# Real-time Data Flow Scheduling for Distributed Control

Anca Hangan, Ramona Marfievici, Gheorghe Sebestyen
*Department of Computer Science, Faculty of Automation and Computer Science*
*Technical University of Cluj-Napoca, Romania*
*E-Mail: {Anca.Hangan, Ramona.Marfievici, Gheorghe.Sebestyen}@cs.utcluj.ro*

## Abstract

*Remote process control and supervision applications developed over the TCP/IP networks require special communication models and techniques, which can guarantee the real-time and safety restrictions inherent to automation systems. This paper presents a reservation-based communication system architecture and a communication model based on data flow analysis that offer a good control over the transmission time of critical data. As part of the model, an analytical method is proposed that allows a-priory evaluation of the required minimum bandwidth necessary to assure the satisfaction of real-time transmission restrictions.*

***Keywords***: *distributed control, reservation-based scheduling, real-time traffic, communication model, quality of service*

## 1. Introduction

Remote process supervision and control systems require a communication infrastructure that supports reliable and safe real-time data transmission. These requirements are usually solved by using dedicated networks and industrial protocols, as presented in [1] and [2]. But these special purpose protocols are incompatible with general-purpose protocols used in local area networks and company intranets. If there are interoperability requirements between distributed control applications and organizational software, the incompatibility of industrial protocols and the TCP/IP stack may be a problem. Local computer networks and TCP/IP protocols, on the other hand, fail to satisfy real-time requirements because they apply the best-effort principle in supplying communication services, and it is quite difficult to use them as infrastructure for real-time applications.

During the last years, a significant research trend emerged for using best-effort (non-deterministic) networks for real-time communication. This trend was caused by technical developments such as the increasing network bandwidth (Gbps) and the development of new Quality of Service (QoS) mechanisms. The need for the integration of distributed control systems with other information systems that do not require special communication services was another cause.

To satisfy real-time communication requirements on TCP/IP networks, solutions that enable a predictable network behavior and that also provide end-to-end delivery time guarantees have to be developed.

In this paper we propose a reservation-based communication system architecture for distributed control applications on TCP/IP networks. As part of the solution, we define a control traffic model and a network bandwidth estimation method.

### 1.1 Related work

Several authors investigated the problem of accommodating real-time traffic on TCP/IP networks and proposed some solutions.

Wijnants and Lamotte present in [3] a method for managing the network bandwidth for multiple client applications. Their communication middleware, the NIProxy, is able to partition available client bandwidth between real-time and non real-time traffic flows by arranging them in a stream hierarchy. This solution gives good results in improving client Quality of Experience, but it does not guarantee any end-to-end timing requirements for real-time traffic.

Another approach is presented in [4], where the authors propose introducing a prioritization mechanism in the TCP/UDP/IP protocol stack, a mechanism which complies with the IEEE 802.1D standard. They evaluate the solution through simulation, using the OPNET simulator, by measuring

end-to-end latency of real-time packets in the presence of FTP traffic on the same network. Their conclusion is that a large part of the end-to-end message latency occurs at the end nodes, assuming that the network bandwidth is large enough to support all the traffic. In the referred paper the authors do not provide a solution for evaluating network bandwidth requirements, they just assume that the bandwidth is large enough.

In [5], Martinez et al. present Earliest Deadline First (EDF) communication scheduler implementation adapted for high-performance networks. The characteristics of high-performance networks enabled them to simplify the calculus of packet deadline, taking into consideration only the previous packet's deadline, packet size and average bandwidth.

A good technique which enables the provision of real-time guarantees is the reservation of resources for each task, in accordance to its requirements. In [6] and [7] reservation techniques are combined with feedback techniques to provide delay and execution time guarantees for tasks that coexists in a shared environment.

Schantz et al. describe two approaches, priority-based and reservation-based, in developing distributed real-time middleware [8]. For both solutions, the communication infrastructure is an IP network. In the priority-based middleware the standard Differentiated Services (DiffServ) mechanism is implemented for network resource management. In the reservation-based middleware the Integrated Services (IntServ) mechanism is implemented. Their main contribution is in the area of middleware implementation. But there are two quite important issues not addressed by this paper: (1) the provision of instruments for evaluating the resource requirements of real-time tasks and (2) the differentiation between hard and soft tasks.

IntServ [9] [10] provides end-to-end per-flow QoS by means of hop-by-hop resource reservation within the IP network but impose a significant burden on the core routers. To reduce the complexity within each core router, alternative schemes, referred to as Measurement Based Admission Control Schemes (MBAC) have been proposed [11]. These schemes replace per-flow states with run-time link load estimates performed in each router. However, MBAC solutions still require significant modification of the existing Internet architecture, as core routers must support load estimation algorithms, and still need to be explicitly involved in per flow signaling exchange.

A completely different approach is provided by DiffServ [12]. In DiffServ, core routers are stateless and unaware of any signaling. While DiffServ easily enables resource provisioning performed in a management plane for permanent connections, their widely recognized limit is the lack of support per-flow resource management and admission control, resulting in the lack of strict per flow QoS guarantees. A number of proposals, presented in the literature, have shown that per flow Distributed Admission Control schemes can be deployed over DiffServ architectures [13] [14]. Although significantly different in implementation, they share the common idea that accept/reject decisions are taken by the network endpoints and are based on the processing of "probing" packets, injected in the network at setup to verify the network congestion status. A "pure" Extended Admission Control (EAC) scheme, called Phantom Circuit Protocol-Delay Variation (PCP-DV) is proposed in [15]. The scheme determines whether a new connection request can be accepted based on delay variation measurements taken on the probing packet at the edge nodes.

Reinemo et al. [16] propose and evaluate three different admission control schemes for virtual cut-through networks, each one suitable for use in combination with DiffServ based QoS scheme to deliver soft real-time guarantees. Two of the schemes assume pre-knowledge of the network's performance behavior without admission control and are both implemented with bandwidth broker. The third is based on endpoint/egress admission control and relies on measurements to assess the load situation. Due to the way the flow control affects latency and the nature of cut-through networks, latency and jitter properties are hard to achieve.

An approach to quantify the impact of end-to-end QoS provisioning through a combination of both intra and inter-autonomous system (AS) traffic engineering (TE) is proposed in [17]. Two offline QoS-aware systems are deployed for this and a direct relationship between intra-AS and inter-AS TE is then established. The interaction between them is analyzed and both the decoupled and integrated approaches are presented.

