Performance Comparison of Selected Object Oriented Middleware Technologies for the .NET Platform

Bjarke N. Rahbek
Department of Engineering
Aarhus University
Aarhus, Denmark
bjarke@au.dk

Christian M. Vraa
Department of Engineering
Aarhus University
Aarhus, Denmark
christian.mark.vraa@post.au.dk

Stefan R. Wagner
Department of Engineering
Aarhus University
Aarhus, Denmark
sw@eng.au.dk

***Abstract*— With the ever increasing demand for interconnecting a large variety of different devices, the sphere of distributed systems is becoming increasingly important. For efficient development of distributed systems, communication depend on stable and high-performance middleware solutions. The middle-ware should seamlessly integrate with the used language and platform to overcome the heterogeneity of the underlying system. The aim of this study is to compare both roundtrip time and payload of the object-oriented .NET middleware solutions: Microsoft's Windows Communication Foundation and ZeroC's Internet Communications Engine. The overall results show that Ice by far outperforms its competitors in both studies by large factors, followed by the binary WCF and the XML WCF at last.**

***Index Terms*— WCF, ICE, XML, middleware, interconnectivity, network, .NET, distributed system.**

I. Introduction

Along with the need for interconnecting just about any device [1] comes the challenge for selecting the appropriate middleware to handle the communication. In a world rapidly moving towards a true implementation of the Internet of Things vision [2], it is not just a matter of connecting devices, it is also essential to look into the performance of the communication.

The nature of distributed systems calls for application level requests of operations from remote objects. The underlying network handles the physical interchange of radio, electrical or optical signals as well as facilitates both error handling and routing. This leaves it up to the middleware to shield the lower level details from the requesting object [3]

With this paper, we focus on the object-oriented middleware technology available for .NET developers, where we compare both time and payload, but also consecutive packet transmission.

The tests are conducted in a simple closed wired client/server setup, using standard off-the-shelf components.

The three ways of object-oriented communicating in a distributed system that we compare are from to different vendors. From Microsoft we have used the Windows Communication Foundation [4] and from ZeroC we used the Internet Communications Engine [5].

A. Windows Communication Foundation

Microsoft's communication framework, Windows Communication Foundation, WCF, is part of the suite of development tools offered by Microsoft all under commercial license. Depending on the setup of the WCF service the languages supported range from any languages/platform that is capable of sending a Hypertext Transfer Protocol (HTTP) request to only .NET-languages on Windows platforms. Two of the main configurations of WCF are [6]:

1) XML

WCF configured to support "plain old XML" (POX) messages [7] with its basicHttpBinding. This allows for a large number of clients to consume it as a simple web service. Along with the flexibility in regard to both choice of platform and programming languages comes the downside of the very verbose format of XML [8]. This binding and a Web Service binding, WSHttpBinding, are among the most commonly used due to their interoperability and legacy benefits. However verbose the basic binding is, the web service binding has a by far larger overhead [9].

2) Binary

WCF can also be set up to use a binary encoding. The Net.Tcp binding is used for performance rather than interoperability [10] and unlike the http bindings, the Net.Tcp binding is connection oriented. With the binary encoding this binding is the obvious choice if throughput is the main driver for decision [11].

B. Internet Communications Engine

ZeroC's Internet Communications Engine, often just called Ice, has its roots in CORBA [12] but is by far less complex [13]. Ice is available both under the General Public License and under a proprietary license. Aside from supporting .NET, Ice also has support for a wide range of languages from C++ and Java to Objective-C for iOS devices and PHP for web. Among the operating systems supported are both Microsoft Windows and Apple's Mac OS X, but Ice is also supported on Linux and Solaris from Oracle.

The aim of the study is to compare WCF and Ice on both roundtrip time and payload performance characteristics.

II. Methods

The test setup - depicted in figure 1 - is, as previously stated, a closed and cabled network, consisting of two laptop

computers with a router connecting them. The client PC is equipped with a 2.4GHz Intel Core i7 and the server PC is equipped with a 2.5 GHz Intel Core i3. The router is a DIR-655 Wireless N Gigabit Router from D-LINK. Running on both computers, the software for the setup was Ice Version 3.5.1 and .Net 4.5. For recording data packages over the network, the SharpPCap library version 4.2.0 was used.

