Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

EVALUATION OF FREQUENT-PATH ADAPTABILITY ASPECT IN ADWEF APPLYING BASIC BUSINESS PROCESSES

[1]FARAMARZ SAFI ESFAHANI, [2]MASRAH AZRIFAH AZMI MURAD
[1]Asstt. Prof., Islamic Azad University, Najafabad Branch, Department of Computer Science, Esfahan, Iran
[2]Assoc. Prof., Masrah Azrifah Azmi Murad, Faculty of Computer Science and Information Technology, University of Putra Malaysia (UPM), 43400, Selangor, Malaysia
Email: [1]fsafi@iaun.ac.ir, [2] masrah@fsktm.upm.edu.my

ABSTRACT

Centralized business process execution in the Service Oriented Architecture (SOA) suffers from lack of scalability. Decentralization of business processes is introduced as an alternative approach to address this shortcoming. However, decentralization methods do not consider adaptability of created fragments with runtime environment. The Adaptable and Decentralized Workflow Execution Framework (ADWEF) introduces frequent-path and proportional-fragment adaptability aspects along with two architectures Type-1 and Type-2 and two decentralization methods HPD and HIPD. The ADWEF has been tested for several boundary-condition and one complex business processes; nonetheless, the behavior of prominent activities of business processes has not been studied yet. This paper introduces If-process, While-process and Flow-process as three basic business processes to study the behavior of their central activities from frequent-path adaptability point of view in terms of response-time and throughput running on architecture Type-1. Evaluations of the basic processes show that considering the frequent-path adaptability provides a range of improvements in response-time and throughput with variable-request-rate/constant-message-size and constant-request-rate/variable-message-size configurations.

Keywords: *Service Oriented Architecture; Business Process Decentralization; Adaptable Business Process Decentrlaization; Self-* Systems; Self-Adaptability;*

1. INTRODUCTION

Scalability is the main motivator for decentralization of business processes in the Service Oriented Architecture (SOA). However, current decentralization methods do not consider adaptability of created fragments with runtime environment. Three adaptability aspects of created fragments with runtime environment are introduced by [1, 2]. In [1], the **frequent-path** and **proportional-fragment** adaptability aspects are introduced. The **frequent-path** considers closely-interrelated activities and encapsulates them in an individual fragment. The **proportional-fragment** decentralizes a business process based on the number of machines dedicated to a workflow engine. The **available-bandwidth** adaptability aspect introduced by [2] creates fragments that are commensurate with runtime environment. In addition to these adaptability aspects, an Adaptable and Decentralized Workflow Execution Framework (ADWEF) is presented in [1, 2] that contains two architectures Type-1 and Type-2 as well as two decentralization methods HPD and HIPD to support the mentioned adaptability aspects. The HPD method decentralizes a block-structured business process based on the levels of business process tree. The HPD method in the first level of process tree is equal to the traditional Centralized method and in the last level of process tree is analogous to the Fully Process Decentralization (FPD). The HIPD method detects closely-interrelated activities in the process tree and encapsulates them in a separate fragment. Simultaneously, each fragment can be decomposed in terms of the levels of process tree. The HIPD fragment in the first level of process tree is called IPD [3-6] which encapsulates a frequent path from process tree root to leaves. The abilities of the adaptability aspects are examined by three boundary-condition and one complex loan application business processes in [1]. However, the main problem is that the behavior of basic as well as primitive activities such as If, While, and Flow

Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

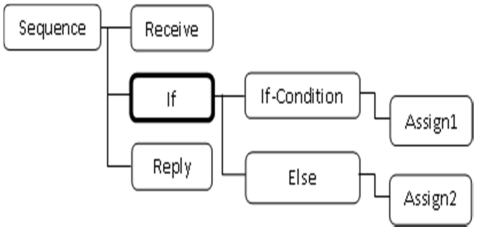
in a simple business process has not been studied yet; therefore, the processes studied in this paper are limited to If-process, While-process, and Flow-process. These three basic processes are used to study their central activities (If, While, Flow) in terms of response-time and throughput in the architecture Type-1 [1] in order to evaluate the frequent-path adaptability aspect.