In [18], several possible algorithms for routing and scheduling which allow coexistence of QoS and best-effort flows are presented. The network algorithm takes into account state imprecision in routers, maxmin bandwidth allocation, and existing link state information.

## 1.2 Our contribution

This paper introduces a communication model, which is based on control data flow analysis and communication scheduling that uses a rate-monotonic algorithm. The communication model solves efficient delivery for short control messages over TCP/IP

networks and facilitates a-priori estimation of required network bandwidth for the reservation mechanism.

A reservation-based communication system architecture is also proposed. Our solution uses Integrated Services/RSVP and the facilities of IPv6 protocol as support for real-time communication. The implemented middleware translates real-time requirements to existing network services and is used to validate both the communication model and the system architecture.

The remainder of the paper is organized as follows: Section 2 presents some specific communication-related issues about distributed industrial control systems. In section 3 we discuss the details of the proposed system architecture. A new data flow-based traffic model is introduced in Section 4. Section 5 describes the method for bandwidth estimation, based on communication scheduling. Section 6 covers some important implementation aspects. The experiments and results analysis are described in Section 7. Section 8 concludes the paper.

## 2. Problem description

Distributed control systems are an integral part of the industrial automation domain. Their functionalities include data acquisition, monitoring and control of industrial processes. While the majority of the control systems are usually located within more confined area (e.g. plant area, company local network) and communications are usually performed using local area network (LAN) technologies that are typically reliable and high-speed, other are geographically distributed (e.g. SCADA systems) and need long-distance communication systems such as the Internet.

Our research has the objective of solving the communication issues of distributed control systems which are deployed in the companies' local TCP/IP networks. Usually, these networks are managed by the companies and the nodes (hosts, switches, routers, servers) are configured and administered by a company internal authority. Such a network has to accommodate two broad categories of traffic: non-real-time and real-time. Non-real-time traffic can adjust to changes in delay and throughput and is generated by applications that include common Internet-based applications, such as file transfer, electronic email, remote logon, network management, and Web access. Real-time traffic does not easily adapt, if at all, to changes in delay and throughput and have requirements that include beside delay and throughput, delay variation and packet loss.

In addition, these corporate networks may be connected to strategic partner networks and to the Internet, thus, making more use of Wide Area Networks (WANs) and Internet to transmit their data to remote stations.

Most control applications must satisfy real-time and safety constraints. A very important parameter in real-time environments is the system response time, defined as the time between the occurrence of an event and the corresponding response. In distributed systems, message delivery time, has a large influence on the system's response time. Network protocols must incorporate message delivery time control mechanisms in order to guarantee maximum delivery time for control messages. These mechanisms assume a deterministic network behavior, which permit a-priori evaluation of maximum message delivery time.

When measuring the performance of a real-time communication system, the following parameters are taken into consideration [19]:

- Deadline miss rate (fraction of all messages that are delivered to late at the destination)
- Delay jitter (the variation of message delays)
- Loss rate (fraction of all messages that are dropped on the route from source to destination)

For hard real-time applications deadline misses are not acceptable, moreover message response time must be guaranteed a-priori. But delay jitter may not cause serious problems as long as deadlines are satisfied. In the case of soft real-time applications deadline misses are tolerable to some extent, but, under some particular conditions, delay jitter may have negative effects. In the case of distributed control systems traffic, one has to deal with both hard and soft real-time requirements. For these reasons, the goal is to minimize both message response time (end-to-end delay) and delay jitter.

In the case of using a TCP/IP network for real-time communication, some important issues may arise. First of all, a maximum response time for packets has to be guaranteed. This can be a serious problem knowing that TCP/IP networks function on a best-effort basis.

Another communication issue is message delivery efficiency. Data transmitted through the network in distributed control systems are quite different compared to data transmitted by usual applications which generate traffic in TCP/IP networks. Control applications use short, unstructured data (e.g. digital signal values). Process control data is generated, mostly, at well determined periods of time. The majority of supervision and control functions involve data acquisition, processing and storage, visualization

of process status and command issuing, which require a short reaction time.

Control applications include different automation and computing devices, which are interconnected. In order to assure interoperability, the communication protocol must allow uniform and transparent access to system's resources and it must be simple enough to allow implementation on devices with limited computing resources.

Last but not least is the issue of real-time and non-real-time traffic coexistence. In the case of remote process control it is quite possible to have both traffic (real-time) generated by the control system and traffic (best-effort) generated by other applications (e.g. office automation) that run in the same network. Network bandwidth has to be managed in order to assure real-time requirements for control traffic and also to assure fair treatment for the best-effort traffic.

The objective of this paper is to take a new approach in solving some of the following communication issues:

- Guarantee packet delivery time in TCP/IP networks in the presence of both real-time and non-real-time traffic
- Assure predictability of network behavior
- Assure transmission efficiency of process control data
- Provide device interoperability and uniform access to process control data

## 3. System architecture

In order to provide a comprehensive communication system architecture based on IP infrastructure so that to meet the challenges of quality of service provisioning for industrial control application we integrate in three major components: (1) industrial control applications and processes; (2) a middleware system (service manager) between the application and the protocol driver; this middleware closely interacts with Internet protocol stack; (3) network infrastructure based on IPv4 or IPv6 protocol.

The first component of the system architecture represents quality of service demanding applications that use a QoS API to send requests to the service manager. These applications generate periodic and aperiodic traffic. The traffic is characterized by packet size, transmitting data rates, priority, and accepted latency.

The middleware bridges the industrial applications and the underlying network systems by dispatching the application requests and returning status and feedback from the underlying system to the application. Examining the application requests and the available network resources, the middleware selects a provisioning service or service level, maps the application QoS to network-specific quality of service, and initiates resource allocation or renegotiates the parameters with the application before the flows' source starts to generate any packets.

The following components were integrated in the proposed middleware:

- Traffic QoS specification
- QoS negotiation
- Traffic and QoS monitoring
- Resource reservation
- Data transfer

### 3.1 Traffic QoS Specification and QoS Negotiation modules

An application which wants to set up a connection in order to transmit packets to another application in the network uses the means of the traffic QoS specification to set up a reservation request first. This module is a generic API so that an application demanding quality of service is isolated from the complexity of the provisioning services.

