Distributed urban traffic applications based on CORBA event services

S.L. Toral* and D. Gregor

Department of Electronic Engineering, E. S. Ingenieros,
University of Seville,
Avda. Camino de los Descubrimientos s/n, 41092, Seville, Spain
Fax: +34-954487373
E-mail: toral@esi.us.es
E-mail: derlis.gregor@gmail.com
*Corresponding author

M. Vargas

Department of Automation and Systems Engineering,
University of Seville,
Avda. Camino de los Descubrimientos s/n, 41092, Seville, Spain
Fax: +34-954487373
E-mail: vargas@esi.us.es

F. Barrero and F. Cortés

Department of Electronic Engineering, E. S. Ingenieros,
University of Seville,
Avda. Camino de los Descubrimientos s/n, 41092, Seville, Spain
Fax: +34-954487373
E-mail: fbarrero@esi.us.es
E-mail: fcortes1@us.es

Abstract: Intelligent transportation systems (ITS) in urban environments are based today on modern embedded systems with enhanced digital connectivity and higher processing capabilities, supporting distributed applications working in a cooperative manner. This paper provides an overview about modern cooperative ITS equipments and presents a distributed application to be used in an urban data network. As a case example, an application based on an embedded CORBA-compliant middleware layer and several computer vision equipments is presented. Results prove the feasibility of distributed applications for building intelligent urban environments.

Keywords: intelligent transportation systems; ITS; event services; embedded systems; ambient intelligence; AmI; CORBA.

Reference to this paper should be made as follows: Toral, S.L., Gregor, D., Vargas, M., Barrero, F. and Cortés, F. (2011) 'Distributed urban traffic applications based on CORBA event services', *Int. J. Space-Based and Situated Computing*, Vol. 1, No. 1, pp.86–97.

Biographical notes: S.L. Toral received his MS and PhD in Electrical and Electronic Engineering from the University of Seville, Spain, in 1995 and 1999, respectively. He is currently an Associate Professor at the Department of Electronic Engineering, US. His main research interests include microprocessor and DSP devices, real-time and distributed systems, open source software projects and embedded operating systems.

D. Gregor received his MS in Electronic Engineering from the University of Seville, Spain, in 2009. He is currently a doctorate student at the University of Seville, Spain. His main research interests include embedded CORBA and cooperative systems.

M. Vargas received his MSc in Computer Science and his PhD in Computer Science, both from the University of Seville, Spain, in 1993 and 2001, respectively. He is currently an Associate Professor at the Department of Automation and Systems Engineering, US. His research interests include vision-based control, robotics, perception and robust control.

F. Barrero received his MSc and PhD in Electrical and Electronic Engineering from the University of Seville, Spain, in 1993 and 1998, respectively. He is currently an Associate Professor at the Department of Electronic Engineering, US. His main research interests include embedded systems and electrical drives for industrial applications.

F. Cortés received his MS in Electronic Engineering from the University of Seville, Spain, in 2007. He is currently an Assistant Professor at the Department of Electronic Engineering, US. His main research interests include wireless CORBA and multi-user environments.

1 Introduction

The public infrastructure distributed through modern cities consists of hundreds of cameras for surveillance purposes, traffic light regulators, and panels and equipments for traffic parameter estimation. These equipments are usually connected to an urban data network, disseminating comprehensive information including real-time data and video streaming (Cook and Das, 2004). This information supports traffic management administrators in making decisions, taking appropriate actions to alleviate congestions, and improving the global performance of the traffic network (Lee et al., 2010). Despite of the computing and networking capabilities of installed traffic equipments, they lack collective intelligence as they are usually intended for transmitting real-time information to the traffic control centre. However, urban environments can be situated in the context of ambient intelligence (AmI) frameworks. The main purpose of AmI is providing transparent support in environments where human activity takes place and, hence, where humans are the primary subject for the provision of an enhanced environment (Velastin et al., 2005). It is a new multi-disciplinary field that brings together sensor fusion, computer vision, middleware, multimedia and many other research fields (Monekosso et al., 2009). Urban environments represent a challenging area for AmI framework due to its high complexity, numerous sources of data and spreading of interesting and non-trivial applications. Today, modern transportation equipments have the potential of providing intelligent transportation systems (ITS) applications, such as advanced traffic management systems, advanced traveller information system, commercial vehicle operation and emergency management system (Lee et al., 2010). They are usually based on embedded computer platforms running an embedded operating system (Barrero et al., 2008; Toral et al., 2009a). This embedded operating system can be seen as the interface between the application and the embedded hardware, and must be tailored to the specific functions of the domain. However, once installed, it facilitates the use of networking and multitasking capabilities (Zhu and Wang, 2003). Many general-purpose operating systems have been modified to decrease their size and to adapt to embedded processors' architectures. Among the different possibilities of available operating systems, Linux is firmly in first place as the operating system of choice for smart gadgets and embedded systems thanks to its scalability, low cost and the superior networking capabilities (Yaghmour, 2003; Toral et al., 2009a). In particular, the Linux IP stack has proven itself to be stable, efficient and well suited for networking applications and services that require high reliability and availability (Spanos et al., 2008). These features can be used to transform the urban environment into a distributed network of embedded systems, offering advance traffic services to traffic administrators, vehicles, users or even other embedded equipments connected to the same urban network.

In this paper, a distributed computing model for an urban network over TCP/IP protocol is presented. This model is based on an open source middleware layer available for a wide variety of embedded devices and processor architectures, allowing urban equipments to offer services to the rest of agents involved in the urban environment, and providing an AmI framework.

The paper is organised as follows. Section 2 details current related work in the field of distributed computing and urban networks. Then, an overview about embedded middleware solutions is provided in Section 3. Afterwards, a road traffic application is illustrated in Section 4, describing the equipments involved and their computing requirements. Finally, implementation experience and obtained results are described in Section 5, and the conclusions are drawn.

2 Related work

Potential benefits of distributed systems applied to urban environments and road networks include increased road network capacity, reduced congestion and pollution, shorter and more predictable journey times, improved traffic safety for all road users, more efficient logistics, improved management and control of the road network (both urban and inter-urban), increased efficiency of the public transport systems, and better and more efficient response to hazards, incidents and accidents (Fuchs and Bankosegger, 2009; Tuominen and Ahlqvist, 2010). The majority of related works in the field of urban environments are usually related to vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communications (Hinsberger et al., 2007). Among these, the FleetNet project (FLEETNET, 2008) and its follow-up project network on wheels (NoW) investigate the integration of the Internet and vehicular networks (Festag et al., 2008). This integration requires mobility support, efficient communication, discovery of services, and support of legacy applications. FleetNet uses an IPv6-based addressing solution to address the vehicles. The proposed architecture contains stationary internet gateways along the road with two interfaces connecting vehicular networks to the internet.

