Towards Ontology-Based Embedded Services .................................................. 100
    G. Mahmoudi, C. Müller-Schloer

TOBAB: A Trend-Oriented Bandwidth Adaptive Buffering in
Peer-to-Peer Streaming System ................................................................. 113
    Sirui Yang, Hai Jin, Xuping Tu

Interference-Aware Selfish Routing in Multi-ratio Multi-channel
Wireless Mesh Networks .............................................................................. 123
    Yanxiang He, Jun Xiao, Naixue Xiong, Laurence T. Yang

Autonomic Group Location Update for Mobile Networks ......................... 133
    Lai Tu, Furong Wang, Fan Zhang, Jian Zhang

Autonomic and Trusted Computing Paradigms ............................................ 143
    Xiaolin Li, Hui Kang, Patrick Harrington, Johnson Thomas

Autonomic K-Interleaving Construction Scheme for P2P Overlay
Networks ................................................................................................... 153
    Khaled Ragab, Akinori Yonezawa

Self Awareness and Adaptive Traffic Signal Control System for Smart
World ......................................................................................................... 163
    Lawrence Y. Deng, Dong-liang Lee

Development and Runtime Support for Situation-Aware Security in
Autonomic Computing ................................................................................ 173
    Stephen S. Yau, Yisheng Yao, Min Yan

Track 2: Trust Models and Trustworthy Systems/Services

A Social Network-Based Trust Model for the Semantic Web ..................... 183
    Yu Zhang, Huajun Chen, Zhaohui Wu

Fuzzy Model Tuning for Intrusion Detection Systems .............................. 193
    Zhenwei Yu, Jeffrey J.P. Tsai

PATROL-F - A Comprehensive Reputation-Based Trust Model with
Fuzzy Subsystems ..................................................................................... 205
    Ayman Tajeddine, Ayman Kayssi, Ali Chehab, Hassan Artail

An Integration Framework for Trustworthy Transactions ......................... 217
    Yonghwan Lee, Dugki Min
An Integration Framework for Trustworthy Transactions

Yonghwan Lee and Dugki Min*

School of Computer Science and Engineering, Konkuk University,
Hwayang-dong, Kwangjin-gu, Seoul, 133-701, Korea
{yhlee, dkmin}@konkuk.ac.kr

Abstract. I/O-driven integration applications, such as EAI, B2Bi, home gateways, and heterogeneous devices integration, need a trustworthy integration framework to process a large volume of I/O-driven transactions with high performance and reliability. This paper proposes an integration framework for trustworthy transactions. This framework employs a design pattern, so-called, the Worker-Linker pattern, that is used for performance stability. According to our experimental results, the pattern helps the integration framework to control the heavy load after the saturation point. Moreover, this framework supports a number of mechanisms for trustworthy integration transactions. In this paper, we describe the mechanisms.

1 Introduction

The evolution of the Internet has brought a new way for enterprises to interact with their partners. Many infrastructures and enterprise information systems have been developed to extend business and value-added services on the Internet. Since markets are rapidly changing, business collaborations tend to be dynamic and require a large volume of integrated transactions. Moreover, heavy workload makes a system unstable in a short time. Therefore, integration systems should provide flexible and trustworthy software architectures and also be afford to process a large volume of transactions stably.

A B2B integration system requires process integration for B2B collaborations [1]. In general, a protocol of B2B collaboration includes a description of message exchange formats, transport binding protocols, business processes, and so on [2, 11]. In ubiquitous environment, I/O-driven integration applications, such as EAI, B2Bi, home gateways, and heterogeneous devices integration, need a trustworthy integration framework to process a large volume of I/O-driven transactions with high performance and reliability. Patterns for efficient I/O-driven distributed applications (i.e., Connector-Acceptor, Reactor, and Proactor patterns) have been proposed to process large volume of transactions [3]. However, how to use the patterns affects the stability of performance and reliability.

In this paper, we propose an integration framework for I/O-driven applications integration. The proposed integration framework copes with a number of software

*Corresponding author. This paper was supported by Konkuk University in 2006.

© Springer-Verlag Berlin Heidelberg 2006
architectural qualities in various aspects. For performance stability, the framework designs its I/O-related modules using a design pattern, called ‘Worker-Linker’, for processing a large volume of transactions. The Worker-Linker pattern is based on the Worker pattern [4] that is generally used for I/O efficiency. The Worker-Linker pattern adds a PPC (Peak Point Control) mechanism to the Worker pattern so that it can control the instability of performance caused by request congestion. For the flexibility and reusability of business integration logics, the Worker-Linker pattern implements business logics as UTL-based components and executes them by a command pattern. The UTL (Unified Transaction Language) is a XML-based language for describing data formats of transactions.

