# Real-time CORBA performance on Linux-RT_PREEMPT

**Manuel Traut**
Linutronix GmbH
Auf dem Berg 3, 88690 Uhldingen, Germany
manut@linutronix.de

**Abstract**

Automation technology lacks an established platform independent, high-level, object oriented real-time capable communication standard, which is based on standard Ethernet hardware and drivers. ACE/TAO is an Open Source implementation of the OMG Real-Time CORBA Specification and might fill this gap. It is designed platform independent, implemented in C++ and provides a standardized communication framework. Real-Time CORBA is already used in industrial environments, e.g. aircraft, naval equipment and others. This paper explains the basics of the ACE/TAO framework and its usage in industrial communication. On the basis of a real-world example - transmission of an 1 KiB data frame - two communication methods are evaluated: the RT-CORBA Remote Procedure Call and the TAO Real-Time Event-Channel. The performance measurement methods are explained in detail. Measurement results under various system loads and a comparison of ACE/TAO on top of a vanilla Linux kernel and a RT_PREEMPT enabled Linux kernel provide a meaningful insight in the capabilities of RT-CORBA. Finally, the paper provides an analysis of functionality which needs to be improved in the operating system to provide real deterministic communication through a standardized framework.

## 1 Introduction

The main communication infrastructure in industrial automation is based on a broad variety of industrial fieldbusses based on CAN, RS422 and RS485 physical layers. Since 2005 ethernet based communication is gaining popularity. Aside of this the usage of PC based hardware with realtime operating systems becomes more wide spread. The systems provide the usual set of interfaces and capabilities known from the IT world with specialized software for machine controls.

Combining PC based hardware and ethernet based real-time capable communication methods utilizing object-oriented middleware could provide a couple of advantages:

- fast development cycles
- ease of maintenance
- flexible reusability of software modules
- connection handling is done by the middleware
- ...

ACE/TAO, a Real-time CORBA implementation, and the RT_PREEMPT patch for the Linux kernel provide such a base environment today.

### 1.1 CORBA

CORBA is a middleware, which allows RPC[1]-based IPC[2] between different operating systems and different programming languages (Figure 1).

The communication interfaces are defined in IDL[3]. The IDL files are compiled into, e.g. c++, java, ..., code which does the (de)serialization of the datatypes. The interface implementations (CORBA objects) are registered with language specific ORB[4]s. Each CORBA process owns one ORB, which handles

[1] Remote Procedure Call
[2] Inter Process Communication
[3] Interface Definition Language
[4] Object Request Broker

the function requests and returns the calculated values.

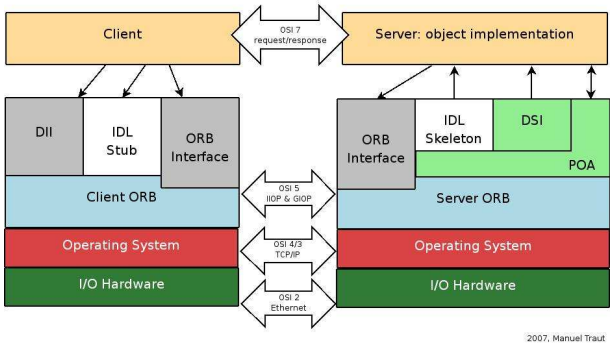


**FIGURE 1:** *CORBA Architecture*

## 1.2 Real-Time CORBA

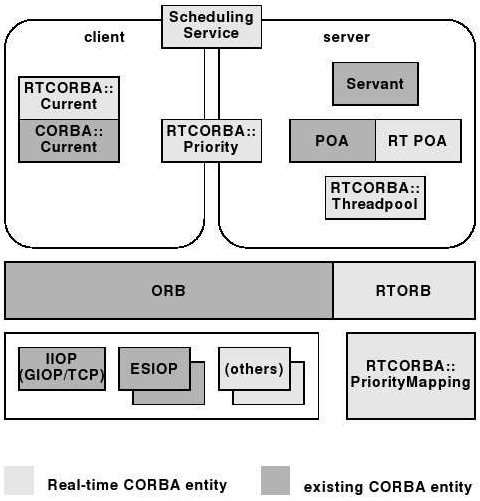


**FIGURE 2:** *Real-time CORBA (source: [1])*

As shown in figure 2, a real-time capable ORB extends a standard ORB with the following features: locating objects in constant time, preallocation of resources, operating system independent priority handling, priority based scheduling.

## 1.3 ACE/TAO

ACE is an open-source c++ framework for platform-independent system- and network-programming. TAO is a Real-time CORBA implementation build on top of ACE (Figure 3).

