Predictibility of Inter-component latency in a Software Communications Architecture Operating Environment

Gael Abgrall*, Frédéric Le Roy*, Jean-Philippe Diguet†, Guy Gogniat† and Jean-Philippe Delahaye‡
*UEB, ENSIETA / DTN, 2 rue Francois Verny, 29806 Brest Cedex 9, France
{gael.abgrall, frederic.le_roy}@ensieta.fr
†UEB, UBS / Lab-STICC, BP 92116, 56321 Lorient Cedex, France
{jean-philippe.diguet, guy.gogniat}@univ-ubs.fr
‡DGA / CELAR, La Roche Marguerite, 35174 Bruz, France
jean-philippe.delahaye@dga.defense.gouv.fr

***Abstract*—This paper presents an in-depth analysis of the behavior of a SCA component-based waveform application in terms of "inter-component" communication latency. The main limitation with SCA, in the context of embedded systems, is the additional cost introduced by the use of CORBA. Previous studies have already defined the major metrics of interest regarding this issue, these are CPU cost, memory requirements and "inter-component" latency. Real-time systems can not afford high latency, in consequence, this paper focuses on this metric. The starting point of this paper is the desire of knowing if the SCA CF does not also bring an overhead. Measurements have been realized with OmniORB as CORBA distribution and OSSIE for SCA implementation. In order to perform these measurements, a SCA waveform composed of several "empty-components" have been created. "Empty-components" are software components compliant to SCA without any signal processing part. The study only focuses on communications between components. The same kind of "inter-component" link has been measured between two components using CORBA without SCA. It is possible to compare the latency values between the two measurements and to show as a result that they are approximately the same. The CORBA bus is really the part which brings an overhead to the system. The final part of this paper introduces a statistical estimation of the latency distributions. It results from measurements performed with various data packet sizes and uses a fitting method based on a combination of Gaussian functions.**

***Keywords*-Software Radio; SCA; CORBA; OSSIE; OmniORB; Latency**

I. Introduction

Current radio communications are evolving to a new kind of software and hardware design. From this perspective, Software Radio (SWR) appears to be the future of the radio communication domain. J.Mitola has defined SWR in [1] as a radio whose channel modulation waveforms are defined in software. Waveforms are generated as sampled digital signals, converted from digital to analog via a wideband Digital to Analog Converter (DAC) and then possibly upconverted from intermediate frequency (IF) to radio frequency (RF). Similarly, the receiver is based on a wideband Analog to Digital Converter (ADC) that captures all of the channels of the software radio node. The receiver then extracts, downconverts and demodulates the channel waveform using software on a general purpose processor. However, current technological issues limit the concept of ideal SWR defined by Mitola. One of the key technological issues is the ADC and DAC converters to reach the high speed and high resolution needed for the high spectrum bandwidth digitalization expected for the SWR digital front end. Another issue concerns the embedded signal processing units, which today are not powerful enough to process RF digitalized data. Nowadays, the technology allows neither developers, nor users to fully exploit SWR due to analog to digital rate restrictions, so it is more appropriate to talk about Software Defined Radio (SDR).

During the past decade several groups have been working on SDR in both military and civil domains. In the civil domain, many telecom operators already use SDR for managing base stations. But they all have developed their own architectures that cannot be published for intellectual property reasons. The main advance in standardisation of the software radio architecture platform appears in the military domain. The JTRS (Joint Tactical Radio System), a program of the US Department of Defense (DoD), has standardized a software architecture for software radio called SCA (Software Communications Architecture) [2], [3]. The concept of this architecture is to optimize radio platform interoperability, software component reusability and portability. The SCA standard defines the software architecture of an Operating Environment (OE) that should be used to optimize radio nodes interoperability. The OE architecture is based on COTS technologies. This architecture is based on the software bus CORBA (Common Object Request Broker Architecture) [4]. Several distributions of OE exist, some of them are free (OSSIE [5], Scari Open) and some are not (Harris dmTK, Scari++).

In SDR systems designed with SCA, processes, methodologies, metrics for performance measurement of the embedded software and the middleware layer as well as the application, are essential to characterize the system mainly based on software. It will also enable the verification of the software behavior and enable the performance enhancement of multi-components distributed software applications. One of the main parameters that impacts SCA SDR system performances is the delay introduced by CORBA communications. This time is a key feature regarding QoS (Quality of Service) especially in real-time systems. This paper is focused on this essential aspect.

Thus our objective, in this study, is to characterise the

978-1-4244-6534-7/10/$26.00 ©2010 IEEE

software component latency behavior of the SCA based radio system software, which is one of the key issues in SCA application performances.

