A Cyber-Insurance Scheme for Water Distribution Systems

PROJECT REPORT

Submitted by

K.JACKULINRANI (510219104002)
P.SOWMIYA (510219104003)
S.SWATHI (510219104004)
K.SWETHA (510219104306)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

ANNAMALAIAR COLLEGE OF ENGINEERING

ANNA UNIVERSITY: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

DEPARTMENT OF COMPUTER ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report **"A Cyber Insurance Scheme for Water Distribution Systems"** is the bonafide work of "**K.JACKULINRANI (510219104002), SOWMIYA.P (510219104003), SWATHI.S (51021910400), SWETHA.K (510219104306)"** who carried out the project work under my supervision.

SIGNATURE

Mr.M.KARTHIKEYAN, B.Tech.M.E.

HEAD OF THE DEPARTMENT

Department of Computer Science
And Engineering,
Annamalaiar College of Engg,
Modaiyur-606 902.

SIGNATURE

Mr.M.KARTHIKEYAN.B.Tech.M.E .,

SUPERVISOR

Department of Computer Science
And Engineering,
Annamalaiar College of Engg,
Modaiyur-606 902.

Submitted for the project viva-voice held on……...

EXTERNAL EXAMINAR **INTERNAL EXAMINAR**

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any work would be incomplete without mentioning those people who made it possible, whose constant guidance and encouragement rounded our efforts with success.

It is a great pleasure for us to acknowledge the assistance and contributions of some people for this effort. First, we would like to thank god for giving us the confidence and power to complete this work successfully. We would like to thank **CHAIRMAN Mr.S.R.SHANMUGAM, Annamalaiar College of Engineering,** for giving us the opportunity to complete our study and the project well in advance.

It is our great pleasure to acknowledge the assistance and contributions of people especially our **PRINCIPAL Dr.R.KANNAN, M.E., Ph.D.,** Annamalaiar College of Engineering, and our **SECRETARY Mr.C.JEEVAGAN, B.Com,** who got the whole thing started and who gave us the chance to express ourselves.

we would like to express my deep gratitude and appreciation to our Project Co-ordinator **Mr.M.KARTHIKEYAN,B.Tech., M.E.,** Head of the Department, **Computer Science and Engineering**, for giving details, guidelines, stimulating suggestions and encouragement which helped me to coordinate my project. We take this opportunity to express my profound gratitude and deep regard to my Supervisor, **Mr.M.KARTHIKEYAN,B.Tech., M.E.,** Department of **Computer Science and Engineering**, for her exemplary guidance, monitoring and constant encouragement throughout the project.

A Cyber-Insurance Scheme for Water Distribution Systems

Abstract

As one of the national critical infrastructures, the water distribution system supports our daily life and economic growth, the failure of which may lead to catastrophic results. Besides the uncertainty from the system component failures, cyber-attacks are vital to the secure system operation and have great impacts on the reliability of the water supply service. Malicious attackers may intrude into the supervisory control and data acquisition (SCADA) system of pump stations in the water distribution networks and interrupt the water supply to the customers. Cyber insurance is emerging as a promising financial tool in system risk management. In this paper, cyber insurance is proposed for the cyber risk management of the water distribution system. A semi-Markov process (SMP) model is devised to model the cyber-attacks against pump stations in the water distribution system. Both the impacts of the independent cyber risks in the individual distribution network and the correlated cyber risks shared across different water distribution networks are evaluated and modeled. A sequential Monte Carlo Simulation (MCS) based algorithm is developed to evaluate the system loss. Cyber insurance premiums for the water distribution networks are designed based on the actuarial principles and potential system losses. Case studies are also performed on multiple representative water distribution networks, and the results demonstrate the validity of the proposed cyber insurance model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

S.NO	ABBREVIATIONS EXPANSION
1	Java VM Java virtual Machine
2	Java API Java Application Programming Interface
3	GUI Graphical User Interface
4	LAN Local Area Network
5	TCP Transmission Control Protocol
6	UDP User Datagram Protocol
7	IP Internet Protocol
8	RMI Remote Method Invocation
9	SDK Software Development Kit
10	ODBC Open Database Connectivity
11	RDBMS Relational Database Management System
12	CRUD Create, Read, Update, Delete
13	DFD Document Flow Diagram

CHAPTER 1

INTRODUCATION

Widespread applications of the information and communication technology (ICT) introduce higher risks on cyber security in modern cyber-physical systems. The real-time monitoring and communication systems are commonly used in the regular operation of water systems. The systems control and data acquisition (SCADA) systems in the water systems are extensively applied to control the automated physical processes which are essential to the drinking water treatment and water distribution systems.

It has become a standard for the operation of medium to large scale drinking water systems and even some small water utilities. With the improvement of operational efficiency, the vulnerabilities of the water system to malicious cyber-attacks are increasing at the same time. A successful cyber-attack on the water network can lead to very serious damage on both the supply side and demand side. Consequently, the SCADA system in the water distribution network has become the primary target of several cyber attacks in the past two decades.

According to the Water Sector Cyber security Brief for States given by the U.S. Environmental Protection Agency (EPA), successful cyber attacks on the controls systems like the SCADA system in the water network have significant impacts on the system performance, such as upsetting the treatment and conveyance processes by opening and closing valves, overriding alarms and disabling pumps or other equipment, deface the utility's website, compromise the email system, install malicious programs like ransom ware to disable the process control operations, etc.

1.1 SYSTEM STUDY

A detailed study to determine whether, to what extent, and how automatic data-processing equipment should be used; it usually includes an analysis of the existing system.

1.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

1.2.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

1.2.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

1.2.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

CHAPTER 2

LITERATURE REVIEW

1.Cyber Security of Water SCADA Systems

This brief aims to perform security threat assessment of networked control systems with regulatory and supervisory control layers. We analyze the performance of a proportional-integral controller (regulatory layer) and a model-based diagnostic scheme (supervisory layer) under a class of deception attacks. We adopt a conservative approach by assuming that the attacker has knowledge of: 1) the system dynamics; 2) the parameters of the diagnostic scheme; and 3) the sensor-control signals. The deception attack presented here can enable remote water pilfering from automated canal systems. We also report a field-operational test attack on the Gigantic canal system located in Southern France.

2. A review of cyber security incidents in the water sector. "Journal of Environmental Engineering"

This study presents a critical review of disclosed, documented, and malicious cyber security incidents in the water sector to inform safeguarding efforts against cyber security threats. The review is presented within a technical context of industrial control system architectures, attack-defense models, and security solutions. Fifteen incidents were selected and analyzed through a search strategy that included a variety of public information sources ranging from federal investigation reports to scientific papers. For each individual incident, the situation, response, remediation, and lessons learned were compiled and described. The findings of this review indicate an increase in the frequency, diversity, and complexity of cyber threats to the water sector.

Although the emergence of new threats, such as ransom ware or crypto jacking, was found, a recurrence of similar vulnerabilities and threats, such as insider threats, was also evident, emphasizing the need for an adaptive, cooperative, and comprehensive approach to water cyber defense.

3. Improving critical infrastructure cyber security

The United States depends on the reliable functioning of its critical infrastructure. Cyber security threats exploit the increased complexity and connectivity of critical infrastructure systems, placing the Nation's security, economy, and public safety and health at risk. Similar to financial and reputational risks, cyber security risk affects a company's bottom line. It can drive up costs and affect revenue. It can harm an organization's ability to innovate and to gain and maintain customers. Cyber security can be an important and amplifying component of an organization's overall risk management. To strengthen the resilience of this infrastructure, the Cyber security Enhancement Act of 20142 (CEA) updated the role of the National Institute of Standards and Technology (NIST) to "facilitate and support the development of" cyber security risk frameworks. Through CEA, NIST must identify "a prioritized, flexible, repeatable, performance-based, and cost-effective approach, including information security measures and controls that may be voluntarily adopted by owners and operators of critical infrastructure to help them identify, assess, and manage cyber risks." This formalized NIST's previous work developing Framework Version 1.0 under Executive Order 13636, "Improving Critical Infrastructure Cyber security," issued in February 20133 , and provided guidance for future Framework evolution.

CHAPTER 3

SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Some existing studies on cyber security characterization have mostly considered the cyber security from a qualitative perspective. So far, very few studies have considered the quantitative assessment of cyber security.

Therefore, it is of great importance to evaluate the impacts of the cyber-attacks on the water distribution network in a quantitative manner. From the systems analysis perspective, there are some similarities between the failures caused by accidental faults such as system components failures or human errors and the failures due to cyber-attacks.

Therefore, we can fully utilize the proposed model and available research in the literature about reliability analysis of critical infrastructures to develop cyber security quantification analysis.

3.1.1 DISADVANTAGE

- An integrated risk evaluation of the water system incorporating the impact of cyber-attacks is very meaningful in enabling more informed decision making.
- One efficient way is to adopt the recently developed cyber insurance policies, which are the policies that provide coverage against the overall system losses from network related issues in cyber security.
- An integrated risk evaluation of the water system incorporating the impact of cyber-attacks is very meaningful in enabling more informed decision

making. One efficient way is to adopt the recently developed cyber insurance policies, which are the policies that provide coverage against the overall system losses from network related issues in cyber security.

- The authors of analyzed the risk management strategies of companies when the risks are interdependent. The study shows how the interdependence of cyber risks reflects on the incentives to invest in security technologies and to buy insurance coverage.

3.2 PROPOSED SYSTEM:

- A modified semi-Markov process (SMP) model incorporating a stochastic cyber risk correlation model is proposed to evaluate the potential cyber security threats against the SCADA system in the water distribution network.
- Based on the proposed model, both the independent cyber risk within the individual water network and the correlated common cyber risks shared across different water networks can be considered and evaluated.
- A Monte Carlo simulation (MCS) based quantitative risk assessment approach is developed to estimate the impact of malicious cyber-attacks on water distribution system reliability.
- A cyber-insurance framework including several actuarial insurance principles is built to manage the risks of water distribution systems considering the financial consequences of cyber security risks.

3.2.1 ADVANTAGE:

- Cyber-attacks can bring potential threats to the water systems. Some existing studies on cyber security characterization have mostly considered the cyber security from a qualitative perspective.
- Reliable water supply is of critical importance to support the economic development and human life. Based on the predictions from report water supply will become one of the most serious national problems by 2025.
- With the increasing risks of substantial economic losses due to cyber-attacks, additional tools to manage the cyber security risks are strongly urged.
- Abnormal operations of the physical components due to cyber-attacks may increase the stress of the system operator as well, so that human errors are more likely to happen in such cases.