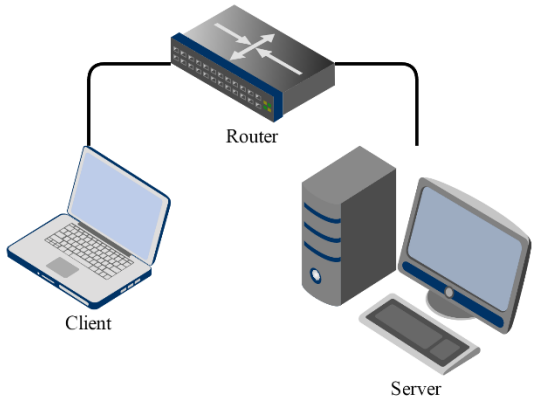

Fig. 1 Test Setup

To be able to test the technologies on as equal terms as possible, generic test data was generated.

1003 instances of each datatype was generated to have the same test data for all the tests, so as dew parameters could influence the results as possible. The reason that it is not exact 1000 is that for the roundtrip test, the first three were not used as data because of the runtime linking of the connection, and the first few will therefore take longer time to sent.

As stated earlier, the test parameters was the roundtrip time and the payload and overhead of the data, so different tests was setup to measure this. The tests are split up into two categories; roundtrip and payload/overhead.

The payload/overhead tests was performed with the SharpPCap library where all the test data was sent, one package at a time. The package data, both total data load and the payload of the packages was recorded. This was performed in three iterations, one for each technology to be tested.

The roundtrip time tests was performed with the .NET internal timing functionality. Two different tests was performed here, one measuring the average time to send and one measuring how much upstart time influenced the roundtrip time. Both tests was performed with only strings. The last one in clusters increasing in size, in the range 1, 5, 10, 25, 50, 100, 250, 500 and 1000 units.

III. Results

This section describes the results from the three different test setups; roundtrip time, payload and clustered average time. The results are shown for all three technology setups. Also a comparison of the overhead is shown.

A. Roundtrip

This subsection shows the results for the average roundtrip time for all setups combined.

The graph in figure 2 shows the overall average roundtrip time for the three different test setups, on different datatypes. It is clearly seen that the blue bars, the XML version of WCF, takes about one millisecond longer for each object to be send, no matter the datatype. To the contrary, Ice shows that it is even faster than both of the WCF roundtrips.

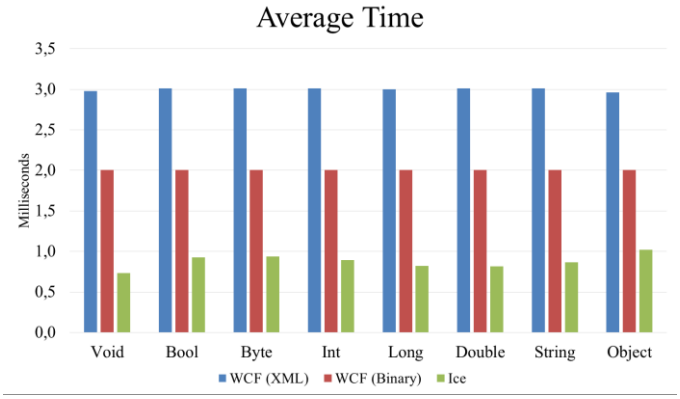

Fig. 2 Average Timer Overview

In the figure, the individual times for the three different setups are shown. Even though both the XML and the binary versions of WCF are significantly slower, they do not wary as much in time. The blue and red bars showing WCF XML and WCF binary respectively, shows that the value of the roundtrip times does not vary much dependent on the datatype. On the other hand, the Ice roundtrip times in figure 2 shows that these times vary depending on datatype, showing that the custom object takes the longest time where Voids take shortest time.

B. Payload

This subsection presents the results for the average payload data for all setups combined and individually.

1) Overview

The average payload of all three setups are shown in figure 3. As the roundtrip time showed, the XML version of WCF is also the worse in this area. The average payload here is more than twice the size of the binary WCFs' payload. Moreover, the payload of Ice is even better and is overall smaller than 100 bytes, which is a factor 5 less than the XML version of WCF.

