A Review on Integration of Protocols and Programming to Distribute Computing

Anu Chaudhary¹ and Puneet Garg²

¹M. Tech. Student, Department of Computer Science Engineering
Ganga Institute of Technology and Management, Kablana, Jhajjar (Haryana)
anu.khatra@gmail.com

²Asstt. Professor, Department of Computer Science Engineering
Ganga Institute of Technology and Management, Kablana, Jhajjar (Haryana)
puneetgarg.er@gmail.com

Abstract

Distributed computing refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into several tasks, each of which is solved by one or more computers, which communicate with each other by passing message. Distributed computing and its applications are day by day becoming very complex. Traditional distributed computing systems are typically complex to implement and costly to maintain. Furthermore, only some comparative work has been done to understand the performance and usability of these systems for their own sake as well of that of new approaches that eventuate. We are adding some of the application that will enhance the present model of distributed computing system through Kompics model. Kompics is a programming framework used for configuring, and executing distributed protocols which actually interact asynchronously by passing data-carrying events. This model raises the level of abstraction to some extent in programming to distributed systems. We study constructs, mechanisms, architectural patterns, as well as programming, concurrency, and execution models that enable programmers to construct and compose reusable and modular distributed abstractions. We believe this is an important contribution because it lowers the cost and accelerates the development and evaluation of more reliable distributed systems. Finally, we add protocol for Problem Solving in Distributed Systems, protocols and framework For Wide Area Network Distributed Computing Systems, security system for distributed computing to enhance the feature of DCS. In this thesis, included protocols are problem solving in distributed and other is framework for wide area network system. And the new enhanced security protocol for distributed computed is to be proposed.

Keywords: Protocol composition, component model, reconfigurable distributed systems architecture, publishes-subscribe event-based systems.

Introduction to Distributed Computing

The distributed computing systems (DCS) are the systems of today and tomorrow. Construction of DCS poses challenges that are related not only to meeting functional requirements but also non-functional requirements, e.g., Quality of Service (QoS). An approach for DCS development must employ formalism in all stages of development from inception to deployment. The Uni Frame Approach (UA) uses iterative development and formalism to design high confidence DCS, with a goal to automate the process of integrating heterogeneous software components to create DCS that conform to certain quality requirements and offer a significant measure of predictability. Distributed computing is a field of computer science that studies distributed systems. A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs. There are many alternatives for the message passing mechanism, including RPC-like connectors and message queues. An important goal and challenge of distributed systems is location transparency.
Kompics Model

The Kompics component model and programming framework was designed to simplify the development of increasingly complex distributed systems. Systems built with Kompics leverage multi-core machines out of the box and they can be dynamically reconfigured to support hot software upgrades. A simulation framework enables deterministic debugging and reproducible performance evaluation of unmodified Kompics distributed systems. Kompics offers a systematic methodology for designing, programming, composing, deploying, testing, debugging, and evaluating distributed systems. These characteristics of the framework, together with a rich library of provided protocols and abstractions, ultimately led to its usage for prototyping, evaluating, and developing a plethora of distributed systems, both within and outside of our research group.

The design of Kompics as well as the principal developer of the reference Kompics implementation in Java. The open source code repository for the Kompics platform, as well as further documentation, videos, and tutorials, were made publicly available at http://kompics.sics.se/.

How Kompics work?

Kompics is a component model targeted at building distributed systems by composing protocols programmed as event-driven components. Kompics components are reactive state machines that execute concurrently and communicate by passing data carrying typed events through typed bidirectional ports connected by channels. This section introduces the conceptual entities of our component model and its Programming constructs, its execution model, as well as constructs enabling dynamic reconfiguration, component life-cycle and fault management. The fundamental conceptual entities in Kompics are events, ports, components, event handlers, subscriptions, and channels. We introduce them here and show examples of their definitions with snippets of Java code. The Kompics component model is programming language independent; however, we use Java to illustrate a formal definition of its concepts.

Back Ground

Earlier models on which distributed computing working is also very robust and almost every feature already been added. But with the advent of new technology Security feature must be improved. Historically, distributed computing has been focused on the problem of distributing computation between several systems that are jointly working on a problem.

In the 1960s, the use of concurrent processes that communicate by message-passing has its roots in operating system architectures studied. In the 1970s, the first widespread distributed systems were local-area networks such as Ethernet, which was invented. ARPA, the predecessor of the Internet, was introduced in the late 1960s, and ARPA e-mail was invented in the early 1970s. E-mail became the most successful application of ARPA, and it is probably the earliest example of a large-scale distributed application. In addition to ARPA, and its successor, the Internet, other early worldwide computer networks included Usenet and FidoNet from 1980s, both of which were used to support distributed discussion systems.

The study of distributed computing became its own branch of computer science in the late 1970s and early 1980s. The first conference in the field, Symposium on Principles of Distributed Computing (PODC), dates back to 1982, and its European counterpart International Symposium on Distributed Computing (DISC) was first held in 1985.

The most often used distributed computing abstraction is the RPC – Remote Procedure Call. RPC allows a remote function to be invoked as if it were a local one. The history of RPC-style distributed computing is fairly complicated. More or less it started with Sun Microsystems’ Open Network Computing (ONC) RPC system in 1987, as the basic communication mechanism for its Network File System (NFS). NFS is now supported on UNIX, Linux, and many other distributed operating systems. NFS is used to access directories and files located on remote computer as if those directories and files were located on the local computer.

The first major effort toward language-independent and platform-neutral distributed computing was taken by the Object Management Group (OMG) in 1989. OMG is a consortium that includes over 500 members. In 1991, OMG delivered the first version of Common Object Request Broker Architecture (CORBA), a distributed objects platform. CORBA allowed programs located in different parts of the
network and written in different programming languages to communicate with each other. The term Object Request Broker (ORB) gained popularity to denote the infrastructure software that enabled distributed objects. In 1996, CORBA version 2 introduced the Internet Inter-ORB Protocol (IIOP) as major enhancements in the core distributed computing model and higher-level services that distributed objects could use. IIOP established CORBA’s dominance in distributed computing for the next 5 years until the advent of Web services. Microsoft started its own distributed computing initiative around 1990. In 1996, Microsoft delivered the Distributed Component Object Model (DCOM), which was closely tied to previous Microsoft component efforts such as Object Linking and Embedding (OLE), non-distributed COM (or OLE2), and ActiveX (lightweight components for web applications). To compete with CORBA, the next year (1997) Microsoft introduced COM+ to bring DCOM much closer to the CORBA model for distributed computing. The Kompics component framework developed in 2008.

Core Concept

Distributed Computing provides illusion of one system while running on multiple systems and transparency of resources. We include Problem Solving in Distributed Systems in which we include Data Decomposition, Task Decomposition (Recursive, Exploratory, and Speculative), Hybrid Decomposition and PROTOCOLS AND FRAMEWORK For Wide Area Network Distributed Computing Systems.

We are going to develop on the behalf of the mentioned two a security system for distributed computing. As the security system is important to focus on enhancement of security.

Review of Literature


Advantages

1) On Spreading Of Important Files on Distributed System, There Is A Main Issue of Security Of Files From Intruders. Our Security Model Will Add Feature To Kompics Model.
2) The Secure File Will Only Be Opened By The Desired User.

Conclusion

At last we can say that with this proposed solution in distributed processing system based on a Kompics model provide security to data in distributed system. In this the security is enhanced to make the data secure in distributed environment. This will help to only display the data required to others which is confidential. As security is mandatory is all aspect.

References