2. EVALATION METHOD AND METRICS

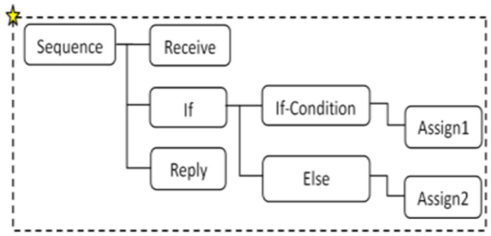
The experiment environment is setup according to [1] based on which Two metrics are considered to be evaluated by the experiments as follows. ***Response-time*** is the time difference between invoking a workflow until receiving a reply from it. ***Throughput*** is the number of completed requests per time unit divided by the total number of requests. The frequent-path evaluation is performed through measuring the response-time and throughput of the fragments. In business process decentralization, process activities and their relations are under focus; therefore, no extra code is considered inside process activities. The fragments created by both HPD and HIPD methods are then compared by both Centralized and FPD methods.

3. EXPERIMENTAL SETUP

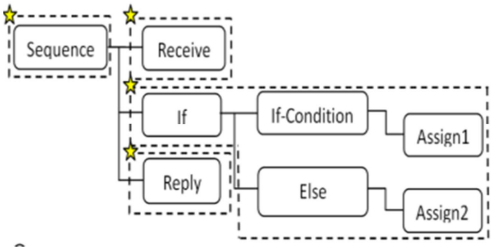
Three basic processes are used to evaluate the behavior of their central activity such as If (Switch), While, and Flow as shown in Figure 1(A), Figure 2(A), and Figure 3(A). In the first step, a basic process is considered for decentralization. Two decentralization methods HPD and HIPD decentralize the basic processes in terms of process tree levels; however, the HIPD recognizes frequent and infrequent paths of previously executed workflows prior to decentralizing processes. For each of the basic processes a path is assumed as frequent. The produced fragments are then encapsulated in agents and then they are deployed into runtime environment based on the architecture Type-1[1] .



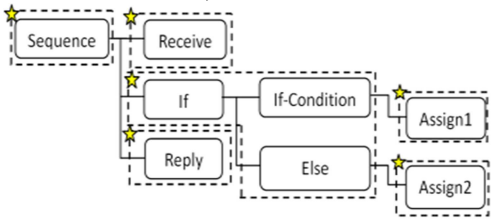
A) Basic If-process



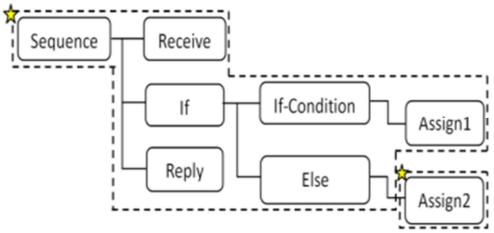
B) HPD0_1



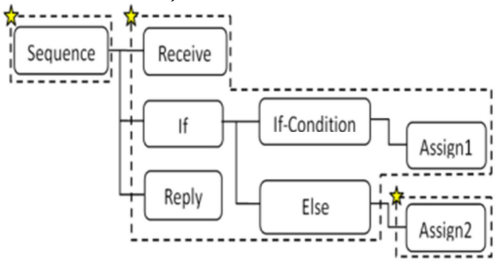
C) HPD1_4



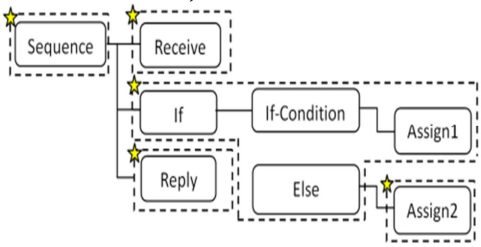
D)HPD2_6= FPD



E)HIPD0_2= IPD



F)HIPD1 _3

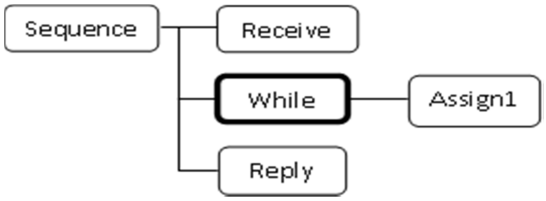


G)HIPD2_5

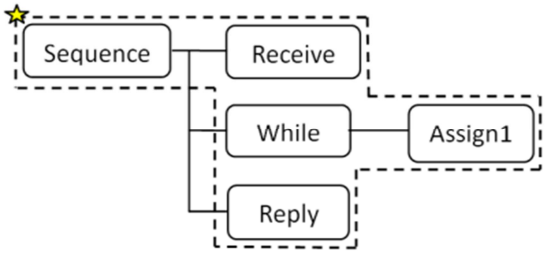
Figure 1: If-process Decentralization Patterns

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

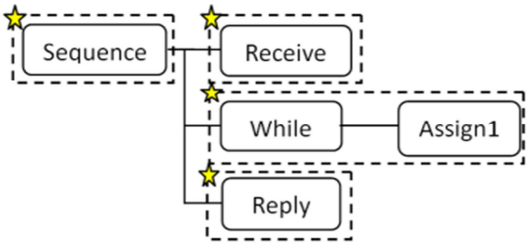
The experiments are performed in *Minimum and Maximum (Min-Max) distributions of* fragments. The *Minimum distribution* applies one machine and in the architecture Type-1, it is equal to applying only one performer agent to execute process fragment. The *Maximum distribution* for the architecture Type-1 means to allocate exactly one machine to each performer agent so that they are able to run different process fragments at the same time. It shows the behavior of a fragment when each fragment is dedicated a server.