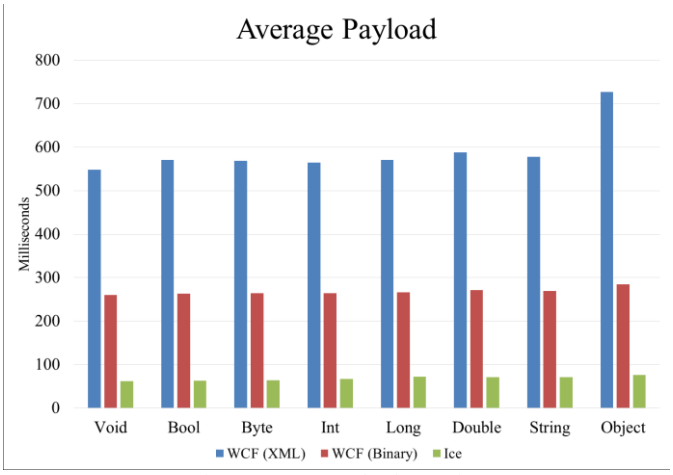

Fig. 3 Average Payload Overview

2) Average Payload Details

Figures 4, 5 and 6 illustrates the individual payloads of the three setups, and all three graphs show that the custom object takes the most bytes to send. The rest of the datatypes are relatively close for all three setups.

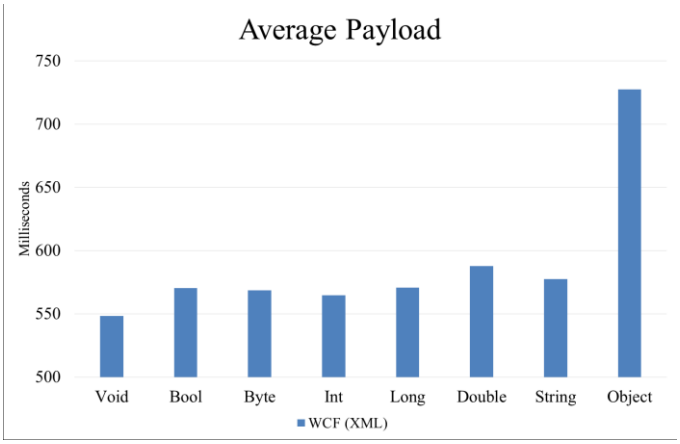

Fig. 4 Average Payload for WCF with Http binding (XML)

The minimum payload of WCF with XML is about 550 bytes, where the largest payload is the custom object with 725 bytes of payload. This is a difference of roughly 175 bytes.

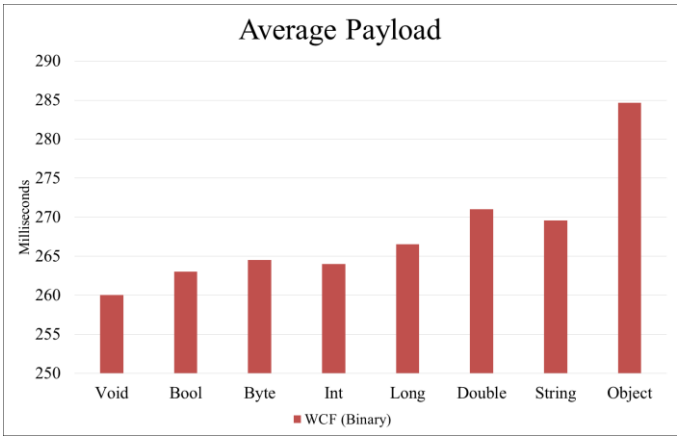

Fig. 5 Average Payload for WCF with Net.Tcp binding (binary)

The minimum payload of WCF over TCP is about 260 bytes, with the maximum payload of about 285 bytes. Again the largest is the custom object. The difference is roughly 25 bytes.

