Run-time Adaptable Business Process Decentralization

Faramarz Safi Esfahani

Department of Software Engineering,
Islamic Azad University, Najaf Abad Branch,
Esfahan, Iran.
fsafi@iaun.ac.ir

Masrah Azrifah Azmi Murad,
Md. Nasir Sulaiman, Nur Izura Udzir
Faculty of Computer Science and Information Technology
University of Putra Malaysia, 43400, Serdang,
Selangor, Malaysia.
{masrah, nasir, izura}@fsktm.upm.edu.my

***Abstract* - BPEL specified business processes in the Service Oriented Architecture (SOA) are executed by non-scalable centralized orchestration engines. In order to resolve scalability issues, the centralized engines are clustered, which is not a final solution either. Alternatively, several decentralized orchestration engines are being emerged with the purpose of decentralizing a BPEL process into fragments, statically. Fully decentralization of a process into its building activities is an example of static fragmentation methods. The fragments are then encapsulated into run-time components such as agents. There are a number of attitudes towards workflow decentralization; however, only a few of them consider the adaptability of produced fragments with a run-time environment. The run-time adaptability can be studied from different aspects such as the proportionality of workflow fragments with number of machines dedicated to a workflow engine or runtime circumstances such as available bandwidth. In our opinion, the SOA suffers from the lack of decentralization adaptability with run-time environments in the orchestration layer. It demands the mapping of run-time circumstances to a suitable fragmentation model. In this paper, a mapping algorithm is presented, which is based on the number of machines and available bandwidth. Evaluation of the presented algorithm for adaptable decentralization demonstrates an improvement of the bandwidth usage compared to a fully decentralized process.**

Keywords-Adaptive Systems; Service Oriented Architecture; Distributed Orchestrate Engine; Self-* Systems; BPEL; Mobile Agents;

I. Introduction

Business processes might be very large, geographic location dependent, long running, carrying a vast number of calculations, manipulating a huge amount of data and will eventually be realized as thousands of concurrent process instances. Such workflows might be found in different areas of industry and even technology. Authors in [1], refer to the applications of business processes in industries such as chain management, online retail, or health care to consist of complex interactions among a large set of geographically distributed services deployed and maintained by various organizations. In addition, an electronic manufacturer is also reported that employs business processes to conduct its operations including component stocking, manufacturing, warehouse, order management and sales forecasting. There exist geographically distributed parties such as a number of suppliers, several organizational departments, a dozen of sales centers, and many retailers. Requests for such processes from different parties all together naturally result in creating thousands of concurrent executing instances. Such number of concurrent requests is a natural fit to this paper. In addition, new software paradigms introduced in the Cloud computing such as software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) are targeted to receive a huge number of requests. Particularly, in the case of orchestrate engine as a service, a huge number of workflow instances might be deployed and requested from various clients all around the world. In order to handle the requests, different number of machines and also resources must be employed. This paper proposes an adaptable and distributed workflow engine, which tackles such an ever-changing environment.

According to the SOA stack [2], business logic layer consists of orchestration and choreography layers. The choreography layer is intrinsically distributed to several distinct workflows communicating with each other and normally run on different workflow engines, whereas the orchestration layer is workflow engine centric. Indeed, a single engine [3] is usually applied to execute a business process and scalability is naturally addressed by replicating orchestration engines, which is not a final solution for scalability problems of centralized engines, entirely [1, 4, 5]. The decentralization of business processes has been introduced as an alternative solution, which is currently based on system analyst and designers' opinion and is carried on at design time, without paying attention to the fact that ever-changing run-time environment raises special requirements on which there is no information at design time.

From this paper point of view and according to [2], business process decentralization methods can be studied from three aspects including fragmentation, enactment and adaptability. A number of decentralized workflow engines have emerged to support these aspects of decentralization. The most challenging area is adaptability, which means the ability of system to reconfigure its component to refrain from or lessen system bottlenecks such as throughput, response time and bandwidth usage. In order to achieve adaptability, the system must be able to react to run-time circumstances and reconfigure itself. A fully process decentralization (FPD) method is applied by [1, 4, 6-10], which decentralizes a business process to activity level fragments. The fragments are encapsulated in run-time components and a third-party middleware is applied to support the communication among fragments. Adaptability also comes about by locating/relocating the run-time components based on system conditions. The FPD negatively produces a number of fragments, which their message passing and resource usage will eventually result in swamping the run-time environment. The main reason is that the fragments are statically produced without considering run-time circumstances. In contrast, there are several dynamic fragmentation methods [11-13], which produce fragments at run-time without considering the run-time circumstances. Thus, the produced fragments may cause violating system thresholds.