The application defines its generic QoS specification in terms of traffic profile which is composed of parameters that characterize the traffic stream or session (source IP address and port number, destination address, transport protocol) and parameters that define quantitatively the network performance requirements (transmitting rates, message size, transmission deadlines, latency), which can be specified using maximal, average and minimal values.

The traffic QoS specification module contains a set of rules for converting the traffic characteristics to parameters in the underlying message model.

Based on the input from the QoS specification module, the QoS negotiation module is responsible for authorizing the request and check if the network is able to support the new connection interacting with the resource reservation module for resource allocation. The goal of this module is to provide optimal quality of service with respect to critical parameters and previous requests.

Application's requests for quality of service parameters can be solved in two ways: positive, in case

the resource reservation module sends a positive acknowledge to the QoS negotiation module that there are enough resources in the network to satisfy the request and the reservation is set along the path, and negative. In case of a negative notification, the application may invoke the QoS negotiation module in order to find what resources and services are available in the network and to adjust the quality of service requirements and start a new negotiating procedure.

### 3.2 Traffic and QoS Monitoring module

In this module components are included for monitoring network resources (available bandwidth, average utilization of a link, delay, jitter) and quality of service related statistics from routers (queue length, number of conforming/exceeding packets in bytes, number of dropped packets, CPU utilization). It also signals significant changes in resource availability.

When an application establishes a network traffic stream, this module starts collecting its performance. It collects data from traffic stream, including quality of service specification, connection times, transmission rates and delays, and communicate the quality of service parameters to the QoS negotiation module in order to determine if there is any quality of service violation. All collected data is stored into a management information base.

### 3.3 Resource Reservation module

The resource reservation module is the ultimate authority for the resource handling in the proposed architecture (Fig. 1). Its main building blocks are admission control and reservation setup. Admission control implements request authorization by checking if the network is able to support the flow and the decision algorithm that nodes use to determine whether a new flow can be granted the requested quality of service with/without impacting earlier guarantees. For these tasks it closely interacts with the main entity, the resource reservation protocol.

Resource ReSerVation Protocol (RSVP) [9] is used for resource reservation signaling. It is designed to enable the senders, receivers and routers of communications sessions to communicate with each other to reserve resources for new flows at a given level of QoS. On the other hand, the reservation protocol is responsible for maintaining flow specific state information at the end nodes and at the nodes along the path of the flow.

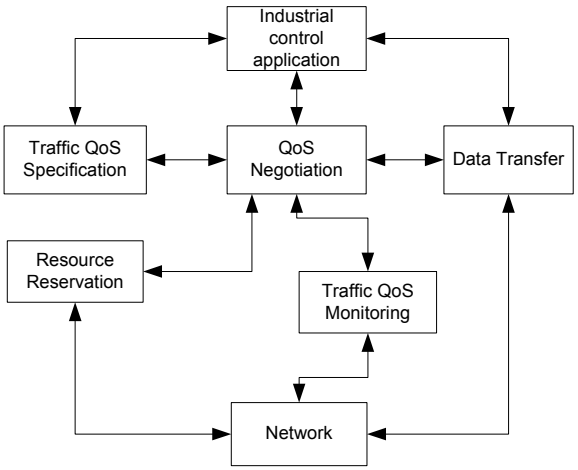



**Figure 1. System architecture – components**

RSVP requests result in resources being reserved in each node along the data path. Given below are the main attributes of this protocol: it requests resources in only one direction (it treats a sender separately from a receiver, although the same application might be running at both the sender and the receiver); is receiver-oriented (the receiver of a data flow initiates and maintains the resource reservation used for that flow); RSVP itself is not a routing protocol, but it is designed to work with the existing routing protocols; RSVP supports both IPv4 and IPv6.

To make a resource reservation at a node [20], our RSVP daemon uses the admission control mechanism. If check fails, the RSVP returns an error notification to the QoS negotiation module that originated the request. If checks succeed, the RSVP daemon sets the parameters.

## 4. The data flow model

In order to make an analytical evaluation of the traffic generated by the distributed control system, it is required to classify this traffic and then, based on the identified types, to define the traffic model.

### 4.1 Traffic classification

Control traffic is generated by data exchanged between the control applications and industrial devices, such as:

- Values obtained through data acquisition, with a well defined frequency (e.g. temperature in an oven, liquid level in a tank, engine state – started/stopped, etc.)
- Commands generated at known periods of time

- Operator commands (e.g. start/stop engine, increase oven temperature to 200 degrees, etc.)
- Process events, alarms, alerts, etc.

The previous categories of data generate periodic and aperiodic network traffic, with real-time constraints.

### 4.2 Model definition

To model the control traffic, we introduce *data flows* [21]. A data flow is the sum of all packets sent through the network that have the same source, destination, content and periodicity. Traffic between control applications and devices connected to the process is a sum of periodic and aperiodic data flows. As an example, consider an application that monitors the temperature in a room. Temperature sensors measure the temperature in the room at the same time, with the periodicity of five minutes and send the data to the process computer. This computer packs the temperature values into packets and sends them to the monitoring application. All these packets containing temperature values create a data flow.

Fig. 2 shows the data flows established between two control applications connected to remote industrial processes, through a TCP/IP network.

Periodic data flows include values obtained through data acquisition, control commands, which occur at well defined periods of time. Aperiodic data flows include commands issued by the application operator, high priority alerts and event signals. A number of parameters are identified for each data flow type. The parameters for periodic data flows are: inter-release period, priority (importance), content (process control data included in the flow), required packet delay (or response time), transmission deadline, source, destination and packet size. The parameters for aperiodic data flows are: priority, content, required packet delay, transmission deadline, source, destination and packet size.

In real-time task modeling, it is a common practice to assume that aperiodic tasks have a minimum inter-release period, which is given by process related parameters. Because all tasks are considered periodic, scheduling and feasibility analysis are simplified. For the same reasons, we choose to make the same assumption (minimum inter-release period) for aperiodic data flows.