CHAPTER 4

SYSTEM SPECIFICATION

4.1 Hardware Requirements:

System	:	Pentium IV 2.4 GHz
Hard Disk	:	40 GB
Floppy Drive	:	1.44 MB
Monitor	:	15 VGA Colour
Mouse	:	Logitech
RAM	:	256 MB

4.2 Software Requirements:

Operating system	:	Windows 7.
Front End	:	JAVA or JSP
Database	:	SQL SERVER 2008
Tools	:	Eclipse IDE

CHAPTER 5

PROJECT DESCRIPTION

MODULES DESCRIPTION

5 .1 Cyber security

Cyber insurance is a risk management technique via which network users' risks can be transferred to an insurance company in return for the insurance premium. With the increasing cyber security risk of critical infrastructures, and in combination with the need for compliance with recently enforced corporate regulation, the demand for cyber insurance has significantly increased.

While the critical infrastructures are becoming more and more dependent on their networked operation systems, more potential vulnerabilities are exposed to the cyber-attacks than before.

5.2 Cyber insurance

Cyber insurance is emerging as a promising financial tool in system risk management. In this paper, cyber insurance is proposed for the cyber risk management of the water distribution system. A semi-Markov process (SMP) model is devised to model the cyber-attacks against pump stations in the water distribution system.

Both the impacts of the independent cyber risks in the individual distribution network and the correlated cyber risks shared across different water distribution networks are evaluated and modeled.

5.3 Water distribution system

Cyber-attacks and physical failures may happen simultaneously to trigger more significant impacts on the overall system. This kind of combinational failures could directly result in the interruption of water services.

Therefore, in this paper the impact due to cyber security threats against the water distribution system is considered in performing the reliability assessment.

Failures in the cyber infrastructure (e.g., the wrong control signals or false system information) could mislead the system operator to make uninformed decisions which may result in system failures.

5.4 Reliability evaluation

Various probabilistic reliability evaluation algorithms have been developed and applied to critical infrastructures. When performing system-level reliability evaluation, a wide spectrum of factors may lead to a system failure.

The physical component failures could be triggered and consequently lead to an overall system failure. However, for modern critical infrastructures, more factors or uncertainties such as cyber-attacks should be taken into consideration when designing effective protection mechanisms to ensure reliable performance of the systems.

Algorithm:

When the same operation software stack is installed on another distribution network, the software vulnerability is shared among them. These shared vulnerabilities can lead to correlated cyber risks of multiple water networks and result in greater loss.

What's more, the information technology infrastructure of various water utilities is dominated by a few similar technologies that may also leave these utilities with the same potential vulnerabilities, which means although the water utilities are not physically connected to each other, they may share correlated common cyber risks.

This kind of cyber risk correlations across various water utilities cannot be modeled through the traditional algorithm of risk analysis. Consequently, a stochastic model is highly needed to identify and formulate the correlation among different water distribution networks due to the similar potential cyber risks they shared.

In most traditional reliability evaluation algorithms, only the physical N-1 or N-k contingencies of system components is considered. The physical component failures could be triggered and consequently lead to an overall system failure. However, for modern critical infrastructures, more factors or uncertainties such as cyber attacks should be taken into consideration when designing effective protection mechanisms to ensure reliable performance of the systems.

CHAPTER 6

SYSTEMS DESIGN

6.1 INTRODUCTION

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

SYSTEM DESIGN

Systems design is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

Overview

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

INTRODUCTION

Purpose and Scope

This section provides a brief description of the Systems Design Document’s purpose and scope.

Project Executive Summary

This section provides a description of the project from a management perspective and an overview of the framework within which the conceptual system design was prepared. If appropriate, include the information discussed in the subsequent sections in the summary.

System Overview

This section describes the system in narrative form using nontechnical terms. It should provide a high-level system architecture diagram showing a subsystem breakout of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high-level context diagram for the system and subsystems, if applicable.

Design Constraints

This section describes any constraints in the system design (reference any trade-off analyses conducted such, as resource use versus productivity, or conflicts with other systems) and includes any assumptions made by the project team in developing the system design.

Future Contingencies

This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans.

Document Organization

This section describes the organization of the Systems Design Document.

Points of Contact

This section provides the organization code and title of the key points of contact (and alternates if appropriate) for the information system development effort. These points of contact should include the Project Manager, System Proponent, User Organization, Quality Assurance (QA) Manager, Security Manager, and Configuration Manager, as appropriate.

Project References

This section provides a bibliography of key project references and deliverables that have been produced before this point.

Glossary

Supply a glossary of all terms and abbreviations used in this document. If the glossary is several pages in length, it may be included as an appendix.

6.2 SYSTEM ARCHITECTURE

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 6, External Interfaces.

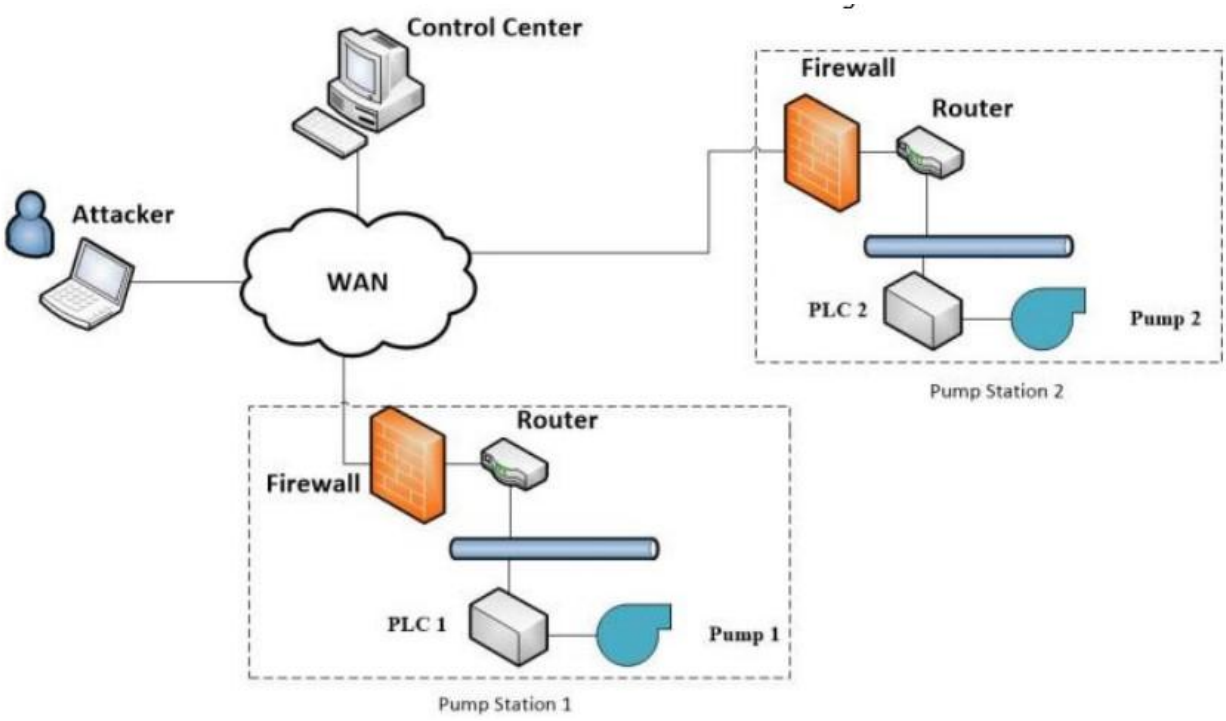


Fig.6.2.1 SCADA Architecture in Water Distribution System

illustrates an example of the cyber structure of SCADA system in the water distribution network, which is used to control and monitor the distribution of the water. The control center and the geographically distributed pump stations are connected through a complex wide area network (WAN). The operators in the control center monitor the statuses of the field devices and send out operation commands. From the intruder's perspective, he/she may launch cyber attacks on either the control center or the distributed pump stations in the network. Once the attacker successfully intrudes into the control network, various malicious actions can be possibly done, such as shutting down pumps, sending out false operation commands, and interrupting the information systems. All these malicious actions will possibly result in serious system failures.

6.4 UML DESIGN

6.4.1USE CASE DIAGRAM:

To model a system the most important aspect is to capture the dynamic behaviour. To clarify a bit in details, dynamic behavior means the behavior of the system when it is running /operating. So only static behaviour is not sufficient to model a system rather dynamic behaviour is more important than static behaviour.

In UML there are five diagrams available to model dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction. These internal and external agents are known as actors. So use case diagrams are consists of actors, use cases and their relationships.

The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system. So to model the entire system numbers of use case diagrams are used. A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the case and will often be accompanied by other types of diagrams as well.