In section II, we give a description of the software used in this study. Then, in section III, we describe our process of latency measurement introduced by the SCA related software and in section IV, we give the corresponding measurement results. Section V presents the main research contribution based on previously described framework and results. This is a statistical model for the estimation of latency within a SCA system. The input of the model is the number of components and channels that can vary according to waveform and use cases. Finally, we conclude and give some perspectives.

II. Software Description

This section gives an overview of a SCA system specifying the software architecture requirements and software components responsible for the deployment, configuration, and control of the waveforms on the hardware. The software infrastructure provided by SCA for the distributed application deployment is based on the CORBA software bus.

A. CORBA

CORBA is a standard defined by the Object Management Group (OMG), which allows software components written in different languages and potentially running on several hosts to communicate with each other. It is based on an Object Request Broker (ORB). An ORB acts as the "glue" for distributed applications running on one or more processors [6]. The most popular ORB are TAO and OmniORB. CORBA is not the only software architecture, there are two other well-known software which are Java RMI (Remote Method Invocation) and Microsoft RPC (Remote Procedure Call). Comparisons between these middlewares have already been performed in [7], [8]. Initially CORBA was specified for large distributed systems consisting of hundred of networked computers in application domains, such as banking and business. CORBA was defined without real-time considerations; communications are asynchronous implying lack of completion. But it provides most of the services required for waveform deployment, configuration and control. Today the adaptation of CORBA to the world of embedded radio systems is still a challenging issue for software industries. For instance, some of the computational resources used in SDR radio such as DSP and FPGA were not capable of CORBA until recently and still need a lot of improvement to reach services available on GPP.

B. SCA

1) Overview: The Software Communications Architecture defined by the JTRS specifies an Operating Environment. It allows for the abstraction between software and hardware. In an ideal view, any waveform (application) can be installed on any platform. The waveform development is a component based software design. For maximum re-use and re-configuration, the SCA defines waveforms and platforms as a set of interconnected components. These components can be re-used and are independent. They encapsulate their behavior and provide certain functionality, exposed through interfaces. This system is divided into three main parts (Figure 1):

- a Core Framework (CF),
- a CORBA ORB,
- an Operating System.

The CORBA ORB and the Operating System depend on the underlying hardware system but all the Core Framework must have the same function. It provides:

- a collection of services used by the waveforms and the other applications,
- a software, which enables the installation, the configuration, the management and the control of waveforms,
- a file system to manage the waveforms,
- hardware interfaces to enable the abstraction of the platform.

As shown in Figure 2, the CF is divided in components. It is also important to know that every entity in SCA has a Universally Unique Identifier (UUID) as defined by the DCE UUID standard adopted by CORBA [2], [4]. This UUID allows for the identifying of every entity when the CF discovers the platform resources (hardware devices and software components) using the CORBA services.

2) SCA waveforms development tools: Due to the complexity of waveform development with SCA (CORBA Design Component and the respect to more than 600,000 of requirements from the SCA specification), it is very difficult to guarantee no syntax errors when creating SCA formatted files without a CAD tool. Two such tools are available today for improving SCA design time:

- SCA Architect by CRC,
- Spectra CX by PrismTech.

Both allow developers to create, manage and run waveforms on a platform.

C. OSSIE

OSSIE is an open source SCA Open Environment developed by the Wireless Team at Virginia Tech [5] and working with the CORBA ORB OmniORB. This open platform is available for workstations running under Linux. The full installation of OSSIE provides some drivers for peripherals (Sound card and Universal Software Radio Peripheral (USRP) [9]) but they are not in the scope of this study. Some development tools are provided too: WaveDev for instance allows developers to generate the entire skeleton of the C++ files needed by the SCA architecture to run a component.

To run a waveform under OSSIE, three commands are necessary:

- "omniNames": starts the ORB (also present for the OmniORB measurement),
- "nodeBooter": runs the Device Manager (beginning of all SCA CF boot up sequence),
- "wavLoader": loads waveforms into the system.

