# ProtEx: A TOOLKIT FOR THE ANALYSIS OF DISTRIBUTED REAL-TIME SYSTEMS

Yves Meylan Aneema Bajpai Riccardo Bettati
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

## Abstract

*Large-scale distributed real-time systems are increasingly difficult to analyze within the Rate Monotonic Analysis framework. This is due partly to their heterogeneity, complex interaction between components, and variety of scheduling and queuing policies present in the system. In this paper we present a methodology to extend the traditional RMA approach by allowing general characterization of workload and flexible modeling of resources. We realize our approaches within ProtEx, a toolkit for the prototyping and schedulability analysis of distributed real-time systems. This toolkit focuses on a wider set of methodologies than the traditional RMA scheduling analysis tools.*

## 1. Introduction

Large-scale distributed real-time systems are increasingly characterized by a number of aspects. First, with the current tendency towards Commercial Off The Shelf (COTS) products, the computation and communication infrastructure is becoming more and more heterogeneous. Typical systems will be deployed across a variety of processor platforms, with different operating systems and different networking technologies. Second, such systems support a variety of paradigms for the interaction among their components. These range from traditional stream-based communication, where data flows through the system in well-defined flows, to publish-subscribe approaches, to highly dynamic method invocations. Next, there is a need for integrated support for timely delivery of service *and* for reliability. Real-time group communication, for example, will be applied in various forms to realize replication. Finally, there is a strong need to build such systems by integrating reusable software components. A number of projects are investigating standard software infrastructures so that components can be re-used in a "plug-and-play" manner [1, 2].

Unfortunately, the development of analysis methodologies to support the design and the verification of these emerging systems has not kept pace. The current technologies are mostly based on the Rate Monotonic Analysis (RMA) methodology. While RMA-based methods have proven to be effective for the analysis and verification of smaller systems, a number of shortcomings limit their usefulness for larger systems. We elaborate on three of them:

First, traditional design and analysis methodologies lack an integrated model for computation and communication. Practical considerations have traditionally led to very different ways of analyzing real-time computation and communication. This artificial separation is awkward (for example, it leads to lower utilization,) and becomes more so as the boundary between the two becomes more fuzzy, as sophisticated communication primitives, for example reliable group communication, evolve.

Second, traditional design and analysis methodologies rely heavily on workload regulation and make make assumptions about worst-case workload; typically they assume periodic workloads. Various forms of regulators make sure that these assumptions are satisfied. For example, rate controllers in packet schedulers enforce a minimum inter-arrival time of packets. Similarly, various forms of sporadic servers ensure that non-periodic real-time workload is executed in a controlled fashion (e.g., [9]). As another example, resource access protocols [13] control the eligibility for execution of the critical sections by appropriately modifying the task's priority. The pervasive use of regulation is problematic for the class of systems described above. It adds run-time overhead and poorly handles integration of COTS components and hardware and software composition.

Third, traditional analysis methodologies make simplistic assumptions about resources. Active resources, such as CPUs or communication links, are typically modeled as constant-rate servers. In real systems, the rate at which jobs can be served is highly variable. Lower-level operating system layers, for example, add various forms of hidden scheduling and priority inversion. Simply assuming a worst-case rate for active resources is a common technique, which unduly reduces resource utilization. Rather, resources must be modeled in a way that allows to flexibly describe the worst-case availability to particular jobs.

Over the last few years our group has developed a number of workload modeling techniques to analyze systems with widely varying workloads [6]. At the same time, we have investigated the applicability of service functions for the general modeling of communication and computation servers [3]. The result of these investigations became a set of techniques that can be used, possibly in conjunction with traditional RMA, or other techniques used in real-time communication [5], to analyze large-scale heterogeneous systems that consist of a wide variety of workloads and servers. It also became apparent that some of these techniques (such as service-function based [12] or integration-based [4]) for end-to-end analysis are superior, to our knowledge, to all currently used methods.

We integrated these techniques into ProtEx, a toolkit for analysis of distributed real-time systems. ProtEx is a toolkit used for prototyping and performing schedulability analysis of distributed real-time system and is well adapted for a networking environment. It gives the possibility to the user to design, prototype, and analyze a system with varying resource and workload characteristics. A number of workload characterization and delay analysis techniques developed by the Texas A&M Real-Time Research Group and by others are integrated in the ProtEx toolkit [3, 4, 6].

