Client-Side Processing Environment Based on Component Platforms and Web Browsers

Adam Piórkowski and Przemysław Szemla

AGH University of Science and Technology,
Faculty of Geology, Geophysics and Environment Protection,
Department of Geoinformatics and Applied Computer Science,
al. A. Mickiewicza 30, 30-059 Kraków, Poland
pioro@agh.edu.pl,pszemla@geol.agh.edu.pl
http://www.geoinf.agh.edu.pl

Abstract. Distributed processing is an important issue of numerical calculations, in particular concerning the problems of time-consuming calculations. Solving this problem requires appropriate software, which is more complicated than the implementation in parallel environments. This article presents a proposal of distributed processing solution based on web browsers. This method, unlike the commonly used, does not require installing any software on the compute nodes. This is achieved through the distribution and execution of the code in the container, which is a web browser.

Keywords: parallel computing, distributed computing, numerical computing, client-side processing, web browsers, component technologies, clusters, domain decomposition.

1 Introduction

Distributed processing is an important method of resolving time-consuming calculations. In the past decades, methods of implementation of distributed processing have changed many times. The basic method of implementation of distributed computing is the design of remote processes, communicating through network mechanisms, for example network protocols such as TCP/IP are being commonly used for that purpose. Unfortunately, this method has significant drawbacks:

- for every application you need to implement a network connection layer,
- there is a full set of distributed settings required – often the static allocation of resources requires creating a new solution for every problem.

PVM (Parallel Virtual Machine) and MPI (Message Passing Interface) are the next methods of communication in parallel and distributed systems. A characteristic feature is the strong binding of a program that uses these technologies [1,2].

A. Kwiecień, P. Gaj, and P. Stera (Eds.): CN 2013, CCIS 370, pp. 21–30, 2013.

The next approach is to create components that allow for distributed processing. Examples of such environments are CORBA (Common Object Request Broker Architecture) and DCOM (Distributed Component Object Model). Both the environments can help to implement a network communication, but are associated with a particular execution platform (system and hardware). Also in this case, the calculation for a specific problem requires a dedicated system [3,4,5].

Another approach allows for a hardware and a system independence. Component environments (Java ME, .NET Framework) allow for providing of mechanisms for calling methods of objects (Java RMI, .NET Remoting). This solution has many advantages, however, it still needs to install software on the computing nodes [6,7].

There are special distributed computing environments [8]. Examples of solutions are the environments: openSSI, XtreemOS and Apache Hadoop [9,10,11].

Distributed processing, as well as parallel processing, requires the ability to decompose the problem. The basic decomposition methods for parallel processing are the functional and the domain decomposition. Functional decomposition requires the ability to implement various functions in the nodes. Domain decomposition usually performs the same code on different nodes, sharing data between these nodes. This approach, although more difficult to implement, allows for much greater scalability than functional decomposition.

The research on distributed computing in component environments and on the calculations in the network [6,7], both on server- and client-side [12,13], has resulted in the concept of system that processes data in browsers (as containers). Therefore, this solution does not require any software installation on the compute nodes. Due to using component environments a code can be run on any computer. Data transmission is performed by using the serialization mechanisms specific to those environments. The idea of applet client-side computing is also a topic of the research [14], but the way of decomposition of numerical problem is different than described in this article.

2 The Concept of the Solution

Flexible environment for numerical computations should meet the demands:

- Portability of code,
- Dynamic adjoining nodes to the system,
- A minimum of complicated configuration of nodes.

Code portability feature can be achieved using technologies such as Java Component VM, NET Framework and Mono. There are environments for different hardware platforms for these technologies.

