# Energy-Efficient Mobile Middleware for SIP on Ubiquitous Multimedia Systems

Felipe Garcia-Sanchez, Antonio-Javier Garcia-Sanchez, and Joan Garcia-Haro

Department of Information and Communication Technologies. Technical University of Cartagena. Campus Muralla del Mar, Plaza del Hospital, 1, 30202. Cartagena, Spain
{felipe.garcia, antoniojavier.garcia, joang.haro}@upct.es

**Abstract.** Currently, different technologies provide solutions to successfully resolve the continuity of the service of a mobile host. The employment of a middleware is a common approach to face this issue, supporting also heterogeneity. In addition to the usual requirements of bandwidth, delay, or CPU load, multimedia applications have special features that are still being studied in a mobile scenario, as the intensive energy consumption. Due to all them, current middleware specifications and services do not directly apply. The Session Initialization Protocol (SIP) is the most frequently employed tool to satisfy mobile multimedia requirements. In this paper, we present and validate the design of a service based on middleware and Publish/Subscribe schemes that minimizes the control data exchange introduced by SIP, reducing the energy consumption in the handoff process. It considers all the relevant factors in the audio/video transmission done by different middleware entities, and it supports their profiles (provider or consumer). Using an intermediate element, a modified event server, decouples the communication edges, adding some improvements: asynchronous connection establishment and extension of mobility to all the terminals involved in the connection. The performance offered by the service proposed is also evaluated, and their results further discussed.

**Keywords:** SIP, middleware, streaming, handoff, publish/subscribe.

## 1 Introduction

Current wireless technologies are evolving to allow the communication among various equipments, supporting the concept '*everyone, everywhere and every time*'. Personal Digital Assistants (PDA), mobile phones, or laptops are examples of terminals that may use these technologies to access all the types of data.

Services based on middleware (software disposed between the applications and the operating system) are widely employed in multimedia and Audio/Video applications, highly distributed and heterogeneous environments. Many interactions among middleware technologies and general-purpose applications may be found in the open literature [1], as well as the usual services provided, e.g. service discovery, QoS parameter negotiation, or security. We consider that higher layer communication protocols must be aware of modifications of the environment as part of the global communication process [2] [3]. Examples of this issue are collaborative environments and embedded systems whose possible interconnection (wireless and mobile) through mobile ad-hoc networks (MANET) is not currently normalized or, at least, not clearly focused. In reference [4], the interesting topic of middleware in mobile networks is outlined.

All the expected capabilities may be designed through middleware, traditional ones as CORBA (*Common Object Request Broker Architecture*) or new approaches as JXTA, uPnP or OSGi, because of their platform independence, the multiple services supported, the interoperability and programming coherence, and the signaling and connection services included in the middleware layer instead of in the application one. However, in this context, mobile multimedia and video applications are not purely developed for their special requirements in throughput, delay, QoS features or handoff energy saving. This is because middleware specifications and services (i.e. standard A/V Streaming [5]) which provide A/V connectivity are not oriented to aspects such as access integration or energy aware of different terminals. For instance, a mechanism to optimize the handoff scheme of an A/V Streaming application when the service is down or when a device changes its location is not included. In these systems, energy consumption is an important issue for mobile terminals, raising the highest values in the handoff process.

Therefore, the heterogeneity and volatility of ubiquitous environments demand new middleware approaches to support mobile multimedia applications, offering generic access. SIP (*Session Initialization Protocol*) [6] [7] protocol is the most extended signaling system to connect and manage services and applications. In this paper, we address how current middleware technologies may face mobile multimedia applications. In particular, we focus on how a multimedia service programmed via middleware may be restored when a terminal changes its place and/or its access to the wired network. With this aim, we propose a system based on generic middleware that supports the SIP signaling operation.

Our contribution consists of developing a communication protocol and a software procedure to guarantee the connection and re-connection of multimedia applications, providing a mechanism to retrieve the signaling information lost in the process and reducing the energy consumption. To implement it, we employ the publish/subscribe paradigm, which helps to solve additional issues, such as scalability or localization, by means of decoupling edges, avoiding the limitations of the client-server scheme in SIP. Other partial objectives are to reduce the signaling information, to decrease the connection establishment delay, and to build an architecture allowing the mobility of providers and consumers. This last goal is applied in symmetric services as direct streaming between final users. We also conduct an evaluation testbed to measure the connection delay incurred in a re-connection between two terminals having into account event propagation in an infrastructure network. The results obtained are discussed and compared with other connection strategies.