Some other projects deal with cooperative systems in order to increase road safety and traffic efficiency. CVIS (cooperative vehicle infrastructure systems, http://www.cvisproject.org/), SAFESPOT (Cooperative systems for Road Safety, http://www.safespot-eu.org/pages/page.php) and COOPERS (COOPerative systEms for Intelligent Road Safety, http://www.coopers-ip.eu/) are initiatives integrated within the European 6th Framework Program (Toulminet et al., 2008).

CVIS aims to design, develop and test the technologies needed to allow cars to communicate securely with each other and with the nearby roadside infrastructure. In order to reach this goal, CVIS uses the international standard communications air-interface, long and medium range (CALM) that is still under development (Han et al., 2006). SAFESPOT aims to improve road safety by conceiving a safety margin assistant which detects critical situations in advance. Safety margin is the time difference between the time of detection of a potential danger and the time of real accident if nothing is done to avoid it. In SAFESPOT, this concept will be tested based on conception of cooperative system which will use V2V and V2I communication and IEEE 802.11p technology. Finally, in COOPERS' vision, vehicles are connected via continuous wireless communication with the road infrastructure. They exchange data and information relevant for the specific motorway segment to increase overall road safety and enable cooperative traffic management. In general, previous developments are mainly focused on the vehicle as the central element of the urban environment. In this paper, a different approach is introduced moving the key role towards the public infrastructure. In the short term, a cooperative computing system among public infrastructure equipments is more viable than V2V communications, which require vehicle manufacturers to agree a common standard and to assume the technological cost of such services. Besides, the amount of urban equipments already installed is huge, and the only needed requirement consists of updating their software. Today, urban infrastructure typically relies on a back-end server for coordinating and integrating data coming from multiple equipments (Yuan et al., 2003). However, they have a bandwidth scaling problem, since the central server can quickly become overloaded with the aggregate sum of messages/requests from the nodes. Also, the server is a single point of failure for the whole system. Some alternative considers the use of agents to support the development of the distributed software (Bramberger et al., 2006) or the use of peer-to-peer (P2P) systems, where individual nodes communicate with each other directly without going through a centralised server (Wang et al., 2010). Although these solutions are well suited for small and closed systems, they exhibit several problems in the field of ITS and urban infrastructure. For instance, ITS equipments are increasing day by day, and the cost of updating the distributed software in existing equipments would be prohibitive. Consequently, some plug and play mechanism in which services might be publicly available should be provided in urban environments. A solution that can overcome these limitations is a middleware layer able to support the heterogeneity of urban equipments, the specificities of embedded processors, and a variety of services. This way, road infrastructure equipments can provide services to other devices and equipment connected to the urban network in a context of AmI, composing complex cooperative services both to vehicles and citizens.

3 Embedded middleware overview

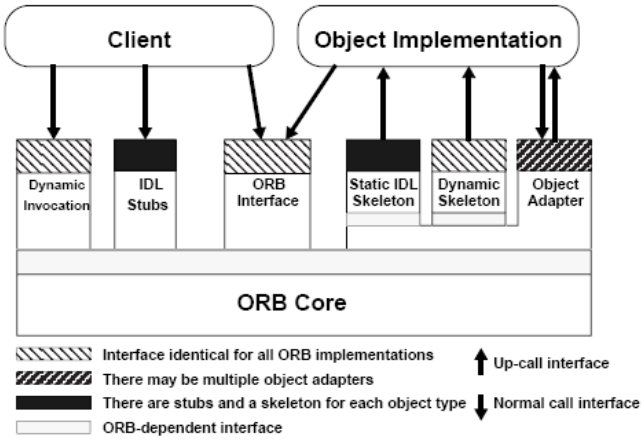
3.1 CORBA overview

CORBA is a framework of standards and concepts for open systems defined by the Object Management Group (OMG) to create distributed client-server applications (OMG, 2003). Methods of remote objects can be invoked transparently in a distributed and heterogeneous environment through an object request broker (ORB). Basically, the ORB handles communications between clients and servers. More specifically, the ORB is responsible for all of the mechanisms required to find the object implementation for the request, to prepare the object implementation to receive the request, and to communicate the data making up the request (Mutlu et al., 2006). Communications are based on the method invocation paradigm such as Java RMI. CORBA manages the heterogeneity of the languages, computers and networks. For interoperability purposes, the CORBA model defines a protocol for inter-ORB communications. The server interfaces are defined with a neutral language called interface definition language (IDL). This language defines the types of objects according to the operations that may be performed on them and the parameters to those operations (Henning and Vinoski, 1999).

To make an application, the designer has to describe the server's interface in IDL. The compilation of this description generates stubs (client side) and skeletons (server side) (Figure 1). The stub's role is to intercept client invocation and to transmit it to the server through the ORB. The skeleton's role is to receive the client's invocations and to push them to the service implementation (Pérez et al., 2003). Stubs and skeletons may be generated in different languages. For example, a stub can be in Java whereas a corresponding skeleton is in C++. The IDL language mapping is defined for many languages such as C, C++, Java, Ada, Python. This is one of the main advantages of using IDL, its ability to work with heterogeneous environments and languages.

Client invocations are received at the server side by an adapter (the portable object adapter) that delivers requests to the adequate object implementation (Figure 1). Stubs and skeletons are not mandatory. Dynamic invocations are possible on the client side [dynamic invocation interface (DII)] or on the server side [dynamic skeleton interface (DSI)]. Hence, it is possible to dynamically create, discover or use interfaces.

Figure 1 CORBA architecture



The CORBA specifications define the general inter-ORB protocol (GIOP) as its basic interoperability framework. GIOP is not a concrete communication protocol that can be used directly to interact ORBs. Instead, it describes how to build reply and request messages as well as other control massages and how to create and fit a particular transport protocol within the GIOP framework. GIOP assumes the underlying transport protocol is connection-oriented, full-duplex, symmetric, provides bytestream abstraction, and indicates disorderly loss of connection. The list of assumptions matches the guarantees provided by the TCP/IP protocol stack (Henning and Vinoski, 1999). GIOP realisation over TCP/IP is internet-IOP (IIOP) and for an ORB to be CORBA compliant, IIOP must be supported.

As a key concept for location transparency, CORBA has introduced the concept of interoperable object references (IORs). An IOR is a specialised object reference. It is used by an application in the same way that an ordinary local object reference is used. An IOR allows an application to make remote method calls on a CORBA object. Once an application obtains an IOR, it can access the respective CORBA object via IIOP, the protocol used between the CORBA infrastructures on both sides to exchange the request and reply messages of the method invocation on the remote object (Becker et al., 2007).

3.2 Embedded CORBA overview

One of the most important advantages of CORBA is its ability to support heterogeneous environments, different vendors' products, and several popular programming languages. This feature justifies the use of CORBA for urban environments where many different embedded systems were installed by different vendors through the time.