A data flow is formally defined as an n-tuple:

$$DF = (T, P, r, D, l, Src, Dest, c) \quad (1)$$

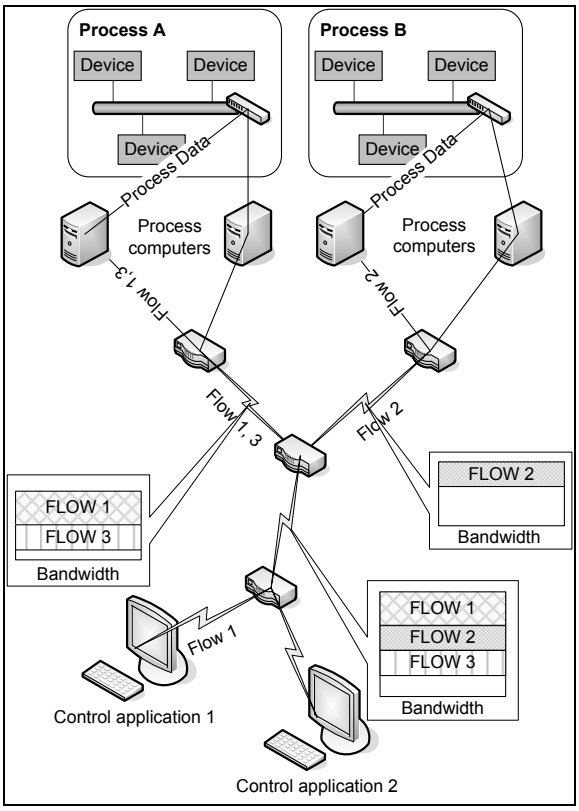



**Figure 2. Data flows between control applications**

The components of the n-tuple are the data flow parameters. $T$ is the inter-release period for periodic data flows and the minimum inter-release period for aperiodic data flows. $P$ is the priority, $r$ is the required packet delay or response time, $D$ is the transmission deadline and $l$ is the size of a data flow packet. The last three components are the source (*Src*), destination (*Dest*) and content ($c$) of the data flow.

### 4.3 Communication optimization

The protocols from the TCP/IP stack are optimized to deliver large packets of data in a best-effort manner. Process control messages, on the other hand, are very short (from a few bytes to hundreds of bytes); they have periodic occurrence and real-time constraints. If very short periodic messages are packed and released in a TCP/IP network, the protocol overhead is very large compared with the payload data. Because, messages with short period of occurrence (e.g. seconds, milliseconds) can create large amount of traffic although few effective data is transmitted, it can be said that the network is inefficiently utilized for data transmission.

```
count = 0;
Foreach process_data _value
{
    DF[count] = Create_data_flow ();
    Set_data_flow_parameters ( DF[count] );
    count++;
}
Sort_data_flows_by_period ();
For ( i=0; i < count-1; i++ )
{
    If ( Periodic ( DF[i] ) &&
          ! Marked_for_delete ( DF [i]))
    {
        j = i+1;
        While ( DF[i].period == DF[j].period )
        {
            If ( Periodic ( DF[j] ) )
            {
                If ( DF[i].src == DF[j].src )
                   and ( DF[i].dest == DF[j].dest )
                {
                    Aggregate ( DF[i], DF[j] );
                    Mark_for_delete ( DF[j] )
                }
            }
            j++;
        }
    }
}
Foreach DF
{
    If (Marked_for_delete ( DF ))
    {
        Delete ( DF );
    }
}
```

**Figure 3. The algorithm pseudocode for the aggregation of data flows**

To optimize the transmission of control packets, we can adopt the strategy of aggregating control data from different process devices into the same data flow.

To organize control data into larger data flows, the following parameters must be considered:

- Data acquisition periodicity, assuming that devices which perform data acquisition with the same period read the sensor values virtually at the same time
- Data priority
- Data source and destination

By performing data aggregation, the data flow packets contain larger amounts of effective data, hence increasing the efficiency of control data transmission.

The specification of data flows for a control system is obtained using the algorithm presented in Fig. 3. First, an array of data flows (*DF*) is created by defining a data flow for each piece of process data. The parameters are set for each data flow. The array of data flows is then sorted by data flow period. To aggregate data flows that have similar characteristics, each periodic data flow is compared to all other periodic data flows that have the same period (*DF[i].period == DF[j].period*). If the data flows have the same source (*DF[i].src == DF[j].src*) and destination (*DF[i].dest == DF[j].dest*), they are put together in the same data flow. The aggregated data flow will contain data from both initial data flows, will have the highest priority (*max(DF[i].priority,DF[j].priority*)) and the smallest deadline (*min(DF[i].deadline,DF[j].deadline*)) of the two data flows. The first data flow will be substituted by the aggregated data flow and the second data flow will be deleted.

For aperiodic data flows, which cannot merge with other flows, transmission efficiency is reduced. A solution for these flows, if their frequency of occurrence is high, is to reserve space for aperiodic data in periodic flows. This space (e.g. a few bytes) is used only if the aperiodic event takes place right before a periodic data flow packet is sent.

## 5. Communication scheduling

Solving the issue of message transmission time control is critical in a distributed control system. The system architecture proposed in this paper uses a TCP/IP network as a communication infrastructure. The main challenge in this case is guaranteeing real-time constraints on a best-effort communication infrastructure. For solving this problem, a bandwidth reservation mechanism is used.

The bandwidth reservation mechanism requires the estimation of network bandwidth for all traffic generated in the system. For this purpose we adopted a method commonly used in the case of real-time tasks, the worst case scenario analysis using the rate-monotonic scheduling algorithm [22].

In our approach, each data flow is a “task” and the network is the “processor”, which has to be shared between all “tasks” in the system. Priorities are assigned to all data flows in a rate-monotonic manner. This way, a data flow which has a lower period will have a higher priority. Periodic and aperiodic data flows are taken into consideration for scheduling.

Aperiodic data flows are considered to have a minimum inter-release period. The next step is to compute the response time for each data flow. The data flow system is feasible if, for each data flow, the response time is less than its transmission deadline (r < D). In this work we consider that transmission deadline for a data flow is equal to its inter-release period (T = D). By obtaining the appropriate values for the response time of all data flows, in the worst case, a maximum value for the required network bandwidth can be derived. The maximum bandwidth value obtained is used to make resource reservations. In this way, it can be guaranteed that actual response time for each data flow will be less or equal than the computed response times.

It is considered that the worst case response time for a data flow happens when a packet has to wait for the transmission of packets that belong to all data flows with higher priority and for one packet with lower priority, but with the largest transmission time. It is also assumed that all data flows start at the same time.