The rest of the paper is organized as follows. Section 2 summarizes current mobility services and the protocols and procedures employed for traditional non-middleware applications. Sections 3 and 4 present the general architecture used as workbench and the protocol proposed. Performance evaluation results are obtained and discussed in section 5. Finally, section 6 concludes the paper.

## 2 Related work

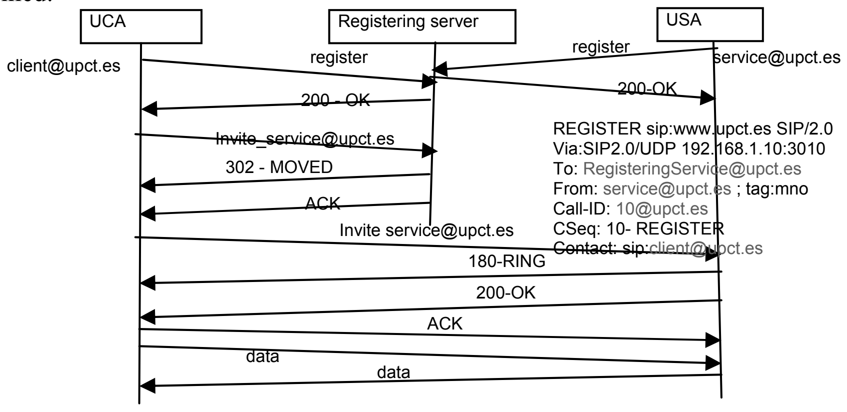
### 2.1 The Session Initialization Protocol subsystem

SIP is a specification [6] that defines the procedure and protocol to support audio/video transmissions in mobile environments. The major SIP features for multimedia applications are the following ones:

- The communication between user A and user B is session-oriented.
- The communication may be in real-time or differed time.
- The system may support video and multimedia applications that require Quality of Service (QoS).
- The identification of users and services is done through a URI (*Universal Resource Identifier*) that facilitates the usability of services. It may be coded in XML (*eXtensible Markup Language*).

The system consists of two different client classes: the User Clients Agents (UCAs) and the User Server Agents (USAs). Messages follow a request-reply structure and implement the current services such as registering and addressing servers, proxies, etc. (see fig. 1) [8].

SIP is implemented by different basic operations like *invite*, *ack*, *bye*, *cancel*, *register*, and *options*. All of them are required to establish the point-to-point communication in a mobile environment, using return codes similar to the http (*Hyperterminal Transmission Protocol*) protocol. Nevertheless, the computing mechanism is not defined.



```
REGISTER sip:www.upct.es SIP/2.0
Via:SIP2.0/UDP 192.168.1.10:3010
To: RegisteringService@upct.es
From: service@upct.es ; tag:mno
Call-ID: 10@upct.es
CSeq: 10- REGISTER
Contact: sip:client@upct.es
```

**Fig. 1**. Resume SIP connection establishment procedure.

### 2.2 Previous middleware mobility works

Currently, the major contributions for SIP implementations are related to minimize the time connection between devices, but omitting the effect on the energy consumption in this process. In the field of mobile devices, the most usual restrictions are both the service connectivity and the energy lifetime. Therefore, the latest interesting works are: (i) the contribution of Zhang et al. [9] that provides a seamless SIP scheme to reduce the time-delay connection and (ii) optimizations of protocols as hierarchical mobile SIP (HMSIP) [10] or RTP (*Real Time Protocol*). Both cases do not take care about energy consumptions and are not oriented to multimedia requirements.

In the field of mobile middleware technology and ubiquitous networking, several efforts have been done to provide multimedia connectivity. We emphasize the latest works of Gehlen et al. [11] and Su et al. [12]. Both provide mechanisms to design mobile devices applications, but without specifications of the signaling mechanism. In the first one, the Publish/Subscribe model [2] is employed to provide context-information about the communication and other subscribed services to the application. Moreover, they use several intermediate middleware services. In the second case, the multimedia system divides users among producers and consumers and a collaborative scenario is developed through events communication. It already performs events written in XML language and offers as implementation example: a SQL database server and clients as middleware agents. The recent work [13] implements SIPHOC (SIP protocol over a MANET network), offering performance evaluations about the signaling and connection delay over different routing protocols and to a wired network.