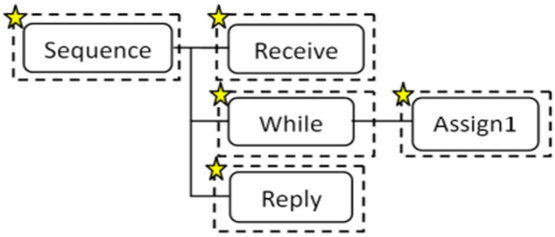
A) Basic While-process



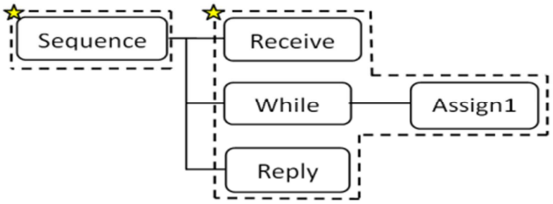
B) HPD0_1=Centralized=IPD



D) HPD1_4



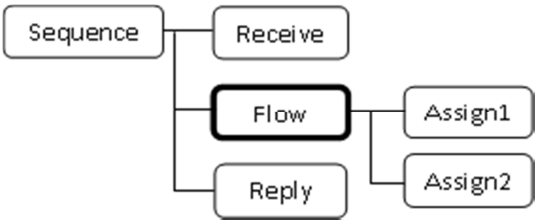
E) HPD2_5=HIPD2_5=FPD



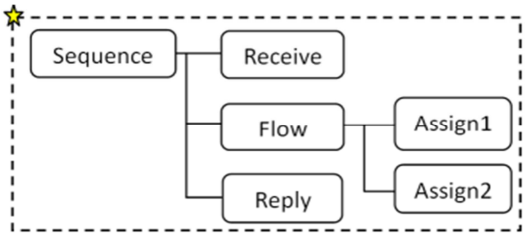
F) HIPD1_2

Figure 2: While-process Decentralization Patterns

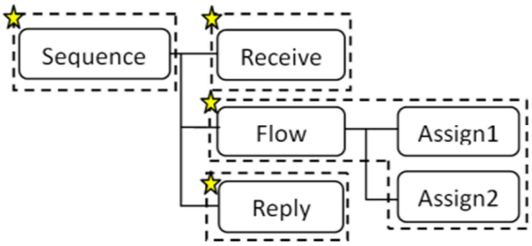
Both evaluation metrics are affected by the request rate and size of messages passed through process activities. Thus, two categories of experiments are designed. In order to omit the effect of request rate on the metrics in the first category of experiments, requests are sent to servers with variable-request-rate and constant-message-size. The second category considers constant-request-rate in presence of variable-message-size with varied values 50KB, 100KB, 150KB, 200KB and 250KB.