In our opinion, the mentioned methods suffer from the following problems. 1) Lack of dynamic criteria to workflow decentralization. 2) Improper selection of activities in decentralized process fragments. On one hand, encapsulating each activity in one run-time component provides a high number of components, along with a plenty of message passing and will eventually increase bandwidth usage. It also results in high response time due to a huge amount of message passing and most importantly low system throughput due to high resource consumption. On the other hand, encapsulating coarser fragments based on static criteria will result in less system flexibility as well as adaptability with run-time environment.

Having an abstract layer in the SOA architecture to realize the adaptability of decentralization with run-time environment can be a solution for the mentioned problems. This layer may seem to be an overhead for executing processes; however, it is a tradeoff among different aspects of system. The negative effects of this layer can also be mitigated by creating the fragments in advance, managing workflow states and etc. There have been experiments [14] that show the gained advantages is eye-catching enough which is a good motivation for presenting this abstract layer. Indeed, the main objective of this work is presenting and implementing the idea of adaptable business process decentralization based on current run-time circumstances. In fact, the current system condition is mapped to a suitable decentralization method. In addition, the contributions of this work are: 1) introducing the idea of run-time adaptable business process decentralization. 2) Presenting a bandwidth adaptable workflow decentralization method.

It is also worth mentioning that this paper focuses on block-structured business processes. In addition, several aspects of workflow management systems are not included in this work such as governance of workflow fragments, managing run-time state of workflow fragments, run-time workflow reconfiguration, transaction, exception handling and sharing workflow fragments interior variables. These are normally implemented by a distributed middleware [11-13] or from dynamic process decentralization view; they demand more attention in future work as well.

II. Background and Related Work

Open World Software Paradigm: Baresi et al., in [15], open a new view towards software development by introducing the idea of open-world software paradigm, which has attracted much attention nowadays. According to this idea, in the open-world paradigm, software is executed in an ever-changing environment; therefore, static design time metrics will not be responsive at run-time. Although the run-time environment changes continuously, it is the software itself, which has to be self-healed and self-adapted to keep the whole system in a safe side. Generally, changes in run-time environment are not predictable at design time; therefore, evolving software at run-time is necessary. Software thus needs to continuously and automatically adapt itself and react to the changes. Systems will need to operate correctly despite of unexpected changes in factors such as environmental conditions, user requirements, technology, legal regulations, and market opportunities. This work brings an example of applying open-world idea to service oriented applications.

A few research works has been performed on self-adapting and self-healing of the service-oriented applications, especially from service composition point of view. However, none of them considers self-adapting of business process decentralization with run-time environment. This paper draws the idea of open-world paradigm into business process composition from decentralization point of view. A decentralized workflow engine is required to be adequately flexible, dynamic and adaptive to handle the changes by providing adaptable fragments. The focus of this work is on implementing a decentralized workflow engine, which creates adaptable fragments from a business process based on run-time environment feedbacks. Adaptability comes about in terms of first) number of dedicated machines to a workflow engine; second) the available bandwidth of media, which connect the workflow engine machines. Based on these adaptability aspects, a bandwidth adaptable algorithm is also presented to choose a suitable decentralization method as well.

Adaptability of Decentralization Based on Workflow Circumstances: Dartflow project [11] has shown an usage of mobile agents in distributed workflow execution. In Dartflow, the workflow model is fragmented dynamically, and the partitions are carried by mobile agents and sent to different sites, which are responsible for them. This work establishes good points for dynamicity of error handling and data sharing among agents in a dynamic workflow system; however, it focuses on the system architecture and does not detail the fragmentation model, adequately.

An abstract and conceptual dynamic workflow fragmentation method is shown by [12], which applies the Petri net formalism. The presented method partitions the centralized process into several fragments step by step, while the process is executed. The created fragments are able to migrate to proper servers, where tasks are performed and new fragments are created and forwarded to other servers (i.e., using mobile agents) to be executed. Dynamism in decentralization is a prominent aspect of this work. The fragmentation method of this work is different from our work in that it considers workflow run-time condition to decentralize a business process, while run-time environment circumstances are applied for the same purpose in this paper. Nonetheless, this method is similar to our work in that the fragments must be prepared beforehand in a fragment pool or must be built on the fly at runtime.