Fig.6.4.1 Use Case Diagram

6.4.2CLASS DIAGRAM:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

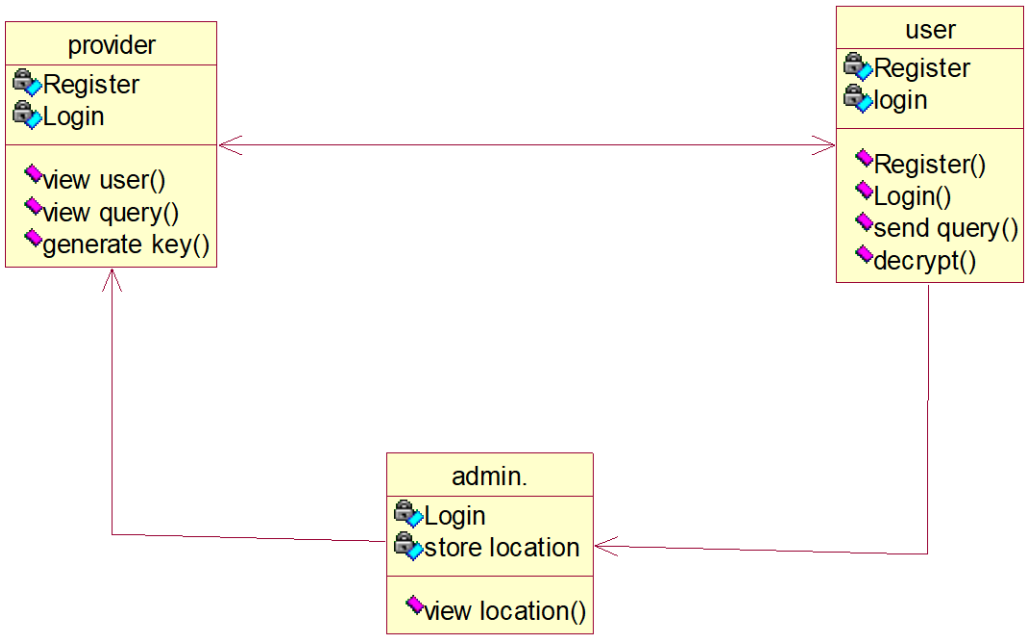


Fig.6.4.2 Class Diagram

6.4.3 SEQUENCE DIAGRAM:

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

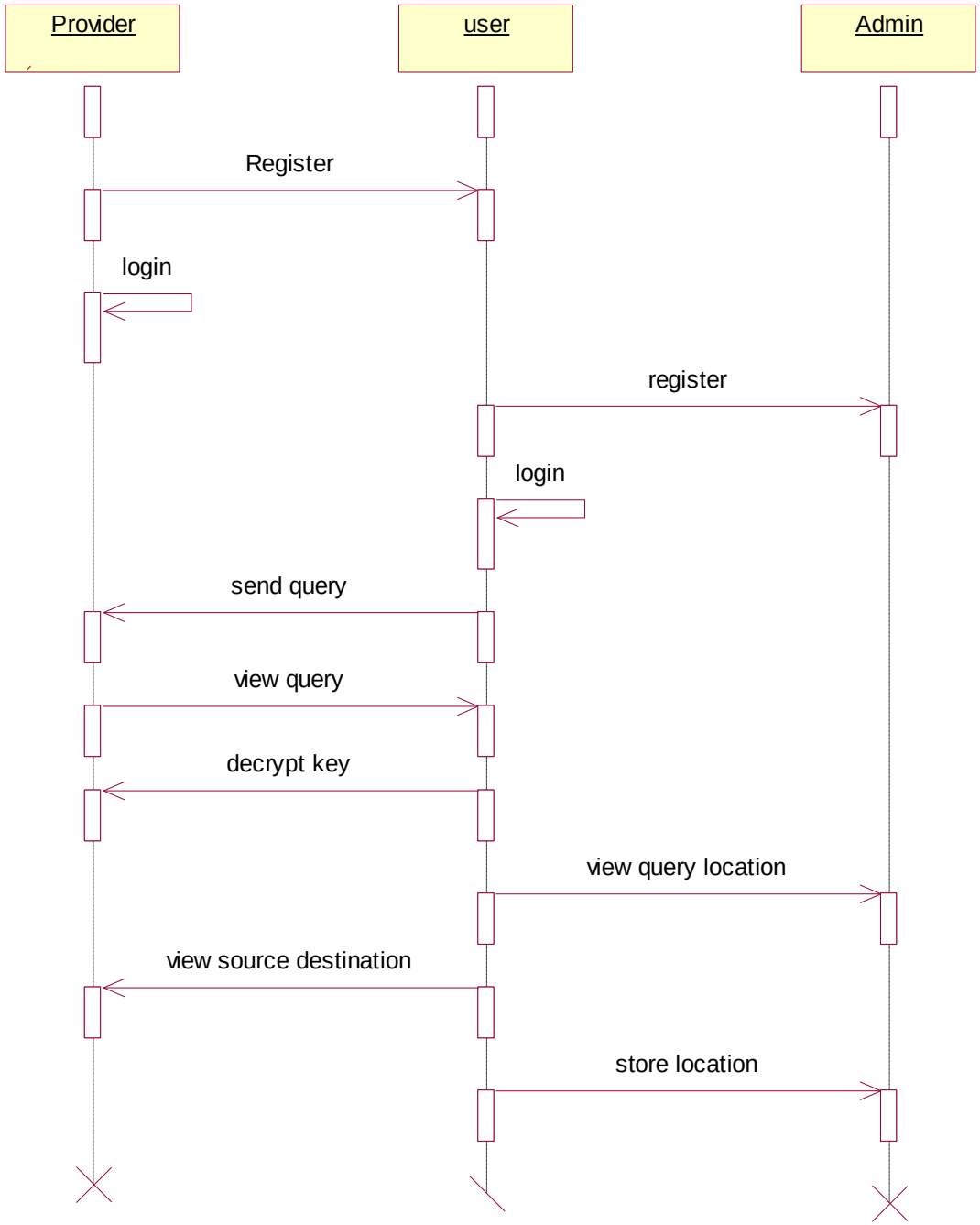


Fig.6.4.3 Sequence Diagram

6.4.4 COLLABORATION DIAGRAM

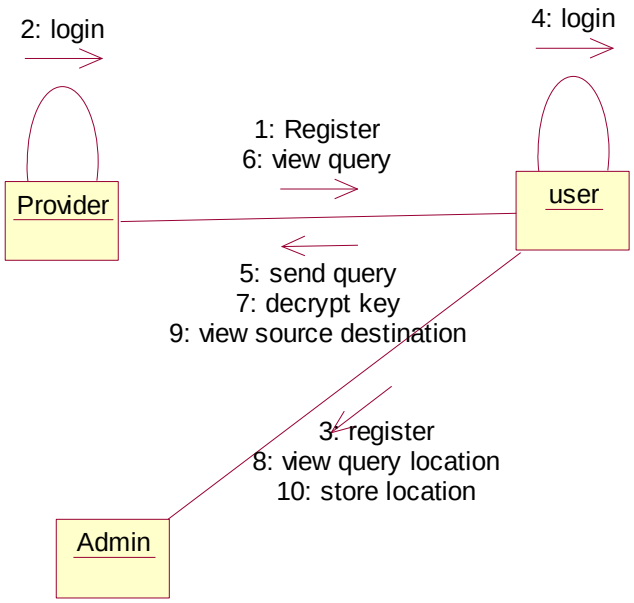


Fig.6.4.4 Collaboration Diagram

6.4.5 ACTIVITY DIAGRAM:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

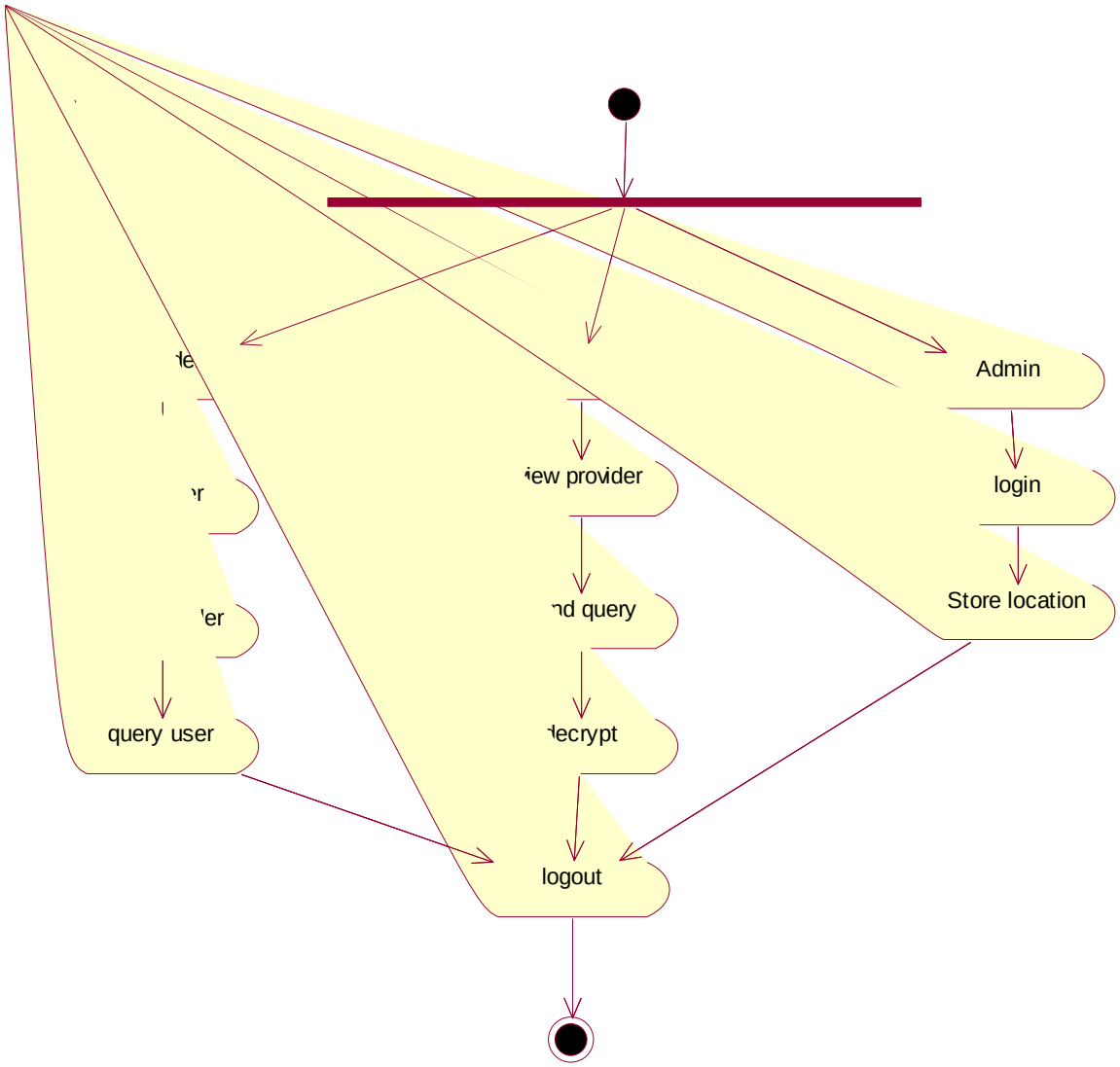


Fig.6.4.5 Activity Diagram

HUMAN-MACHINE INTERFACE

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design .

Inputs

This section is a description of the input media used by the operator for providing information to the system; show a mapping to the high level data flows described System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in File and Database Design, may be referenced. Include data element definitions, or refer to the data.

dictionary. Provide the layout of all input data screens or graphical user interfaces (GUIs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary. This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

- ✓ Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
- ✓ Description of any access restrictions or security considerations
- ✓ Each transaction name, code, and definition, if the system is a transaction-based processing system

Outputs

This section describes of the system output design relative to the user/operator; show a mapping to the high-level data flows described in System outputs include reports, data display screens and GUIs, query results, etc. The output files are described in Section 3 and may be referenced in this section.

The following should be provided, if appropriate:

- ✓ Identification of codes and names for reports and data display screens

- ✓ Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
- ✓ Description of the purpose of the output, including identification of the primary users
- ✓ Report distribution requirements, if any (include frequency for periodic reports)
- ✓ Description of any access restrictions or security considerations

DETAILED DESIGN

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be

Hardware Detailed Design

A hardware component is the lowest level of design granularity in the system. Depending on the design requirements, there may be one or more components per system. This section should provide enough detailed information about individual component requirements to correctly build and/or procure all the hardware for the system (or integrate COTS items).