In addition to the Woker-Likner pattern for performance stability, the framework employs several other mechanisms for trustworthy integration transactions: processes-based fault tolerance and load balance, automatic process scheduling, separation of service processes from management processes, communication simulation of unavailable communication partners, and testcall checker for automatic self-detecting connection fails.

This paper is structured as follows. Related work is in the next section. Section 3 presents the overall software architecture of the proposed integration framework. Section 4 explains the design aspects of the framework. Section 5 draws a conclusion.

2 Related Works

Some B2B protocols are developed to support B2B e-commerce: ebXML [5], RosettaNet [6], and e-Service [8]. To support multiple B2B protocols, a B2B engine masters the communication between trading partners [7]. Those B2B applications mostly focus more on application level or business level and less on system level like I/O efficiency and performance stability [9].

An Acceptor-Connector pattern has been proposed for improving I/O efficiency [14]. The Acceptor-Connector pattern can be classified into a Reactor pattern and a Proactor pattern by methods of multiplicity for processing I/O events. The Reactor pattern processes events synchronously but the Proactor pattern asynchronously. Accordingly, the Proactor pattern can achieve higher performance than the Reactor pattern. However, the Proactor pattern should provide synchronization for shared resources. In asynchronous I/O, a pair of event and a reference to an event handler puts into a queue. When an event happens, the event handler processes it.

Even though the Acceptor-Connector pattern can achieve efficiency of I/O, if request congestion happens at certain points, the Acceptor-Connector pattern may have the tendency of performance instability. Performance stability is important to e-business integration and device integration or convergence in many ubiquitous environments [10]. To solve these problems in ubiquitous environments, this paper provides the Worker-Linker pattern-based integration framework for improving efficiency of I/O and stability of performance.
3 Software Architecture of Integration Framework

The proposed integration system is an integration communication middleware providing stable and trustworthy integration transactions between source systems and target systems. The integration system has the responsibility of processing trustworthy integration transactions between source systems and target systems. It is largely comprised of the Admin Console, the System Management, the Gateway, the UTL Processor, and UTL Server. The Admin Console supports the development and management toolkits for coders and system managers. The System Management is in charge of monitoring and managing system-related resources and the Gateway has the responsibility of core functionality, such as network-related details, protocol conversion, message routing. The UTL Processor and the UTL Server is in charge of creating, caching, deploying, and executing UTL components for processing integration business logics. Figure 1 shows the software architecture of the integration system.

![Software Architecture of the Integration System](image)

The Gateway is comprised of the Adapter, the Message Controller, the Dispatcher, and the Routing Manager. The Adapter is in charge of processing communication-related details of sources systems or target systems, such as connection, listening, and conversion of protocols. The Message Controller has the responsibility of controlling flow of messages. If the Message Controller receives messages from the source adapters or target adapters, it decides the next destination of those messages based on configurations for each task in the Admin Console. If business logics (i.e., data format transformation, data validation, etc) are configured for a specific task, the Message Controller invokes the Dispatcher to perform the business logics before or after communication with the target systems. To process business logic, the Dispatcher gets a UTL component from the UTL Processor and executes it. The UTL Processor is in charge of creating and caching UTL components. If a UTL component does not exist
in the UTL Processor, the UTL Processor gets the related information for creating UTL components from the UTL Server and then creates UTL components.

An UTL is an xml-based language for defining data formats of header, request, and response block in a message. The header, request and response element defines data formats of header, request, and response block respectively. An UTL file is saved to the UTL Repository by the UTL Server. The UTL component is generated from the framework after a developer develops an UTL class according to the hook methods defined by the framework for processing business logics.

The Admin Console has a development and a management tool. In view of a development tool, the Admin Console is in charge of managing source codes and various parameters according to the concerned task unit. In other word, the Admin Console categories tasks in a tree form, provides the related source code (i.e., UTL Code) to the task, edits them, compiles them, packages them, distributes them, and tests them via a tool. Also the Admin Console manages setting information which is necessary for each task. Systematic integration processed by the framework and tools together takes the overall system development process rapidly. In view of management tool, the Admin Console manages all resources related to the framework and monitors them.