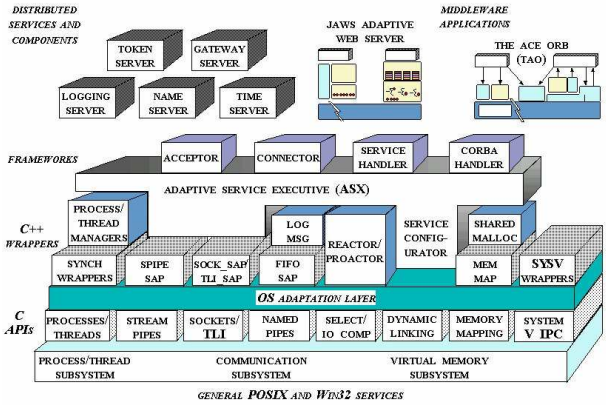


**FIGURE 3:** *ACE/TAO framework (source: [2])*

The ACE/TAO package is available for all important operating systems. The framework can be trimmed for embedded systems: Each application described in this paper consumes less than 1 MByte of RAM. Also the consumed CPU time is suprisingly low.

# 2 Performance Measurements

The measurements were made on embedded systems (Intel Mobile CPU 600 MHz, 512 MB RAM, Intel e100 NIC) with digital I/0 ports. A square-pulse generator is connected to a digital input of the *supplier system* and to channel 1 of the oscilloscope. The digital outputs of the *receiver systems* #I and #II are connected to channel 2 and 3 of the oscilloscope.

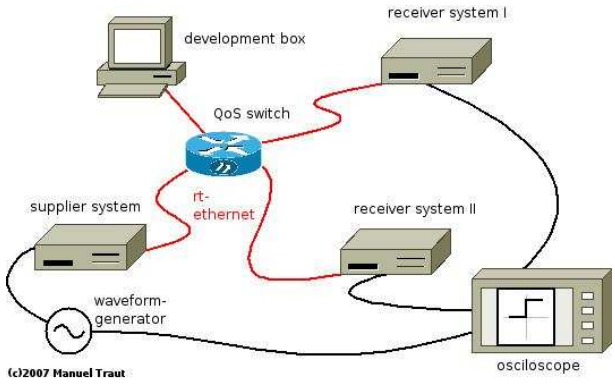


**FIGURE 4:** *Measurement environment*

## 2.1 RPC

Each receiver hosts an object, for writing values to its digital output:

```
module benchmark{
```

```
  interface Put{
    void Port( in short portNo,
               in short value,
               in string data   );
  };
};
```

The *Port* function is called by the *supplier* as soon as the state of one of its digital inputs changes. The real end to end latency is measured with the oscilloscope.

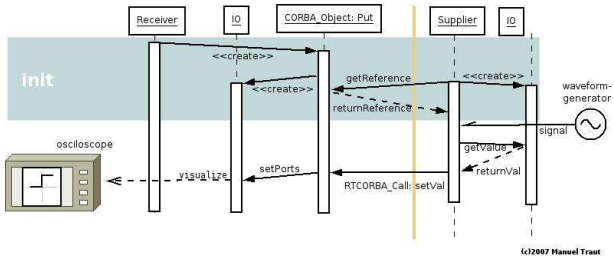


**FIGURE 5:** *RPC measurement: sequence diagramm*

Figure 6a shows a latency histogram measured on system running a RT_PREEMPT enabled kernel. Extra system load is generate by heavy disk I/O, network traffic on non-prioritized NICs and five low priority CPU hogs. The RPC *data* string has zero length.

The latency is the total time of the RPC execution. The RPC execution is triggered by the rising and the falling edge of the square-wave generator connected to the digital input of the supplier system. The RPC results in toggling the output on the receiver system. The input and the output are monitored by a digital oscilloscope, which provides histogram generation functionalities.