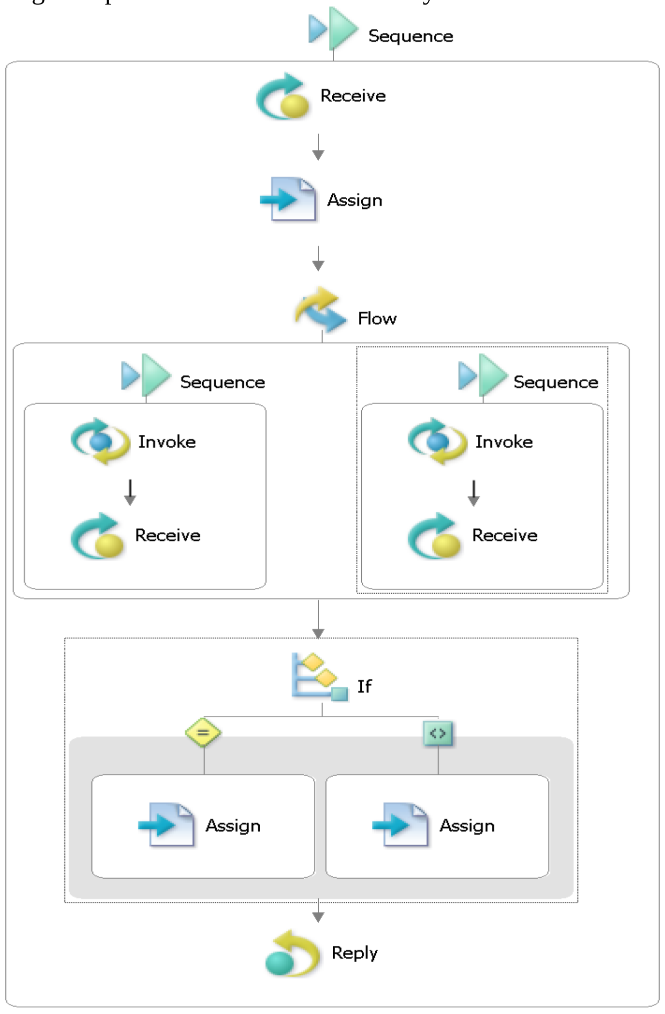


Figure 1: BPEL View of Loan Application Process

In [13], a decentralized workflow model is presented for inter-organizational workflow decentralization, where inter-task dependencies are enforced without requiring to have a centralized WFMS. This work is different from our work in that it considers a criterion to decentralize a business process from inter-organizational point of view which is fitted to choreography layer of service oriented architecture. The produced fragments can be considered as inputs of our work for dynamic decentralization. This work is also an inter-organizational version of the research paper [12], which partitions the workflows on the fly. It is also different from our work in that a workflow is partitioned based on workflow run-time conditions, while this paper is targeted to use run-time environment circumstances for decentralization purposes.

In [16], the ever-changing legislation of governments, customers' needs and other changes in environment of business processes are introduced as the main reasons of implementing various forms of business process applications in different organizations. This research work looks for a way of providing highest level of flexibility as well as adaptability to the changes in run-time environment. The decentralization methods introduced in this current paper require flexible architectures to support dynamic fragmentation, which execute different execution forms of a business process at runtime.

Adaptability of Decentralization Based on Run-time Circumstances: This study is also an extension of our previously published papers, which were totally on introducing dynamic criteria for business process decentralization. In [17, 18], mere idea and motivations of using a mining method for intelligent process decentralization (IPD) were introduced and the improvement of only response time was *mathematically* shown for several sample BPEL processes. Moreover, [19] showed an SLA driven aspect of the IPD as well. Furthermore, hierarchical process decentralization criterion (HPD) and its composition with the IPD for two different case studies were shown in [14, 20]. In this current research study, the HPD method is considered as a decentralization method, which decentralizes a business process based on the hierarchy of activities. For instance in $level_0$ of the process tree, the whole process is considered as a fragment that is equal to a Centralized process. In the $level_{n-1}$, which is the last level of the process tree, each process activity is considered as an individual fragment and it is analogous to the FPD method. The middle layers of the process tree provide coarser fragments. Based on the level of decentralization, different numbers of fragments are produced. This paper presents a method for process decentralization, which applies the number of workflow engine machines and available bandwidth to choose a suitable level of decentralization in the process tree.