Dynamic adjoining nodes to the system is an important feature. Static assignment of nodes can be a cause of problems if one node fails. The system is waiting for a result that nodes had to provide. If the system can dynamically add computational units, overcoming this problem is available (Fig. 1). Minimum complicated configuration is another challenge. The best solution involves that

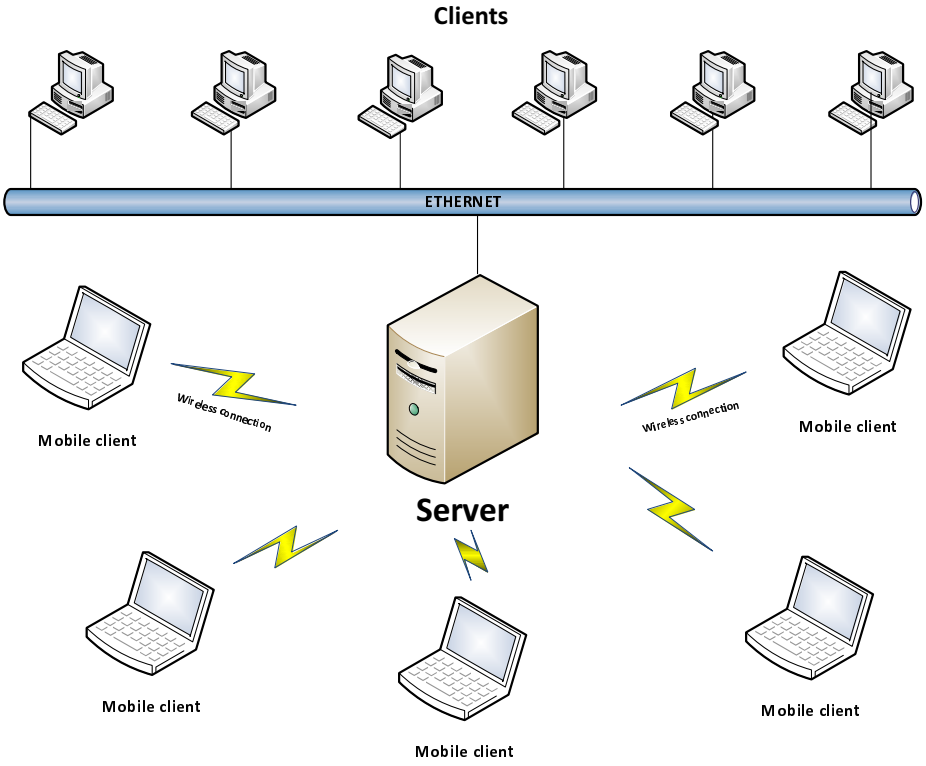

Fig. 1. Schema of the network for the proposed system

a client should only report to the system access to participate in the calculation. Other activities, such as the transfer of input and output data and executable code should not occupy his attention.

The proposed solution that satisfies these assumptions is the client-side application, in particular an applet executed inside the container, which is a Web browser (Fig. 2). The use of Java-applet or MS Silverlight (Moonlight for Linux, Mac) allows to run precompiled and managed code in browser [13]. Attaching nodes to the system can be made at any time, by user actions or service (e.g. screensaver). Once connected, the system should maintain connections to objects. By using browser system should not require any configuration at the client side.

The scheme should have an object interface, that allows for easy consolidation of its classes with classes of a numerical problem solution. An analogy to the Parallel. For loop (.NET 4.0) can be used – Distributed.For. Such a loop would allow for the automatic division of the data domains to computational units in the system. This applies only to numerical problems that allow for the proper domain decomposition (Fig. 3).