Several implementations of the CORBA specification exist today, both in the commercial and the open source domain. A detailed comparison of several CORBA implementations can be found in Puder (2004). When choosing a CORBA implementation, several considerations must be taken into account:

- Supported platforms: When working with heterogeneous embedded systems, it is important to choose a CORBA implementation able to work properly in the majority of them. This is particularly important for urban environments, where ITS equipments from different vendors have been installed during a long period of time. In particular, system-on-chip (SoC) architectures support is very interesting as they have been popularised for multimedia processors typically used in surveillance and monitoring applications.
- Supported programming languages and prerequisites: The supported programming languages and prerequisites to successfully compile the CORBA implementation must be also taken into account. ITS equipments are usually based on great variety of processors and operating systems. In this case, the heterogeneity of installed equipments requires the application software to be adapted to the existing system.
- Supported features: Depending on the final application, additional features like objects passed by value, asynchronous messaging, CORBA component model, etc. should be considered.
- Licences: Commercial and open source CORBA implementations can be found nowadays, and the licences under which they are released can affect the development of the project. In the case of urban environments, open source licences are preferred by public authorities to achieve vendor independence.

CORBA is usually targeted for general-purpose computing and are not suitable for resource-limited devices. That is the reason why numerous studies are focused on adapting the CORBA technology to resource constrained real-time systems. For instance, the real-time CORBA (RT-CORBA) (OMG, 2001) and minimum CORBA (OMG, 2002) specifications have been defined to reduce CORBA requirements removing some of the CORBA services. Based on these specifications, several studies have adapted it to different devices and environments. For instance, Rinner et al., (2007) developed a lightweight middleware for a set of smart cameras, implementing several domain specific services like allocation, reconfiguration and quality of service (QoS), and Gal et al. (2002) proposed the use of aspect-orientation in real-time systems for distribution, timeliness and dependability domains.

The main drawback of these reduced implementations is that they only achieve configurability using an IDL compiler, which means existing equipments should be reprogrammed to cooperate with new installed equipments. In general, current embedded middleware systems provide very limited availability of self-adaptation and self-configuration mechanisms (Mohamed and Al-Jaroodi, 2008). However, not all the embedded devices exhibit the same limitations in terms of memory or CPU processing capabilities. Urban equipments are frequently

based on industrial PCs, containing enough memory and power processing. For instance, smart cameras tend to be based on modern multimedia processors, which are a type of processor, specifically designed for the creation and distribution of digital media and also including increasing processing and networking capabilities. In this paper, we propose the development of a ITS cooperative system based on a full CORBA implementation, able to use CORBA standard services like the name or the event service.

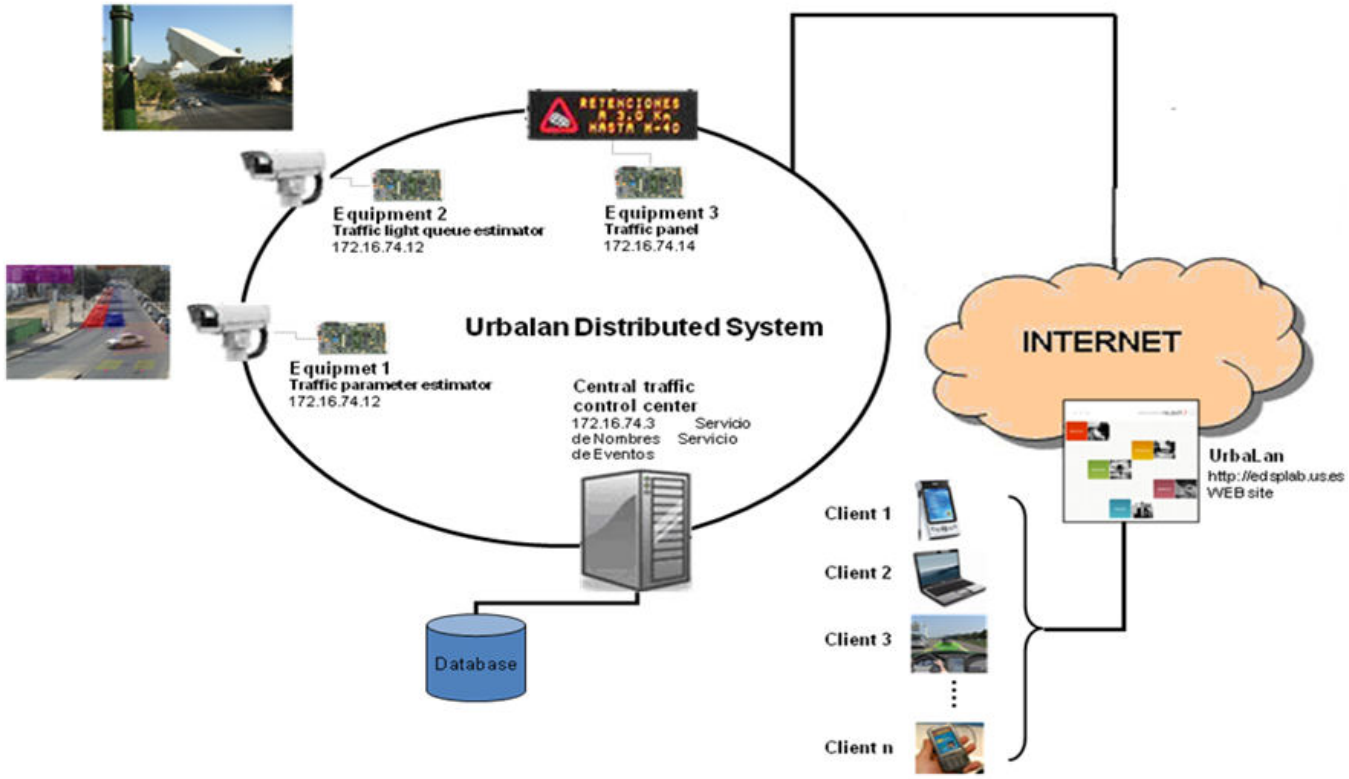
4 Road traffic application

Urban traffic control is extremely complex and dynamic. Real-time traffic parameter estimation and control operation constitutes a challenging part of an urban traffic control system (UTCS) (Choy et al., 2003). UTCS are usually based on a centralised network. Consequently, data are received and managed by the traffic control centre to solve a particular situation. For instance, control systems must have the capability to handle unforeseeable changes in traffic flow, such as accidents, and be able to optimise the traffic flow by adjusting traffic signals and by coordinating operations of each signal.

In this paper, we propose an agent paradigm based on the notion of autonomous, internally-motivated entities that inhabit dynamic environments (Roozemond, 2001). Autonomy should be understood as the ability to function as an independent unit or element over an extended period of time, performing a variety of actions necessary to achieve pre-designated objectives while responding to stimuli (Zeigler, 1990).