ProtEx performs end-to-end or single-server schedulability analysis for a defined set of tasks based on a selected analysis methodology. It provides the user with the ability to incrementally work on an application prototype before going into the implementation, testing, and integration phases of the software life cycle. By performing extensive schedulability analysis during the design and prototyping phase, ProtEx can ultimately save time during the next phases of the application development.

In this paper we first motivate the need for general workload characterization and for flexible resource modeling for end-to-end analysis of distributed real-time systems in Section 2. Section 3 gives an overview of the general methodologies applied in ProtEx. We describe the modeling of the system resources and of the workload. We also describe the schedulability analysis techniques used for single node and end-to-end analysis. A number of schedulability analysis tools, most of them based on RMA, are available. We describe two of them in Section 4. We conclude in Section 5.

## 2. Challenges in the Modeling of Distributed Real-Time Systems

In order to meet the requirements for analysis methods for real-time systems, novel forms of modeling resources and workload must be used. In this section, we propose such methods, based on general characterizations of workload and flexible resource modeling.

**General Workload Characterization.** Traditionally, work on schedulability analysis in endsystems focuses on periodic tasks, where the inter-arrival time of requests is fixed to be the period of the task. Non-periodic workload is typically transformed into periodic workload by either one of the following three ways: (i) by treating the non-periodic tasks as periodic tasks with the minimum inter-arrival time being the period, (ii) having server, which look like periodic tasks to the rest of the system, execute the non-periodic tasks (e.g. [9]), or (iii) splitting the non-periodic tasks each into collections of periodic tasks of different sizes and periods. In all three cases, well-known schedulability analysis methodologies for periodic workloads can be used.

Applying the same methodologies for distributed real-time systems, where tasks execute across multiple endsystems, shows poor results, even for periodic workloads. While the arrival of instances of a periodic task may indeed be periodic at the first processor, the completion of these instances almost certainly is not. If no special action is taken, and the completion of an instance indicates that the second processor can go ahead, the "arrival" of instances of the tasks on the second processor is not periodic.

By appropriately synchronizing, or regulating, the execution of the tasks on the processors, excessive bursts can be eliminated, which increases schedulability. The task execution can be made to adhere to simple workload descriptors, which makes rigorous schedulability analysis possible. A number of such synchronization schemes were presented in [10]. They allow the use of traditional schedulability analysis methods for periodic workloads.

Appropriate regulation reduces the worst-case end-to-end response times as compared to systems without regulation. However, regulation adds overhead to the system and increases the average end-to-end response time for tasks. In addition, it is of limited applicability when the workload is inherently aperiodic.

To deal with systems that have limited or no support for workload regulation, we use workload *re-characterization*: instead of regulating the workload after each processor to conform to a predetermined descriptor, we use general descriptors and compute the descriptor of the workload after each processor. The methods for this depend on the scheduling policies and the other workload present on the processor. Our group has applied these techniques in various forms at network level, and we are using some of these within ProtEx for the analysis of systems with both network and endsystem elements.

It is important to note that we do not envision workload re-characterization as a replacement for regulation. Rather, it is complementary and can be naturally combined with it,

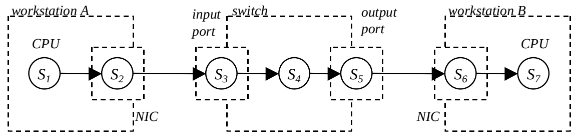

**Figure 1. Simple Server Graph**

allowing for an integrated analysis methodology.

**Flexible Resource Modeling.** A model for processors must reflect that resources are not ideal. (i) Service to particular tasks may be interrupted or delayed because of various forms of priority inversion. (ii) Processors may be controlled by a variety of different scheduling policies. (iii) A modeling methodology must lend itself to effectively and accurately model hierarchical compositions of processors, be this a collection of processors, or processors in combination with operating system layers incorporating resource managers, or processors controlled by multi-level schedulers.

To achieve this, we make available a rich variety of different server types, for example FIFO, static-priority, EDF, and others). In addition, we generalize the concept of service rate, which is traditionally used in processor modeling. We allow processors to be modeled by service functions, a well-known method to model service to traffic streams in networks [12]. In addition, we take advantage of the flexible workload modeling described above as a means to compose systems with multiple different server types.