In order to compute the data flow response time ($r_i$) the following variables are taken into consideration:

- The delay caused by the devices found on the network path ($t_{delay}$)
- The transmission time of packets that belong to the data flow ($C_i$)
- The data flow's inter-release period ($T_i$)
- The data flow's priority ($Pi$)
- The transmission time of packets that belong to data flows with higher priority
- Maximum transmission time of packets that belong to data flows with lower priority
- The number of hops from source to destination ($nHops$)

Response time for each data flow is computed using the following set of equations:

$$\begin{cases} r_i^{\,0} = t_{delay} + nHops * (C_i + \sum_{P_j > P_i} C_j) \\ r_i^{t+1} = t_{delay} + nHops * (C_i + \sum_{P_j > P_i} \left\lceil \frac{r_i^t - C_i}{T_j} \right\rceil * C_j + \\ \quad + Max\{C_k \mid P_k < P_i\}) \end{cases} \quad (2)$$

The response time is obtained through iteration, until $r_i^{t+1} = r_i^t$. There are two important conditions which have to be imposed:

- All response times have to be less than the corresponding deadlines
- Network utilization has to be less than 100%

The bandwidth value is gradually increased (with 10%) while the response time and utilization are computed, until these requirements are met.

In the first iteration, the response time of a data flow is computed by summing up the transmission time, the delay caused by the network devices such as switches and routers, and the transmission time of all other data flows which have higher priority. In subsequent iterations, the response time equation has two new components, the maximum transmission time of data flows which have lower priority ($Max\{C_k \mid P_k < P_i\}$) and the sum of transmission time of all packets of higher priority that are likely to be released while the packet is being transmitted through the network. The number of packets with higher priority that influence the response time of the data flow equals the number of packets that are released in the response time of a packet from the data flow. To decrease the number of iterations, the time interval when the actual packet is being transmitted ($C_i$) is subtracted from the response time, as in that time interval packets from other data flows can not be transmitted.

The transmission time for packets which are included in a data flow is computed as follows:

$$C = \frac{Packet\_length}{Bandwidth} \quad (3)$$

The delay caused by network devices can be approximated by using a mean round-trip time of a probe packet sent on the same route on which the bandwidth reservation will be made.

As the response time is computed recursively, the computation time could be a problem. In our case, because bandwidth requirements are assessed and reservations are made before starting the control system, a larger computation time is not an issue.

## 6. Implementation details

To validate the proposed system architecture, the data flow model and the method of estimating network bandwidth, a prototype of a distributed control system, was developed. The prototype includes the

communication middleware, the industrial application and the industrial device simulator (for the provision of industrial process data).

The communication middleware runs on a network infrastructure based on TCP/IP stack, with IPv6 as a network protocol. The solution adopted [23] is to use the IPv6 Traffic Class and Flow Label fields. The Traffic Class field enables a source to identify desired traffic-handling characteristics of each packet relative to other packet from the same source. The intent is to support various forms of services. In case of IPv6 standard [24], a flow is defined as a sequence of packets sent from a particular source to a particular destination for which the source desires some special handling by the intervening routers. From the source's point of view, a flow is just a sequence of packets that are generated from a single application instance at that source and have the same transfer service requirements. From the router's point of view, a flow is a sequence of packets that share attributes that affect how these packets are handled by the router. In principle, all of a user's requirements for a particular flow could be defined in an extension header and included in each packet, but for our implementation, we leave the concept of flow open to include a wide variety of requirements and adopt the flow label, in which the flow requirements are defined before traffic generation and a unique flow label is assigned to the flow.

The RSVP module is designed as a state machine. The objects defined in this module represent:

- RSVP sessions
- State information extracted from PATH message and information from RESV message
- Reservations installed in an outgoing interface
- Information about a previous hop in a session, i.e. the last reservation that has been sent to this hop

For each RSVP session all relevant information is bundled and the destination address and port is saved. From each PATH message all relevant information is held, i.e., the sender's address and traffic specification, routing information. For each reservation requested from a next hop, reservation specification is held, i.e., the FlowSpec, which determines the amount of resources that are requested, depending on the service class.

The industrial control application uses all the facilities offered by the communication middleware and implements the following functionalities:

- Remote process control and visualization
- Input and output data flow definitions for devices participating in the industrial process
- Control data flow through commands sent to devices connected to processes
- Specification and negotiation of resources needed for communication with other devices
- Receive and process data flows from industrial devices
- Register data flow delay time

The operator can visually create the diagram of the industrial process, by dragging the symbols of different types of devices on the control board. Next, the operator has to specify input data flows (data received from devices connected to the process) and output data flows (commands sent to devices) in order to establish communication parameters.

After the definition of data flows, the negotiation process for resources starts. Input and output data flows are analyzed and, as a result, bandwidth needed to satisfy real-time communication constraints is computed. The application sends a query asking for the available bandwidth and round-trip time to destination process. The response time can be guaranteed only for the data flows having the period less than the delay caused by the network devices (e.g. switches, routers). If the available bandwidth is insufficient, data flows having the smallest period are deleted, data is recomputed and application begins the resource reservation process. After the negotiation and reservation process, the application can start to send and receive data flows.

The industrial device simulator sends periodical data flows (requested by the control application) containing process values randomly generated from a predetermined range and receives periodical data flows representing commands from the control application for devices connected to the process. Devices cannot negotiate resource reservations for generated data flows nor to specify quality of service parameters. The control application connected to these devices is responsible for the negotiation and bandwidth reservation.

An important issue encountered during the implementation of both the industrial control application and the device simulator is the specification of data flows. In order to assure the device and application interoperability, data flow parameters and content are specified using XML. In this way messages between applications and devices are interpreted easier and the access to process and control data is uniform.

```
<Flow>
    <ID> data_flow_ID </ID>
    <SrcIP> source_IP </SrcIP>
    <DestIP> destination_IP </DestIP>
    <Per> data_flow_period </Per>
    <Pri> data_flow_priority </Pri>
    <Name> data_flow_symbolic_name </Name>
    <Content>
        XML_content_specification
    </Content>
</Flow>
```

**Figure 4. Data flow specification in XML**

Fig. 4 shows an example of a periodic data flow specification in XML.

## 7. Experiments

We conducted two sets of experiments. First, we used a simulator to validate the proposed method for network bandwidth estimation. Second, we performed some tests using the implemented control system prototype, which was deployed on our experimental infrastructure.

### 7.1 Estimation of network bandwidth

For the first set of experiments we measured the response time, jitter and packet loss for multiple periodic real-time data flows, which were released in a simulated TCP/IP network. The main objective of these experiments was to check if the computed network bandwidth value guarantees the required response time and low jitter for all data flows.