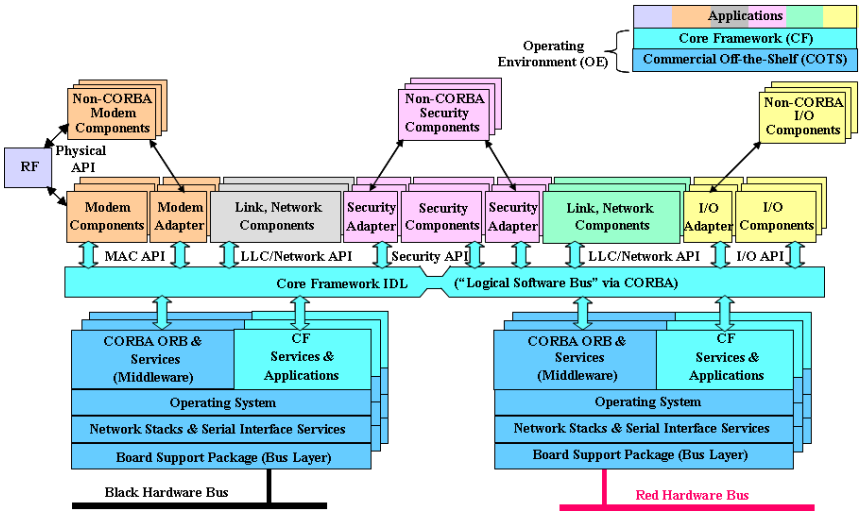

Figure 1. SCA Software Structure

Some new features are now available with OSSIE (like the Eclipse Plug-In to create and manage components and waveforms) but this study did not used them.

III. Latency Profiling

Managing a waveform with distinct components brings the need of using a protocol for "inter-component" communications. CORBA bus has been held back to realize this function. But adding communication layers in a system will inevitably increase the time required to bring data from one component to another. Time measurements on software bus have already been done in different ways [10]–[12]. The main difference between this approach and the others ones is the presence of signal processing components. These measurements are specifically focused on the communication overhead. The measurements realized in this paper do not include signal processing, the only metrics which is analysed is the inter-component latency.

A. Measurements without SCA

The first realised measurement does not use OSSIE but only OmniORB to clearly show what CORBA really gives to a SCA OE. CORBA is based on a Client/Server architecture, so the application consists of a client requesting services on a server (both of them are executed by the same computer). The server response is a string of randomised characters, which has a defined length identical for OSSIE measurements. To measure the time elapsed between emission and reception, an assembly command can be used: RDTSC (ReaD Time Stamp Counter [13]). This counter is incremented at the CPU frequency speed, so the result of the measurement has a good accuracy on actual processor (i.e. more than 1 GHz). The measurement process is shown in Figure 3. The latency measurement is realized by calculating the time difference between step 3 and step 5.

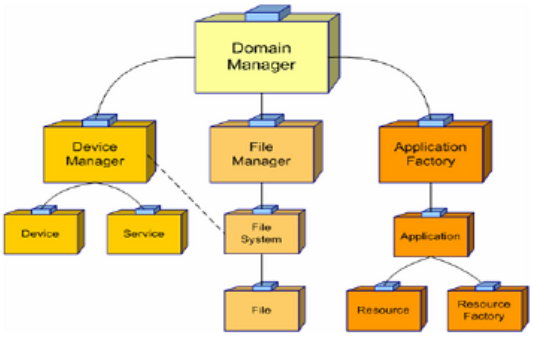

Figure 2. SCA Architecture [3]

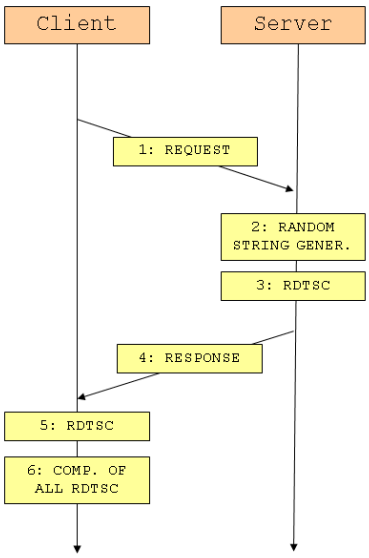

Figure 3. OmniORB measurement process

B. Measurements with OSSIE

Measuring latency with OSSIE is realized in the same way as for OmniORB measurements. OSSIE allows the creation of waveforms. Multiple tests are possible by creating several waveforms. All these cases allow for the

measurement of the inter-component latency with different configurations. The idea is to create multi channels and/or multi inter-component links as proposed in Figure 4, where OmniORB is limited to one channel and one inter-component link in this study. All the waveforms are composed of several components:

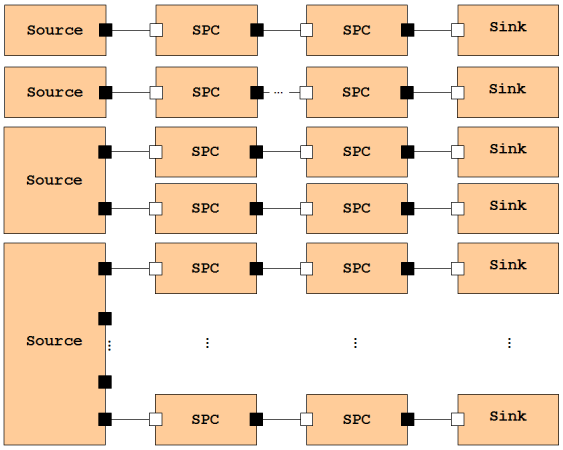

Figure 4. Waveforms created for the measurements

- a source to generate the packet to transmit,
- at least two ”SPC” (Signal Processing Component), these components are here to simulate the introduction of signal processing stages but because of the interest of this study, there is no signal processing in it,
- one or more sinks (one per channel) to receive the data.