III. HPD Decentralization of Loan Process

Unfortunately, there is no standard business process for benchmarking BPEL processes. Nonetheless, a loan application business process has been applied in [1, 10, 21, 22] that also fits our research. The loan process illustrated in Figure 1 is decentralized based on the HPD decentralization approach as shown in Figure 2. It makes us able to study several simple and structured BPEL activities together. The loan process consults with two external web services, which send a credit report for the loan applicant. In order to refrain from approbating risky loans, the loan request is accepted, when both web services confirm the applicant's credit. According to the HPD decentralization method, the loan process can be decentralized based on the levels of the process tree. The $level_0$ or HPD0 contains only one fragment, which is equal to the Centralized model. The $level_1$ which is analogous to HPD1 provides six fragments, HPD2 in the $level_3$ decentralizes the process to ten fragments and finally HPD3, which is the finest fragmentation model and is also called the FPD, fully decentralizes the loan process into sixteen fragments each of which contains only one activity.

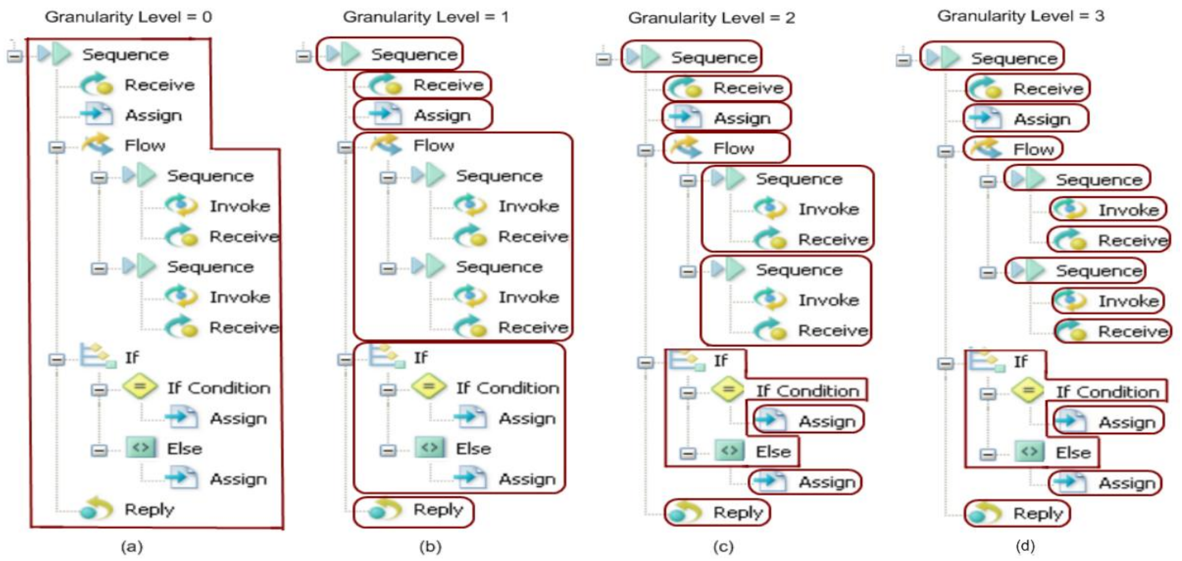


Figure 2: HPD Decentralization of Loan Application Business Process

IV. Adaptable Process Decentralization Framework

A run-time environment is the subject of many changes during the execution of software applications, which may result in violating system thresholds. The adaptability of software with requirements of its execution environment is important in that it helps the run-time environment to refrain from catastrophic events.

Distributed systems are used to resolve the scalability issues of centralized models. However, distribution may result in performance bottleneck due to the communication among system components and continuous changes in a distributed environment. Adaptability of a distributed system, i.e. a decentralized orchestrate engine, with its environment is of high importance to avoid approaching system thresholds and bypassing bottlenecks.

Figure 3 shows the main phases of an Adaptable and Decentralized Workflow Execution Framework (ADWEF) to support the adaptable decentralization of business processes. The central part of the framework is a feedback data repository, which can be initialized by different parties such as a system administrator, a distributed workflow engine, configuration files, monitoring devices and software, etc. Based on the data provided in the feedback repository, a decentralization decision maker may be able to decentralize/re-decentralize a new/running business process. Re-decentralization may occur due to violating system thresholds. Making decision on how to decentralize a business process, the decision maker submits required information to the workflow decomposer which is able to fragment a business process to workflow fragments. Through a process of deployment, the produced fragments are encapsulated into runtime components (i.e., agents) and they will be deployed into the machines dedicated to a distributed workflow engine. Dynamic architectures, which support the execution of dynamic fragments, have to implement collaborative components to reinforce each of the phases specified in the framework.