The simulation study was performed on Network Simulator (NS-2) [25], version 2.33. The simulation results were evaluated for different scenarios using the topology depicted in Fig. 5.

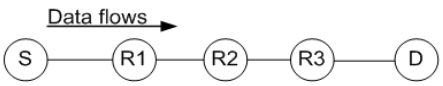



**Figure 5. The topology used for simulations**

**Table 1. Parameter settings for periodic real-time data flows**

| *Flow* | *Period (ms)* | *Packet size (B)* |
|---|---|---|
| F1 | 10 | 300 |
| F2 | 120 | 300 |
| F3 | 50 | 300 |
| F4 | 75 | 300 |
| F5 | 520 | 300 |

**Table 2. Response times (1st scenario)**

| *Flow* | *Response time (ms)* | | |
|---|---|---|---|
| | *Measured maximum* | *Measured average* | *Measured minimum* |
| F1 | 50.66 | 28.87 | 25.33 |
| F2 | 50.66 | 33.01 | 25.33 |
| F3 | 50.66 | 32.98 | 25.33 |
| F4 | 50.66 | 30.48 | 25.33 |
| F5 | 50.66 | 33.03 | 25.33 |

The topology consists of 5 nodes. These nodes are connected with full-duplex bidirectional links. All links have the same available bandwidth and propagation delay. In this paper it is assumed that per link delay is negligible. Constant-Bit-Rate (CBR) agents were attached to the source node (S) and used to generate periodic, fixed size packet traffic in the network. User Datagram Protocol (UDP) was used as transport layer protocol to minimize the overhead of establishing a connection. Five periodic data flows were defined, having the same source (S) and destination (D). The parameter settings are summarized in Table 1.

Two scenarios were simulated. Measurements were made to compare data flows' response times with the corresponding deadlines and to observe to what extent the jitter affects the response time of packets.

In the first scenario the network bandwidth was set to the minimum value which can accommodate all defined data flows (379 Kbps). Results analysis revealed that the average measured response times were acceptable in the case of data flows which had larger periods, but for the other data flows response times were very often greater than the corresponding deadlines. Jitter measurements showed that even if the average value was quite small, the maximum value was very large, approximately equal to the measured minimum response time. For this scenario no packets were lost.

For the second scenario, equations (2) were used to derive the maximum bandwidth needed by the set of data flows in order to satisfy the deadlines. The computed maximum bandwidth was 2106 Kbps. As expected, a considerable difference can be observed between the measured maximum response time and the maximum computed response time. This difference is due to the fact that the worst-case scenario does not occur during simulation time, thus the resulting network utilization is low. All the deadlines were satisfied and the average delay jitter is very small for the flow with the largest period. There was no packet loss.

For both scenarios, measured values can be found in Tables 2-5 and the comparison between data flows

in terms of response time and jitter are shown in Fig. 6-9.

### 7.2 Tests performed using the prototype

To test the distributed control system prototype, two PCs connected in a local network were configured as traffic source and destination. A static route consisting of another two PCs which played the role of routers was established between these nodes. The network infrastructure was based on IPv6 protocol. The communication middleware, the control application and the device simulators were deployed on the test infrastructure.

A process schema containing monitoring elements connected to two data flows was specified in the control application. The first data flow (Flow 1) has a 2 seconds period and 270 byte packet size. The second data flow (Flow 2) has a 0.5 second period and the same packet size. After starting the remote control application and the device simulators, response time for all packets was measured.

**Table 3. Response times jitter (1st scenario)**

| *Flow* | *Response time jitter (ms)* | | |
|---|---|---|---|
| | *Measured maximum* | *Measured average* | *Measured minimum* |
| F1 | 25.33 | 3.22 | 0 |
| F2 | 18.68 | 3.58 | 0 |
| F3 | 19 | 4.33 | 0 |
| F4 | 24 | 4.41 | 0 |
| F5 | 25.33 | 3.59 | 0 |

**Table 4. Response times (2nd scenario)**

| *Flow* | *Response time (ms)* | | | |
|---|---|---|---|---|
| | *Measured maximum* | *Measured average* | *Measured minimum* | *Computed* |
| F1 | 6.078 | 3.103 | 3.039 | 9.12 |
| F2 | 6.078 | 3.839 | 3.039 | 68.4 |
| F3 | 6.078 | 3.870 | 3.039 | 27.36 |
| F4 | 6.078 | 3.447 | 3.039 | 41.04 |
| F5 | 6.078 | 3.724 | 3.039 | 86.64 |

**Table 5. Response times jitter (2nd scenario)**

| *Flow* | *Response time jitter (ms)* | | |
|---|---|---|---|
| | *Measured maximum* | *Measured average* | *Measured minimum* |
| F1 | 3.039 | 0.114 | 0 |
| F2 | 2.279 | 0.547 | 0 |
| F3 | 2.279 | 0.484 | 0 |
| F4 | 3.039 | 0.815 | 0 |
| F5 | 3.039 | 0.002 | 0 |

Measurements were made in two cases. In the first case, the communication middleware was used to make network bandwidth reservations before starting the traffic. In the second case, no reservations were made for the real-time traffic. For both data flows, response time measured during tests was less than the maximum allowed response time, in the case of reservations, presented in Table 6.

The measurements showed that the proposed system architecture, traffic model and method of data flow scheduling are able to satisfy the control system's requirements and guarantee a maximum delivery time. They also showed that the analytical evaluation of the response time is an upper limit to the measured time parameters.

If no reservations were made, for both data flows, measured response time fluctuated between a minimum of 0.367 seconds and a maximum of 0.617 seconds, as can be observed in Table 7. Packets of Flow 1 have the same priority on the network as packets of Flow 2, even though Flow 2 requires a better response time. Real-time requirements were not satisfied, because for Flow 2 the maximum measured response time was greater than the computed maximum response time.

Fig.10 and Fig. 11 show charts that compare the computed response time for the two data flows with the measured response time, on both test scenarios.