To the best of our knowledge, there is not any document or paper related to the operation of SIP signaling in a multimedia implementation based on middleware. The closest documents are related to publish/subscribe schemes to facilitate mobility to middleware applications. However, protocols SIP and SDP (*Session Description Protocol*) are the most widely signaling protocols used for mobility in wireless networks. Therefore, our work is aimed at enhancing the middleware operation for wireless environments through XML event propagation, including the SIP operations. Our system includes the following features:

- A/V Streaming connection establishment through event propagation, using the Publish/Subscribe model.
- Mobility is facilitated in both edges (provider and consumer).
- Optimizing the number of remote invocations, to reduce the messages transmission and, consequently, the energy consumption.

## 3 Mobile Architecture

The Event Service (ES) [14] specification is the simplest tool to propagate events in a middleware environment, employed as starting point in the signaling architecture. However, it does not offer the singular features required by a streaming transmission. In fact, the modification of ES must support the following requirements:

- It must provide the connection and create the channels for the flow transmission.
- The edges, the video providers, and the consumers, must have the same interface, and their behavior for mobility has to be the same. The streaming provider and the consumer may be located in any place of the network.
- The system must be scalable, to generalize the architecture for a global network, or through Internet.
- For our implementation, we define these new elements: *Bidirectional-Event Services* (B-ES), *Hybrid* interfaces for all the edges, and using a federated policy of events transmission (see fig. 2).

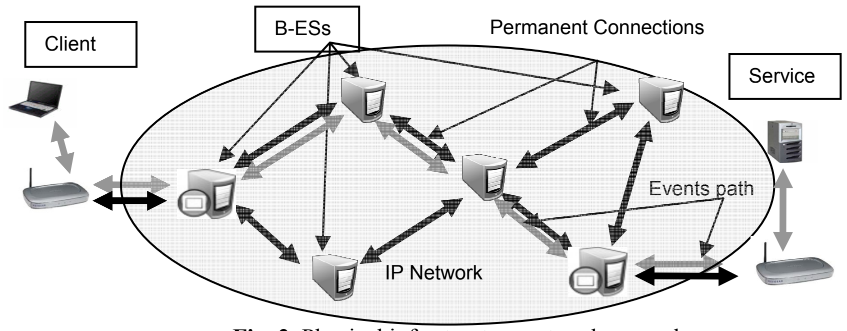


**Fig. 2**. Physical infrastructure network example.

The design considers all the terminals as the same type of elements, they are *hybrids*. Any edge may send and receive events, and, at the same time, they may choose their functionality of sender of a video/audio flow or receiver, or even both profiles. This section shows the architecture at high-level description. In particular, fig. 3 presents the architecture and the interactions among the different elements.

### 3.1 Hybrid interfaces

Usually, the different edges of the video or multimedia applications are divided into providers (or senders) and consumers (receivers). The *Hybrid* interface presents different features providing the classes and methods required for both, providers and consumers, and open methods to add future properties. Now, instead of the sender and receiver edges, the edges are *Hybrid* with a different task but with the same interface. This *Hybrid* interface facilitates the connection between an A/V Stream application (in general, any application that uses it as interface) and a B-ES.

This interface implements the connection to the B-ES through the *request* method. It sends a packet (XML format) that includes different aspects such as the identification (URI) of the requested edge, the type of connection, transport protocol to use for the flow, etc. This information is useful not only for the connection itself, but also for locating the edge. The headers identify different aspects, such as origin machine, port, etc. All this information is saved by the B-ES in a database.

The additional functions are focused on managing the events and on receiving the connection responses. They provide the methods to send, receive, and manage events, and for creating A/V flows channel. The A/V channels are created when both edges wanting to intercommunicate are connected to the same B-ES.