V. Adaptable Decentralization Decision Maker Unit

This section elaborates an adaptable decentralization decision maker unit. Adaptability may come along with different criteria such as memory usage, bandwidth usage, throughput, etc. Nevertheless, having all of them together is impossible due to confliction of goals. This section also opens discussions on considering run-time circumstances in business process decentralization. Figure 4 also presents a decentralization decision maker unit, which offers a suitable level of decentralization based on receiving two parameters from run-time environment including the number of available machines and available bandwidth. The decision maker unit is implemented using a Fuzzy approach in this paper.

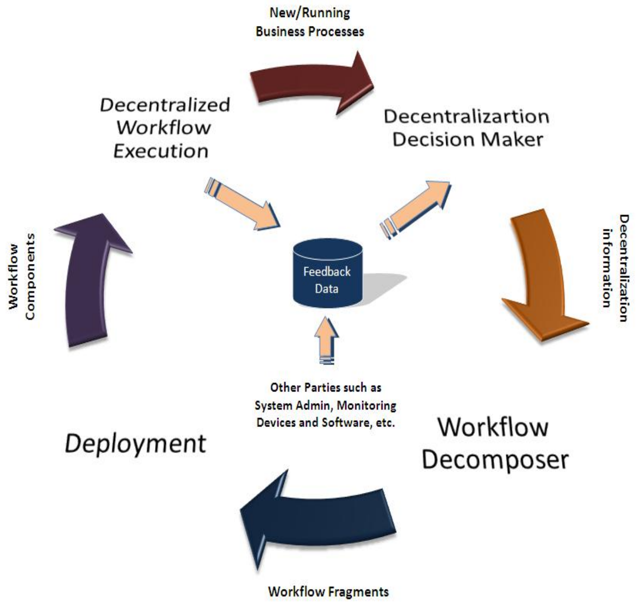

New/Running Business Processes
Decentralized Workflow Execution
Decentralizartion Decision Maker
Decentralization information
Workflow Components
Feedback Data
Other Parties such as System Admin, Monitoring Devices and Software, etc.
Deployment
Workflow Decomposer
Workflow Fragments

Figure 3: Adaptable and Decentralized Workflow Execution Framework (ADWEF)

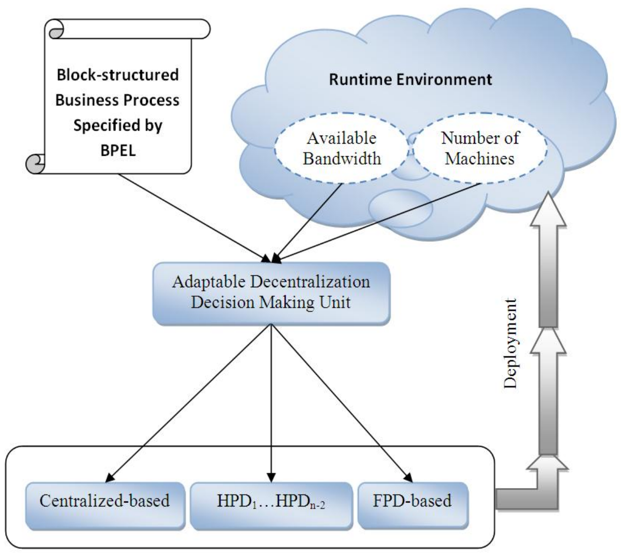


Figure 4: Decentralization Decision Maker

```
name: fuzzyGranularity;
input:
  ps (Process Specification),
  bw (Bandwidth),
  nom (Number of Machines);
output:
  granularityLevel;
begin
  | fsa = fragmentSetArray (ps, "HPD");
  | fp = findFragmentProportionality (fsa, nom);
  | fpl = findFragmentProportionalityLevel (fp)
  | granularityLevel =  findFuzzyGranularity (fsa, fpl, bw);
  | return granularityLevel;
end;
```

Figure 5: Adaptable Fuzzy Decentralization Algorithm

VI. Bandwidth Adaptable Decentralization

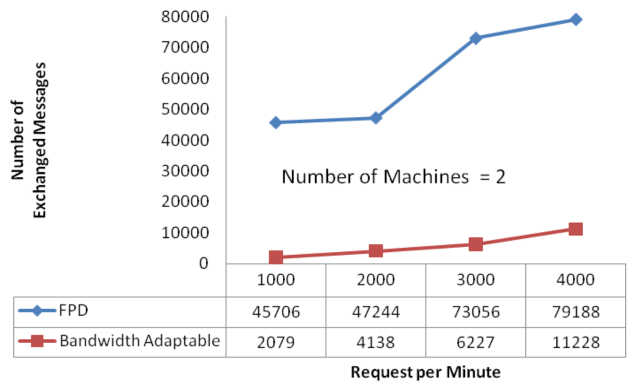
Bandwidth is important in that it is independent from other parameters such as response time and throughput; however, response time and throughput are highly affected by the available bandwidth. A busy communication media may increase the latency of communication among components, which results in increasing response time and decreasing throughput, consequently.