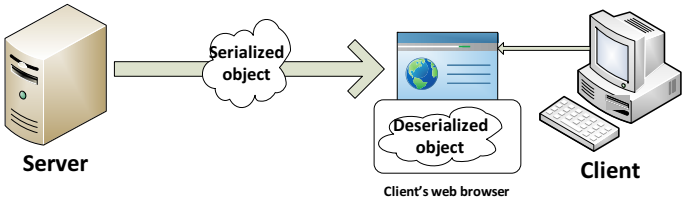

Fig. 2. Web browser as a container for an applet with code

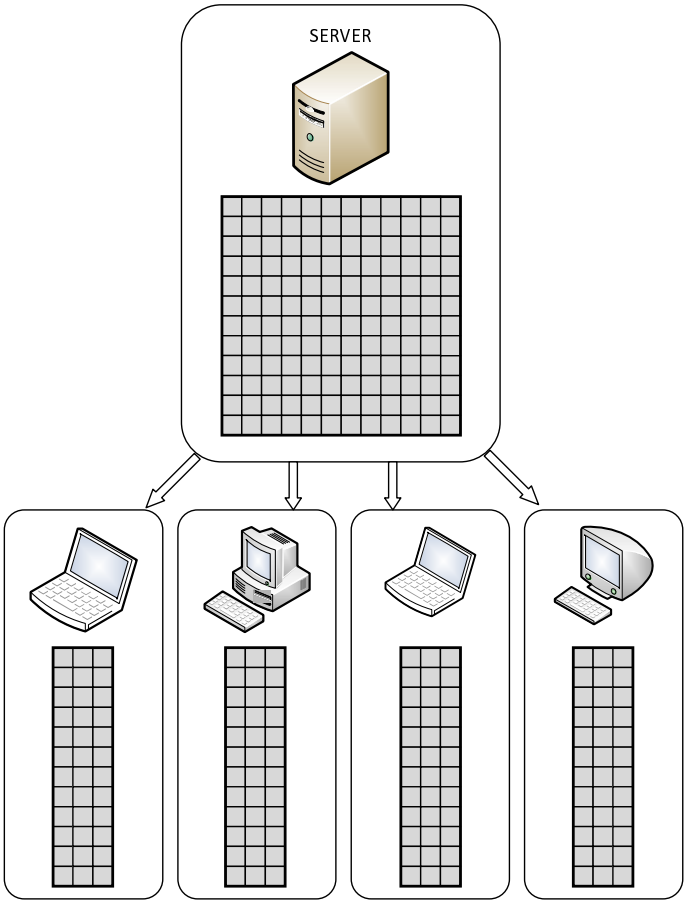

Fig. 3. Schema of the domain decomposition

3 Principles of Implementation

The loading of the numerical code at the client side takes place at the applet downloading. Next the data should be passed. The way of passing data is to

use serialization feature for component objects (Fig. 2). After deserialization the application can start processing the data at the range pointed by server for the current node. The output values are returned to the server. The action's scenario is shown on the Fig. 4. The first version of presented solution was implemented for Java VM environment.

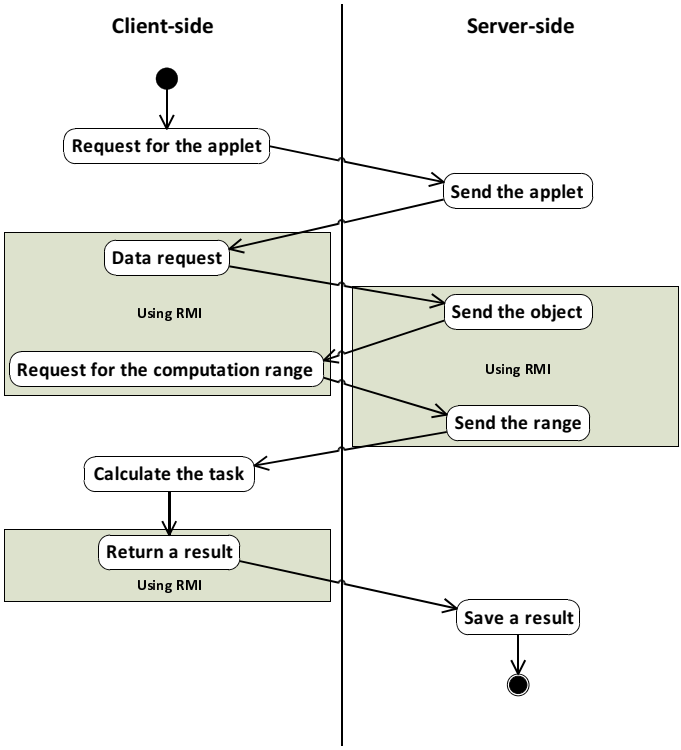

Fig. 4. Action schema

3.1 Implementation of a Server

The server of an application is a three-threaded [15]:

- starting thread – it initiates a connection with RMI register,
- programmer's thread – the code written by a programmer that uses a solution for an implementation of own numerical problem,
- main thread – it controls the programmer's thread.

The Figure 5 shows a class diagram a server of solution [15]. There are four parts of server:

– *server* package – it implements two of the threads mentioned above,
– *program* package – it contains a code of numerical calculations, it uses the *DistributedFor* function, that allows to divide a problem into domains based on parts of input data, the *reduce* class merges the parts of solution delivered by nodes,
– *common* package – this package is shared by the server and the client, it contains an implementation of *ICalculation* interface, used for calculations,
– *distributed* package – it provides all interfaces and abstract classes, e.g. *IReduce*, *ICalculation* and *ACalculation*.