**Table 6. Measured response time for experimental data flows with reservations**

| | *Flow 1* | *Flow 2* | *Flow 1* | *Flow 2* |
|---|---|---|---|---|
| *Computed bandwidth* | 9 kbps | | | |
| *Available bandwidth* | 100 kbps | | 64 Mbps | |
| *Measured RTT* | 1.5 ms | | 0.65 ms | |
| *Computed maximum response time* | 0.745 s | 0.497 s | 0.744 s | 0.496 s |
| *Measured response time* | 0.685 s | 0.372 s | 0.677 s | 0.367 s |

**Table 7. Measured response time for experimental data flows without reservations**

| | *Flow 1* | *Flow2* |
|---|---|---|
| *Computed bandwidth* | 9 kbps | |
| *Available bandwidth* | 100 Mbps | |
| *Measured RTT* | 0.4 ms | 0.4 ms |
| *Computed maximum response time* | 0.744 s | 0.496 s |
| *Maximum measured response time* | 0.617 s | 0.617 s |
| *Minimum measured response time* | 0.367 s | 0.367 s |

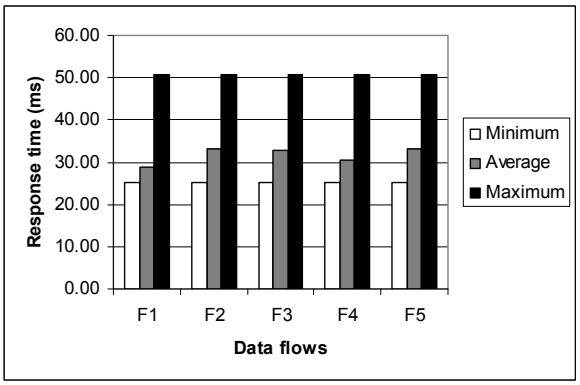



**Figure 6. Response times (1st scenario)**

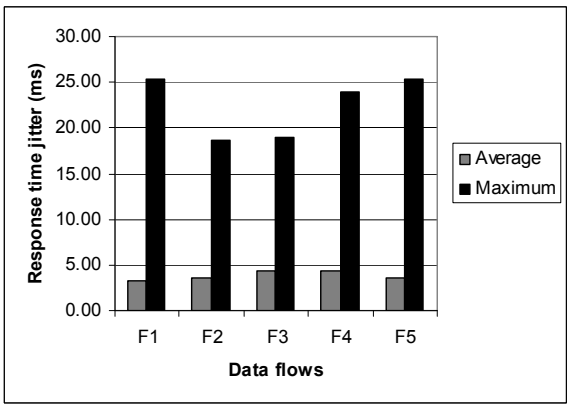



**Figure 7. Response times jitter (1st scenario)**

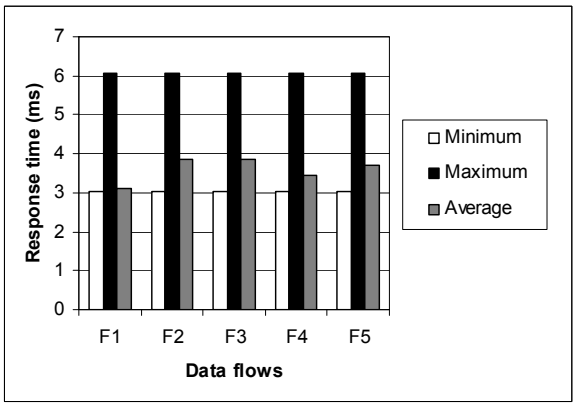



**Figure 8. Response times (2nd scenario)**

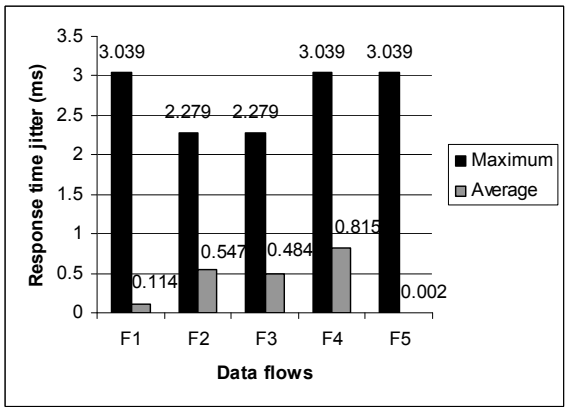
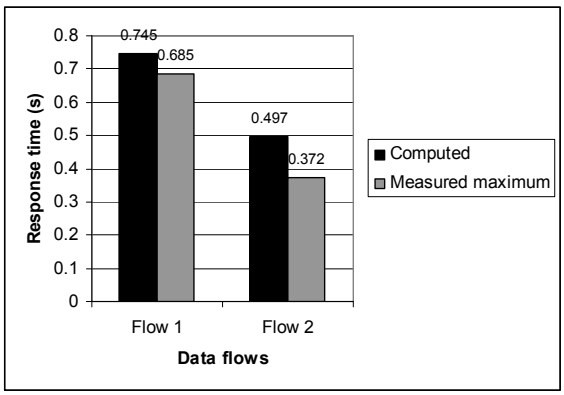
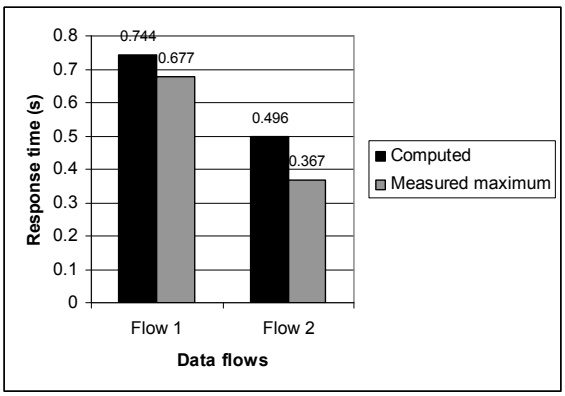



**Figure 9. Response times jitter (2nd scenario)**




**(a) Available bandwidth = 100kbps**




**(b) Available bandwidth = 64Mbps**

**Figure 10. Response time measurements using reservations**

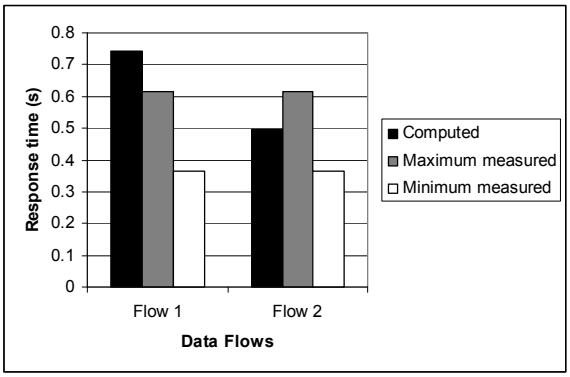



**Figure 11. Response time measurements without using reservations**

## 8. Conclusion

This paper presents a new approach in solving network delivery time control and data delivery efficiency in distributed control systems, on TCP/IP infrastructures. We introduce the data flow traffic model which provides the basis for communication scheduling, and a reservation-based communication system architecture. Our solution uses Integrated Services/RSVP and the facilities of IPv6 protocol as support for real-time communication.