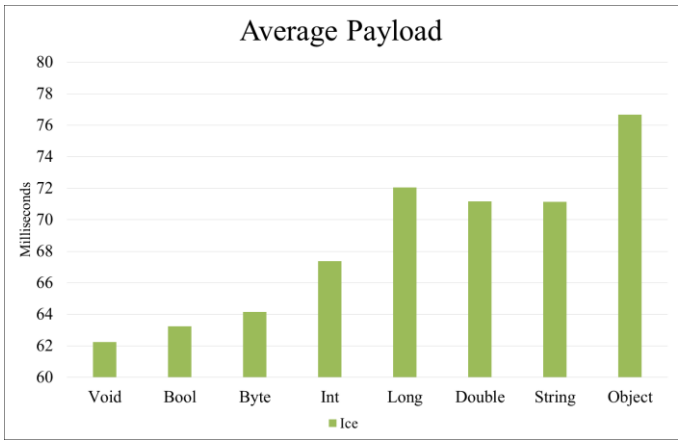

Fig. 6 Average Payload for Ice

At last, the smallest payload in Ice is about 62 bytes and the largest is about 76 bytes. A difference of roughly 14 bytes.

C. Overhead

A comparison of the average overhead of the sent messages are shown in table 1. All the messages are sent as strings.

TABLE I. OVERHEAD

	Description	***Bytes***
XML	WCF with BasicHttpBinding	162
Binary	WCF with Net.Tcp binding	68
ICE	Object transfer over ORB	~34

As seen in table 1 the overhead is considerably larger on the XML version of WCF. The binary version is a little more than halved, whereas the overhead in Ice is about halved once again. The reason for the varying average for Ice is that it is not exact. When many messages are sent consecutively in Ice, it will clutter them together, and therefore it is hard to capture each packet separately.

D. Consecutive series of calls

At last figures 7 and 8 shows the data of the last setup, where a number of consecutive calls were made. The initial start of figure 7 shows that both WCF setups has a much slower upstart than Ice. As the curve evens out, it is seen that as more objects are sent, the binary version of WCF is getting closer to Ice on the average. Figure 8 is the same data, zoomed in on the right hand side of the graph.

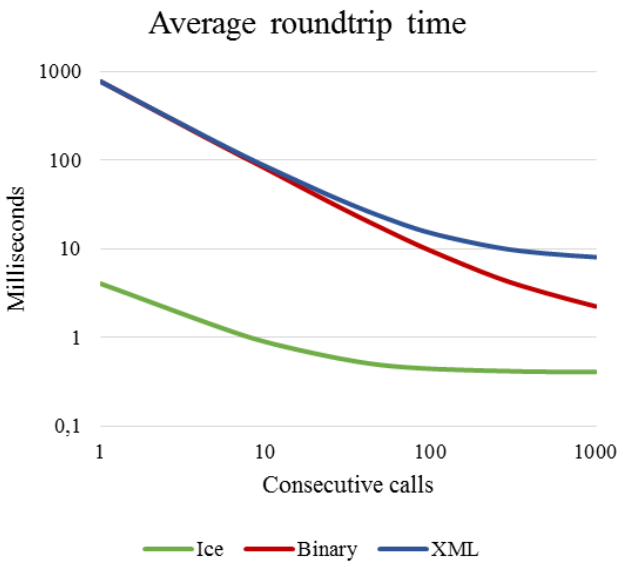

Fig. 7 Average Roundtrip Time Overview

Ice is by far outperforming both WCF setups in terms of speed and startup time, and the two WCF setups seem to have the same startup time.

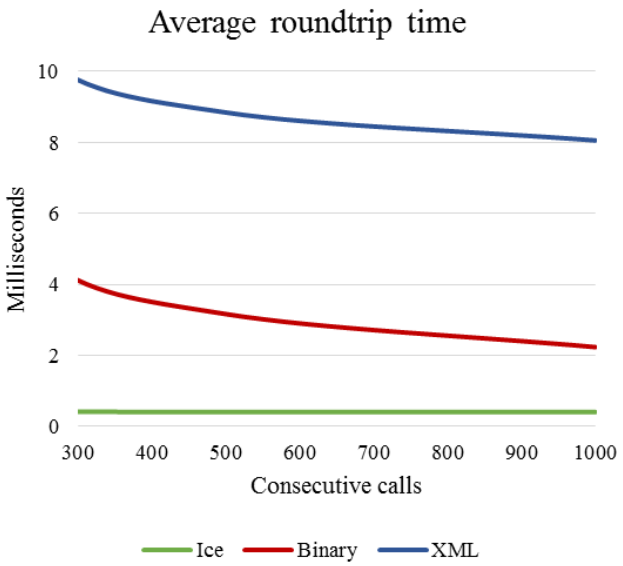

Fig. 8 Average Roundtrip Time – 200-1000 calls

Over time the binary WCF setup is getting closer to Ice's speed, where the XML WCF setup is far behind.