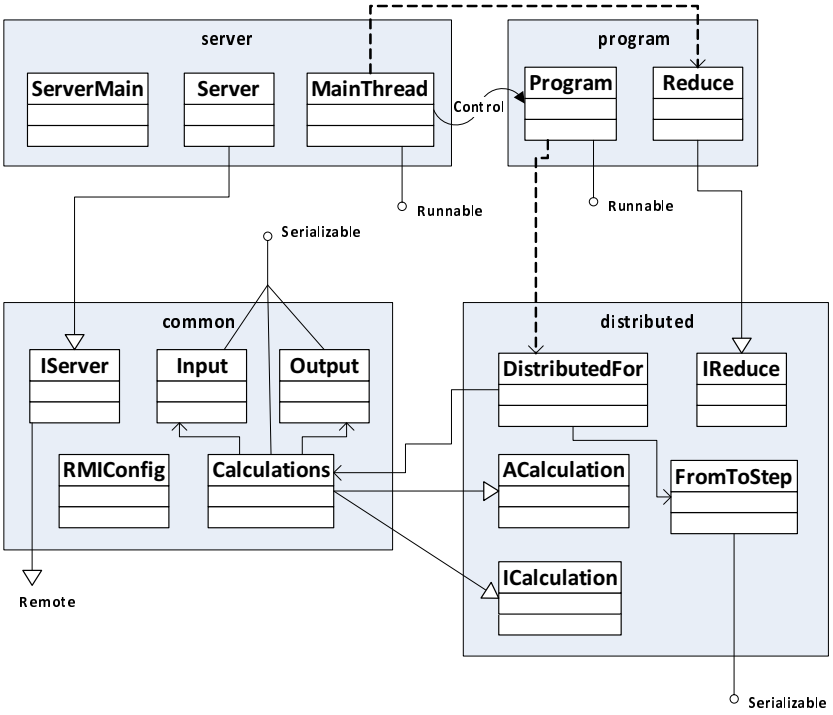

Fig. 5. Class diagram of a server

3.2 Implementation of a Client

The main code of calculations is realized as a self-signed applet (Swing technology). It contains the *client* package, also shares *common* and *distributed* packages, excluding *DistributedFor* class and *IReduce* interface.

At the start of connection the applet takes the calculation class, takes the current state and data of this class by deserialization of server data and gets the range of calculations (From, To, Step). Next the processing is performed and the results are uploaded back to the server. This procedure is repeated for the next range for calculations.

4 Tests

To assess a performance of proposed solution the tests have been carried out [15]. 8 PC (Intel i7 Core 2600, 4 GB DDR3, 1333 MHz) were used as client nodes and the ninth PC as a web server. The Gigabit Ethernet connection with switch was used. As a test algorithm a median filter with mask of 101×101 for gray scale images was implemented (as an example of time-consuming calculations). The tests were performed for two images:

- Picture A of 1920×1200,
- Picture B of 1200×900.

The results of tests are presented in the table (Table 1) and on the plot (Fig. 6).

Table 1. Execution time of distributed image processing [s] (and speedup)

	Serial	1 node	2 nodes	4 nodes	8 nodes
Pic A, 1920×1200	381	391 (102 %)	197 (51 %)	103 (27 %)	65 (17 %)
Pic B, 1200×900	150	151 (101 %)	77 (51 %)	42 (28 %)	22 (14 %)