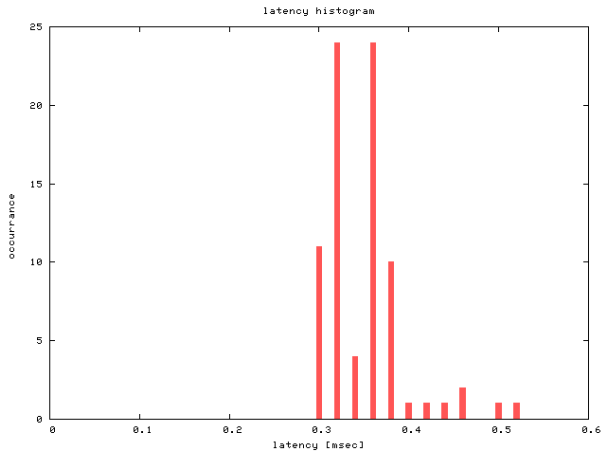


**FIGURE 6:** *Latency histogram RT_PREEMPT, system load*

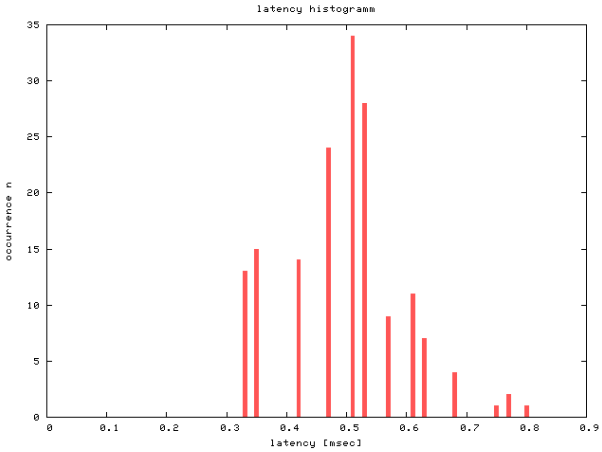


**FIGURE 7:** *Latency histogram Vanilla, without system load*

The results of the same measurement on a none RT_PREEMPT enabled kernel are shown in figure 6b (without system load) and 6c (with system load).

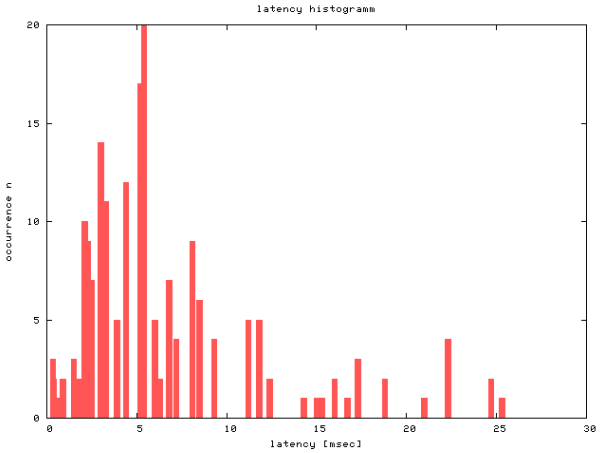


**FIGURE 8:** *Latency histogramm Vanilla, system load*

The same measurement under identical system load was made once again with both receiving systems active. Receiving system #I hosts the object for setting port values with lower priorization than receiving system #II. On the supplier system the square-waveform generator is connected to two digital input ports. Changes on the first port are commited to the higher prioritized *receiver* on system #II; changes on the second port are sent to the lower prioritized receiving system #I. Figure 7 shows, that no priority inversion occours.

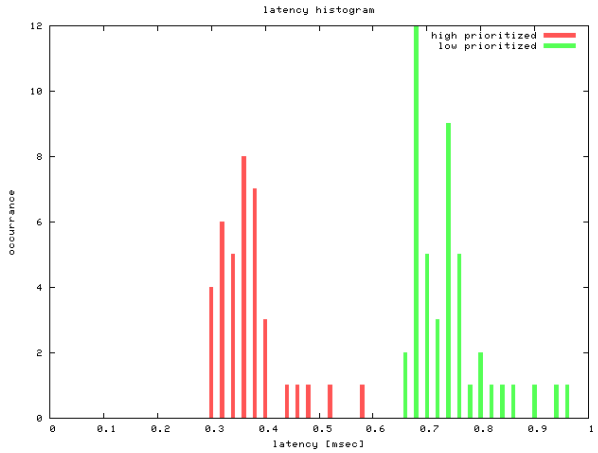


**FIGURE 9:** *Latency histogramm, two priorizations*

To simulate a higher data transmission rate, the parameter *in string data* is read in from a text file, so its length can be changed after compile time by editing the text file. Figure 8 shows the larger the process data image is the larger is the difference between immediate and worst case latency.

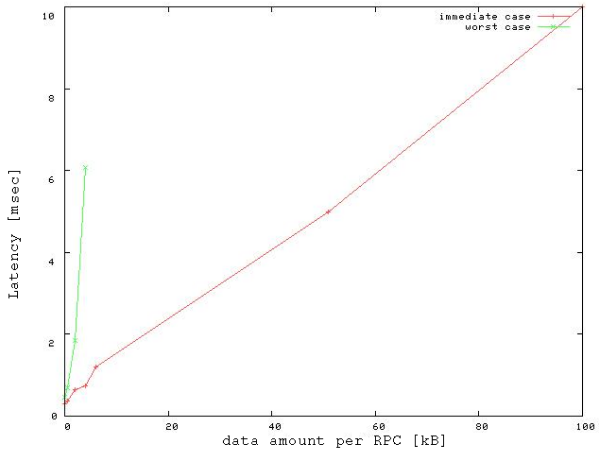


**FIGURE 10:** *dependency between process data image size and latency*

## 2.2 TAO Real-time Event-Channel

The TAO Real-time Event-Channel is a Messaging Service. Suppliers are sending messages to the Messaging Service. A client can subscribe for messages at the Messaging Service.