If there are many components or if the component documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each component and its functions, adequately. Industry-standard component specification practices should be followed. For COTS procurements, if a specific vendor has been identified, include appropriate item names.

Include the following information in the detailed component designs (as applicable):

- ✓ Power input requirements for each component
- ✓ Signal impedances and logic states
- ✓ Connector specifications (serial/parallel, 11-pin, male/female, etc.)
- ✓ Memory and/or storage space requirements
- ✓ Processor requirements (speed and functionality)
- ✓ Graphical representation depicting the number of hardware items (for example, monitors, printers, servers, I/O devices), and the relative positioning of the components to each other
- ✓ Cable type(s) and length(s)
- ✓ User interfaces (buttons, toggle switches, etc.)

Hard drive/floppy drive/CD-ROM requirements

Software Detailed Design

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

- ✓ A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing,

logic, interfaces to other modules, interfaces to external systems, security requirements, etc. explain any algorithms used by the module in detail

- ✓ For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
- ✓ Data elements, record structures, and file structures associated with module input and output
- ✓ Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
- ✓ Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section

Internal Communications Detailed Design

If the system includes more than one component there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. This section should provide enough detailed information about the communication requirements to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):

- ✓ The number of servers and clients to be included on each area network
- ✓ Specifications for bus timing requirements and bus control
- ✓ Format(s) for data being exchanged between components
- ✓ Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances

between components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location

- ✓ LAN topology

EXTERNAL INTERFACES

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

Interface Architecture

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams If appropriate, use subsections to address each interface being implemented.

Interface Detailed Design

For each system that provides information exchange with the system under development, there is a requirement for rules governing the
interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

- ✓ The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined

- ✓ Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
- ✓ Format(s) for error reports exchanged between the systems; should address the disposition of error reports.

Graphical representation of the connectivity between systems, showing the direction of data flow

- ✓ Query and response descriptions If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

SYSTEM INTEGRITY CONTROLS

Sensitive systems use information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of State programs, or the privacy to which individuals are entitled. Developers of sensitive State systems are required to develop specifications for the following minimum levels of control:

- ✓ Internal security to restrict access of critical data items to only those access types required by users
- ✓ Audit procedures to meet control, reporting, and retention period requirements for operational and management reports

CHAPTER 7

SYSTEM IMPLEMENTATION

7.1 Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Distributed
- High performance
- Multithreaded
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

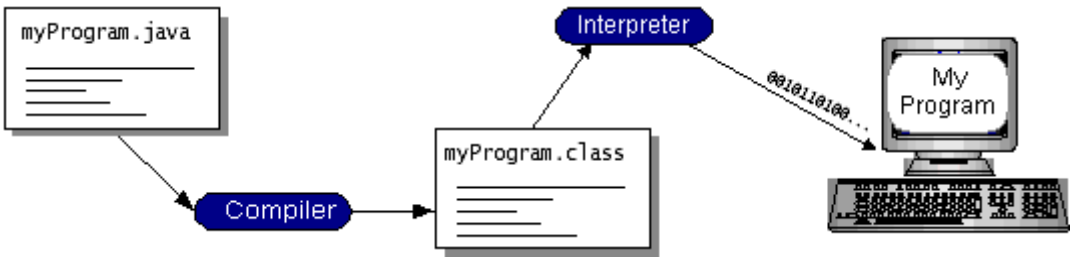


Fig.7.1 java VM

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

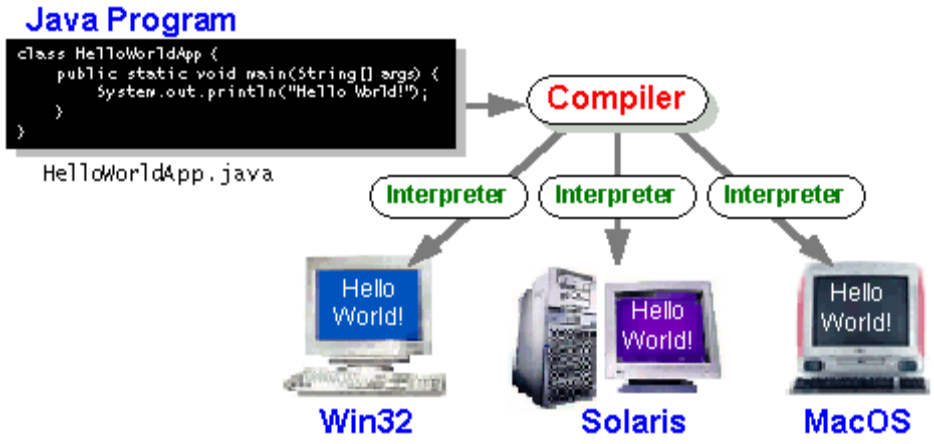


Fig .7.2 compiler

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

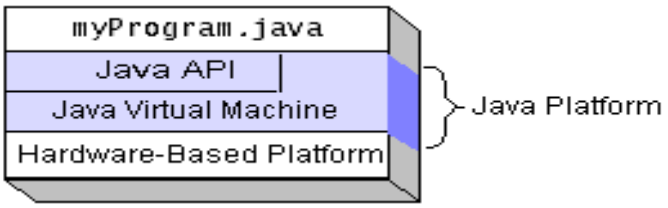


Fig.7.3 java API

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servers run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans™, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

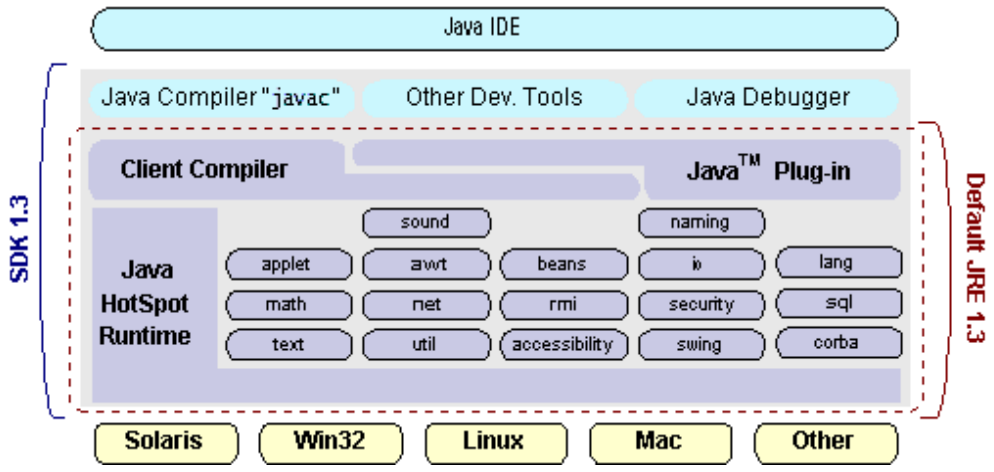


Fig.7.4 java JRE

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.

- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. **SQL Level API**

   The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2 . SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

JDBC must be implemental on top of common database interfaces
The JDBC SQL API must "sit" on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

2 **Provide a Java interface that is consistent with the rest of the Java system**

Because of Java's acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

3 **Keep it simple**

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

4 **Use strong, static typing wherever possible**

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT's, INSERT's, DELETE's and UPDATE's, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

- Simple Architecture-neutral
- Object-oriented
- Portable
- Distributed
- High-performance
- Interpreted
- Multi threaded
- Robust
- Dynamic

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed.

You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it's a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make "write once, run anywhere" possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:

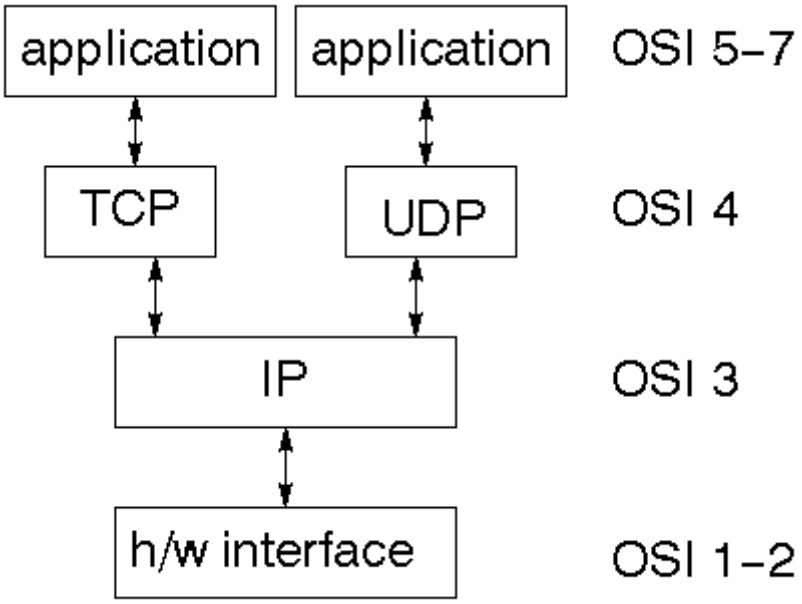


Fig.7.5 User Datagram Protocol

TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

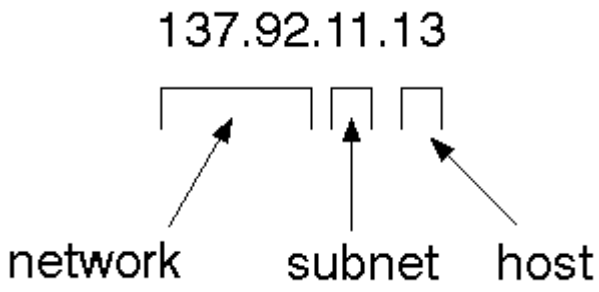
Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address



The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

Non-functional requirement

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. Non-functional requirements add tremendous value to business analysis. It is commonly misunderstood by a lot of people. It is important for business stakeholders, and Clients to clearly explain the requirements and their expectations in measurable terms. If the non-functional requirements are not measurable then they should be revised or rewritten to gain better clarity. For example, User stories help in mitigating the gap between developers and the user community in Agile Methodology.

Usability:

Prioritize the important functions of the system based on usage patterns.

Frequently used functions should be tested for usability, as should complex and critical functions. Be sure to create a requirement for this.

Reliability:

Reliability defines the trust in the system that is developed after using it for a period of time. It defines the likeability of the software to work without failure for a given time period.

The number of bugs in the code, hardware failures, and problems can reduce the reliability of the software.

Your goal should be a long MTBF (mean time between failures). It is defined as the average period of time the system runs before failing.