### 3.2 Bidirectional-Event Service

The B-ES is a service presented in reference [14]. It may be developed on different programming paradigms such as JXTA, uPnP, etc., according to the applications requirements, supporting XML events. This service propagates the events from any of the *Hybrid* connected to it to the *Hybrids* included in their respective *Event Channel* [14] where the original *Hybrid* is connected. The structure of the B-ES is distributed according to the next levels:

*Stream Dispatcher* or *event deliverer*. It delivers the events that arrive, and selects their destination. It composes the different *Event Channels* for the event propagation (each service needs at least an *Event Channel* created inside the B-ES), and it orders the creation of the A/V Flow through the respective interfaces. Moreover, it has access to the two different databases: one dedicated to the usual operation where the different *Hybrids* are registered with their features and destination edges and another one specially devoted to the mobility service, where the different *Hybrid* mobile terminals are registered with their current location.

*Hybrid Administrators* are required to manage the different connections of the *Hybrids* (edges) to the B-ES, as the typical specification. They add one more task: the order to create the A/V flows.

Finally, the *ProxyHybrid* Interface is the tool to connect the different *Hybrids*, or, for federated structures, the connection of other B-ES. It should be noted that the traditional policy of Publish/Subscribe based on Suppliers and Consumers is changed for a point-to-point structure based on *Hybrids*. Moreover, the *ProxyHybrid* has to receive, send, and propagate events. The definition of the *ProxyHybrid* allows to include it with other traditional interfaces, supplier and consumer if necessary.

## 3.3 Connection establishment and static operation

This section presents the general operation of the B-ES in coordination with the edges that require the service. Fig. 3 represents the usual sequence of connection setup. This process obtains energy efficiency to simplify the number of invocations, decreasing transmissions, and assuming that the message transmission is highly more costly than the CPU processing, and reception is less costly than transmission.

The edges must execute the method *request* that sends a packet with their own and the desired destination (*SIP register*). This message includes the original location of the edges, communication port, etc. This information is saved in the *Location* database of the B-ES. Moreover, the destination information is saved in the *Hybrid* database. This information is maintained until a *destroy* (*SIP bye*) event happens. When two edges are connected to the same B-ES (or for federated events to two interconnected B-ES) the B-ES executes two methods, one *accept* (*SIP invite*) for *Hybrid*, and a *complete* one for the second *Hybrid*. Then, both edges know that the event channel is open, and they may send events to the other edge. In the messages associated to *accept* and *complete*, the information includes the references of the destination edge of the A/V Stream Flow. In the example of Fig. 3, the packet for *Service* includes the information of *Client*, and for *Client* includes the information of *Service*. Finally, the one that receives the *accept* (*Service*) executes the active connection.

The rest of the connection establishment of the A/V stream flow is independent of the event transmission and propagation through the B-ES. Therefore, from this moment the B-ES is limited to delivering the possible events between applications.

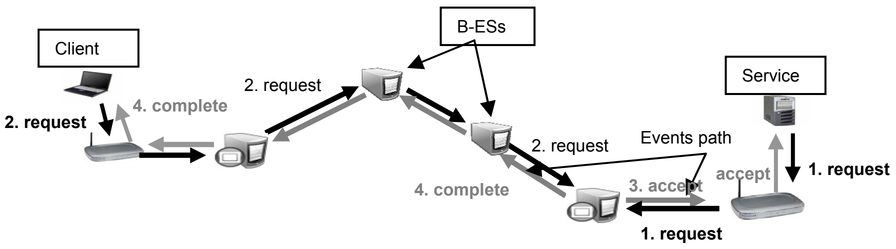


**Fig. 3**. Connection establishment scheme.

The event propagation is similar to the one provided for the traditional ES. However, an interesting point is that the event propagation is bidirectional, both *Hybrids* may send and receive events. This is possible thanks to the *ProxyHybrid* Class that listens to its connected *Hybrid* and to the *Event Channel* performed in the B-ES. Moreover the *Hybrid* components in the edges must pay attention to the events received from the *Event-Channel* and the events generated in their own edge. On account of that, a design condition is that only a push model is supported to guarantee the right attention to all the events. The pull model may be considered as a push event in the opposite direction.