Three electronics equipments have been designed to illustrate a heterogeneous distributed environment application, Figure 2, using CORBA services to provide the distributed functionality. Each of the equipments executes a main road-traffic application to obtain real-time data or to operate over the public infrastructure, and it is connected to the urban network. An embedded middleware layer is also included in urban equipments of Figure 2, so they can provide services both to the control traffic centre as well as to other urban equipments connected to the data network, or even citizens connected to internet. The implemented urban network benchmark of Figure 2 considers two computer vision-based traffic equipments which provides real-time traffic data and a third equipment driving a traffic panel. All of them can be configured and operated from the traffic control centre. Real-time data and video streams are delivered through the urban network and messages can be remotely updated in the traffic panel. However, middleware services are available for the digital world (equipments owning to the same network or even citizens accessing through a public website). For instance, traffic panel can be automatically updated if a traffic incidence is detected by some other equipment in the network without waiting for the control traffic centre to operate.

Figure 2 Proposed distributed urban environment (see online version for colours)

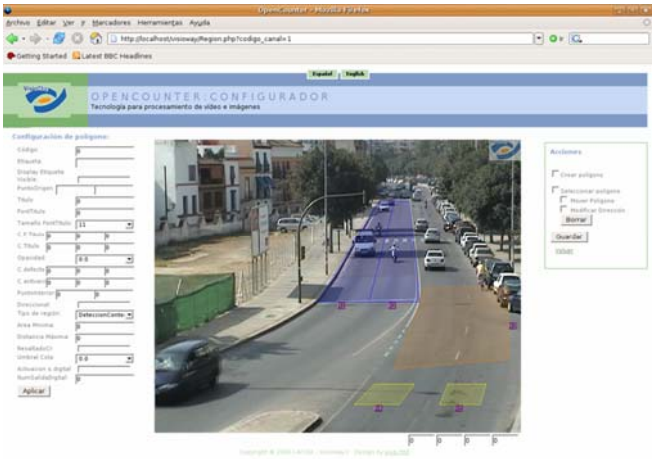


4.1 Equipment 1: traffic parameter estimator

This equipment is devoted to the detection of several traffic parameters based on computer vision techniques (http://www.visioway.com). This equipment provides data about traffic flow and incident detection (Toral et al., 2009b). It is based on an embedded video processor system and a video camera. The complete configuration of the scene can be made via the HTTP server running on a multimedia processor i.MX21 from Freescale (Figure 3).

Traffic data are collected over the detection areas drawn on the scene of Figure 3. Each of the areas or regions is a user-configurable polygon with an arbitrary shape or size, and an associated functionality.

Figure 3 Equipment configuration website (see online version for colours)



For instance, they allow the estimation of useful traffic information such as traffic flow per lane, lane vehicle counting, lane average speed, lane occupancy, etc.

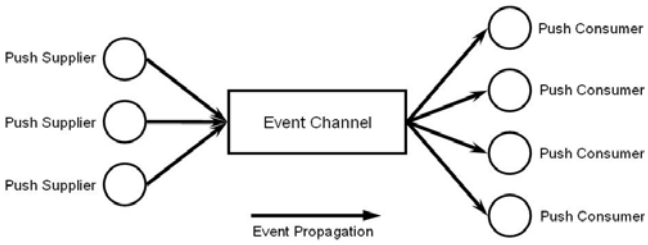
Region-based data estimation techniques require the detection of moving vehicles and short-term still vehicles. Specific algorithms are behind the described traffic-data estimation, like background subtraction or shadow removal algorithms, as well as more basic and general algorithms like edge detection, neighbourhood operations, image labelling or image thresholding. The most prominent algorithms among those implemented in the system are the background subtraction and the shadow removal algorithms.

The background subtraction technique allows extracting a moving object from an image sequence obtained using a static camera. It is based on the estimation of the so called background-model of the scene. This model is used as a reference image which is compared to each recorded image. Consequently, the background-model must be a representation of the scene after removing all the non-stationary elements, and must be permanently updated to take into account the changing lighting conditions or any change in the background texture (Toral et al., 2009b, 2009c). Surveys and comparisons of different algorithms for background subtraction can be found in Piccardi (2004). Shadow removal is a main concern when dealing with images in outdoor or unstructured environments. This is the case of traffic scenes, where the shadows of vehicles may be cast over neighbour regions. Shadows have some particular properties that can be exploited in order to eliminate them or, at least, to reduce its presence in the image [Prati et al., (2003), p.30; Cucchiara et al. (2003)]. The adopted solution for shadow detection and removal is inspired by the work described by Jaques et al. (2005).

In addition to this main functionality, the equipment also includes several CORBA services to make this functionality available to the rest of equipments connected to the same network. The first one is the name service, to facilitate access to the system's resources. All CORBA objects are registered within the name service, allowing them to be found and used by any other object in the system. Thus, each object that participates in a CORBA-based computation accesses the name service at least once: either to register itself via its IOR, or to locate other CORBA objects' IORs (Cheuk Lung et al., 1999). One of the OMG's common object services is a name service standard, called CosNaming, which defines how to resolve object names in a CORBA environment.

The second implemented service is the event service. The CORBA event service manages the synchronous and asynchronous propagation of messages (called events) among distributed objects (Singh et al., 2002). It is based on the traditional policy of publish/subscribe, where event suppliers and consumers are decoupled: suppliers can generate events without knowing the identities of the consumers and consumers can receive events without knowing the identities of the suppliers. The event service is suitable for distributed environments because there is no requirement for a centralised server or dependency on any global service (Garcia-Sanchez et al., 2005). CORBA specifies two approaches to initiating the transfer of events between suppliers and consumers. These approaches are called the push model and the pull model. In the push model (Figure 4) suppliers initiate the transfer of events by sending those events to consumers. In the pull model, consumers initiate the transfer of events by requesting those events from suppliers (Aleksy and Korthaus, 2005). The first one has been implemented, being event consumers passive elements used for providing information.

Figure 4 Push model suppliers and consumers communicating through an event channel



The dual service of the event service is the CORBA notification service, which was developed to address the limitations of the CORBA event service. The event service provides no support for filtering events or for specifying delivery requirements. The primary goal of the CORBA

notification service is to enhance the event service by providing event filtering and QoS. Clients of the notification service subscribe to events they are interested in by associating filter objects with the proxy objects through which the clients communicate with the Notification Service. With the notification service, each channel, each connection, and each message can be configured to support QoS (Coulouris et al., 2005).