Create a requirement that data created in the system will be retained for a number of years without the data being changed by the system.

It’s a good idea to also include requirements that make it easier to monitor system performance.

Performance:

What should system response times be, as measured from any point, under what circumstances?

Are there specific peak times when the load on the system will be unusually high?

Think of stress periods, for example, at the end of the month or in conjunction with payroll disbursement.

CHAPTER 8

SYSTEM TESTING

8.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

8.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

8.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

8.4 SYSTEM TESTING

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

8.5 WHITE BOX TESTING

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

8.6 BLACK BOX TESTING

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

CHAPTER 9

SCREENSHOT

CODING:

```
import javax.swing.*;

import java.awt.*;

import java.awt.event.*;

import java.sql.Connection;

import java.util.Random;

public class Startup extends JFrame {

private static final long serialVersionUID = 3l;

public static JTextArea jta;

static JPanel panel,err[];

JPanel main;

JScrollPane jsp;

Container c;

JButton send,clear;

String serverIp;

static JCheckBox jcb;

String processor;
```

```
static JLabel lbl1,lbl2;

JMenuBar jmb;

JMenu menu;

JMenuItem upload,cls,path2,path1,buffer,temp,Network,receiver;

public Startup() throws Exception{

super(" Attacks  ");

Global.ACCESS_KEY = new Random().nextInt(1000);

//UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.W
indowsLookAndFeel");

c = getContentPane();

c.setBackground(Color.darkGray);

c.setLayout(null);

panel= new JPanel();

panel.setBounds(50,50,400,360);

panel.setBackground(Color.black);

panel.setLayout(null);

c.add(panel);

jta = new JTextArea();

jta.setForeground(Color.BLACK);
```

```
jta.setLineWrap(true);

jta.setFont(new Font("Italic",Font.BOLD,14));

jta.setEditable(false);

jsp = new JScrollPane(jta);

jsp.setBorder(BorderFactory.createTitledBorder("------- Welcome     Screen -------
"));

jsp.setBounds(50,20,310,200);

jta.setText("  Attacks (Choose your form using Main Menu)");

panel.add(jsp);

clear = new JButton("Exit");

GUI.button(clear,"Exit");

clear.addActionListener(new Exit());

clear.setBackground(Color.white);

clear.setForeground(Color.black);

clear.setBounds(55,265,270,25);

panel.add(clear);

jmb = new JMenuBar();

setJMenuBar(jmb);

menu = new JMenu("Main Menu");
```

```
jmb.add(menu);

cls = new JMenuItem("Sender");

menu.add(cls);

Network = new JMenuItem("Network 1");

menu.add(Network);

path1 = new JMenuItem("Network 2");

menu.add(path1);

path2 = new JMenuItem("Network 3");

menu.add(path2);

buffer = new JMenuItem("Topology");

menu.add(buffer);

temp = new JMenuItem("Attacker");

menu.add(temp);

receiver = new JMenuItem("Receiver");

menu.add(receiver);

upload = new JMenuItem("Exit");

menu.add(upload);

GUI.forMenuBar(jmb);

GUI.forMenu(menu,'M');
```

```
GUI.assign(cls,'S');

GUI.assign(Network,'T');

GUI.assign(path1,'Q');

GUI.assign(path2,'P');

GUI.assign(buffer,'B');

GUI.assign(receiver,'R');

GUI.assign(temp,'G');

GUI.assign(upload,'E');

cls.addActionListener(new Clear());

Network.addActionListener(new Net1());

path1.addActionListener(new path1x());

path2.addActionListener(new path2x());

buffer.addActionListener(new bufferx());

temp.addActionListener(new tempx());

receiver.addActionListener(new receiverx());

upload.addActionListener(new Uploads());

main = new JPanel();

main.setBackground(Color.black);

main.setLayout(new FlowLayout());
```

```
main.setBounds(25,225,350,50);

panel.add(main);

err = new JPanel[12];

for(int i=0;i<err.length;i++) {

err[i] = new JPanel();

err[i].setBorder(BorderFactory.createRaisedBevelBorder());

main.add(err[i]); }

setSize(500,500);

setLocation(300,200);

setVisible(true);

setResizable(false);

setDefaultCloseOperation(3);}

public class Clear implements ActionListener{

public void actionPerformed(ActionEvent e){

try {

//dispose();

new Sender();}

catch (Exception e2)

e2.printStackTrace();}}}
```

```
public class Net1 implements ActionListener{
public void actionPerformed(ActionEvent e){
try {
//dispose();
new Network1();}
catch (Exception e2) {
e2.printStackTrace();}}}
public class path1x implements ActionListener{
public void actionPerformed(ActionEvent e)
{
try {
//dispose();
new Network2();}
catch (Exception e2) {
e2.printStackTrace();}}}
public class path2x implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
```

```
try {
//dispose();
new Network3();
}
catch (Exception e2) {
e2.printStackTrace();}}}
public class bufferx implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
try {
//dispose();
new topology();
}
catch (Exception e2) {
e2.printStackTrace();}}}
public class tempx implements ActionListener
{
public void actionPerformed(ActionEvent e)
```

```
{
try {
//dispose();
new Attackers();}
catch (Exception e2) {
e2.printStackTrace();}}}
public class receiverx implements ActionListener{
public void actionPerformed(ActionEvent e)
{
try {
//dispose();
new Receiver();
}
catch (Exception e2) {
e2.printStackTrace();}}}
public class Exit implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
```

```
try {
System.exit(0);}
catch (Exception e2) {
e2.printStackTrace();
}
}
}
public class Uploads implements ActionListener
}
public void actionPerformed(ActionEvent e,
{
try
{
dispose();
}
catch(Exception e1)
{
e1.printStackTrace();
}}}
```

```
public static void main(String args[])  throws Exception

{

try {

for(javax.swing.UIManager.LookAndFeelInfo info

javax.swing.UIManager.getInstalledLookAndFeels())

{   if

 ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

Break;

}}

} catch (ClassNotFoundException ex) {


java.util.logging.Logger.getLogger(Startup.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Startup.class.getName()).log(ja

a.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Startup.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
```

```
catch (javax.swing.UnsupportedLookAndFeelException ex) {
java.util.logging.Logger.getLogger(Startup.class.getName()).log(java.util.logging.
Level.SEVERE, null, ex);
}
new Startup();
for(int i=0;i<11;i++) {
err[i].setBackground(Color.red);
Thread.sleep(300);
err[i].setBackground(Color.gray);
}
}
}
```