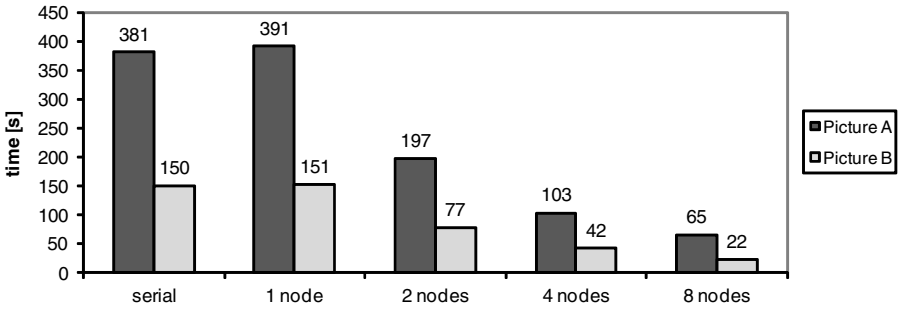

Fig. 6. The results of tests

The results of tests proved that proposed solution is an effective way of parallelization. There was a very small network overhead in case of Picture B, estimated as a comparison of serial mode and calculations on a single node, although the transmitted data after decompression was not negligible. An important factor was the speed of the network. The proposed solution can process in different environments, so the network overhead should be taken into account. The problem of network speed in cluster computations was considered in [16].

5 Conclusions and the Future Work

The proposed solution meets the conditions of flexible environment for client-side distributed numerical calculations. This way of processing enables to set a distributed environment that contains a dynamic number of nodes, that are based on different hardware and software platforms. Numerous problems and applications can be processed at the client side. It is also possible to provide computational services by individual clients. The future work involves the implementation for MS .NET framework and MS Silverlight. Although Java based applets will most likely be a very common solution due to mobile market penetration, MS Silverlight allows to easier implementation using provided interfaces, especially a compression of transmitted data stream. It reduces a network overhead [17,18]. Further improvements (like compression of input data) should allow to reach better efficiency because in case of Picture B processing, simple extrapolation of the measured data for current implementation shows less improvement in execution time in case of 16 processes and significant drop in efficiency for larger numbers. For 8 nodes the speedup is 17 % instead 12.5 %. Another problem is memory allocation – we tested max. 2 GB of input data – such amount of data needs changes in VM configuration. More detailed and extensive testing is needed to accurately assess the solution.

An important issue is the use of the proposed method in real numerical problems. The research on algorithms for distributing task for distributed systems involves time-consuming algorithms used in Geophysics.

The ray tracing technique can easily and accurate simulate the seismic wave propagation in geological medium and can provide synthetic times of first arrivals. This data can then be easily used in further calculations, like solving the inverse problem. However, precise ray tracing algorithms are time consuming. To reduce computational time the parallel approach is recommended [19]. Another interesting time-consuming problem is the calculation of water percolation through a soil resulting from local differences of piezometric pressure distribution [20].

Inversion of seismic tomography data using stochastic method is common computational problem [21,22]. During calculation many velocity distribution were tested by comparing received and estimated travel times of seismic waves. Additional computational problem is a long time for estimation travel times.

The proposed solution is not limited to use in Geophysics. It can be also used for solving time-consuming numerical problems in other domains, like simulation of energy production and distribution for electricity market [23].

Acknowledgments. This work was co-financed by the AGH – University of Science and Technology, Faculty of Geology, Geophysics and Environmental Protection, Department of Geoinformatics and Applied Computer Science as a part of statutory project.