4.2 Equipment 2: queue detector

The equipment detects whenever the queue of vehicles in front of a red light goes beyond a fixed threshold. The limits of the queue region can be remotely configured by drawing virtual detectors on the image of the scene, as shown in Figure 3. The blue regions in front of the traffic light delimit the queue detection regions. The service consists of delivering the current state of the queue of vehicles in the selected traffic light. Vehicle queues are detected using a combination of short-term and long-term background models of the scene (Zheng et al., 2006). The same than previous equipments, several CORBA services have been implemented to be executed with the main application.

4.3 Equipment 3: traffic panel

Panels can be automatically updated using services provided by other equipments connected to the urban network. For instance, a traffic panel can automatically warn about an incident event or traffic delay, whenever this information is provided by previous equipments.

5 Implementation and results

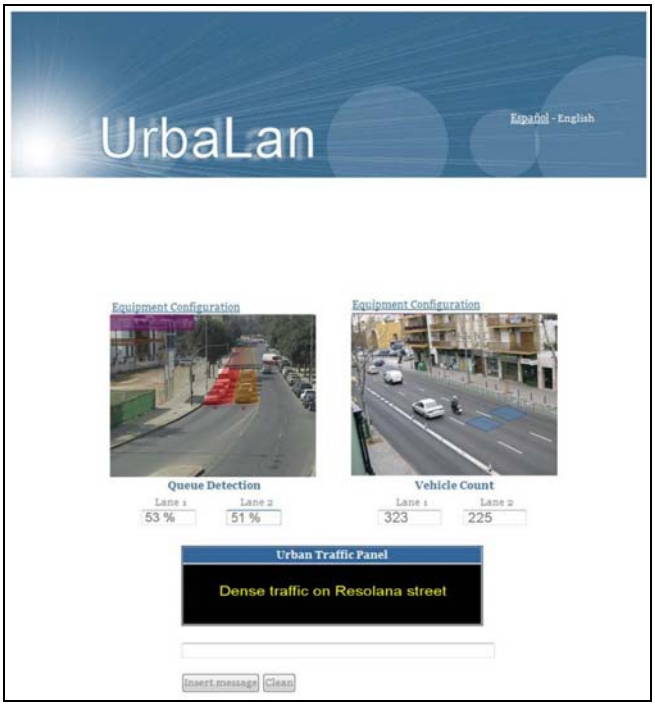
Several implementations of the CORBA specification can be found, both in the commercial and the open source domain. However, the intended application imposes several restrictions to the final choice. First, traffic equipments are based on embedded devices using a variety of processor architectures. Consequently, the selected CORBA implementation should cover a wide variety of software and hardware platforms. Second, open source implementation are preferred to reduce costs and to take advantage of their communities of support (Toral et al., 2009d). This last consideration is relevant taking into account the heterogeneity of involved devices in a typical urban network and the necessity of adapting the middleware layer to their particular features.

In accordance with these previous considerations, the selected CORBA implementation is TAO. TAO is a C++ ORB that is compliant with most of the features and services defined in the CORBA 3.x specification, including the RT-CORBA specification (Gokhale and Schmidt, 1998). TAO can be downloaded from the internet, and freely used and redistributed without developer or run-time licensing costs. Documentation and developers forums are also provided through its website.

TAO was cross compiled using the ARM gcc-3.3.2 and the library glibc-2.3.2, and was installed on the traffic equipments based on an ARM926EJ-S processor core, operating at 266 MHz. This processor is running an ARM Linux kernel 2.4.20. CORBA services are implemented as a middleware layer over the operating system, and they are called through the urban data network using the standard IIOP, built on top of TCP/IP.

Figure 5 shows a small application of cooperative computing among several urban traffic equipments. This figure illustrates the traffic control centre, where equipments can be individually managed. In particular, cameras can be remotely configured using the configuration interface of Figure 3, and traffic panels' messages can be manually updated using the corresponding button of Figure 5.

Figure 5 Traffic control centre (see online version for colours)



The implemented middleware layer allows equipments to work in a cooperative way using the CORBA event service, which is defined within the CORBA object services (COS) component (CosEventService). Using the CosEventService service, an event supplier can asynchronously send messages to one or several event consumers, allowing event delivery, and distributed group communication. The push model is used in this case example, so events are generated whenever an embedded system spread over the city detects a traffic incident or data requires the generation of valuable information. An object that performs the role of an event consumer must implement the PushConsumer interface, meanwhile an event supplier must implement the PushSupplier interface.

Algorithm 1 RtecEventChannelAdmin module

```
module RtecEventChannelAdmin
{
interface ProxyPushSupplier :
        CosEventComm : : PushSupplier
  {
        void connect_push_consumer (
        in CosEventComm:: PushConsumer
                push_consumer ,
        raises ( AlreadyConnected, TypeError );
  };

interface ProxyPushConsumer :
        CosEventComm : : PushConsumer
  {
        void connect_push_supplier (
        in CosEventComm : :
                PushSupplier push_supplier ,
        raises ( AlreadyConnected, TypeError );
  };
interface ConsumerAdmin
{
        ProxyPushSupplier
                obtain_push_supplier ( ) ;
}

interface SupplierAdmin
{
        ProxyPushConsumer
                obtain_push_consumer ( ) ;
}

interface EventChannel
{
        ConsumerAdmin for_consumers ( );
        SupplierAdmin for_suppliers ( );
void destroy ():
}
};
```

Although the CosEventService model seems to address many common needs of event-based, real-time applications, in practice, however, the standard CORBA event service specification lacks other important features required by real-time applications such as real-time event dispatching and scheduling, periodic event processing, and efficient event filtering and correlation mechanisms. The real-time event service included as part of TAO alleviates these limitations. TAO's RT event service augments the CORBA event service model by providing source-based and type-based filtering, event correlations, and real-time dispatching. In the defined application, consumers and suppliers connect to each other using the second alternative defined in the RtecEventChannelAdmin module (Algorithm 1).

The EventChannel interface defines three administrative operations: an operation for adding consumers, an operation for adding suppliers and an operation for destroying the channel.

An object which implements the PushConsumer interface must first acquire an object reference to an EventChannel using the name service Algorithm 2. The PushConsumer must then invoke the for_consumers() method on the EventChannel to obtain a reference to a ConsumerAdmin object. The obtain_push_supplier() method on the ConsumerAdmin object provides a reference to a ProxyPushSupplier. The PushConsumer registers itself with the EventChannel by invoking the connect_push_consumer() method on the ProxyPushSupplier. After this point, whenever the EventChannel has an event, it will invoke the push() method on the PushConsumer to supply the event.

The setup sequence for a PushSupplier is very similar to that for a PushConsumer. The PushSupplier will invoke the for_suppliers() method on the EventChannel and obtain a SupplierAdmin object (Algorithm 3). The obtain_push_consumer() method on the SupplierAdmin object provides a reference to a ProxyPushConsumer. Next, the connect_push_supplier() method on the ProxyPushConsumer object must be invoked. This will register the PushSupplier with the EventChannel. Whenever the PushSupplier has an event, it must call the push() method on the ProxyPushConsumer. This will send the event to the EventChannel, which will in turn invoke the push() methods of any PushConsumers which are attached to it.