CHAPTER 9

SCREENSHOT

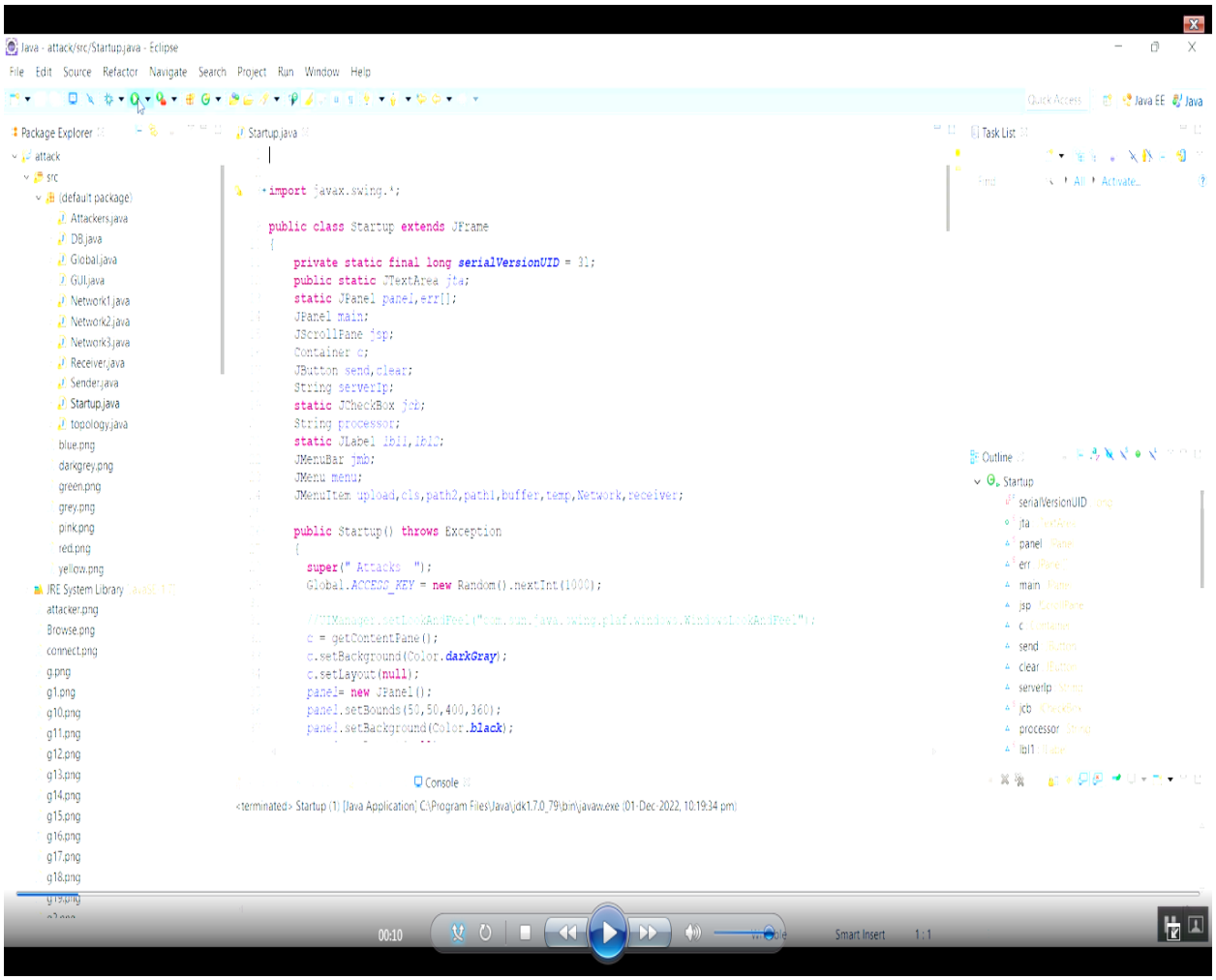


Fig .9.1Coding for cyber insurance Scheme for Water Distribution system

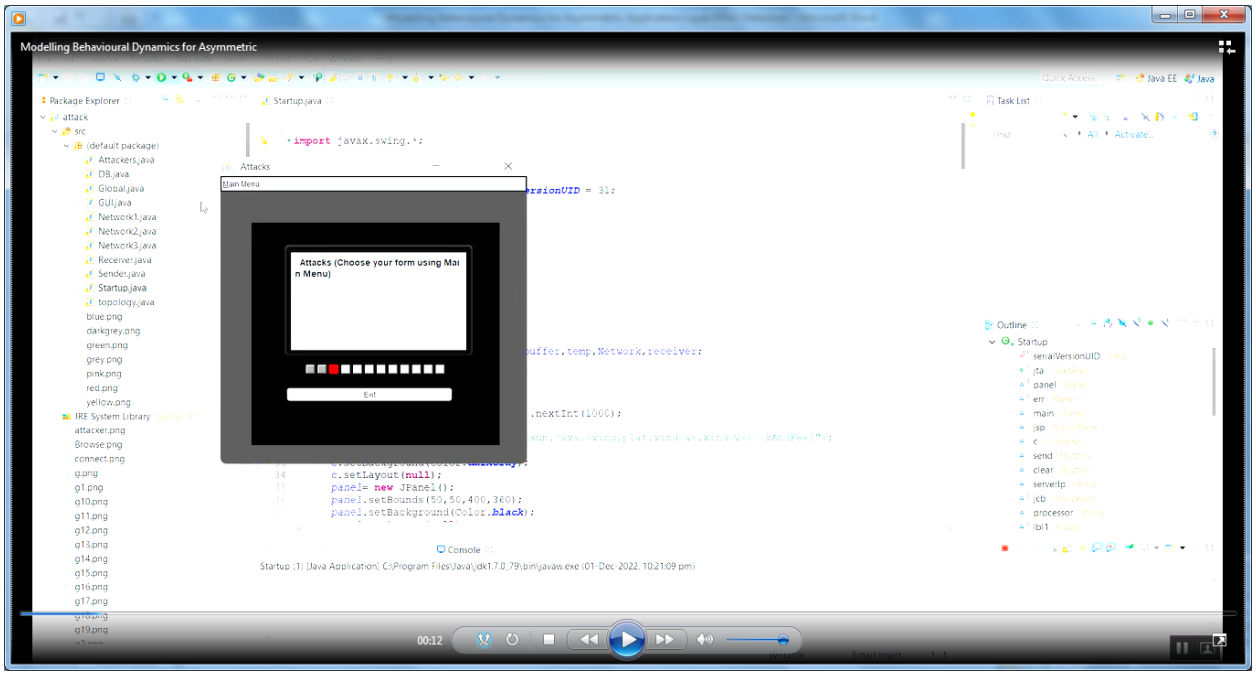


Fig.9.2 Running for this code view in the choosing from the main menu of attacks

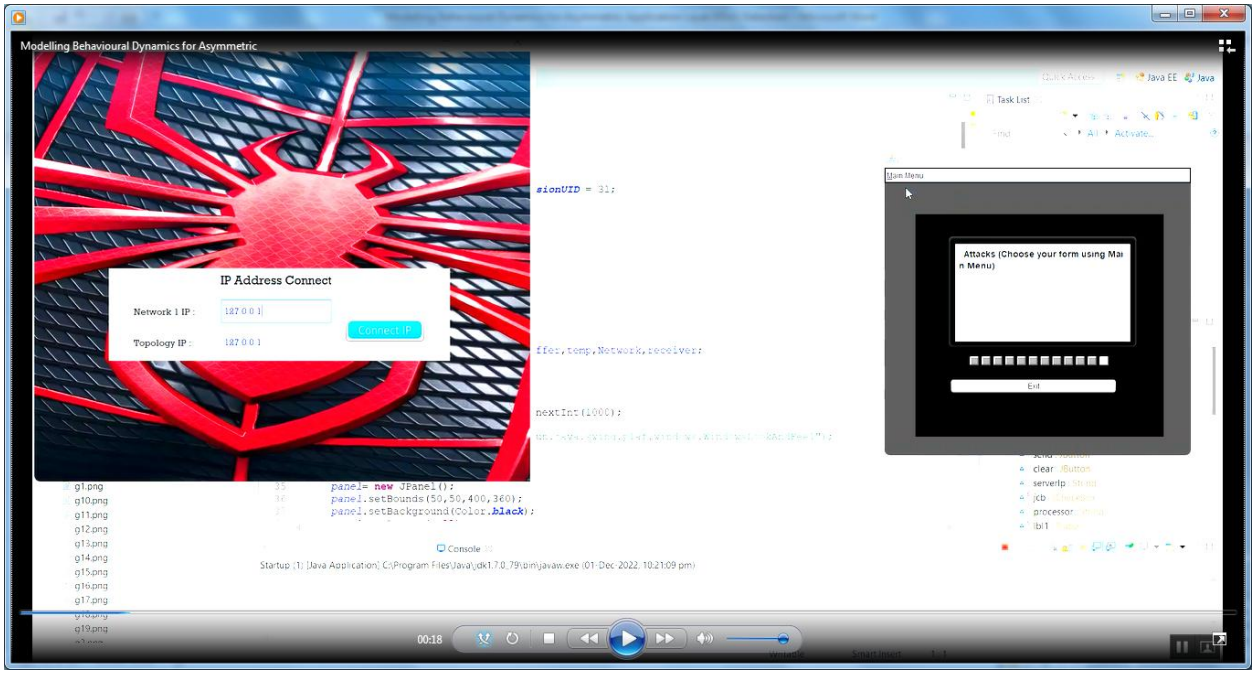


Fig 9.3Choose your form using main menu in the attacks of the sender connected to the address

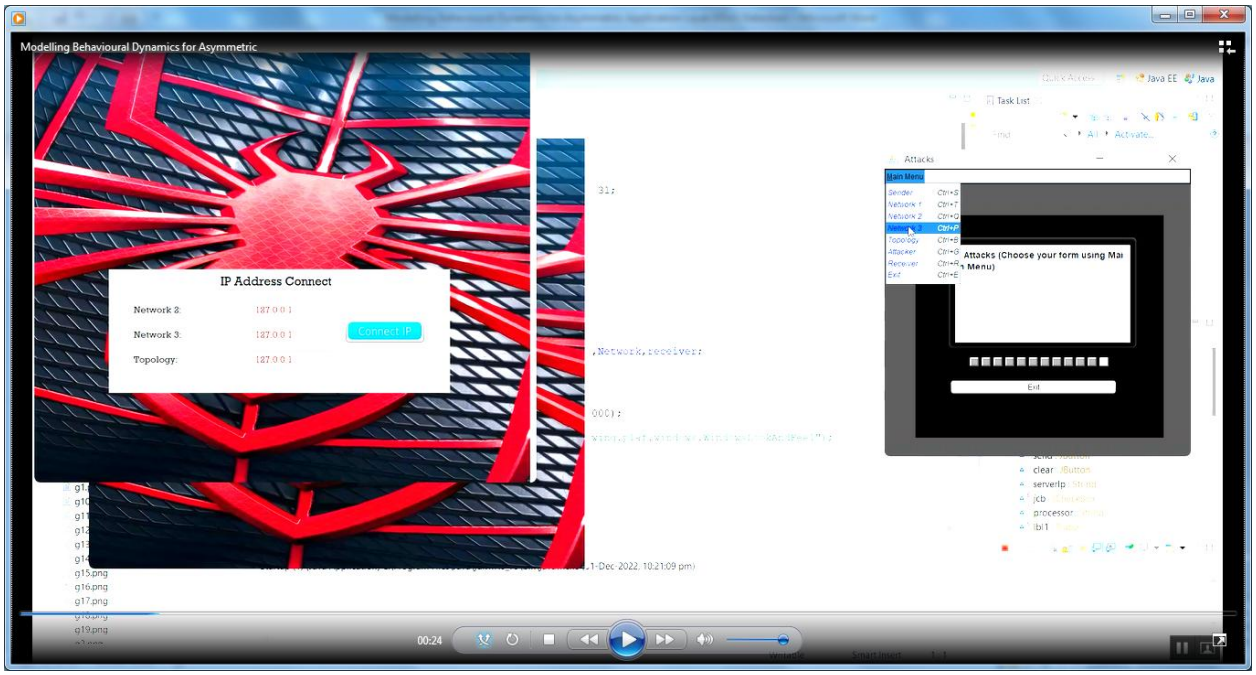


Fig .9.4 Choosing the main menu sender connect to the IP address for the connected in the attacker