An OSSIE component is composed of several parts as shown in Figure 5:

- the main part where all the signal processing are done,
- the ”communication” part where data are exchanged between other components, this part consists of two functions: ”Get()” for receiving data and ”Push_Packet ()” to transmit data.

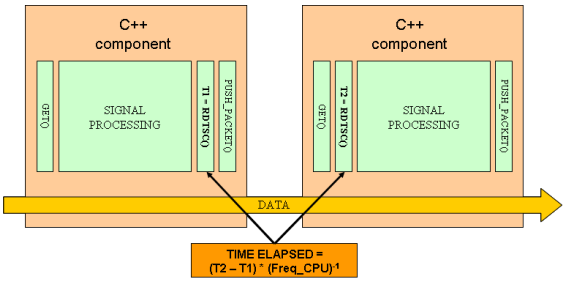

Figure 5. OSSIE Component Architecture and measurement process

Introducing RDTSC measurements just after the Get function and just before the Push_Packet function gives the elapsed time to transfer the data from one component to another.

The results of all these measurements provide a better knowledge of the waveform behavior with a complete

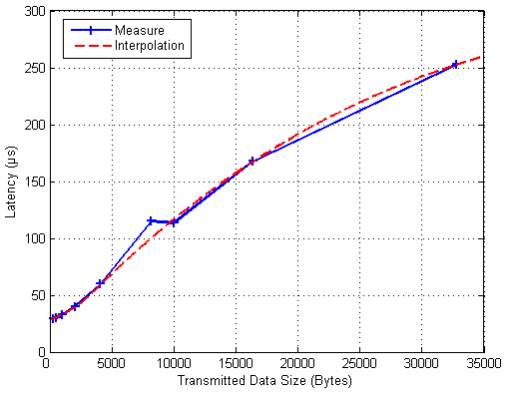

Figure 6. OmniORB latency measurement

abstraction of signal processing stages. The behavior of the system might be totally different depending either on the number of components in a channel or on the number of channels itself.

IV. Results

All this study has been performed on the same computer (Intel Pentium 4 @ 3 GHz with 1 GB of memory) running with Linux Fedora Core 6. Versions of OmniORB and OSSIE were 4.1.0 and 0.6.1 respectively. Note that new versions for both softwares since these measurements have been taken. This section presents the results obtained for both OmniORB and OSSIE. A comparison between these two solutions is also given. A finer analysis is shown in the next section.

A. OmniORB Results

The measurement process has been explained earlier in this paper. Measurements have been realized for different sizes of transferred data. OSSIE works with packets, for this study, a ”RealShort” packet type has been chosen. It corresponds to 2 Bytes. In OmniORB, this kind of abstraction does not exist, the possible data types are the standardized data contained in the C++. To have the same amount of transmitted data in both measurements, a string of chars (1 Byte) has been sent with size equivalent to an ”OSSIE packet”. For example, an OSSIE packet of 8,192 samples (RealShort type, 2 Bytes) is equivalent to a string of chars with 16,384 chars transmitted on OmniORB in pure CORBA packet string.

This measurement has been done with transmitted data size of 256 Bytes to 32,768 by increasing in power of 2 and also with 10,000 Bytes to have one point out of the powers of two. The data rate is set to 320,000 ”packets” per second, this value is the same for all the measurements. This rate has been chosen to guarantee the work integrity with the previous studies realized [11], [12]. Other rates have been used to make some comparison in OSSIE but the main part of the measurements is only with 320 k samples per second (kSps). Every set of measurements is on 5,000 samples. Once the mean calculated, a curve can be exploited as shown in Figure 6.

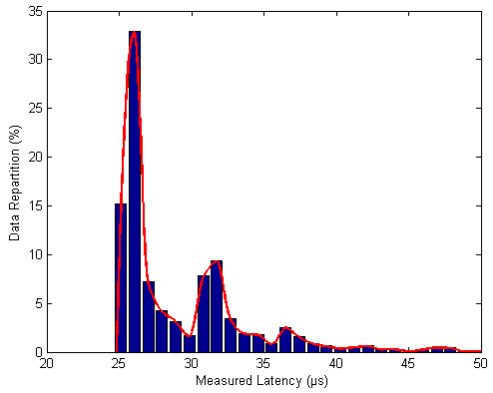

Figure 7. Distribution of the measured latency for data size of 256 Bytes (OmniORB)