A) Basic Flow-process



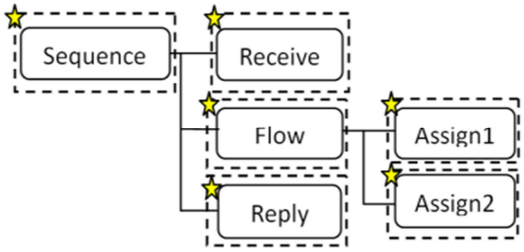
B) PD0=Centralized=HIPD0_1=IPD



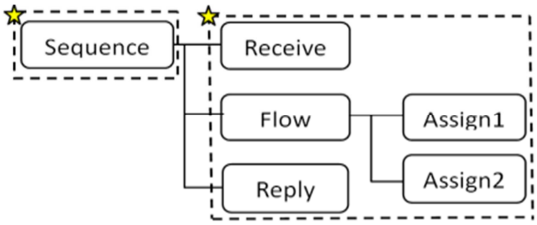
C) HPD1_4

Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



D) HPD2_6=FPD



E) HIPD1_2

Figure 3: Flow-process Decentralization Patterns

4. EXPERIMENTAL RESULTS

A. If-process Experiments

Figure 1 shows the response-time improvement trend of the FPD by other decentralization methods with variable-request-rate/constant-message-size. It is clear that the Centralized and HIPD fragments improved the FPD. It depicts that the average improvement percent of the FPD was around 98.92%, 98.85% and 98.81% by the Centralized, IPD and HIPD1_3_1 fragments. In comparison, HPD1_4_1 improved the average response-time about 57.01%. Figure 2 shows the throughput improvement trend of the FPD by other decentralization methods with variable-request-rate/constant-message-size. It is clear that the Centralized and HIPD fragments improved the FPD. It also depicts that the average throughput of the FPD was improved around 38.97%, 38.66% and 37.02%. In comparison, HPD1_4_1 improved the average throughput around 5.76%. Figure 3 shows the improvement trend of the FPD response-time by other methods with constant-request-rate/variable-message-size. The average improvement of response-time in request rate 500 by the Centralized, IPD and HIPD1_3_1 was 97.97%, 98.06% and 97.96%, respectively. In comparison, HPD1_4_1 improved the average response-time about 49.79%. Figure 4 shows the percent of throughput improvement for the decentralization methods compared to the FPD fragments with constant-request-rate/variable-message-size. In request rate 500, the average improvement of throughput by the Centralized, IPD and HIPD1_3_1 was 88.72%, 70.20% and 70.28%. In comparison, HID1_4_1 improved the FPD around 8.56%.

B. While-process Experiments

Figure 5 shows that the response-time of the FPD with variable-request-rate/constant-message-size was improved by the Centralized=IPD and HIPD1_2_1 around 99.72%, 99.53%. In comparison, HPD1_4_1 improved the average response-time of the FPD about 86.78%. Figure 6 shows the throughput improvement trend of the FPD by the other fragments of the basic While-process with variable-request-rate/constant-message-size**.** It is obvious that the Centralized, HIPD1_2_1 and HPD1_4_1 fragments improved the FPD throughput. It also depicts that the average throughput of the FPD was improved around 45.46%, 37.98% and 14.63%, respectively. Figure 7, shows the improvement trend of the FPD response-time by the other While fragments with constant-request-rate/variable-message-size. The average improvement of response-time in request rate 500 by the Centralized and HIPD1_2_1 was 99.99% and 96.52%. In comparison, HPD1_4_1 improved the average response-time about 83.61%. Figure 8 shows the improvement trend of the FPD throughput by the other fragments with constant-request-rate/variable-message-size. The average improvement of throughput in request rate 500 by the Centralized and HIPD_1_2_1 was 96.52% and 45.40%. In comparison, HPD1_4_1 improved the average response-time about 16.52%.

C. Flow-process Experiments

Figure 9 shows the response-time improvement trend of the FPD by the other decentralization methods with variable-request-rate/constant-message-size**.** It is clear that the Centralized and HIPD fragments improved the FPD. It also depicts that the average response-time improvement of the FPD was around 99.52% and 99.10% by the Centralized and IPD. In comparison, HPD1_4_1 improved the average response-time about 78.09%. Figure 10 also shows the throughput improvement trend of the FPD by other decentralization methods with variable-request-rate/constant-message-size**.** It is clear that the Centralized and HIPD fragments improved the FPD. It also depicts that throughput improvement of the FPD was around 65.10%, 34.64% and 10.38% by the Centralized, HIPD1_2_1 and HPD1_4_1. Figure 11 shows the trend of the FPD response-time improvement by the other methods with constant-request-rate/variable-message-size. The average

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

improvement of response-time in request rate 500 by the Centralized and HIPD1_2_1 was 99.98% and 94.01%. In comparison, HPD1_4_1 improved the average response-time about 67.85%. Figure 12 shows throughput improvement percent of the FPD by the other decentralization methods with constant-request-rate/variable-message-size. In request rate 500, the average improvement of throughput by the Centralized and IPD was 90.20% and 38.92%. In comparison, and HIPD1_4_1 improved the FPD around 10.16%.

5. CONCLUSION AND FUTURE WORKS

According to the experimental results, the Centralized method demonstrated the best response-time and throughput compared to the other methods. The fragments provided based on frequent-path adaptability showed better results compared to the fragments of the FPD method. The results were very close to the Centralized method. It also showed that in one single machine the less number of fragments showed better results that proved proportional-fragment adaptability in one single machine. In future, the same experiments can be repeated for the architecture Type-2. In addition, several numbers of machines can be applied to substantiate fragment proportionality aspect of adaptability for the basic processes.