## 3.4 Federated Bidirectional-Event Service

Although an alone B-ES node is enough to perform an end-to-end service, desired scalability requires a more complex configuration. The B-ES, as the primitive ES standard, may use centralized or federated architecture. In the federated architecture, different B-ES may be subscribed to other B-ES as common *Hybrids* and propagate connections and events to other edges or B-ES. The familiar structure of a federated ES may be observed in reference [15]. Basically, a B-ES uses the *ProxyHybrid* interface as a real *Hybrid* interface. It is able to propagate and receive events. It is similar to a supplier/consumer architecture where the *ProxySupplier* connects to the *ProxyConsumer* of the ES but changing the role of *ProxySupplier* as a *Hybrid*, and *ProxyConsumer* as *ProxyHybrid*. The connection of two Edges through a federated B-ES architecture is as follows:

1. Both edges connect to a different B-ES known by each one of them, and both subscribe for the other edge.
2. A B-ES (or both) sends an event to the rest of the B-ES network. The event is propagated through the network, arriving at the destination B-ES. If both events arrive at the remote B-ES, one operation is discarded when the time-to-live (configurable parameter of time of event living in the B-ES without response until it is discarded) of the event expires. The system offers the traditional features, such as control of event replications or event time-to-live.
3. The request packet of the event gives the information of the original edge for the A/V flow connection (URI), but the information of the message header comes from the subsequent B-ES. This provides the appropriate information to connect the A/V flow between Hybrids, and the information to propagate events through the federated architecture.

## 4 SIP using B-ES nodes

The upgraded design of the SIP protocol to the B-ES architecture is developed through the implementation of B-ES nodes that supports the SIP signaling and operation. The basic idea behind is to identify the tasks that the SIP registering server performs to adapt the methods required by the protocol to the B-ES working diagram. There are two initial conditions: the B-ES must be located in a fixed position that may be known through a simple call to a DNS (*Domain Name Service)* or similar service, and the disconnection of both edges *Hybrids* may not be performed simultaneously. In that case, the B-ES loses the references and the re-connection is not possible. Therefore, the system provides additional structures for the *Hybrid* interface:

- The *void identify (Hybrid peerHybrid)* method encapsulates the actions to generate the identification of a *Hybrid* previously connected to the B-ES (as *invite* does in SIP). It saves these identifications in the *Location* database and compares the URI provided by the *identify* operation with the stored ones in this database. When the comparison succeeds, the B-ES decides that this *Hybrid* is the same as the one connected before. Then, the following steps are done to manage the connection of the new flow and propagate the data events. In fact, *identify* is similar to *request*, adding the check and managing the connection of an event channel previously created.
- The *null* event. This event is managed between *Hybrid* and B-ES (not propagated) and aimed at confirming regularly the success of the connection between them when many events fail or events are not generated in a pre-established time or threshold.

To perform the SIP operation, the handoff process must be designed. Therefore, we distinguish two major cases, (1) when the *Hybrid* in movement receives the flow, and (2) when it transmits. As mentioned in section 2, we take advantage of the location discovery notification of the A/V flow rate sensitiveness to the movement, since rates are reduced (or they stop) when the location is close to change. Therefore, the location discovery of the case (1) may be obtained by this analysis. However, in the case of (2), the *Hybrid* does not receive the flow, and this task may not be implemented. Therefore, we propose two different mechanisms to resolve the location discovery in the source edge (using reactive strategies, applied in streaming services):

1. To regularly send *null* events from *Hybrid* to the B-ES, receiving ACK messages when the connection succeeds and no response otherwise.
2. To directly send the *null* events (they are sent by the B-ES when it does not detect activity for the *Hybrid)*. In this case, the B-ES acts as an active element.

In our implementation, we choose a combined solution: first, the sending *Hybrid* transmits *null* events when the last event exceeds a threshold, and secondly the B-ES is able to send *null* events when it does not detect activity. Finally, when the *Hybrid* sends consecutive *null* events without any response or it receives a reply from the B-ES, it starts the re-connection process.

## 4.1 Re-connection for the consumer mobility

The mobile behavior of the *Hybrid* that receives the flow may be denoted as the consumer mobility. It may be considered a typical SIP operation with a reactive handoff algorithm. This is the most extended case, with high relevance when the service is located in the wired network. The re-connection process is shown in fig. 4.

