was also impossible to compare the performance of the implementations since no MIH implementation was available when these tests were performed. Our approach emphasized standardized ways for consumers to receive trigger from a variety of sources. TRG framework is also fully implemented and tested in a laboratory environment with several operating systems. Easy application registration to TRG permits them to get the information they want from different sources. Event generation, on the other hand, is by its very nature a distributed process and, without a central agent, all sources and consumers are forced to create a fully meshed topology. By introducing TRG, event collection becomes straightforward and trigger distribution standardized. That is why we propose that instead of using only the services provided by the IEEE 802.21 MIH functionality future mobile systems should use also TRG alongside 802.21 services. IEEE 802.21 can be, for example, the source entity that provides the lower layer information to TRG.

7. Concluding Remarks and Future Work

This paper presented a novel TRG framework for managing mobility-related triggers and its functionalities for collecting information from various event sources origi-

nating not only from the lower layers of the protocol stack (physical, data link, and network), but also from the upper layers and processing the collected events in a standardized trigger format. By using the defined mechanism, TRG framework enables easy and efficient use of cross-layer and cross-domain information. This framework was implemented and evaluated by performing tests in a real environment with several operating systems (Linux, FreeBSD, Windows, Linux Familiar for the PDA and Maemo Linux for the Nokia tablet) to prove its robustness and measure its performance.

The TRG framework experimentations with the performance test and evaluations showed that the implemented TRG functionalities are very promising. TRG can run efficiently in small device with very limited processing power and can enable lossless session handovers between devices. Stress tests showed that the TRG filtering mechanism does not cause delay for processing time and TRG can be used to filter and disseminate large numbers of triggers from several information sources.

Our Triggering management framework is currently integrated with Mobile IP [1] and HIP [2] protocols and is also a part of the Ambient Networks Architecture [15] and prototype as discussed in [16−18]. TRG and MIP integration with the use of network information a.k.a cascaded triggering presented in [24] showed the benefits of using TRG for the Mobile IP in the case when networks will be congested. HIP integration with TRG and test evaluations presented in [25] showed as well that TRG processing have only a small factor (less than 9%) to the total; trigger collection, processing and dissemination process.

Next steps will be to run a complete test with these integrated mobility protocols in real heterogeneous environments with the WLAN, 3G/HSDPA, WiMAX access technologies. Tests will benefit of the cascaded functionality of TRG when TRG can be located both at the terminal and network side as discussed in [24]. For example, TRG sources at the network side can monitor the network capacity load and other QoS metrics in overlapping networks and based on this information, the network side TRG can send triggers to the terminal initiating or even forcing a vertical handover.

While the tests and evaluations are made in a real testbed environment, a simulation environment will be built to fulfil the tests and analysis of the performance and scalability. In a forthcoming study we will also map the trigger management framework with the recently finalized standard by the IEEE 802.21 working group [12].

8. References

[1] D. Johnson, C. Perkins, and J. Arkko, "Mobility support in IPv6," Series Request for Comments, No. 3775. IETF, June 2004.

[2] A. Gurtov, "Host Identity Protocol (HIP): Towards the secure mobile Internet," Wiley and Sons, pp. 328, June 2008.

[3] J. Mäkelä, K. Pentikousis, M. Majanen, and J. Huusko, "Trigger management and mobile node cooperation," in M. Katz and F. H. P Fitzek (Editors), Cognitive wireless networks: Concepts, methodologies and visions inspiring the age of enlightenment of wireless communications, Springer-Verlag, pp. 199−211, 2007.

[4] J. Mäkelä and K. Pentikousis, "Trigger management mechanisms," Proceedings of the Second International Symposium on Wireless Pervasive Computing, San Juan, Puerto Rico, pp. 378−383, February 2007.

[5] P. Prasad, W. Mohr, and W. Konhuser, "Third generation mobile communication systems," Boston, MA, Artech House Publishers, 2005.

[6] J. Eisl (Editor), "Ambient networks D4.2: Mobility architecture & framework," EU-project IST-2002-507134-AN/WP4/D4.2, 2005.

[7] P. Vidales, J. Baliosian, J. Serrat, G. Mapp, F. Stajano, and A. Hopper, "Autonomic system for mobility support in 4G networks," IEEE JSAC, Vol. 423, No. 12, pp. 2288−2304.