REFERENCES

[1] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "Adaptable Decentralized Service Oriented Architecture," *Journal of Systems and Software*, vol. 84, pp. 1591-1617, 2011.

[2] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "Run-time Adaptable Business Process Decentrlaization," in *IARIA/eKnow 2011*, Gosier, Guadeloupe, France, 2011.

[3] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "A Case Study of The Intelligent Process Decentralization Method," in *WCECS*, San Francisco, USA, 20-22 October, 2009.

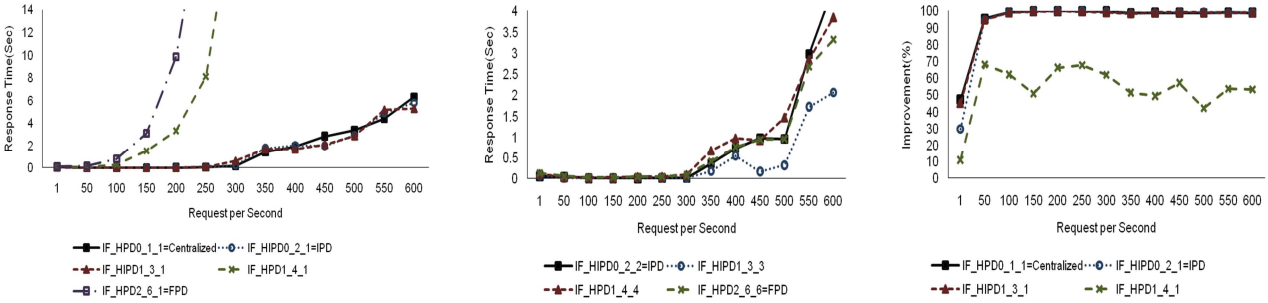
[4] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "Using Process Mining To Business Process Distribution," in *SAC2009*, Hawaii/Honolulu, 2009, pp. 1876-1881.

[5] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "SLA-Driven Business Process Distribution," in *IARIA/eKnow2009*, Mexico, 2009.

[6] F. Safi Esfahani, M. A. Azmi Murad, M. N. Sulaiman, and N. I. Udzir, "An Intelligent Business Process Distribution Approach," *Journal of Theoretical and Applied Information Technology*, vol. 4, pp. 1236-1245, 31st December 2008.

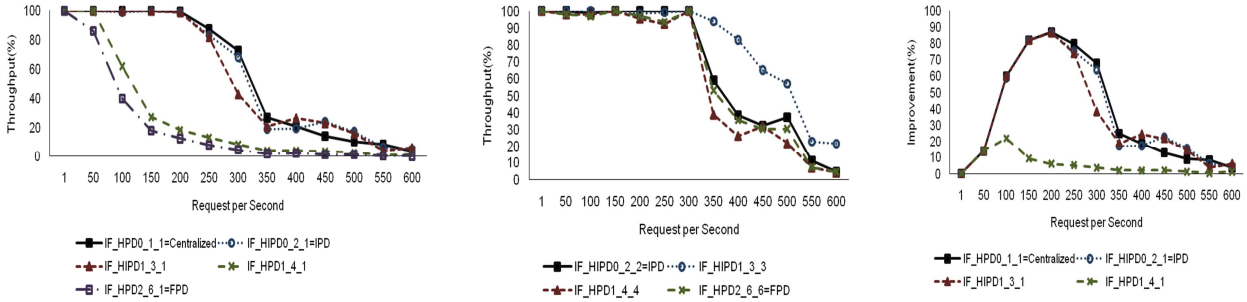
Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



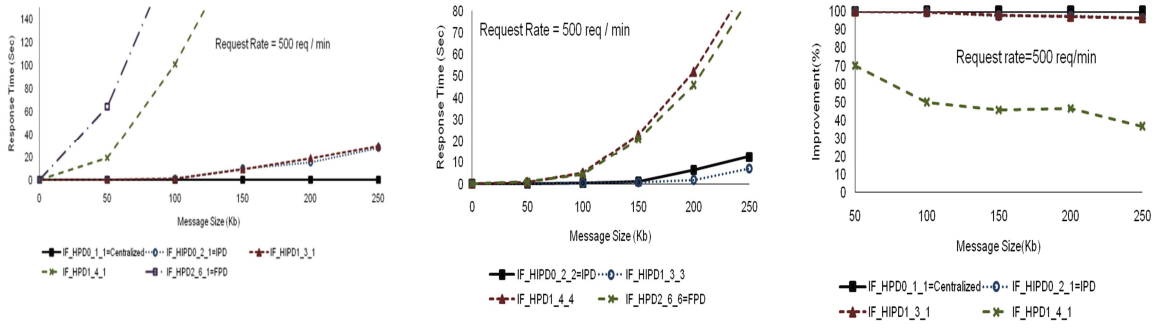
Minimum Distribution | Maximum Distribution | If-process Response-time Improvement Comparison