## 3. Schedulability Analysis in ProtEx

### 3.1. System Model: Server Graphs

For analysis purposes, the system is decomposed into its basic resource components, which we call *servers*. Some servers can be mapped onto real hardware components (such as CPU, I/O ports, busses,) while others are logical in nature (such as workload regulators or servers for critical sections). We describe the collections of resources in form of a server graph, whose $m$ nodes represent the available servers $S_1, S_2, \ldots, S_m$, and the edges describe the connectivity among servers.

Figure 1 depicts a possible representation of a server graph of two workstations connected by an ATM switch. In this example, we model the ATM switch as a collection of input ports ($S_3$), the switch fabric ($S_4$), and a collection of output ports ($S_5$). Each worksthation is described by a server representing the CPU and possible memory management and DMA machinery to the network interface card ($S_1$ and $S_7$) and one server representing the network interface card and the link to the ATM switch ($S_2$ and $S_6$). System designers can model systems at higher levels using hierarchical compositions of server graphs.

We distinguish three classes of servers, depending on how they affect the analysis:

*Constant Delay Servers:* The delay for such servers is independent of other workload on the server. Examples are physical links in a switched network or non-blocking fabrics in switches. For delay computation purposes, constant delay servers can be easily eliminated by appropriately adapting the end-to-end delay requirements of the workload: After the deadline for each workload that uses a particular constant delay server is reduced by the constant delay added by that server, the server itself can be deleted from the server graph.

*Variable Delay Servers:* The amount of delay offered by a variable delay server to a particular workload depends on all the workload on that server. Virtually all servers for which contention can occur belong to this class, and delay analysis deals mostly with variable delay servers. Examples are CPUs and output ports in switches with output port queuing. Variable delay servers may perturb the workload, typically making it more bursty, as it leaves the server and proceeds to the next. If no workload regulator is in place at the next server, this increase in burstiness of workload arrival must be taken into account during the analysis.

*Regulator Servers:* These server can be used to model workload regulators, for example periodic or sporadic servers to handle sporadic workloads on processors, or traffic shapers on switches or routers. Similarly, different processor synchronization methods in end-to-end systems [10] can be realized through regulator servers.

The system represented in Figure 1 consists of four variable delay servers ($S_1$, $S_5$, $S_6$, and $S_7$) and three constant delay servers ($S_2$, $S_3$, and $S_4$). There are no regulator servers in this example. The connectivity among components is described using ports. Figure 1 illustrates how the first workstation is connected to the switch by connecting the two ports `ws1.port1` and `switch.portIN`. Figure 2 gives a textual representation of the same server graph. Two server classes are defined (`workstation` and `ATMswitch`), and their instantiations (`wsA`, `wsB`, and `switch`) are then connected using the appropriate ports.

### 3.2. Workload Characterization: Task Graphs and Arrival Functions

We model the workload as a set of n tasks $T_1, T_2, \ldots, T_n$, independently of whether it is computation workload in the end systems or routers, or traffic in the network. Each task $T_i$ consists of a (typically infinite) sequence of invocations. All invocations of portion $j$ of Task $T_i$ form the subtask $T_{i,j}$. We say that $T_{i,j}$ executes on Server $S_{i,j}$. Each subtask has a worst-case execution time of $e_{i,j}$ time units, meaning that each invocation executes for

```
SERVERGRAPH sgexample:   # Two workstations connected using a single switch

# Definition of component classes
CLASS SERVERGRAPH workstation   # Definition of the workstation component
  SERVER cpu:
    TYPE = VARIABLE;  POLICY = static_priority; # other parameters...
  END;
  SERVER nic:
    TYPE =CONSTANT;  # other parameters...
  END;
  # Definition of the connectivity within the workstation
  cpu  -> nic;
  nic  -> port1
END;

CLASS SERVERGRAPH ATMswitch    # Definition of the ATM switch component
  SERVER input_port:
    TYPE = CONSTANT; # other parameters...
  END;
  SERVER switch_fabric:
    TYPE =CONSTANT;  # other parameters...
  END;
  SERVER output_port:
    TYPE =VARIABLE;  POLICY = FIFO; # other parameters...
  END;
  # Definition of the connectivity within the switch fabric
  port1         -> input_port;  input_port  -> switch_fabric;
  switch_fabric -> output_port; output_port -> port2;
END;

# Definition of the instances
SERVER wsA, wsB OF CLASS workstation;
SERVER switch   OF CLASS ATMswitch;

# Definition of connectivity for the server graph using ports
wsA.port1 -> switch.portIN;  switch.portOUT -> wsB.port1;
```

**Figure 2. Example Resource Graph**

no more than $e_{i,j}$ time units on $S_{i,j}$.