As soon as the value of the digital input of the Supplier changes, the new value of the digital input port is send to the Messaging Service. The Messaging Service sends this value, to all subscribed clients. The clients write the values from the messages to the digital outputs.

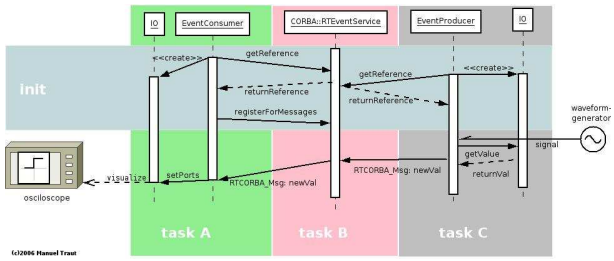


**FIGURE 11:** *Event-Channel measurement: sequence diagramm*

The TAO Real-time Event-Channel is an additional CORBA application the data has to pass, so the latency should be approximately two times the latency of the RPC measurement. Figure 10 shows, that this is a correct assumption.

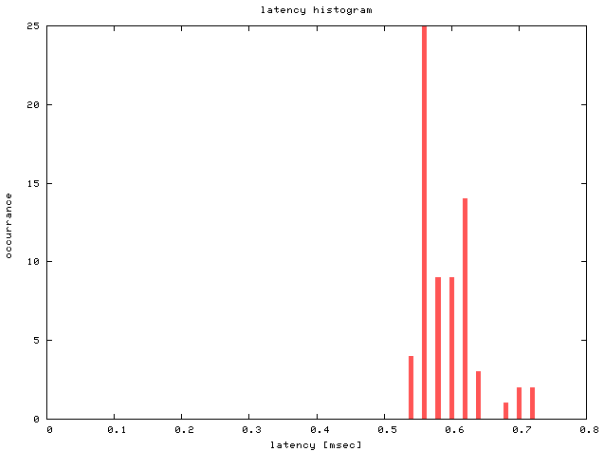


**FIGURE 12:** *Latency histogramm, TAO RT Event-Channel*

# 3 Needed operating system functionalities

A TAO ORB can only schedule its CORBA requests correctly when the underlying operating system has real-time capabilities with a deterministic scheduling latency. The programming model expects a priority based scheduling algorithm. The operating system latencies of Linux can be tested with cyclictest. cyclictest is an utiliy which determines the deviation of the expected time line of a periodic timer. This takes the full chain of timer interrupt, scheduler invocation, context switch and return to the user space application into account. On a given test system the maximum latency with a RT_PREEMPT enabled kernel was 26$\mu s$, with a vanilla kernel the maximum latency increased to 39.6$ms$.

Another requirement for deterministic communication is the priorization of device I/O. Right now

this is not fully implemented in the RT_PREEMPT kernel. The priority of interrupt service handlers is only configurable per interrupt line, which causes problems if the interrupt line is shared between a high priority and a low priority device. The same applies for the networking soft interrupt which handles all network interfaces in the same queue.

The kernel community has already recognized the importance of real-time networking capabilities and solutions for this problem are already discussed.

Not all NICs (especially their firmware and drivers) are suitable for real-time networking. Many modern NICs don't request an interrupt for each received package. They only request an interrupt if a defined amount of data is received, or a defined timeout is over. These cards cause high latencies especially when the transfered data packages are small.

# 4 Conclusion

Real-time communicaion based on object-orientated middleware over standard ethernet hardware is a promising solution. Various problems have been identified, but resolving those is not trivial.

Interestingly enough of these problems are not restricted to the communication requirements of the automation industry. The increasing demands on deterministic networking for other application areas e.g. telecommunication provide additional momentum for improving the deterministic behaviour of network communication in the Linux kernel.

ACE/TAO and RT_PREEMPT enabled Linux are providing a useful and solid environment today with further improvements in the foreseeable future.

[3] is a good resource for further informations.

# References


[1] *Real-time CORBA Specification*,2005, OMG

[2] *Overview of ACE*,2007
*http://www.cs.wustl.edu/schmidt/ACE-overview.html*

[3] *TAO technical documents*,2007
*http://www.cs.wustl.edu/schmidt/corba-research-realtime.html*