[8] S. Ishihara, K. Koyama, G. Miyamoto, and M. Kuroda, "Predictive rate control for realtime video streaming with network triggered handover," IEEE WCNC, Vol. 3 No. 13−17, Las Vegas, Nevada, USA, pp. 1335−1340, 2005.

[9] H. Chaouchi and P. Antunes, "Pre-handover signalling for QoS aware mobility management," International Journal of Network Management, Vol. 14, No. 6, pp. 367−374, 2005.

[10] E. Casalicchio, V. Cardellini, and S. Tucci, "A layer-2 trigger to improve QoS in content and session-oriented mobile services," Proceedings of 8th ACM international Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Montral, Quebec, Canada, pp. 95–102, 2005.

[11] C. Kappler, P. Mendes, C. Prehofer, P. Pöyhönen, and D. Zhou, "A framework for self-organized network composition," Lecture Notes in Computer Science, No. 3457, Springer, pp. 139−151, 2005.

[12] IEEE Std 802.21™-2008, IEEE standard for local and metropolitan area networks-Part 21: Media independent handover services, IEEE, January 2009.

[13] R. Giaffreda, K. Pentikousis, E. Hepworth, R. Agüero, and A. Galis, "An information service infrastructure for Ambient Networks", Proc. 25th International Conference on Parallel and Distributed Computing and Networks (PDCN), Innsbruck, Austria, February 2007, pp. 21−27.

[14] OASIS eXtensible Access Control Markup Language, OASIS specification, Available: http://www.oasis-open.org/committees/xacml/.

[15] N. Niebert, A. Schieder, J. Zander, and R. Hancock (Eds.), "Ambient networks; co-operative mobile networking for the wireless world," Wiley & Sons, 2007.

[16] P. Pääkkönen, P. Salmela, R. Aguero, and J. Choque, "An integrated ambient networks prototype," Proceedings

SoftCOM 2007, Split, Croatia, pp. 27−29, September 2007.

[17] C. Simon, R. Rembarz, P. Pääkkönen, et al., "Ambient networks integrated prototype design and implementation," Proceedings 16th IST Mobile Summit, Budapest, Hungary, pp. 1−5, July 2005.

[18] K. Pentikousis, R. Agüero, J. Gebert, J. A. Galache, O. Blume, and P Pääkkönen, "The ambient networks heterogeneous access selection architecture," Proceedings First Ambient Networks Workshop on Mobility, Multiaccess, and Network Management (M2NM), Sydney, Australia, pp. 49−54, October 2007.

[19] J. Makela, R. Aguero, J. Tenhunen, V. Kyllönen, J. Choque, and L. Munoz, "Paving the way for future mobility mechanisms: A testbed for mobility triggering & moving network support," Proceedings 2nd International IEEE/Create-Net Tridentcom, Barcelona, Spain, March 2006.

[20] E. Tuulari and A. Ylisaukko-oja, "SoapBox: A platform for ubiquitous computing research and applications," Lecture Notes in Computer Science 2414, Pervasive Computing, Zurich, CH: Springer, pp. 125−138, August 2002.

[21] C. Pinho, J. Ruela, K. Pentikousis, and C. Kappler, "A protocol for event distribution in next-generation dynamic networks," Proceedings Fourth EURO-NGI Conference on Next Generation Internet Networks (NGI), Krakow, Poland, pp. 123−130, April 2008.

[22] C. H. Lwi, H. Mohanty, and R. K. Ghosh, "Causal ordering in event notification service systems for mobile users," Proceedings of International Conference on Information Technology: Coding and Computing (ITCC), Vol. 2, pp. 735−740, 2004. Conference Distributed Computing Systems Workshops (ICDCS 02), pp. 639−644, 2002.

[23] H. A. Duran-Limon, G. S. Blair, A. Friday, T. Sivaharan, and G. Samartzidis, "A resource and QoS management framework for a real-time event system in mobile ad hoc environments," Proceedings of International Workshop Object-Oriented Real-Time Dependable Systems (WORDS), pp. 217−224, 2003.

[24] M. Luoto and T. Sutinen, "Cross-layer enhanced mobility management in heterogeneous networks," Proceedings of International Conference on Communications, Beijing, China, May 2008.

[25] P. Pääkkönen, P. Salmela, R. Aguero, and J. Choque, "Performance analysis of HIP-based mobility and triggering," Proceedings of IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, 2008.