Algorithm 2 Pseudo code of event consumer

```
BeginProcess Bind_to_Event_Channel_and_wait_for_events
        obj<-resolve_initial_references ("Event_Service");
        ec<-EventChannel;
        If ec=NULL Then
                return 1;
        Ifnot
                consumer_impl<-create_consumer;
                If consumer_impl=NULL Then
                        return 1;
                Ifnot
                        activate (consumer, poa,
        consumer_impl, consumer_deactivator);
                EndIf
        EndIf
        ec->for_consumers();
consumer_admin->obtain_push_supplier();
supplier->connect_push_consumer(consumer.in());
EndProcess
```

Algorithm 3 Pseudo code of event supplier

```
BeginProcess Bind_to_Event_Channel_and_push_events
        obj<-resolve_initial_references ("Event_Service");
        ec<-EventChannel;
        If ec=NULL Then
          return 1;
        Ifnot
          ec->for_suppliers();
          supplier_admin->obtain_push_consumer();
          consumer->connect_push_supplier();
          send_events(consumer.in());
        EndIf
        ec->destroy();
EndProcess
```

The main difference between a PushSupplier and a PushConsumer is that a PushSupplier explicitly invokes a push() method on a ProxyPushConsumer to send an event, while the PushConsumer never directly invokes a method to receive an event. Instead, the EventChannel will invoke the PushConsumer's push() method to send it an event.

In the detailed case example, the traffic parameter estimator and the traffic queue estimator are event providers, while the traffic panel is an event consumer. When the event channel is set, the message of the traffic panel is automatically updated depending on the circumstances detected by the cameras. For instance, incidents or traffic jams are detected by event providers and this information is delivered as an event to all the subscribed equipments of the urban network.

The CosEventService and the real-time event services have been compared in terms of latency measurements, from the side of the supplier and the side of the consumer. In both cases, latency is measured as a function of the number of supplied events. Supplier latency time includes initialisation of the ORB and argument analysis, obtaining the reference from the event channel, the reference from SupplierAdmin used for the supplier connection, obtaining the reference to a ProxyPushConsumer, and, in the case of RT supplier, publishing the SourceIB and the event type.

On the other hand, the consumer latency time includes:

- Initialisation of the ORB which represents the interface to the ORB functions, the POA Manager and analysing arguments.
- Obtaining the POA object reference 'RootPOA'.
- Getting the specific reference to event channel. The consumer is created and registered with the POA.
- Activating the servant with the POA and returning the object reference.
- Getting the ProxyPushSupplier for connecting as a consumer. The consumer who wants to connect to the event channel obtains the object reference from the supplier through the consumer_admin interface. In the case of RT Event_Service, the consumer should provide some information on QoS using the object ACE_ConsumerQOS_Factory.

Figure 6 and Figure 7 compare the latency times of CosEventService and RT Event Service as a function of the number of events from the side of the supplier and the consumer. The RT event service exhibit less latency times than CosEventService in the supplier side, and they remain approximately constant in average. This is not the case of the consumer side, where additional data concerning event filtering and requirements must be processed.

Obviously, the presented example can be generalised to all the equipments connected to the urban traffic networks, and event channels can be remotely set or removed from the traffic control centre. The result is that equipments are working in a cooperative environment, and services are available to new equipments connected to the same network.

Figure 6 Event supplier latency time (see online version for colours)

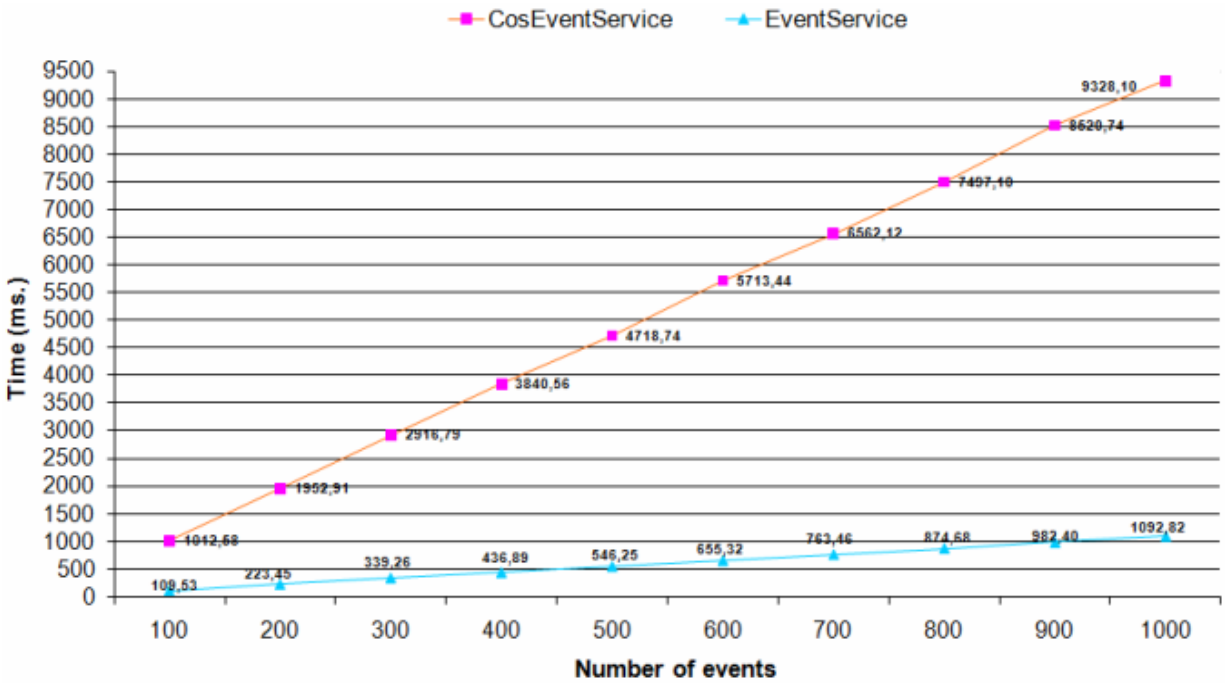
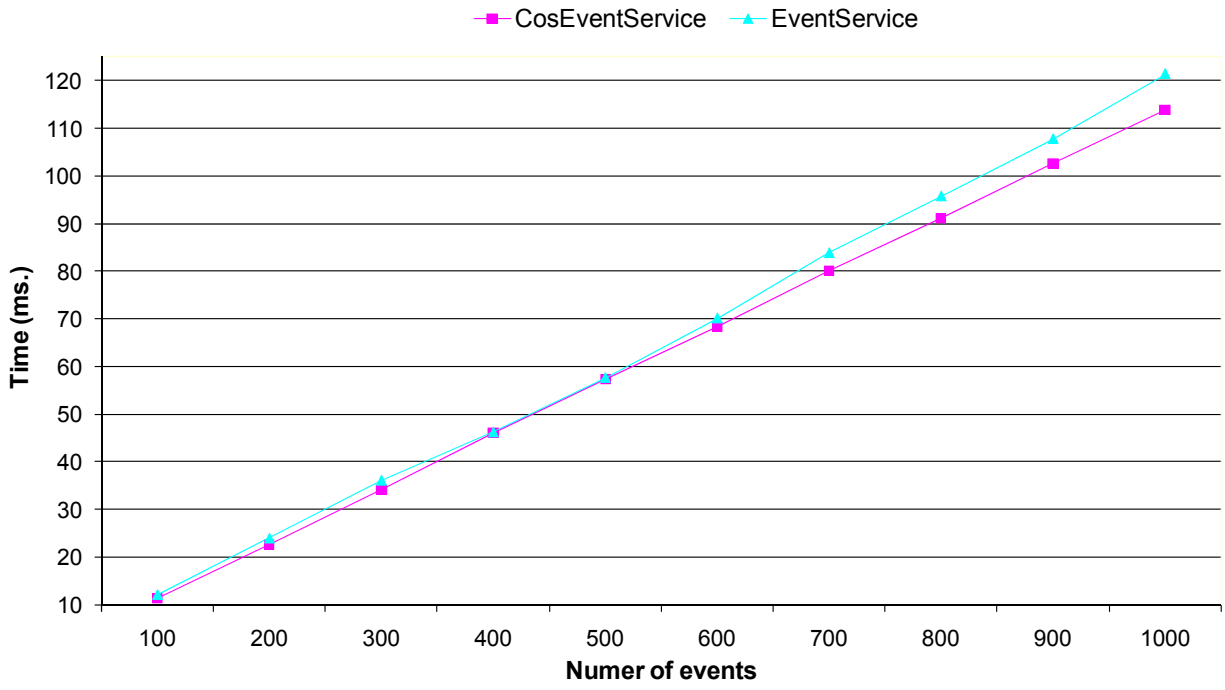