The experimental results show that the implemented prototype satisfies the real-time constraints. This proves the validity of the proposed communication model and the method for computing the required network bandwidth. Moreover, the analytical evaluation of response time demonstrated to be an upper limit for measured delivery time.

## 9. References

[1] L. B. Fredriksson, "Controller area networks and the protocol CAN for machine control systems", Mechatronics, vol. 4, no. 2, pp. 159-192, 1994

[2] S. Koubias and G. Papadoupoulous, "Modern fieldbus communication architectures for real-time industrial applications", Computers in Industry, vol. 26, no.3, pp. 243-252, Aug. 1995

[3] M. Wijnants, W. Lamotte, "Managing client bandwidth in the presence of both real-time and non real-time network traffic", 3rd International Conference on Communication Systems Software and Middleware and Workshops, (COMSWARE), Bangalore, Jan. 2008, pp. 442-450

[4] T. Skeie, S. Johannessen, O. Holmeide, "Timeliness of real-time IP communication in switched industrial Ethernet networks", IEEE Transactions on Industrial Informatics, Volume 2, Issue 1, Feb. 2006, pp. 25 – 39

[5] A. Martínez Vicente, G. Apostolopoulos, F. J. Alfaro, J. L. Sánchez, J. Duato, "Efficient Deadline-Based QoS Algorithms for High-Performance Networks," IEEE Transactions on Computers, vol. 57, no. 7, Jul., 2008, pp. 928-939

[6] C. Lu, Y. Lu, T. Abdelzaher, J. Stankovic, S. Hyuk Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers", IEEE Transactions on Parallel and Distributed Systems, vol. 17, no. 9, Sep. 2006, pp. 1014-1027

[7] L. Abeni, L. Palopoli, G. Lipari, J. Walpole, "Analysis of a Reservation-Based Feedback Scheduler", IEEE Real-Time System Symposium (RTSS), Austin, Texas, 2002, pp. 71

[8] R.E. Schantz, J.P. Loyall, C. Rodriguez, D.C. Schmidt, Y. Krishnamurthy, I. Pyarali, "Flexible and Adaptive QoS Control for Distributed Real-time and Embedded Middleware", Proceedings of Middleware 2003, ACM/IFIP/USENIX international middleware conference, Rio de Janeiro , Brazil, 2003, pp. 374-393

[9] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification", RFC2205, September 1997

[10] J. Wroclawski, "The use of RSVP with IETF Integrated Services", RFC2210, Sept. 1997

[11] S. Georgoula.; P. Trimintzios, G. Pavlou, K.H. Ho, **"**Heterogeneous real-time traffic admission control in differentiated services domains**",** IEEE Global Telecommunication Conference, (GLOBECOM), Volume 1, 28 Nov.-2 Dec. 2005

[12] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC2475, Dec. 1998

[13] F. Kelly, R. Key, S. Zachary, "Distributed Admission Control", IEEE Journal on Selected Areas in Communications, Vol. 18, Dec. 2000, pp. 2617-2628

[14] L. Breslau, E. Knightly, S. Shenker, I. Stoica, H. Zhang, "Endpoint Admission Control: Architectural Issues and Performance", in Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 2000, Stockholm Sweden, pp. 57-69

[15] G. Bianchi, F. Borgonova, A. Capone, L. Fratta, C. Petrioli**, "**Endpoint admission control with delay variation measurements for QoS in IP networks", ACM SIGCOMM Computer Communication Review, vol. 32, no. 2, April 2002, pp. 61 - 69

[16] S.-A. Reinemo, F. O. Sem-Jacobsen, T. Skeie, O. Lysne, "Admission Control for diffServ Based Quality of Service in Cut-through Networks", 10th International Conference on High Performance Computing (HiPC 2003), ed. by Timothy Mark Pinkston and Viktor K. Prasanna, pp. 118-129, Heidelberg, Springer. Lecture Notes in Computer Science

[17] K.-H Ho, M. Howarth, N. Wang, G. Pavlou, S. Georgoulas, **"**Two Approaches to Internet Traffic Engineering for End-to-End Quality of Service Provisioning", 1st EuroNGI Conference on Next Generation Internet Networks - Traffic Engineering, Rome, Italy, 18-20 April 2005, pp. 135 – 142

[18] K. Nahrstedt, S. Chen, "Coexistence of QoS and Best Effort Flows - Routing and Scheduling", Proceedings of 10th Tyrrhenian International Workshop on Digital Communications: Multimedia Communications, Ischia, Italy, Sept. 1998

[19] Liu, J. W.S., *Real-Time Systems*, Prentice Hall, 2000

[20] S. Shenker and L. Breslau, "Two Issues in Reservation Establishment", ACM SIGCOMM '95 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, Cambridge, 1995, pp. 14-26

[21] A. Hangan, R. Marfievici, Gh. Sebestyen, "Reservation-Based Data Flow Scheduling in Distributed Control Applications" – The Third International Conference on Networking and Services, ICNS 2007, 19-25 June 2007, Athens, Greece, in Proceedings of the Third International Conference on Networking and Services, IEEE Computer Society Washington, DC, USA, 2007, ISBN: 0-7695-2858-9, pp. 10-15

[22] C.L. Liu, J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", Journal of the ACM, Vol. 20, No. 1, January 1973, pp.46-61

[23] R. Marfievici, Gh. Sebestyen, A. Pop-Bidian, "Industrial Control Communication Framework based on an IPv6 Infrastructure" – International Multi-Conference on Computing in the Global Information Technology – Challenges for the Next Generation of IT&C, ICCGI 2006, Bucharest, Romania, 2006

[24] R. Banerjee, The Internet Protocol version 6 (IPv6): issues and challenges, Technical Report, Computing Science Laboratory, Oxford University, Feb. 2002

[25] ***, The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/