A sample implementation of decision making unit is shown in Figure 5. It shows the *fuzzyGranularity* algorithm as well as its input parameters including process specification, current bandwidth and number of machines. The level of decentralization will be the output parameter. At first, the business process specification is sent to *fragmentSetArray* by identifying the method of fragmentation i.e. HPD; second, fragment proportionality is determined by *findFragmentProportionality*. It determines that the number of produced fragments in which decentralization level is closer to the number of machines. The closer number is called fragment proportionality. Then, the level of a business process tree, which satisfies the fragment proportionality, is returned by the next method *findFragmentProportionalityLevel*.

Finally, by having all the fragments, available bandwidth and fragment proportionality level, granularity level is calculated by the *findFuzzyGranularity* method. This method receives fragment set array (*fsa*), fragment proportionality level (*fpl*) and available bandwidth (*bw*) as input and returns the level of decentralization in process tree as output. At first, bandwidth is segmented dynamically to a set of Segments (S_i) and then; *bw* is fuzzified using a singleton function. For each segmented bandwidth S_i a new rule is created using a singleton function such that $S_i \rightarrow$ *Singleton (fsa[i].fragmentNo())*. The created rules are executed using a rule engine. Output is defuzzified later and finally a crisp value is calculated. The crisp value determines a suitable level of process tree, which is the final result of *findFuzzyGranularity* method.

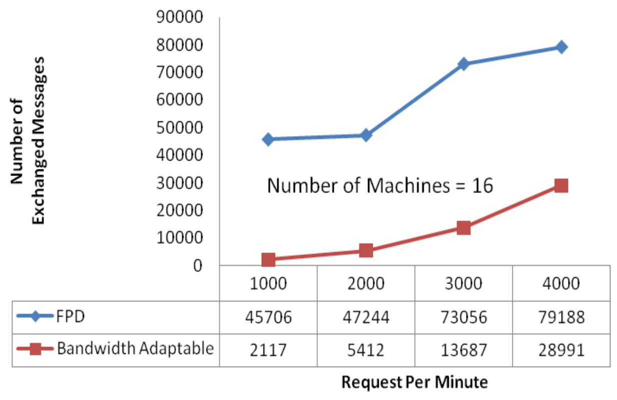
VII. Experimental Setup and Evaluation

The focus of this section is evaluating the behavior of the bandwidth-adaptable fuzzy decentralization decision making unit. The algorithm is expected to adapt decentralization of processes with current available bandwidth. It considers the fragment proportionality of decentralization with number of machines as well.



	1000	2000	3000	4000
FPD	45706	47244	73056	79188
Bandwidth Adaptable	2079	4138	6227	11228

Figure 6: Comparing exchanged messages by fragments of FPD and the presented algorithm using two machines.



	1000	2000	3000	4000
FPD	45706	47244	73056	79188
Bandwidth Adaptable	2117	5412	13687	28991

Figure 7: Comparing exchanged messages by fragments of FPD and presented algorithm using sixteen machines.

In order to implement the experiments, WADE/JADE [23-26] platform was selected and installed on a network with sixteen machines. The created fragments using the HPD method were encapsulated in WADE/JADE agents and deployed to network machines. Accordingly, the experiment was repeated for two and sixteen machines. A client sent requests with specific rates of 1000, 2000, 3000 and 4000 per minute. During the experiments, sniffer software monitored the number of messages exchanged among the fragments. Receiving a request from a client, a number (between 0-100) was generated with exponential distribution, which was the simulation of available bandwidth. The available bandwidth along with the fragment proportionality parameter was sent to a fuzzy algorithm to recommend a suitable level of decentralization. The same experiments were repeated without the presence of bandwidth adaptable algorithm, which was analogous to applying the FPD method.