Figure 1: If-process Response-time Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions



Minimum Distribution | Maximum Distribution | If-process Response-time Improvement Comparison

Figure 2: If-process Response-time Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions

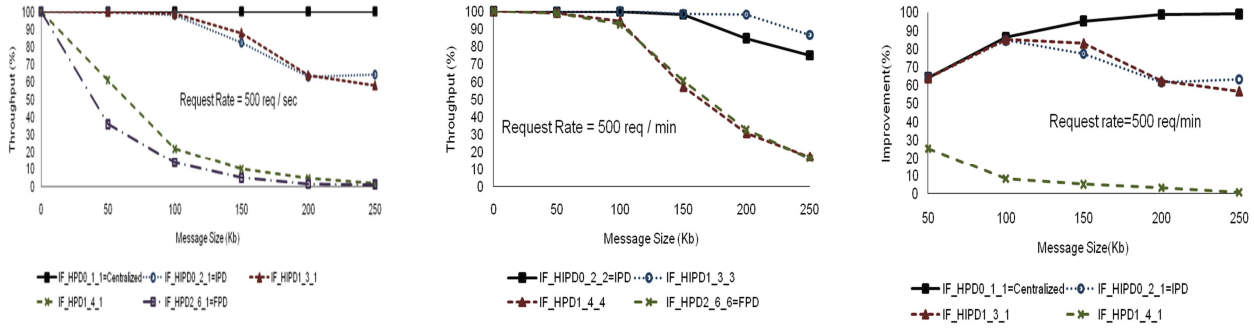


Minimum Distribution | Maximum Distribution | If-process Response-time Improvement with Request Rate 500

Figure 3: If-process Response-time Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions

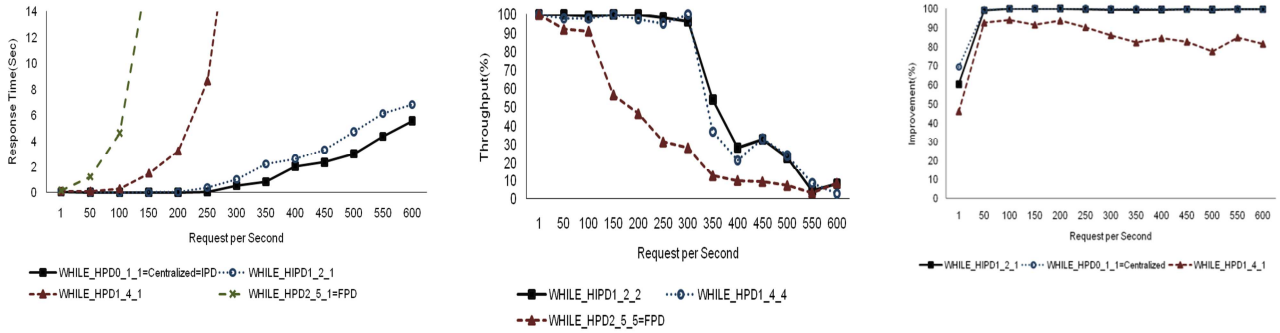
Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



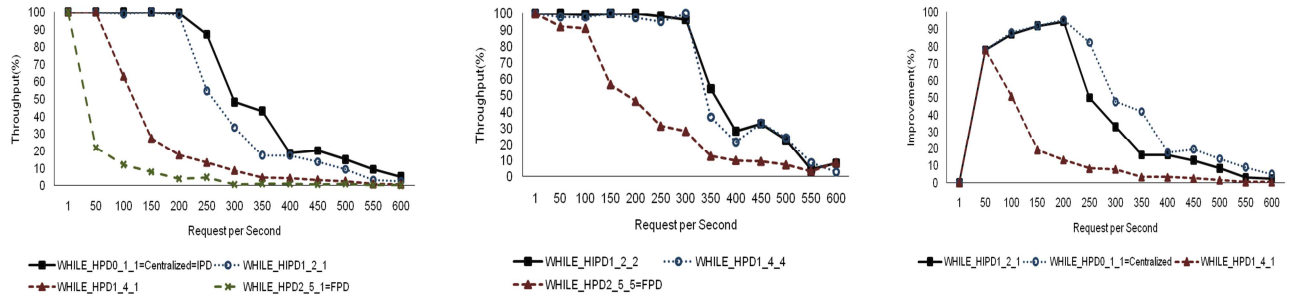
Minimum Distribution | Maximum Distribution | If-process Throughput Improvement with Request Rate 500