An invocation of task $T_i$ on a server can trigger one or more invocations on one or more subsequent servers. Subjobs belonging to the same invocation are therefore in a dependency relation to each other that can be represented by a directed graph, which we call the task graph $G_i$ for a given Task $T_i$.

In order to allow for specification of non-periodic tasks and for uniform description of arrivals of tasks to servers in the system, we generalize the traditional periodic workload model by using *arrival functions*. The arrival function $F_{i,j,ARR}(t)$ of subtask $T_{i,j}$ is defined as the maximum number of invocations of $T_{i,j}$ released during any interval of length $t$. A strictly periodic task arrival would therefore be represented by the arrival function $F_{i,j_ARR}(t) = \lceil t/p \rceil$, where $p$ is the period of the task. Using this notation, the arrival function of Task $T_i$ is $F_{i,1,ARR}(t)$. Arrival functions thus provide a deterministic, time-invariant, way to bound general arrivals of tasks to the system.

## 3.3. Schedulability Analysis

A number of different methods can be used to analyze the overall system, and the designer can pick the most appropriate method depending on the types of servers in the system and the system topology. All method rely explicitly or implicitly on the same approaches to analyze single-server systems. These approaches are then expanded to allow the analysis of end-to-end systems.

### 3.3.1 Single-Node Analysis

A number of delay formulas exist for workload that is defined by general arrival functions for a number of servers types. In its most general form, the delay $d_{k,j}$ for task $T_k$ on Server $S_j$ is given by the following formula:

$$d_{k,j} = max_{m>0}(F^{-1}_{k,j,DEP}(m) - F^{-1}_{k,j,ARR}(m)) \ ,$$

where $F_{k,j,ARR}$ is the arrival function defined earlier, and $F_{k,j,DEP}$ is the equivalent departure function of Task $T_k$ from Server $S_j$, that is, the maximum number of invocations of Task $T_k$ finishing on Server $S_j$ during any time period $t$. In its most general form, the departure function $F_{k,j,DEP}(t)$ can be derived from the arrival functions $F_{;j,ARR}(t)$ of all tasks on the server, and the service functions $S_{k,j}(t)$ of the server: $F_{k,j,DEP}(t) = \lfloor S_{k,j}(t)/e_{k,j} \rfloor$. The service function $S_{k,j}(t)$ for Task $k$ on Server $j$ specifies the minimum amount of service Task $k$ receives over any interval of length $t$. In order to use these general formulas, the service function must be derived for each server type. Such service functions exist for FIFO and Preemptive Static-Priority Servers. Approximations exist for Non-Preemptive Static-Priority Servers [3]. For the various realizations of Generalized Processor Sharing Servers, the derivation of these functions is straightforward.

The use of these elaborate formulas is only necessary when the task arrival is bounded by a general arrival function. When tasks can be modeled as periodic, for example traditional time-demand analysis can be used to determine the local delay at a server.

### 3.3.2 End-to-End Analysis

The simplest form of end-to-end analysis partitions the system into isolated servers, computes the local delay on each server, and then computes the end-to-end delay by summing up all local delays along the critical path of a task. This method is called *Decomposition-Based Analysis*, and has been first described in [11].

In order to compute the local delays at a server, the arrival functions for all tasks at that server must be known. These arrival functions are identical to the departure functions on the previous servers. Decomposition-based analysis can therefore easily be performed after the servers have been topologically ordered as defined by the task graph. If the task graph contains cycles, an iterative approach is used that terminates whenever the solution converges or when a deadline is missed.

Decomposition-based analysis is simple and suitable for systems with arbitrary topologies and server types. The drawback of this method is that it tends to overestimate the end-to-end delay suffered by the traffic. This is because it assumes that a task suffers the worst-case delay at every server along its connection path [4].

Better methods exist for special cases of workloads and servers. If service functions for all servers in the system exist, servers can be clustered by convoluting the service functions of the individual servers to generate service functions of aggregated servers [12]. The end-to-end analysis is then performed by performing a single-server analysis on the aggregated server. We call this method for end-to-end analysis the *Service Curve method*.