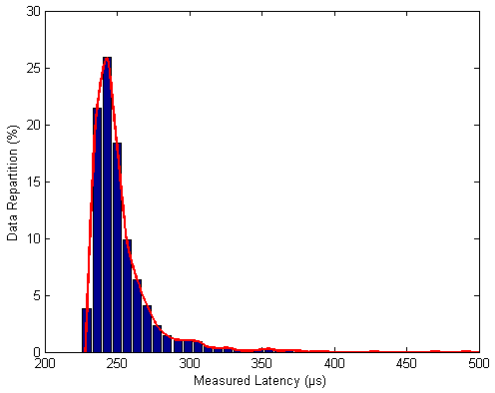

Figure 8. Distribution of the measured latency for data size of 32,768 Bytes (OmniORB)

In this figure, the latency increases when the size of the transmitted data grows. Some particular points can be observed for transmitted data sizes of 8,192 and 10,000 Bytes where the latency decreases slowly to continue normally for the next points. The solid curve shows the measured data and the dashed one shows a basic fitting of the results without the two points mentioned earlier. It can be noticed that the progression is not linear, it is basically a logarithmical curve.

Instead of looking to the latency mean of each measurement, it appears to be more interesting to look at the repartition of each one of the measurements by creating histograms as done in Figures 7 and 8. The repartition of the data is totally different between the two campaigns of measurements. For small data size (256 Bytes in Figure 7), a "pseudo-periodic" comportment can be observed: there is a maximum for 26 µs, another peak is visible at 31 µs and a last one at 36 µs. These three values are separated by 5 µs each.

The second histogram represents the measurement for a high amount of data transmitted (32768 Bytes in Figure 8). In comparison to a small data size, there still is a maximum peak but the rest of the curve does not present any other peak.

These measurements realized with OmniORB give a first relevant comparison point regarding the aim of this study: characterising the latency in a SCA system.

After measuring the latency with OmniORB, it is interesting to add a software layer (OSSIE) to see how the system will then behave.

B. OSSIE Results

The same kind of latency measurement has been made with OSSIE. Thanks to "WaveDev" (included with OSSIE), the development of components and waveforms is very simple. Thus, several waveforms have been created to have a large panel of measurements as explained earlier with Figure 4.

1) Link Number Variation: In a first approach, the measurement parameter is the number of components in

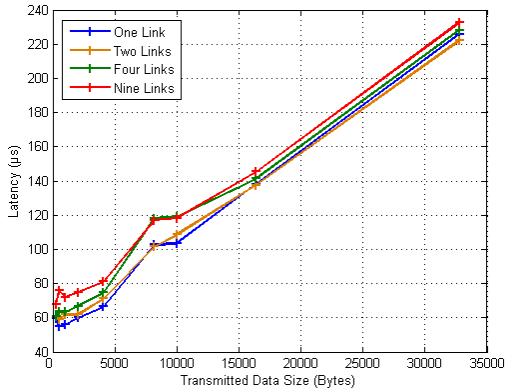

Figure 9. Latency measurements with component variation in a single channel waveform (OSSIE)

a single channel waveform. The results of these measurements are presented Figure 9. The first noticeable remark is the similitude with the OmniORB measurement (Figure 6). The look of the curves is approximately the same with an identical peak between 5,000 and 10,000 Bytes of data transmitted. By focusing on the OSSIE measurements, they are all nearly identical with an increase of the latency when the number of links is higher. The difference seems to be negligible (about 10 µs) but it is important not to forget that in a waveform with nine links like this one, the total latency is higher than for a single link waveform:

- for one link, the latency is about 226 µs for 32,768 Bytes of data transmitted,
- for nine links, the average latency on one link is about 232 µs but the total latency of the waveform is 232*9 which equals 2 ms.

The difference between these two measurements shows the importance of taking care of the inter-component latency. Added to the time spent in signal processing, these 2 ms are significant regarding the global system latency.

Some measurements have also been done with different data rates to see if a difference could appear. The default data rate for this study is 320 kSps, to perform the comparison, measurements at 100 kSps and 500 kSps

have been realized. The difference is weak, so it has been decided to continue to perform the study with only one data rate.