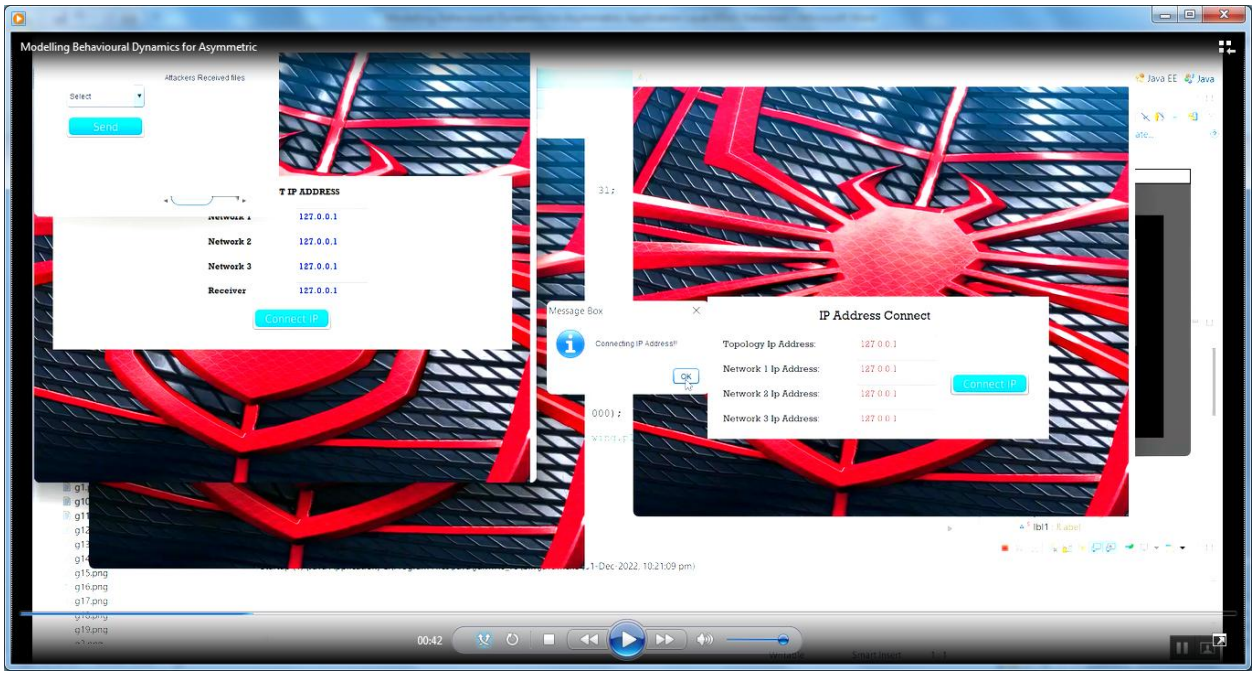


Fig.9.5choosing the main menu for the network 1 is connected to the IP address

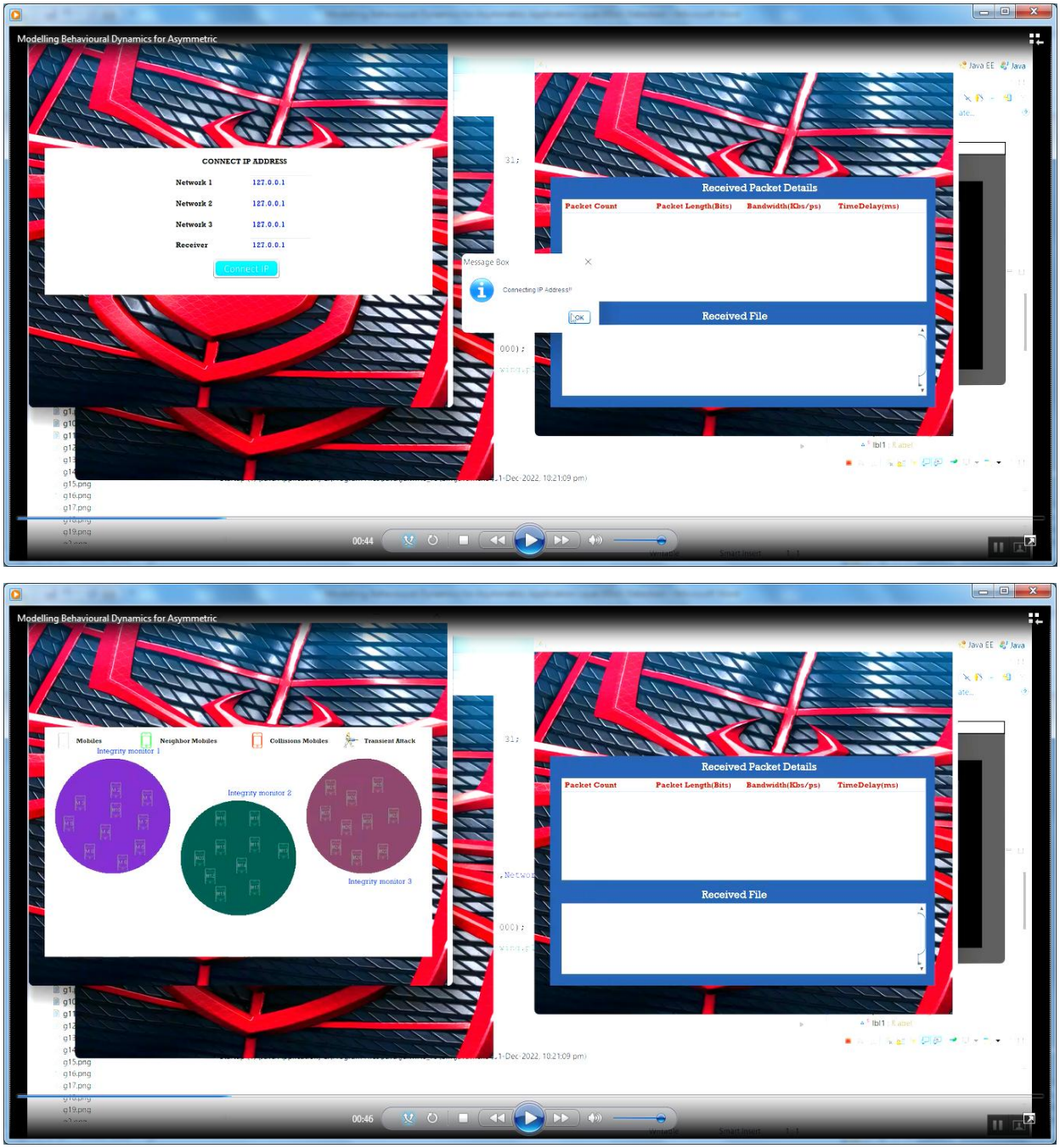


Fig.9.6 Select the mobile and choose the file from the customer is connected to the address and view in the mobile attacker

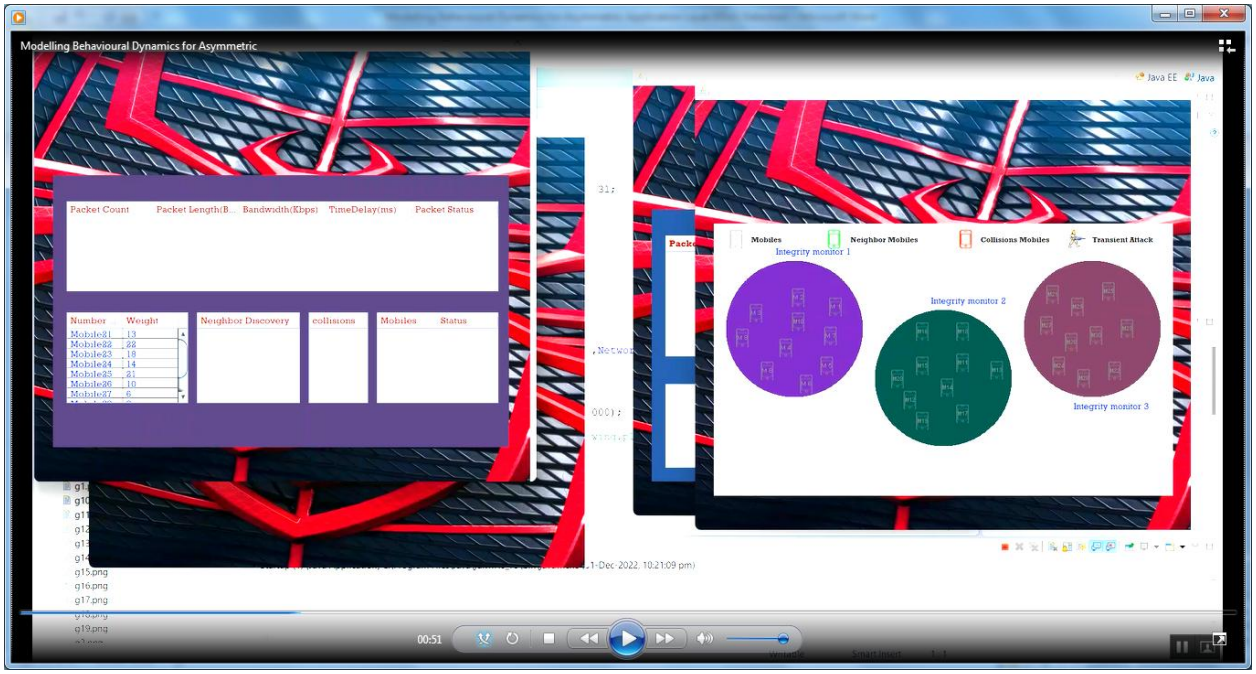
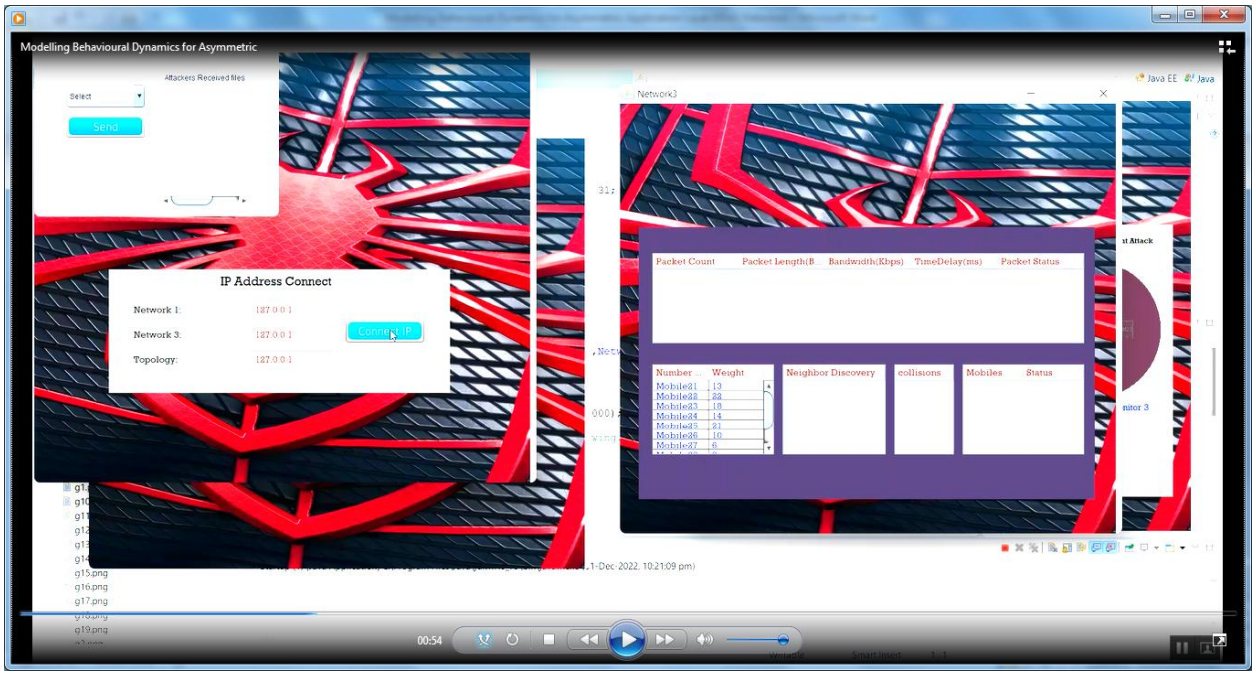


Fig 9.7 There are the packet of mobile number is view in the display the attacker choosing in the mobile number