Figure 4: If-process Throughput Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions



Minimum Distribution | Maximum Distribution | While-process Response-time Improvement Comparison

Figure 5: While-process Response-time Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions

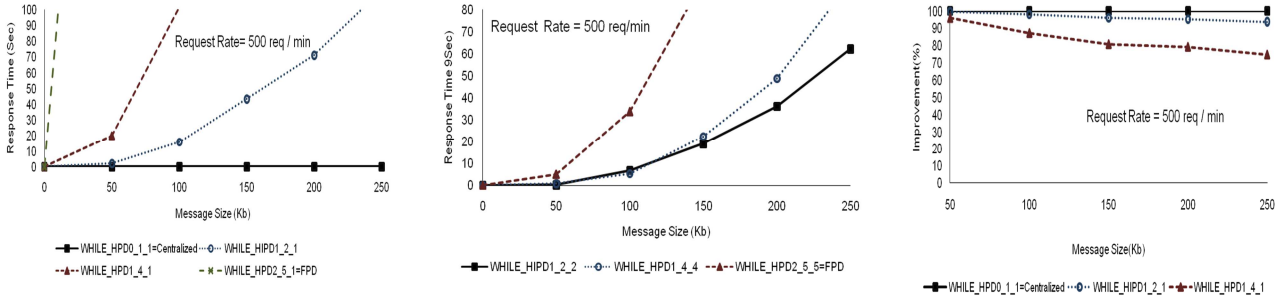


Minimum Distribution | Maximum Distribution | While-process Throughput Improvement Comparison

Figure 6: While-process Throughput Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions

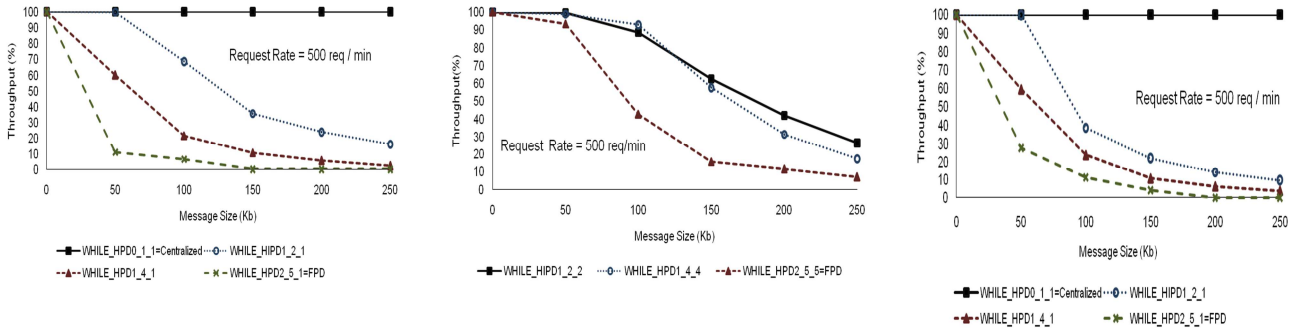
Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



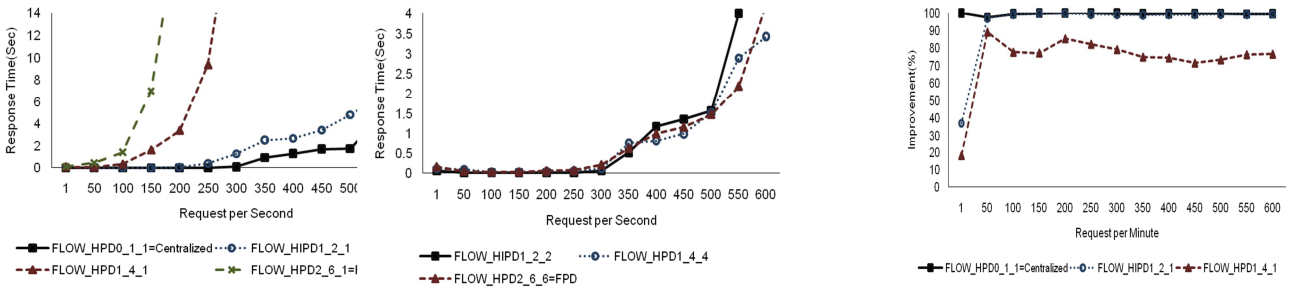
Minimum Distribution | Maximum Distribution | While-process Response-time Improvement with Request Rate 500