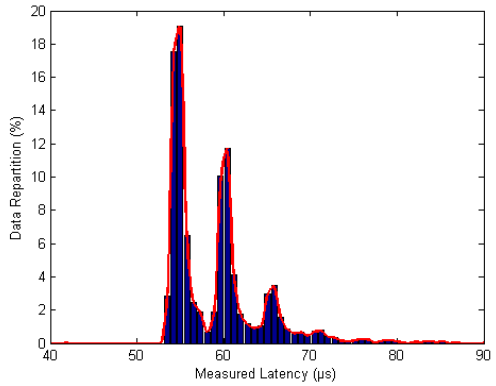

Figure 10. Distribution of the measured latency for data size of 256 Bytes on an inter SPC component link

It is interesting to look at the histograms as was previously done for OmniORB. The Figures 10 and 11 show the histograms for one component waveform but they have been realized with different sizes of packets. The waveform in Figure 10 corresponds to a packet size of 256 Bytes (128 RealShort) while there are 32,768 Bytes (16,384 RealShort) in a packet for the waveform in Figure 11.

By looking at these two figures, some similarity with those realized with OmniORB only in Section 4.1 can be found. The same pseudo-period is present on the 256 Byte data size measurement. With OSSIE using OmniORB, it is comforting to find the same behavior again.

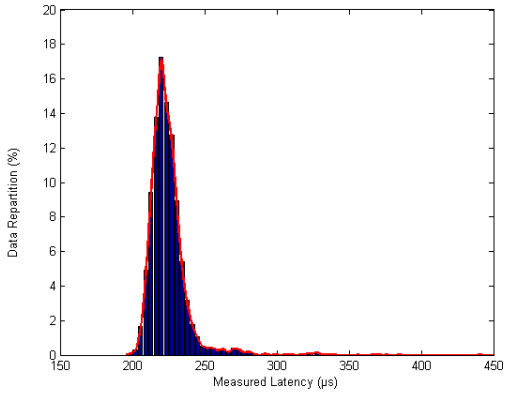

Figure 11. Distribution of the measured latency for the data size of 32,768 Bytes on an inter SPC component link (OSSIE)

When other components are added in a waveform (obviously inter-components links are added too), the latency mean does not vary a lot as shown in Figure 9. The same report can be done for histograms, Figures 7 and 10 are approximately the same.

These measurements show the limits of separating signal processing functions into several components. By only having an elementary signal processing function, the reusability of a component increases considerably but as can be seen with these measurements, a high number of inter-component links in a channel induces the presence of a large amount of latency. The size of a component (in terms of signal processing) must be a trade-off between reusability and a minimal number of components in the waveform.

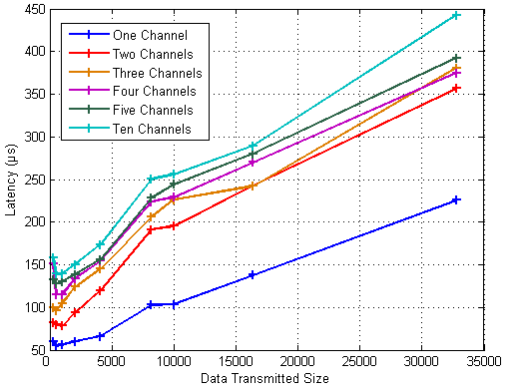

Figure 12. Latency measurement with channel number variation with one link per channel (OSSIE)

2) Channel Number Variation: A second kind of measurement performed with OSSIE is to add additional channels as shown in Figure 4.

Like for the component number variation, the latency mean of each measurement has been calculated and the result is presented in Figure 12. It shows that when the number of channels increases, the latency increases too. It makes a huge difference with Figure 9, where the variation of the latency in function to the number of components in a single channel is low.

The gap between a single channel waveform and the other waveforms shows once again the importance of designing a waveform with concern of not only what is inside the component, but also what is outside it.

C. OmniORB and OSSIE Comparison

It is interesting to notice that OSSIE does not bring more latency than OmniORB itself as shown in Figure 13. For small data size, communication time with OSSIE is longer than communication time with OmniORB only. When the size of the data transmitted increases (i.e. superior to 10,000 Bytes), a reversal can be observed even if the difference is very low: 12 µs on a total time of 250 µs which represents less than 5%.

The explanation of this observation can be the fact that OSSIE is using a ”faster” method to transmit data than the one used to realize this study.

V. Statistic Estimation

Being able to measure the inter-component latency is not enough. The ideal result for designers is the possibility to predict this latency. Thus, the purpose of this section is to propose a statistical model to know, while designing a waveform, the cost of the inter-component latency and