Servers can be aggregated in special cases even when no service functions are provided. The *Integrated Analysis method* described in [4] aggregates pairs of FIFO or Static-Priority servers, and can be used to significantly improve the performance of decomposition-based analysis.

## 4. Related Work

A number of prototyping and schedulability analysis tools for distributed real-time systems exist. We elaborate on two: Tri-Pacific offers a product suite of tools that include Rapid Rma, Rapid Sim, and Rapid Build [7]. This toolkit is based on PERTS [14], and uses a RMA approach. It allows the designers to test, simulate, and execute software models against various design scenarios and evaluate how different implementations might optimize the performance of a system.

TimeSys' TimeWiz [8] is another schedulability analysis tool that allows the user to build prototypes and validate them before implementation by analyzing and simulating the timing behavior of the system. This tool can analyze real-time applications to be run on network elements. As for TriPacific's toolkit, TimeWiz is based on the Rate Monotonic Analysis.

The main strength of these tools is that they provide convenient mechanisms for integrating and using multiple different tool characteristics, such as workload extraction and analysis, into a single development environment. These tools also offer support for end-to-end analysis and simulation. However, they focus on a single schedulability analysis methodology approach, namely Rate Monotonic Analysis.

## 5. Conclusion

The initial ProtEx toolkit version has been developed to establish an infrastructure for large scale distributed real-time system prototyping and analysis. Resource and workload definitions with general characterization is a central aspect of the tool. The real-time software designer can use varying workload representation through service and arrival curves and specific schedulability analysis methodologies.

We have shown earlier [3, 4] that this type of delay computation along with an appropriate methodology such as decomposition-based or integrated-based analysis generates excellent results for the schedulability analysis in terms of worst-case execution time and system utilization. Additionally, we have also built a framework that allows the user to hierarchically define and use resources and tasks for a given real-time application. Through clearly defined software module de-coupling, our tool is scalable for large scale distributed real-time system analysis.

## References

[1] T.Abdelzaher, *et al*. "ARMADA Middleware Suite." Proceedings of the IEEE Workshop on Middelware for Distributed Real-Time Systems and Services. San Francisco, December 1997.

[2] V.Fay Volfe *et al*. "Real-Time CORBA." Proceedings of the Third IEEE Real-Time Applications Symposium, June 1997.

[3] C. Li, R. Bettati, W. Zhao. "Response Time Analysis for Distributed Real-Time Systems with Bursty Job Arrivals." Proceeding of ICPP, 1998

[4] C. Li, R. Bettati, W. Zhao. "New Delay Analysis in High Speed Networks." Proceedings of ICPP, 1999.

[5] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact Admission Control in Networks with Bounded Delay Services," *IEEE/ACM Trans. Networking*, vol. 4, pp 885–901, Dec. 1996.

[6] A. Raha, S. Kamat, W. Zhao. "Guaranteeing End-to-End Deadlines in ATM Networks." Proceedings of the IEEE International Conference on Distributed Computing Systems, May 1995

[7] "Tri-Pacific Software Inc" Real-Time Scheduling Solutions. 31 Mar. 2000.
URL http://www.tripac.com/

[8] "TimeSys Corporation" Real-Time - Real Solutions. 31 Mar. 2000.
URL http://www.timesys.com/

[9] S. Ramos-Thuel and J. Lehoczky. "On-Line Scheduling of Hard Deadline Aperiodic Tasks in Fixed-Priority Systems." Proceedings of the IEEE Real-Time Systems Symposium, Phoenix, AZ, December 1992.

[10] J.Sun and J.W.-S. Liu. "Synchronization Protocols in Distributed Real-Time Systems." Proceedings of the International Conference on Distributed Computing Systems. Hong Kong, May 1996.

[11] R.L. Cruz. "A Calculus of Network Delay, part I,II: Network Analysis." IEEE Trans. on Inform. Theory, 37(1), Jan 1991.

[12] R.L. Cruz. "Quality of Service Guarantee in Virtual Switched Network." IEEE Journal on Selected Areas in Communication. Vol 13, no.6, 1995.

[13] R. Rajkumar. "Synchronization in Real-Time Systems - A Priority Inheritance Approach." Kluwer Academic Publishers, 1991.

[14] J. W.-S. Liu, *et al*. "PERTS: A Prototyping Environment for Real-Time Systems." Proceedings of the Real-Time Systems Symposium, Raleigh-Durham, N.C., Dec. 1993.