The System Management is comprised of the System Manager and the Resource Collector. The System Manager is in charge of scheduling and managing the framework-related resources. For improving the qualities like performance and scalability, main modules (i.e., The Adapter, The Gateway, and The Dispatcher) may be deployed to different processes or hardware boxes in distribute system. The each main module has a thread called the State Manager. The thread gets resources-related information (i.e., process state, CPU, memory, I/O, etc) and sends it to the Resource Collector. The Resource Collector collects all resource information from the main modules and sends to the System Manager. The System Manager uses it for balancing load of each main module.

4 Mechanisms for Trustworthy Integration Transactions

Our integration framework takes into account many mechanisms for trustworthy integration transactions. In this section, we introduce a number of mechanisms to be used for trustworthy integration transactions.

4.1 Worker-Linker Pattern for Performance Stability

To process a large volume of transactions in performance stability, the integration framework designs many I/O-related modules (i.e., the Adapter, the Message Controller, the Dispatcher, etc) by the Worker-Linker pattern proposed by this paper. Figure 2 shows a sequence diagram of the Worker-Linker pattern. The Linker is similar to the Connector of the Acceptor-Connector pattern. After the Acceptor receives messages from the Linker, it puts them a queue and wake up a waiting Worker thread. The Worker thread gets them from the queue and gets the delay time from the WorkerManager. After the Worker thread sleeps for the delay time calculated by the
WorkerManager, the Worker thread executes business logics by using a command pattern. Finally, the Worker increments the number of transactions processed by the Worker threads.

![Sequence Diagram of the Worker-Linker Pattern](image)

Fig. 2. Sequence Diagram of the Worker-Linker Pattern

The Worker-Linker pattern is the pattern reconstructed with existing patterns, such as the Worker, Command, and Acceptor-Connector Pattern. The Worker pattern is similar to the Consumer-Producer pattern in that the Acceptor puts a message into a buffer and the Worker gets the message from the buffer. A Worker pattern is generally used for I/O efficiency. The Worker-Linker pattern adds a PPC (Peak Point Control) mechanism to the Worker pattern [14] so that it can control the instability of performance caused by request congestion. The Worker-Linker Pattern also uses UTL-based components for improving flexibility and reusability of business integration logics. To execute business integration logics, the Worker-Linker pattern uses a command pattern. In other words, UTL components are implemented as command objects. Accordingly, the integration framework can easily add new business logics by implementing just command objects.

### 4.2 Separation of Service Process Set from Management Process Set

To achieve reliability and high performance, the integration system separates service process set from management process set. Figure 3 shows the separation of service process set from management process set.

The service process set means the processes required for processing online transactions. The management process set means the processes for managing the integration system. In Figure 3, the integrated system may create separate management process for each management-related module, such as the System Manager, Resource Reporter, Log Dispatcher, and Log Server. Separation of service process set from management process set prevents the integration system from stopping service
processes due to the failure of a management process. In other words, a service process can support online transaction service regardless of the failure of a management process.

4.3 Multiple Processes-Based Fault Tolerance and Load Balance

To support large volume of transactions and session failover, the integration framework supports the following mechanisms: multi-thread, connection pooling, multiple processes based-session failover and load balance. In Figure 4, the Source Adapter...
can distribute a request to the Message Controller A, B or C by specific scheduling algorithms (i.e., Round Robin, FIFO, etc). Moreover, if the state of the Message Controller process A is fail, the Source Adapter can invoke another process B or C. All processes in the Gateway process have a thread called the State Manager. The thread gets the process-related resource information and sends them to the Resource Collector. The Resource Collector collects all resource information from all related processes and sends them to the System Manager. The Source Adapter uses them for fault tolerance and load balance.

4.4 Automatic Process Management by Scheduling Mechanism

The integration framework supports a web-based Admin Console for automatic process management by a scheduling mechanism. In view of management, the Admin Console classifies nodes and categories process in each node in a tree form, provide the statistics of all resources related with each process, stop and start the processes. In view of integration business logics, The Admin Console categories tasks in a tree form, provides the related source codes (i.e., Java code in URLT script) to the task, edits them, compiles them, packages them, distributes them, and tests them via a tool. Also the Admin Console manages setting information which is necessary for each task. Figure 5 is the steps for starting and stopping processes by a scheduling mechanism. A system manager inputs schedule information to the Admin Console and starts scheduling. Concurrently, the System Manager collects the status information of all processes. The status information of all processes is used for process management and balancing load of processes. When it is the time configured in the schedule, the System Scheduler sends start or stop request to the System Manager. The System

![Diagram showing automatic processes management by scheduling mechanism](image)
Manager gets information of processes (i.e., IP, Port, etc) from the Process Configuration and sends start or stop command to each process.