Both Figure 6 and Figure 7 show the results of the experiments. The fully process decentralization method (FPD) was neutral to the number of machines and bandwidth fluctuations; thus, a constant number of messages were passed among the agents. In contrast with the FPD, the bandwidth adaptable algorithm, decentralized the loan process based on the generated bandwidth and number of machines. As shown in Figure 2, different versions of the loan process were created at run-time. The adaptable algorithm reduced the number of exchanged messages in both cases due to considering the adaptability of fragments with number of machines and available bandwidth. In the case that enough bandwidth was available, the algorithm considered only the adaptability with number of machines. When the bandwidth was not wide enough, the algorithm shrunk the fragments and created more centralized fragments.

VIII. CONCLUSION

An adaptable business process decentralization framework was introduced as a solution for decentralizing large scale business processes. The main problem was that current decentralization methods did not consider the adaptability of business process decentralization with run-time environment circumstances such as available bandwidth in this paper.

By decentralizing a business process based on the number of machines, the extra communication cost of inter-fragment communication is omitted. If there is only one machine to execute processes, there is no need to create several fragments running on one machine. In other words, a multi-thread process would be more effective. In the case of several machines, the proportionality of produced fragments with number of machines dedicated to a distributed engine reduces the number of fragments and consequently communication among them. On the other hand, the available bandwidth of a network may directly affect the response time and throughput of workflows. Imagine a busy communication media in a distributed workflow engine. Under such circumstances, creating a fully fragmented process may result in a huge amount of exchanging messages among the fragments and then system approaches/violates its thresholds, consequently. Obviously, the less number of fragments provides better outcomes in this case. As a matter of fact, the process is shrunk to refrain from violating the thresholds. After ameliorating the run-time conditions, current processes and/or new processes, may be expanded/re-expanded by creating more fragments.

Indeed, in this paper, an adaptable process decentralization framework was introduced to create fragments proportional to both the number of machines and current available bandwidth. The evaluation of bandwidth adaptable algorithm showed that this algorithm was able to execute a process with considerably less number of exchanged messages compared to the FPD method. The fragments of the FPD exchanged ten to twenty times more messages compared to the bandwidth adaptable algorithm.

It is worth mentioning that the adaptable decentralization decision making unit opens more discussions on finding more adaptability metrics and more intelligent algorithms to achieve better decentralization outcomes. Considering the network capacity, accumulative bandwidth, the relation of workflow activities, etc can be instances of such adaptability metrics. Furthermore, algorithms are required to decentralize graph-structured business processes and mapping them to run-time circumstances. Currently, our main focus is on developing a Fuzzy algorithm to integrate the HPD and HIPD methods to provide a more adaptable decentralization approach.

ACKNOWLEDGMENT

Faramarz Safi Esfahani is also a PhD candidate in University of Putra Malaysia, 43400, Serdang, Selangor, Malaysia.

REFERENCES

[1] Guoli Li, Vinod Muthusamy, and Hans-Arno Jacobsen, “A Distributed Service Oriented Architecture for Business Process Execution,” ACM Transactions on the Web (TWEB), vol. 4, no. 1, article 2, 2010.

[2] Paolo Bruni, Marcos Henrique Simoes Caurim, Alexander Koerner, Christine Law, et al., Powering SOA with IBM Data Servers, ISBN. 738494542, IBM, 2006, p. 754.

[3] “Workflow management coalition: process definition interchange,”, http://www.wfmc.org, 2011.

[4] Mirkov Viroli, Enrico Denti and Alessandron Ricci, “Engineering a BPEL orchestration engine as a multi-agent system,” Journal of Science of Computer Programming, vol 66, issue 3, 2007, pp. 226-245 2007.

[5] Roberto Silveira Silva Filho, Jacques Wainer and Edmundo Roberto Mauro Madeira “A fully distributed architecture for large scale workflow enactment,” International Journal of Cooperative Information Systems, vol. 12, no. 4, 2003, pp. 411-440.

[6] Giancarlo Fortino, Alfredo Garro, Wilma Russo, “Distributed Workflow Enactment: an Agent-based Framework,” Proc. WOA2006, 2006, pp. 110-117.

[7] Li Guo, Dave Robertson, Yun-Heh Chen-Burger, “A Novel Approach for Enacting the Distributed Business Workflows Using BPEL4WS on the Multi-Agent Platform,” Web Intelligence and Agent Systems, 2005, pp. 657-664.

[8] Daniel Wutke, Daniel Martin and Frank Leymann, “Model and Infrastructure for Decentralized Workflow,” Proc. ACM/SAC 2009, ACM, 2008, pp. 90-94.