Figure 7 Event consumer latency time (see online version for colours)



6 Conclusions

This paper introduces the advantages of distributed computing in road traffic networks. The public infrastructure based on modern electronic equipments and devoted to road traffic monitoring and control, can take advantage of the increasing processing and networking capabilities of embedded processors. A simple road traffic network has been developed to prove the validity of the proposed model. The distributed applications have been performed using an open source middleware implementation of CORBA. In particular, event services have been used to deliver critical information to other subscribed devices. Results show the feasibility of the proposed solution and highlight the possibility of creating new value added services capable of transforming urban environments into intelligent environments.

Acknowledgements

This work has been supported by the Spanish Ministry of Education and Science (Research Project with reference DPI2007-60128) and the Consejería de Innovación, Ciencia y Empresa (Research Project with reference P07-TIC-02621).

References

Aleksy, M. and Korthaus, A. (2005) *Implementing Distributed Systems with Java and CORBA*, Springer, Heidelberg.

Barrero, F., Toral, S.L. and Gallardo, S. (2008) 'eDSPLab: remote laboratory for experiments on DSP applications', *Internet Research*, Vol. 18, No. 1, pp.79–92.

Becker, C., Staamann, S. and Salomon, R. (2007) 'Security analysis of the utilization of Corba object references as authorization tokens', *10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing, ISORC '07*, IEEE, Santorini Island, Greece, pp.196–203.

Bramberger, M., Doblander, A., Maier, A., Rinner, B. and Schwabach, H. (2006) 'Distributed embedded smart cameras for surveillance applications', *Computer*, Vol. 39, No. 2, pp.68–75.

Cheuk Lung, L., da Silva Fraga, J., Farines, J-M., Ogg, M. and Ricciardi, A. (1999) 'CosNamingFT-a fault-tolerant CORBA naming service', *Proceedings of the 18th IEEE Symposium on Reliable Distributed Systems*, IEEE, Lausanne, Switzerland, pp.254–262.

Choy, M.C., Srinivasan, D. and Cheu, R.L. (2003) 'Cooperative, hybrid agent architecture for real-time traffic signal control', *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, Vol. 33, No. 5, pp.597–607.

Cook, D. and Das, S. (2004) *Smart Environments: Technology, Protocols and Applications*, 1st edition, Wiley-Interscience, Hoboken, NJ.

Coulouris, G.F., Dollimore, J. and Kindberg, T. (2005) *Distributed Systems: Concepts and Design*, Pearson, Essex, England.

Cucchiara, R., Grana, C., Piccardi, M. and Prati, A. (2003) 'Detecting moving objects, ghosts, and shadows in video streams', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 10, pp.1337–1342.

Festag, A., Noecker, G., Strassberger, M., Libke, A., Bochow, B., Torrent-Moreno, M., Schnaufer, S., Eigner, R., Catrinescu, C. and Kunisch, J. (2008) '(NoW) network on wheels: project objectives, technology and achievements', *5th International Workshop on Intelligent Transportation (WIT)*, IEEE, Hamburg, Germany, pp.211–216.

FLEETNET (2008) *FleetNet*, http://www.et2.tu-harburg.de/fleetnet/english/vision.html, [accessed on 12/10/2009].

Fuchs, S., Bankosegger, D. (2009) 'Developing value networks for I2V co-operative services: an Austrian example', *IET Intelligent Transport Systems*, Vol. 3, No. 2, pp.216–224.

Gal, A., Spinczyk, O. and Preikschat, W. (2002) 'On aspect-orientation in distributed real-time dependable systems', *Proceedings of the Seventh International Workshop on Object-Oriented Real-Time Dependable Systems*, IEEE, San Diego, California, pp.261–267.

Garcia-Sanchez, F., Garcia-Sanchez, A., Pavon-Mariño, P. and Garcia-Haro, J. (2005) 'A CORBA bidirectional-event service for video and multimedia applications', *Lecture Notes on Computer Science*, Vol. 3760, pp.715–731.

Gokhale, A. and Schmidt, D. (1998) 'Measuring and optimizing CORBA latency and scalability over high-speed networks', *IEEE Transactions on Computers*, Vol. 47, No. 4, pp.391–413.

Han, S.W., Song, S.K. and Youn, H.Y. (2006) 'CALM: an intelligent agent-based middleware architecture for community computing', *Proceeding of the 4th Workshop on Software Technologies for Future Embedded and Ubiquitous and Systems and 2nd Intl. Workshop on Collaborative Computing, SEUS-WCCIA'06*, IEEE, Gyeongju, Korea, pp.89–94.