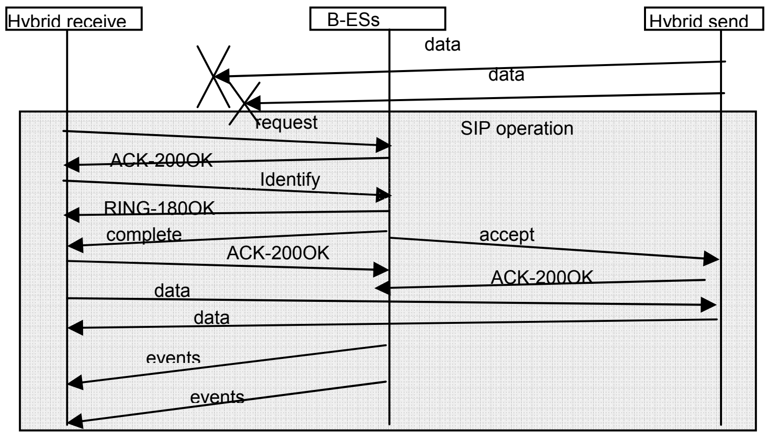


**Fig. 4**. Time diagram for the re-connection of consumers.

When the activity on the A/V flow decreases, the *Hybrid* decides the re-connection operation. Then, as every *Hybrid* that requires a service, it must send the packet *request* to the B-ES node. The next step, is to execute the *identify* method, sending its URI. At this point, the B-ES compares this URI with the stored ones in the *Location* database. Therefore, the B-ES carries out two tasks: (a) It performs the actualization in its own database and (b) It propagates the events queued in the server provided by the other edge representing the service to the moving *Hybrid*. When the events propagation is taking place, the edges may execute their end-to-end connection. Then, when the connection establishment and the events delivered are finished, the service is restored.

## 4.2 Re-connection for the supplier mobility

The mobile behavior of the *Hybrid* that sends the flow may be denoted as the supplier mobility. This case appears when the service source is in charge of a mobile host. The re-connection process is shown in fig. 5.

When the *null* events do not arrive to the B-ES, the *Hybrid* decides the re-connection operation. Then, it sends the packet *request* to the B-ES node. While the connection establishment to the B-ES is performed, the *Hybrid* sends the packet *identify*. The B-ES compares its URI information with the one stored in the *Location* database and determines the service identification. Then, it searches for the *Hybrids* connected to it and, by means of the *accept* and *complete* methods, it orders the connection establishment as explained above in section 3. Finally, the service may be supplied.

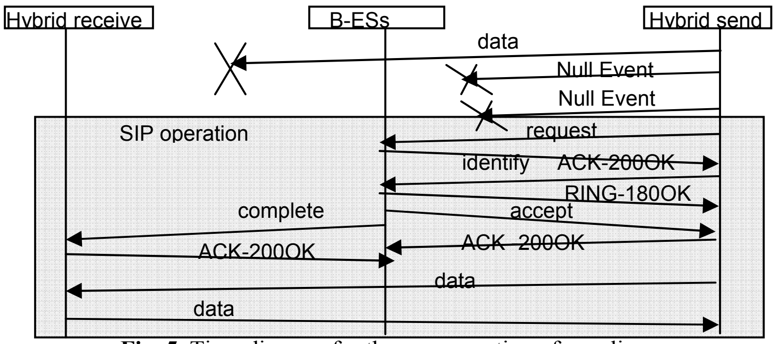


**Fig. 5**. Time diagram for the re-connection of supplier.

## 5 Performance evaluation

Our implementation is evaluated by simulation. Therefore, we develop a tool based on the OMNeT++ framework. It provides an appropriate environment to obtain several metrics of interest, in particular the connection delay and power consumption.

The evaluation is only performed for the consumer mobility, because the supplier mobility procedure does not offer a real estimation of the connection delay. In the supplier case, delays depend on the variable time to estimate the location change.

We simulate the scenario where a centralized B-ES node interconnects several (or different) *Hybrids* components. This scenario consists of a fixed B-ES node which the *Hybrids* must find inside a wired local area network (i.e. 100 Mbps Ethernet), and radio interfaces that implement an IEEE 802.11b (11 Mbps) network. Moreover, *Hybrids* are moving hosts with a constant speed *v*. The evaluation testbed is completed with an implementation of the SIPHOC protocol [13] connected to the wired network (one hop). All values are referenced to the native SIP procedure.