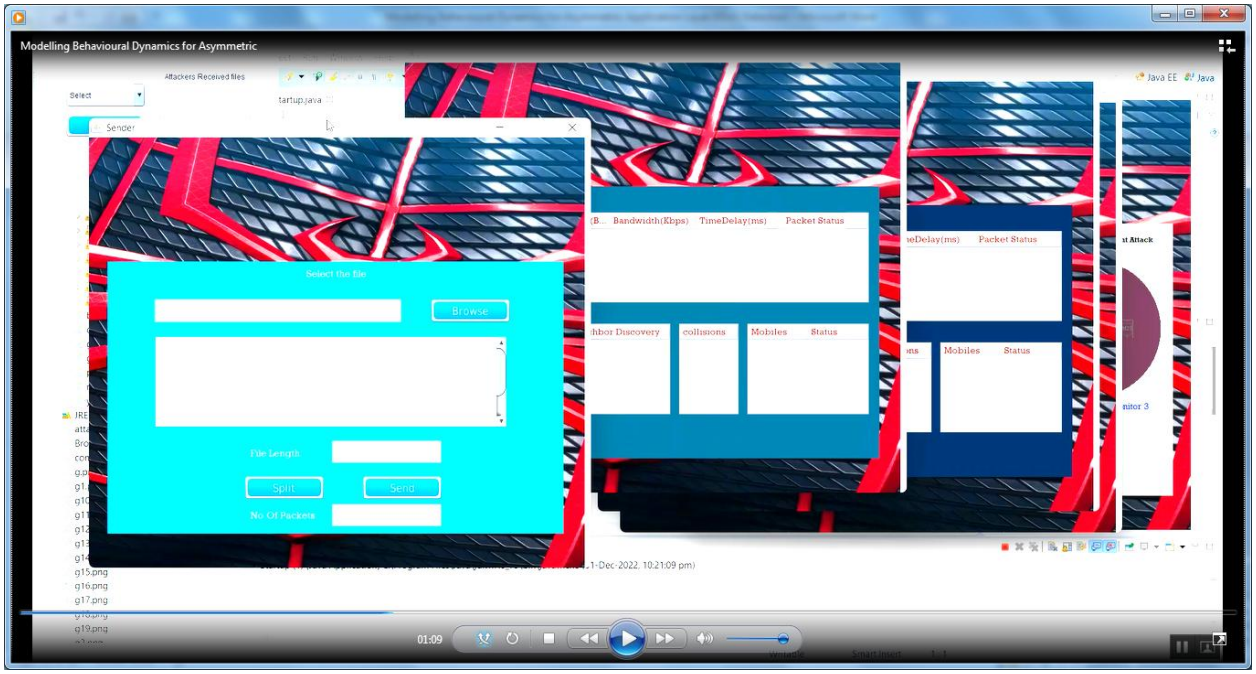
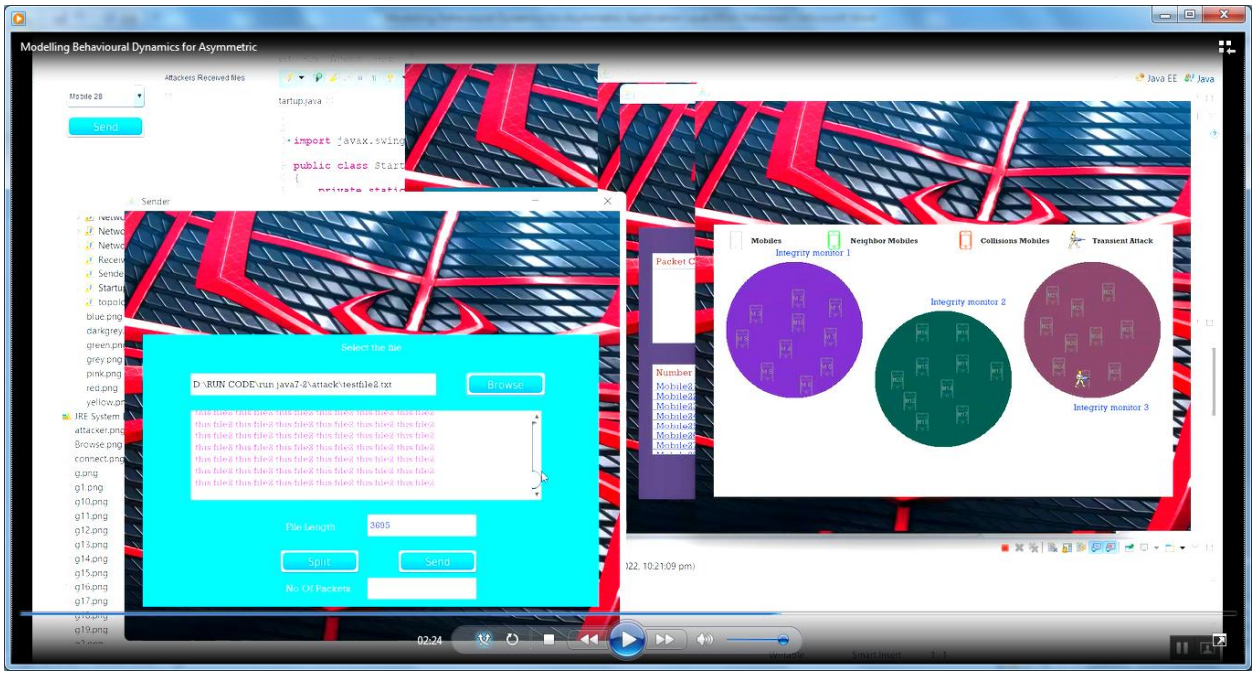


Fig 9.8The following fig is choosing the file for browser in send to the customer split the file send to the package of reciver.



`

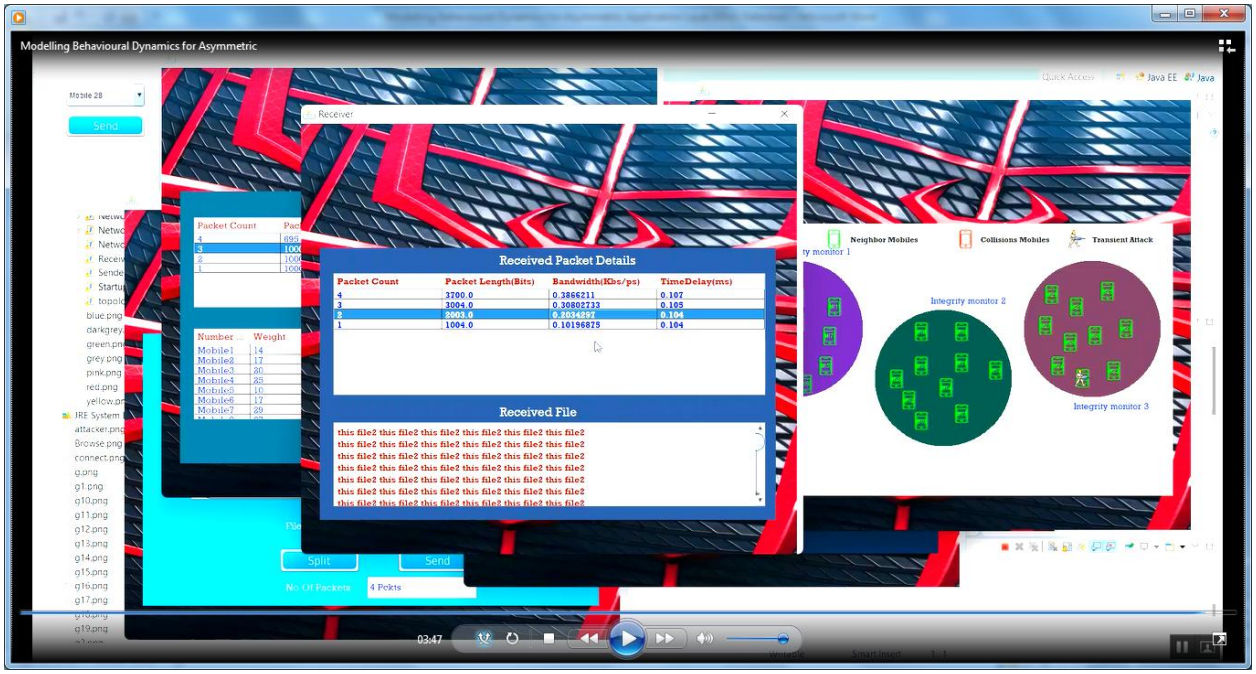


Fig 9.10 The output of cyber insurance for the customer in the attacker of received packet details then the customer detail show in the attacker

CHAPTER 10

10.1 CONCLUSION:

This paper proposes a modified semi-Markov process (SMP) incorporating a cyber-risk correlation model to evaluate the potential cyber security threats against the SCADA system in the water distribution network. By applying the proposed approach, both the independent cyber risks within one individual water network and the correlated cyber risks across different water utilities can be considered. A sequential Monte Carlo simulation-based algorithm is also developed to assess the overall system loss and the failure event duration considering two types of cyber-attack scenarios against the water distribution networks. The total cyber-insurance premium can be estimated based on the designed actuarial principles. The results of the case studies indicate that higher system reliability and more advanced self-protection mechanism can reduce the cyber-insurance premium of the water utilities with the proposed actuarial principles.

REFRANCE

[1] S. Amin, X. Litrico, S. Sastry and A. M. Bayen, "Cyber Security of Water SCADA Systems—Part I: Analysis and Experimentation of Stealthy Deception Attacks," in IEEE Transactions on Control Systems Technology, vol. 21, no. 5, pp. 1963-1970, Sept. 2013.

[2] S. Diop, P. M'mayi, D. Lisbjerg, and R. Johnstone, “Vital water graphics: an overview of the state of the world's fresh and marine waters,” vol. 1: UNEP/Earthprint, 2002.

[3] J.H. Germano, Cybersecurity risk & responsibility in the water sector. American Water Works Association. (2018)

[4] L.A. Gordon, L.P. Martin, and T. Sohail "A framework for using insurance for cyber-risk management." Communications of the ACM 46.3 (2003): 81-85.

[5] A. Hassanzadeh, A. Rasekh, S. Galleli, M. Aghashahi, R. Taormina, A. Ostfeld, and M. Banks, "A review of cybersecurity incidents in the water sector." Journal of Environmental Engineering 146.5 (2020): 03120003.

[6] E. J. Lee and K. J. Schwab, "Deficiencies in drinking water distribution systems in developing countries," Journal of water and health, vol. 3, pp. 109-127, 2005.

[7] B. Obama, "Executive order 13636: Improving critical infrastructure cyber security." Federal Register 78.33 (2013): 11739.

[8] S. Reed, Was BWL prepared for a ransomware attack? Lansing State Journal. (2016)

[9] U.S. Environmental Protection Agency (EPA) Water Sector Cyber security Brief for States. EPA. (2018)

[10] W. Yurcik, and D. Doss, "Cyber insurance: A market solution to the internet security market failure," in Workshop on Economics and Information Security (WEIS), Berkeley, CA. 2002.