4.5 Communication Simulation for Unavailable Communication Partners

As integration systems have the characteristics of performing integration testing with other partners. The status of other partners should be considered. If some partners are not available for integration testing, the delay caused by unavailable partners may increases project risk. For this purpose, the proposed integration framework provides the communication simulation for supporting integration testing with unavailable partners. While on-line transactions are processed, the Target Adapter saves the processed data to the simulation DB for late usage. Figure 6 is the steps for performing simulation jobs. When the Target Adapter detects connection failure, it gets integration test-related data from the simulation DB and responses instead of the Target Application.

Fig. 6. Simulation DB-based Integration Testing during Abnormal Transaction

4.6 Testcall Checker for Automatic Detecting and Notifying of Connection Fail

The integration framework supports the testcall checker for automatic self-detecting connection failure and notifying the status of system fail to a system manager. Figure 7 is the steps for detecting and notifying the status of connection failure using the TestCall Checker.

When connection time-out happens in the Target Adapter, the Target Adapter calls the TestCall Checker that checks the status of the Target Application. If the TestCall Checker does not receive any response messages from the Target Application during 3 times retry, it notifies the System Manager that connection time-out happens in the Target Application. The System Manager responses fail messages to the Source Adapter and the Gateway instead of the Target Adapter. Finally, the Source Adapter sends SMS messages to a system manager.
5 Conclusions

I/O-driven integration applications, such as EAI, B2Bi, home gateways, and heterogeneous devices integration, need a trustworthy integration framework to process a large volume of I/O-driven transactions with high performance and reliability. This paper describes architecture of the integration framework and the Worker-Linker pattern for a large volume of transactions. The Worker-Linker pattern adds PPC (Peak Point Control) mechanism to an existing Worker pattern with I/O efficiency so that it can control the instability of performance caused by request congestion. In addition to the Worker-Linker pattern for performance stability, the integration framework employs several other mechanisms to achieve trustworthy integration transactions: processes-based fault tolerance and load balance, automatic process scheduling, separation of service processes from management processes, communication simulation of unavailable communication partners, and testcall checker for automatic self-detecting connection fails.

References


Welcome to the proceedings of the Third International Conference on Autonomic and Trusted Computing (ATC 2006) which was held in Wuhan and Three Gorges, China, September 3-6, 2006.

Computing systems including hardware, software, communication and networks are growing with ever increasing scale and heterogeneity, and becoming overly complex. The complexity is getting more critical along with ubiquitous permeation of embedded devices and other pervasive systems. To cope with the growing and ubiquitous complexity, autonomic computing focuses on self-manageable computing and communication systems that perform self-awareness, self-configuration, self-optimization, self-healing, self-protection and other software operations to the maximum extent possible without human intervention or guidance.

Any autonomic system must be trustworthy to avoid the risk of losing control and to retain confidence that the system will not fail. Trust and/or distrust relationships in the Internet and pervasive infrastructure-based global computing exist universally in the course of dynamic interaction and cooperation of various users, systems and services. Trusted computing targets computing and communication systems as well as services that are available, predictable, traceable, controllable, assessable, sustainable, dependable, persistable, security/privacy protectable, etc. A series of grand challenges exist to achieve practical self-manageable autonomic systems with truly trustworthy services.

The ATC 2006 conference provided a forum for engineers and scientists in academia, industry, and government to address the most innovative research and development including technical challenges and social, legal, political, and economic issues, and to present and discuss their ideas, results, work in progress and experience on all aspects of autonomic and trusted computing and communications. ATC 2006 as a conference came from the First International Workshop on Trusted and Autonomic Ubiquitous and Embedded Systems (TAUES 2005) held in Japan, December, 2005, and the International Workshop on Trusted and Autonomic Computing Systems (TACS 2006) held in Austria, April, 2006.

There was a very large number of paper submissions (208), representing 18 countries and regions, not only from Asia and the Pacific, but also from Europe, and North and South America. All submissions were reviewed by at least three Program or Technical Committee members or external reviewers. It was extremely difficult to select the presentations for the conference because there were so many excellent and interesting submissions. In order to allocate as many papers as possible and keep the high quality of the conference, we finally decided to accept 57 papers for presentations, reflecting a 27% acceptance rate. We believe that all of these papers and topics not only provided novel ideas, new results, work in progress and state-of-the-art techniques in this field, but also