An important constraint in the wireless communication is that not all the A/V packets arrive in time, being discarded in that case. Therefore, the condition to disconnect and to initialize a re-connection process is imposed to 20 consecutive packets (frames for video case) lost (typically 1 sec.). The simulation reproduces two different cases: first, when the *Hybrids* change their location from the fixed network to the wireless one; and second, when moving from a wireless network to other wireless network, changing the IP of the mobile host. Additional parameters are the average arrival rate of events $\lambda = 0.5$ and an average rate of services lifetime of 0.0003 (mean service duration up to 15 minutes). Services are typical MPEG-4 video at a medium bitrate of 150 Kbps. Moreover, to measure the energy consumption, we select the power parameters of the *SDIO RW31 card* for PDA's: sleep mode 50 μA, reception 80 mA, transmission 138 mA and power supply 3.6 V.

We modify the range of different parameters: the number of *Hybrids* in mobile hosts connected to the arriving access point (bandwidth) and the packet (event) generation rate $\lambda$ are evaluated. The mobile speed is fixed to 5 meters per second with linear course for hosts. For the first scenario we obtain the average values for the connection delay presented in fig. 6 left. From these values, some considerations arise:

- Both implementations offer better performance than the SIP protocol as expected.

- The re-connection time in the B-ES case is slightly increasing to the SIPHOC protocol. The packet sizes (*request*) in the B-ES architecture are bigger, delaying the time connection, but the results are smaller than other multimedia middleware solutions (e.g. [15]).
- Energy consumption (Fig. 7 left) reveals that this implementation minimizes the power lost in the handoff process. Notice that when the number of events (signaling or handoff) is higher, the implementation B-ES performs best results in comparison to the other options, due to the smallest number of packets delivered.

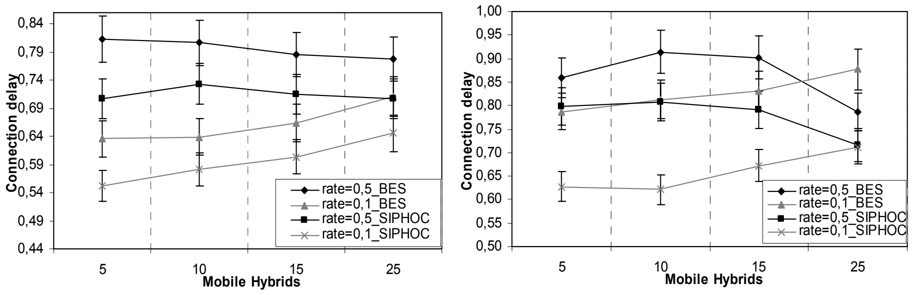


**Fig. 6**. Connection delay average referenced to native SIP (95% interval confidence, Batch Mean Method) in B-ES systems: left, wired-wireless handoff, right, wireless-wireless handoff.

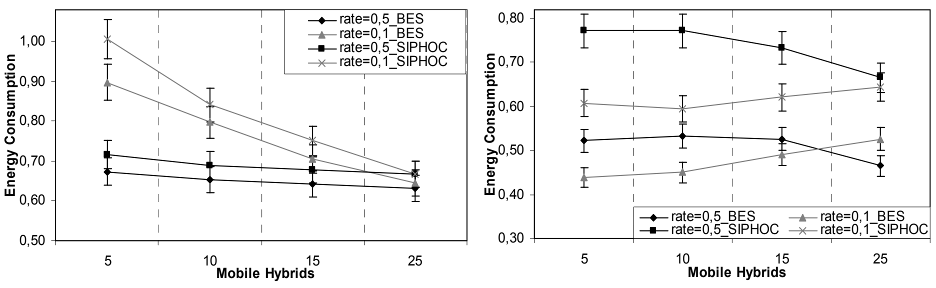


**Fig. 7**. Energy Consumption average referenced to native SIP (95% interval confidence, 30 min. simulation): left, wired-wireless handoff, right, wireless-wireless handoff.