[9] Gustavo Alonso, Fabio Casati, kuno Harumi and Machiraju Vijay, “Exotica/FMQM: A Persistent Message-Based Architecture for Distributed

Workflow Management," Proc. IFIP WG 8.1 Workgroup Conference on Information Systems Development for Decentralized Organizations (ISD095), 1995, pp. 1-18.
[10] Vinod Muthusamy, Hans-Arno Jacobsen, Tony Chau and Allen Chan, "SLA-driven business process management in SOA," Proc. CASCON, Toronto, Canada, 2009, pp. 86-100.
[11] Ting Cai, Peter A Gloorand and Saurab Nog, "DartFlow: A workflow management system on the web using transportable agents," Technical Report PCS-TR96-283, Dartmouth College, Hanover, NH, 1996, http://www.cs.dartmouth.edu/~jcrespo/cms_file/SYS_techReport/156/TR96-283.pdf.
[12] Wei Tan and Yushun Fan, "Dynamic workflow model fragmentation for distributed execution," Comput. Ind., vol. 58, no. 5, 2007, pp. 381-391; DOI http://dx.doi.org/10.1016/j.compind.2006.07.004.
[13] Vijayalakshmi Atluri, Soon Ae Chun, Ravi Mukkamala and Pietro Mazzolen, "A decentralized execution model for inter-organizational workflows," Distrib. Parallel Databases, vol. 22, no. 1, 2007, pp. 55-83; DOI http://dx.doi.org/10.1007/s10619-007-7012-1.
[14] Faramarz Safi Esfahani, Masrah Azrifah Azmi Murad, Md. Nasir Sulaiman and Nur Izura Udzir," Adaptable Distributed Service Oriented Architecture," Elsevier Journal of Systems and Software (JSS), in press.
[15] Luciano Baresi, Elisabetta Di Nitto and Carlo Ghezzi, "Toward open-world software: Issue and challenges," Computer, vol. 39, no. 10, 2006, pp. 36-43.
[16] Yiwei Gong, Marijn Janssen, Sietse Overbeek and Arre Zuurmond, "Enabling flexible processes by ECA orchestration architecture," Proc. ICEGOV, 2009, pp. 19-26.
[17] Faramarz Safi Esfahani, Masrah Azrifah Azmi Murad, Md. Nasir Sulaiman and Nur Izura Udzir, "Using Process Mining To Business Process Distribution," Proc. SAC2009, ACM, 2009, pp. 1876-1881.
[18] Faramarz Safi Esfahani, Masrah Azrifah Azmi Murad, Md. Nasir Sulaiman and Nur Izura Udzir, "An Intelligent Business Process Distribution Approach," Journal of Theoretical and Applied Information Technology, vol. 4, 2008, pp. 1236-1245.
[19] Faramarz Safi Esfahani, Masrah Azrifah Azmi Murad, Md. Nasir Sulaiman and Nur Izura Udzir, "SLA-Driven Business Process Distribution," Proc. IARIA/eKnow2009, IEEE, 2009, pp.14-21.
[20] Faramarz Safi Esfahani, Masrah Azrifah Azmi Murad, Md. Nasir Sulaiman and Nur Izura Udzir, "A Case Study of the Intelligent Process Decentralization Method," Proc. WCECS, IAENG, 2009, pp 269-274.
[21] Rania Khalaf and Frank Leymann, "E Role-based Decomposition of Businesses using BPEL," Proc. IEEE International Conference on Web Services (ICWS'06), 2006, pp. 770-780.
[22] Mangala Gowri Nanda, Satish Chandra and Vivek Sarkar, "Decentralizing execution of composite web services," ACM SIGPLAN Notices, vol. 39, no. 10, 2004, pp. 170-187.
[23] Fabio Bellifemine, Agostino Poggi and Giovanni Rimassa, "JADE–A FIPA-compliant agent framework," Proc. PAAM99, 1999, pp. 97-108.
[24] Giovanni Caire, Danilo Gotta and Massimo Banzi, "WADE: a software platform to develop mission critical applications exploiting agents and workflows," Proc. 7th international joint conference on Autonomous agents and multiagent systems: industrial track, Richland, SC, 2008, pp. 29-36.
[25] Krzysztof Chmiel, Maciej Gawinecki, Pawel Kaczmarek, Michal Szymczak, et al., "Efficiency of JADE agent platform," Journal of Scientific Programming, vol. 13, no. Number 2/2005, 2005, pp. 159-172.
[26] Bellifemine Fabio, Caire Giovanni and Greenwood Dominic, Developing Multi-Agent Systems with JADE, WILEY, 2007.