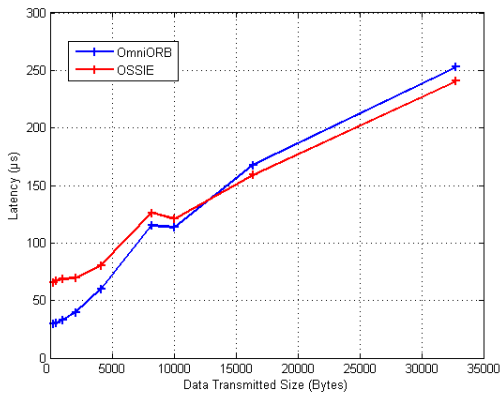

Figure 13. Comparison between OmniORB and OSSIE on one inter-component link

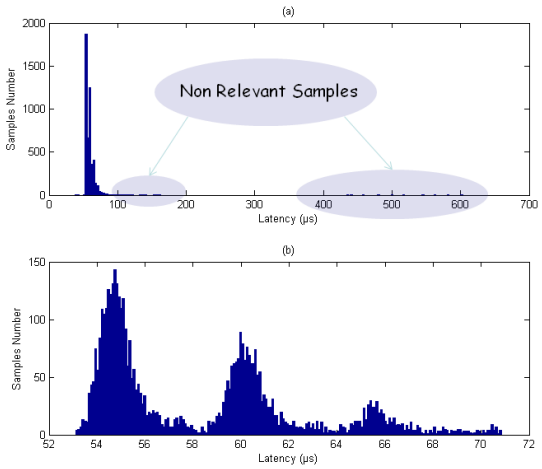

Figure 14. Comparison between a graph without preprocessing (a) and a graph with it (b)

being able to choose the right component number, and the corresponding granularity to find a trade-off between the reusability of the component and the global latency of the system.

A. Homogeneity of Latency in a Waveform

The first performed test consists in testing if all the communications inside a waveform are similar or not. To get this information, the Kolmogorov-Smirnov Test [14] has been used. This test is useful to know if it is sufficient to perform the other tests on one distribution or on many distributions. If all the distributions are similar, the assumption done for one is also valuable for the others.

The results of the test are interesting because for each waveform the assumption of similar distributions is verified. The following analyses are performed only on one link of the waveforms by taking the assumption verified in this section.

B. Operating System Noise Measurement

Running this kind of measurements on a complex operating system can distort the measurements in several ways. Even if the test machine was installed with the minimal software to carry out the measurement, it is still possible to have a process, which runs at the same time and takes some resources. Thus, a kind of pre-processing on the data was decided on and the samples that were not significant for the measurement (i.e. latency values which are more than twice as great as the latency mean) were removed. Removal of these samples does not mean that they are irrelevant because they are present in most of the system. But after comparison of tests with and without these samples, removing them allows for better accuracy in the statistical model. Figure 14 shows data repartition before and after this pre-processing. The relevant samples are kept and the others are deleted.

C. Statistical Model

As previously seen, Figures 10 and 11 show the repartition of the latency on an inter SPC component link in an OSSIE waveform. Looking at the histograms gives a first indication of the behavior of the latency but having a deeper analysis can be very helpful. To realise this, two MATLAB tools have been used (dfittool and cftool). The choice of the tool used depends of the nature of the curves. The histogram of Figure 10, which presents several peaks, does not correspond to distributions present in ”dfittool”. By using ”cftool”, the result of the fitting gives an equation like this:

$$Eq = a_1 * e^{-\frac{x-b_1}{c_1}} + \ldots + a_6 * e^{-\frac{x-b_6}{c_6}} \quad (1)$$

This equation corresponds to the sum of six Gaussians and the result can be seen in Figure 15. In this Figure, many curves can be observed. The red one is the result of the fitting process, which was given the equation mentioned above. The two dashed curves represent this equation too but only with a precision of 95%. It gives an idea how well the test curve fits with the calculated ones. This equation gives a fast estimation of the latency evolution; this is relevant information for designers who do not have any metric today.

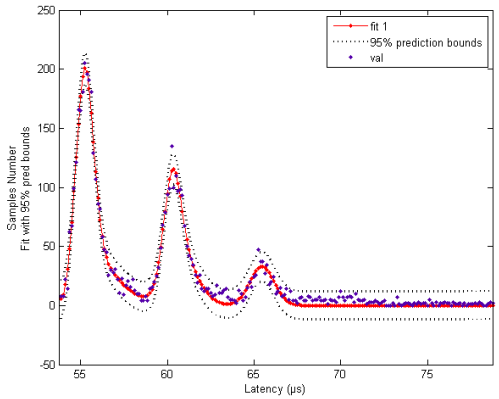

Figure 15. Fitting of Figure 10 histogram

The size of transmitted data in this test is very low,

it is not representative of an operational waveform. The histogram of Figure 11 represents the measurement for data size, which could be implemented in a real system. By using "dfittool" on this distribution, two well-known distributions appear to be similar to the under-test one.