For the second case (from wireless to wireless network) we obtain a mean value for the connection delay and energy consumption, that are shown in figs. 6-7 right, are higher than in the previous case. Regarding the results, the average value increases again with the number of *hybrids* and with the generation rate of events, but not in the same proportion (due to the bandwidth limitation of both wireless networks). The values associated to higher number of mobile hosts can be explained by the bandwidth saturation to manage too much simultaneous streaming and *hybrid* traffic.

## 6 Conclusions and future work

In this paper, a middleware architecture providing mobility of multimedia applications is presented. It is based on the SIP protocol and the publish/subscribe paradigm, propagating events. The integration of the SIP protocol, together with middleware computing extended by means of event propagation and optimizing the energy consumption is the most relevant contribution. Therefore, this implementation supports the SIP signaling system and manages mobility of multimedia nodes in a wireless network. It overcomes the limitations in the behavior of these services, including the reduction of overhead, flexibility, and separation between flow data and signaling. The energy consumption decreases thanks to the connection-oriented operation of the middleware layer and the minimization of the number of connections. The B-ES architecture decouples the communication edges and facilitates the connection among them. Although remote ends are designed in a similar way, mobility tasks are adapted according to the status of the stream flows (sent or received), thus optimizing, the SIP implementation. As future works, we propose to extend the study to large scale networks and mobility prediction and the event propagation system to support other data signaling such as those required in collaborative environments, for instance.

**Acknowledgment.** This work has been funded by the Spanish Ministerio de Educacion y Ciencia with the project grant TEC2007-67966-01/TCM and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia (Plan Regional de Ciencia y Tecnología 2007/2010)".

## References

1. Chambers D., et al.: Stream Enhancements for the CORBA Event Service. Proceedings of the 9th ACM Multimedia Conference, pp. 61-69, Ottawa, Canada (2001).
2. Carzaniaga, A., et al: Achieving scalability and expressiveness in an Internet-scale event notification service. Proc. Of ACM (PODC), pp. 219-227, Portland, OR (2000).
3. Sivaharan T., et al.: GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. OTM 2005, LNCS vol. 3760, pp. 732–749, (2005).
4. Mandato D., et al.: CAMP: A Context-Aware Mobility Portal. IEEE Communications Magazine, 40 (1), 91--97, (January 2002).
5. Mungee S., et al..: The design and performance of a CORBA Audio/Video Streaming Service. IEEE Proc. of the Hawaiian Int. Conf. is System Science, pp. 8043-8059, (2001).
6. IETF Draft: Session Initialization Protocol. RFC. 3261, 2002.
7. Kalmanek C., et al..: A Network-Based Architecture for Seamless Mobility Services. IEEE Communication Magazine 44 (6), 103--106, (June 2006).
8. Huang C.M., et al.: A Novel SIP-Based Route Optimization for Network Mobility. IEEE Journal of Selected Areas in Communication 24 (9), 1682--1691 (2006).
9. Zhang J., et al.: A SIP based Seamless-handoff (S-SIP) Scheme for Heterogeneous Mobile Network. IEEE WCNC Proceeding, pp. 3949-3954, Hong Kong (2007).
10. Chahbour F., et al.: Fast Handoff for Hierarchical Mobile SIP Networks. Proc. World Academy of Science, Engineering and Technology, vol. 5, pp. 34-39, Czech R. (2005).
11. Gehlen, et al.: A Mobile Context Dissemination Middleware. ITNG, pp. 165-170, (2007).
12. Su X., et al.: Middleware for Multimedia Mobile Collaborative System, Proc. of 3rd IEEE Wireless Telecommunication Symposium, pp. 112-117, USA (2004).
13. Suedi P., et al.: SIPHOC: Efficient SIP Middleware for Ad Hoc Networks. ACM/IFIP/ /USENIX 8th Int. Middleware Conf., LNCS 4834, pp. 60-79, Newport B., USA (2007).
14. Garcia-Sanchez F., et al: A CORBA Bidirectional-Event Service for Video and Multimedia Applications. CoopIS/DOA/ODBASE, LNCS 3760, pp. 715-731, (2005).
15. Caporuscio M., et al.: Design and Evaluation of a Support Service Publish/Subscribe Applications. IEEE Trans. on Software Engineering 29 (12), 1059--1071, (2003).