IV. Discussion

When choosing between these selected middleware technologies, a few parameters have to be in mind, as not only the hardcore performance facts are to be evaluated. If there, for example, is a need for legacy support, the obvious choice is WCF with a Http binding (XML), even though the performance tests shows that this is by far the slowest solution of the selected ones.

To overcome the heterogeneity without the massive loss in performance, the options are between WCF Net.Tcp and Ice.

WCF is intended for communication between WCF clients, and as this is a Windows technology the support for devices is limited, where Ice supports true coverage over a large variety of devices, e.g. iOS, Windows, Android, Mac and Linux.

Choosing Ice is also choosing not to support any and all devices compared to the WCF Http binding where any device that can make a simple http request can consume the service. This is very important in some applications, e.g. the Internet of Things vision [14].

V. Conclusion

When concluding on the results, a note should be taken that it is based on the topic object-oriented middleware, so only performance of the technologies is taken into account. So within the scope of the paper, Ice is by far outperforming the other two options.

Even though Ice outperformed WCF, other issues like legacy support and interoperability are important issues as well. Not to mention the use of open standards and not be tied to one vendor with a proprietary format.

VI. Future Work

It could be interesting to do the same study in a 'real-world' scenario over both Wi-Fi networks and 4G mobile networks, to study another angle of this. The stability and throughput of the application could change the results of the study significantly.

Also CERN has through an evaluation of recognized middleware products documented that a solution of iMatix has an even higher performance [15] than ZeroC's. The iMatix's ZeroMQ could be interesting to compare with the solutions selected in this study.

References

[1] Lu Tan, L et al (2010), *Future internet: The Internet of Things*", 3rd International Conference on Advanced Computer Theory and Engineering (1-4244-6539-7, 978-1-4244-6539-2), (p. V5)

[2] L. Atzori et al., *The Internet of Things: A survey*, The International Journal of Computer and Telecommunications Networking, Volume 54, Issue 15, 28 October 2010, Pages 2787–2805.

[3] G. F. Coulouris, *Distributed Systems: Concepts and Design*, Addison-Wesley, 2011, ISBN 0-13-214301-1

[4] Microsoft (2015). *What Is Windows Communication Foundation (.NET Framework 4.5)* [Online]. Available: https://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx

[5] ZeroC, Inc. (2015). *The Internet Communications Engine (Ice)* [Online]. Available: https://zeroc.com/ice.html

[6] D. Chappell, *Introducing Windows Communication Foundation – An Early Look*, September 2005, Microsoft Corporation

[7] C. Pautasso, *RESTful Web Services: Principles, Patterns, Emerging Technologies* in Web Services Foundations, Springer-Verlag New York, 2014, pp. 31-51

[8] S. Sakr, *XML compression techniques: A survey and comparison*, Journal of Computer and System Sciences, Volume 75, Issue 5, August 2009, Pages 303–322

[9] C. McMurty et al., *Windows Communication Foundation Unleashed*, Sam Publishing, 2007, ISBN: 0672329484

[10] Kola Siva Tharun et al. (April 2013), *Advantages of WCF Over Web Services*, International Journal of Computer Science and Mobile Computing

[11] Microsoft (2015), *Choosing a Transport (:NET Framework 4.5)*, [Online]. Available: https://msdn.microsoft.com/en-us/library /ms733769(v=vs.110).aspx

[12] Michi Henning. 2008, *The rise and fall of CORBA*, Commun. ACM 51, 8 (August 2008), pp. 52-57. DOI 10.1145/ 1378704.1378718

[13] M. Henning (Jan. 2004), *A new approach to object-oriented middleware*, Internet Computing, IEEE (Volume:8 , Issue: 1), pp. 66-75

[14] A. Dunkels and J. P. Vasseur, *IP for Smart Objects*, IPSO Alliance White Paper No. 1, Sept. 2008

[15] A. Dworak at el. (2012), *The new CERN Controls Middleware*, International Conference on Computing in High Energy and Nuclear Physics 2012, Journal of Physics: Conference Series 396 (2012) 012017.