Henning, M., Vinoski, S. (1999) *Advanced CORBA(R) Programming with C++*, 1st edition, Addison-Wesley Professional, NY.

Hinsberger, A., Wieker, H., Riegelhuth, G. and Zurlinden, H. (2007) 'Benefits and technology of an intelligent roadside unit system for vehicle to infrastructure and infrastructure to centre communication', *14th World Congress on Intelligent Transport Systems*, ITS, Beijing, China, pp.1–8.

Jacques, J.C.S., Jung, C.R. and Musse, S.R. (2005), 'Background subtraction and shadow detection in grayscale video sequences', *Proc. of the XVIII Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI'05)*, IEEE, Natal, Rio Grande do Norte, Brazil, pp.189–196.

Lee, W-H., Tseng, S-S. and Shieh, W-Y. (2010) 'Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system', *Information Sciences*, Vol. 180, Nos. 1–2, pp.62–70.

Mohamed, N. and Al-Jaroodi, J. (2008) 'Characteristics of middleware for networked collaborative robots', *The 2008 International Symposium on Collaborative Technologies and Systems*, ACM, Irvine, California, USA, pp.524–531.

Monekosso, D., Remagnino, P. and Kuno, Y. (2009) *Intelligent Environments: Methods, Algorithms and Applications*, Springer, UK.

Mutlu, U., Edwards, R. and Coulton, P. (2006) 'QoS aware CORBA middleware for bluetooth', *2006 IEEE Tenth International Symposium on Consumer Electronics, ISCE '06*, IEEE, St. Petersburg, Russia, pp.1–7.

Object Management Group (OMG) (2001) *Real-Time CORBA 2.0*, available at http://www.omg.org, [accessed on 06/10/2010].

OMG (2002) *Minimum CORBA 1.0*, available at http://www.omg.org, [accessed on 06/10/2010].

OMG (2003) 'Common object request broker architecture, (CORBA/IIOP)', Technical report, formal/02-11-03.

Pérez, C., Priol, T. and Ribes, A. (2003) 'A parallel CORBA component model for numerical code coupling', *The International Journal of High Performance Computing Applications*, Vol. 17, No. 4, pp.417–429.

Piccardi, M. (2004) 'Background subtraction techniques: a review', *Procs. 2004 IEEE International Conference on. Systems, Man and Cybernetics*, IEEE, The Hague, Netherlands, Vol. 4, pp.3099–3104.

Prati, A., Mikic, I., Trivedi, M.M. and Cucchiara, R. (2003) 'Detecting moving shadows: algorithms and evaluation', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 25, No. 7, pp.918–923.

Puder, A. (2004) 'MICO: an open source CORBA implementation', *IEEE Software*, Vol. 21, No. 4, pp.17–19.

Rinner, B., Jovanovic, M. and Quaritsch, M. (2007) 'Embedded middleware on distributed smart cameras', *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2007*, IEEE, Honolulu, Hawai, USA, Vol. 4, pp.1381–1384.

Roozemond, D.A. (2001) 'Using intelligent agents for pro-active, real-time urban intersection control', *European Journal of Operational Research*, Vol. 131, No. 2, pp.293–301.

Singh, K., Nair, G. and Schulzrinne, H. (2002) 'Optimization of signaling traffic in centralized conference using SIP', *Proc. Second WSEAS International Conference on Multimedia, Internet and Video Technologies (WSEAS ICOMIV 2002)*, IKN, Koukounaries, Skiathos, Greece, pp 2931–2936.

Spanos, S., Meliones, A. and Stassinopoulos, G. (2008) 'The internals of advanced interrupt handling techniques: performance optimization of an embedded Linux network interface', *Computer Communications*, Vol. 31, No. 14, pp.3460–3468.

Toral, S.L., Martínez-Torres, M.R. and Barrero, F. (2009a) 'Virtual communities as a resource for the development of OSS projects: the case of Linux ports to embedded processors', *Behavior and Information Technology*, Vol. 28, No. 5, pp.405–419.

Toral, S.L., Vargas, M. and Barrero, F. (2009b) 'Embedded multimedia processors for road-traffic parameter estimation', *Computer*, Vol. 42, No. 12, pp.61–68.

Toral, S.L., Vargas, M., Barrero, F. and Ortega, M.G. (2009c) 'Improved sigma-delta background estimation for vehicle detection', *Electronics Letters*, Vol. 45, No. 1, pp.32–34.

Toral, S.L., Martínez-Torres, M.R. and Barrero, F. (2009d) 'An empirical study of the driving forces behind online communities', *Internet Research*, Vol. 19, No. 4, pp.378–392.

Toulminet, G., Boussuge, J. and Laurgeau, C. (2008) 'Comparative synthesis of the 3 main European projects dealing with cooperative systems (CVIS, SAFESPOT and COOPERS) and description of COOPERS demonstration site 4', *11th International IEEE Conference on Intelligent Transportation Systems, ITSC 2008*, IEEE, Beijing, China, pp.809–814.

Tuominen, A. and Ahlqvist, T. (2010) 'Is the transport system becoming ubiquitous? Socio-technical roadmapping as a tool for integrating the development of transport policies and intelligent transport systems and services in Finland', *Technological Forecasting and Social Change*, Vol. 77, No. 1, pp.120–134.

Velastin, S.A., Boghossian, B.A, Lo, B.P.L., Jie, S. and Vicencio-Silva, M.A. (2005) 'Prismatica: toward ambient intelligence in public transport environments', *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 35, No. 1, pp.164–182.

Wang, Y., Velipasalar, S. and Casares, M. (2010) 'Cooperative object tracking and composite event detection with wireless embedded smart cameras', *IEEE Transactions on Image Processing*; Vol. 19, No. 10, pp.2614–2633.

Yaghmour, K. (2003) *Building Embedded Linux Systems*, O'Reilly, CA.

Yuan, X., Sun, Z., Varol, Y. and Bebis, G. (2003) 'A distributed visual surveillance system', *Proc. IEEE Conf. Advanced Video and Signal Based Surveillance*, IEEE, Miami, Florida, pp.199–204.

Zeigler, B.P. (1990) 'High autonomy systems: concepts and models', *Proceedings AI, Simulation, and Planning in High Autonomy Systems*, IEEE, Tucson, Arizona, pp.2–7.

Zheng, J., Wang, Y., Nihan, N.L and Hallenbeck, M.E. (2006) 'Detecting cycle failures at signalized intersections using video image processing', *Computer-Aided Civil and Infrastructure Engineering*, Vol. 21, pp.425–435.

Zhu, L-X., Wang, F-Y. (2003) 'Component-based constructing approach for application specific embedded operating systems', *Proceedings of the 2003 IEEE Intemational Confuence on Intelligent Transportation Systems*, IEEE, Shangai, China, Vol. 2, pp.1338–1343.