References

1. Sunderam, V.S.: PVM: A Framework for Parallel Distributed Computing. Concurrency: Practice and Experience 2(4), 315–339 (1990)
2. Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message-Passing Interface. MIT Press (1994)
3. Object Management Group. The Common Object Request Broker: Architecture and Specification. OMG Document, Ver. 2.0 (1995)
4. Onderka, Z.: The Efficiency Analysis of the Object Oriented Realization of the Client-Server Systems Based on the CORBA Standard. Schedae Informaticae 20, 181–194 (2011)
5. Onderka, Z.: DCOM and CORBA Efficiency in the Wireless Network. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2012. CCIS, vol. 291, pp. 448–458. Springer, Heidelberg (2012)
6. Kowal, A., Piorkowski, A., Danek, T., Pieta, A.: Analysis of selected component technologies efficiency for parallel and distributed seismic wave field modeling. In: Proceedings of the 2008 International Conference on Systems, Computing Sciences and Software Engineering (SCSS), part of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering, CISSE 2008, Bridgeport, Connecticut, USA. Innovations and Advances in Computer Sciences and Engineering, pp. 359–362. Springer (2010)
7. Piorkowski, A., Pieta, A., Kowal, A., Danek, T.: The Performance of Geothermal Field Modeling in Distributed Component Environment. In: Sobh, T., et al. (eds.) Innovations in Computing Sciences and Software Engineering. Proceedings of the 2009 International Conference on Systems, Computing Sciences and Software Engineering (SCSS), part of the International Joint Conferences on Computer, Information, and Systems Sciences, and Engineering (CISSE 2009), Bridgeport, Connecticut, pp. 279–283. Springer (2010)
8. Czerwinski, D.: Numerical performance in the grid network relies on a Grid-Appliance. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2011. CCIS, vol. 160, pp. 214–223. Springer, Heidelberg (2011)
9. `http://hadoop.apache.org/`
10. Krauzowicz, Ł., Szostek, K., Dwornik, M., Oleksik, P., Piórkowski, A.: Numerical Calculations for Geophysics Inversion Problem Using Apache Hadoop Technology. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2012. CCIS, vol. 291, pp. 440–447. Springer, Heidelberg (2012)
11. Kim, H., Kim, W., Lee, K., Kim, Y.: A Data Processing Framework for Cloud Environment Based on Hadoop and Grid Middleware. In: Kim, T.-H., Adeli, H., Cho, H.-S., Gervasi, O., Yau, S.S., Kang, B.-H., Villalba, J.G. (eds.) GDC 2011. CCIS, vol. 261, pp. 515–524. Springer, Heidelberg (2011)
12. Piorkowski, A., Plodzien, D.: Efficiency Analysis of the Server-Side Numerical Computations. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2009. CCIS, vol. 39, pp. 225–232. Springer, Heidelberg (2009)
13. Szatan, P., Piorkowski, A., Danek, T., Pieta, A.: Client side web-based simulations of geophysical phenomena. Mineralia Slovaca 43, 187 (2011)
14. Jin, H., Sullivan, G.F., Masson, G.M.: Distributed Applet-Based Certifiable Processing in Client/Server Environments. In: Proceedings of the the 7th Symposium on the Frontiers of Massively Parallel Computation (FRONTIERS 1999), p. 44. IEEE Computer Society, Washington, DC (1999)

15. Szemla, P.: Web based environment for distributed calculations. Engineering Thesis, WGGiOS, AGH (2013)
16. Wrzuszczak-Noga, J., Borzemski, L.: Comparison of MPI Benchmarks for Different Ethernet Connection Bandwidths in a Computer Cluster. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2010. CCIS, vol. 79, pp. 342–348. Springer, Heidelberg (2010)
17. Piorkowski, A.: Methods of creating database applications in.NET environment. In: Kwiecien, A. (ed.) Computer Networks 2007, Computer Networks – Applications and Uses, vol. 2, pp. 195–202. WKiL, Warsaw (2007)
18. Flak, J., Gaj, P., Tokarz, K., Wideł, S., Ziębiński, A.: Remote Monitoring of Geological Activity of Inclined Regions – The Concept. In: Kwiecień, A., Gaj, P., Stera, P. (eds.) CN 2009. CCIS, vol. 39, pp. 292–301. Springer, Heidelberg (2009)
19. Szostek, K., Leśniak, A.: Parallelization of the seismic ray trace algorithm. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 411–418. Springer, Heidelberg (2012)
20. Onderka, Z., Schaefer, R.: Markov chain based management of large scale distributed computations of earthen dam leakages. In: Palma, J.M.L.M., Dongarra, J. (eds.) VECPAR 1996. LNCS, vol. 1215, pp. 49–64. Springer, Heidelberg (1997)
21. Dwornik, M., Pięta, A.: Parallel Implementation of Stochastic Inversion of Seismic Tomography Data. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 353–360. Springer, Heidelberg (2012)
22. Danek, T., Slawinski, M.A.: Bayesian inversion of VSP traveltimes for linear inhomogeneity and elliptical anisotropy. Geophysics 77(6), R239–R243 (2012)
23. Pałka, P.: Multilateral negotiations in distributed, multi-agent environment. In: Jędrzejowicz, P., Nguyen, N.T., Hoang, K. (eds.) ICCCI 2011, Part II. LNCS, vol. 6923, pp. 80–89. Springer, Heidelberg (2011)