Figure 16 shows the result of this analysis. The two possible distributions are the T-Location Scale distribution and the Generalized Extreme Value distribution. They are not exactly similar to the measured values but they are closer than the other distributions available in the MATLAB tool. The equations of these two distribution are presented here (respectively T-Location Scale distribution (2) and Generalized Extreme Value distribution (3)):

$$f\left(x|\,\Gamma,\upsilon,\mu,\sigma\right) = \frac{\Gamma\left(\frac{\upsilon+1}{2}\right)}{\sigma\sqrt{\upsilon\pi}\Gamma\left(\frac{\upsilon}{2}\right)}\left[\frac{\upsilon+\left(\frac{x-\mu}{\sigma}\right)^2}{\upsilon}\right]^{-\left(\frac{\upsilon+1}{2}\right)} \tag{2}$$

$$f\left(x|\,k,\mu,\sigma\right) = \left(\frac{1}{\sigma}\right)exp\left(-\left(1+k\frac{(x-\mu)}{\sigma}\right)^{-\frac{1}{k}}\right)\times \left(1+k\frac{(x-\mu)}{\sigma}\right)^{-1-\frac{1}{k}} \tag{3}$$

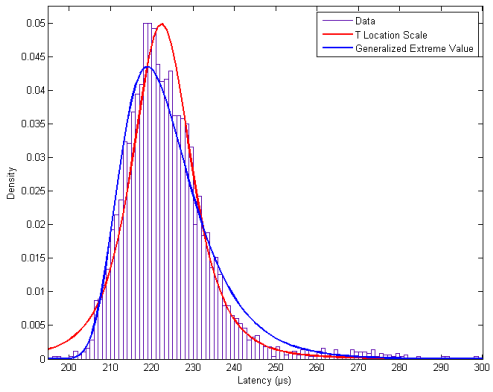

Figure 16. Fitting of Figure 11 histogram

VI. Conclusion

This study shows the behaviour of inter-component communications in a SCA OE and demonstrate the necessity of having good knowledge about this to design efficient waveforms. In a waveform with many components, this latency can exceed 2 ms, which is a really significant overhead. The comparison of communications between OmniORB and OSSIE also shows that OSSIE does not bring some extra latency to the system. Good knowledge of the communications in a SCA architecture requires a good control of CORBA. The real challenge is to find the good trade-off between component reusability and system latency. A second result that must be underlined is the impact of the choice regarding the size of data transmitted. Finally, we propose in this paper a first model that enable one estimate the best suitable known distribution. It provides SWR developers with a tool to anticipate the communication behavior at design time. Obviously, estimated latency values on the test platform are not usable on other platforms. But the process is simple and can be implemented easily. Alternately, the measurement process could be used in a different way. Measuring inter-component latency during run-time may allow the system to dynamically reconfigure parts which do not respect real-time system limitations. It may bring SDR closer to Cognitive Radio.

VII. Future Work

The good knowledge of SCA through these OSSIE experimentations will be used to go further and to investigate implementation solutions of CORBA over a Network On Chip (NoC). The idea is to take advantage of this kind of architecture to increase the CORBA performances and especially the mapping of the GIOP (General Inter-ORB Protocol) onto the NoC powerful hardware transport mechanisms.

References

[1] J. Mitola III. Software radio architecture: A mathematical perspective. *IEEE Journal on Selected Areas in Communications*, 17(4), April 1999.

[2] *JTRS SCA v2.2.2*. http://sca.jpeojtrs.mil/.

[3] J. Bard, V.J. Kovarik. Jr. *Software Defined Radio: The Software Communications Architecture*. WILEY, 2007.

[4] Object Management Group, CORBA. http://www.corba.org/.

[5] Virginia Tech, Open Source SCA Implementation :: Embedded. http://ossie.mprg.org.

[6] Rhapsody. *CORBA Development Guide*.

[7] A. Buss, L. Jackson. Distributed Simulation Modeling: A Comparison of HLA, CORBA and RMI. *Winter Simulation Conference*, 1998.

[8] M.B. Juric, I. Rozman, M. Hericko. Performance Comparison of CORBA and RMI. *Information of Software Technology*, 42(13), September 2000.

[9] Ettus Universal Software Radio Peripheral. http://www.ettus.com/.

[10] P.J. Balister, M. Robert, J.H. Reed. Impact of the use of corba for the inter-component communication in SCA based radio. *SDR Forum Technical Conference*, 2006.

[11] T. Tsou, P.J. Balister, J.H. Reed. Latency profiling for SCA software radio. *SDR Forum Technical Conference*, 2007.

[12] G. Abgrall, F. Le Roy, J.P. Delahaye, J.P. Diguet, G. Gogniat. Comparative study of two software defined radio environments. *SDR Forum Technical Conference*, 2008.

[13] P. Work, K. Nguyen. *Measure Code Sections Using the Enhanced Timer*. Intel Corporation.

[14] F.J. Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253), March 1951.