Figure 7: While-process Response-time Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions



Minimum Distribution | Maximum Distribution | While-process Throughput Improvement with Request Rate 500

Figure 8: While-process Throughput Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions

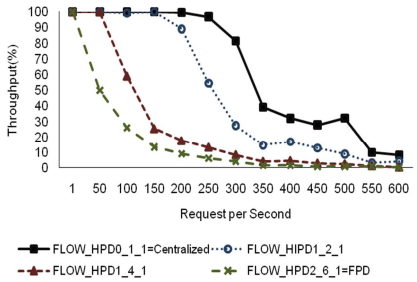


Minimum Distribution | Maximum Distribution | Flow-process Response-time Improvement Comparison

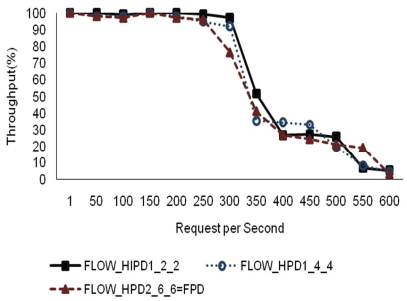
Figure 9: Flow-process Response-time Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions

Journal of Theoretical and Applied Information Technology
15 September 2012. Vol. 43 No.1

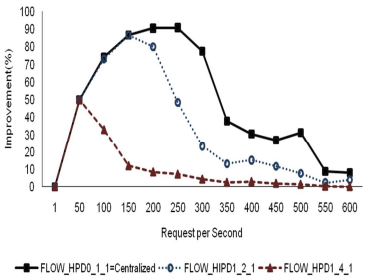
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195



Minimum Distribution

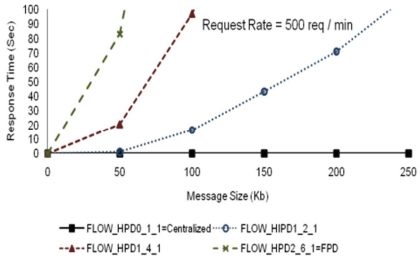


Maximum Distribution

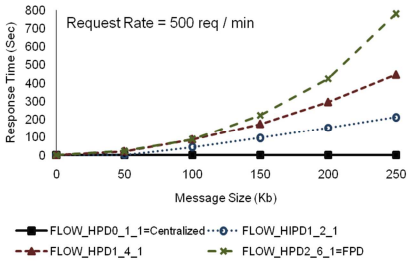


Flow-process Throughput Improvement Comparison

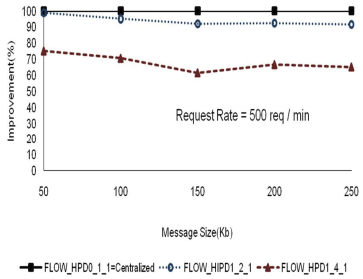
Figure 10: Flow-process Throughput Comparison, Variable-Request-Rate, Constant-Message-Size, Min-Max distributions



Minimum Distribution

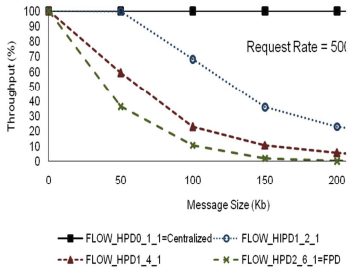


Maximum Distribution

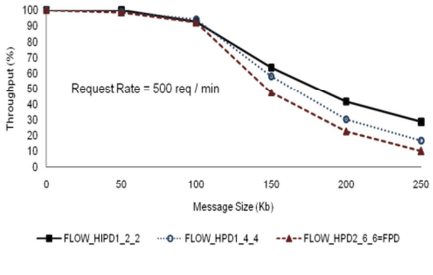


While-process Response-time Improvement with Request Rate 500

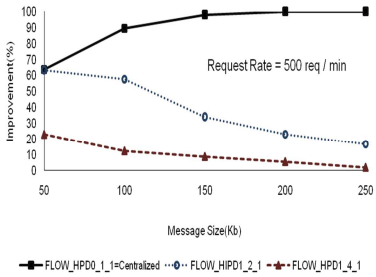
Figure 11: Flow-process Response-Time Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions



Minimum Distribution



Maximum Distribution



Flow-process Throughput Improvement with Request Rate 500

Figure 12: Flow-process Throughput Comparison, Constant-Request-Rate, Variable-Message-Size, Min-Max distributions