ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

The Geometric Efficient Matching Algorithm For Firewalls

Priyanka Harish Pachkore*, C.M. Jadhav
Bharat Ratna Indira Gandhi College Of Engineering, Kegoan, Solapur, India
priyankaspophalikar@gmail.com

Abstract

Given a geographic query that is composed of query keywords and a location, a geographic search engine retrieves documents that are the most textually and spatially relevant to the query keywords and the location, respectively, and ranks the retrieved documents according to their joint textual and spatial relevance's to the query. The lack of an efficient index that can simultaneously handle both the textual and spatial aspects of the documents makes existing geographic search engines inefficient in answering geographic queries. In this paper, we propose an efficient index, called IR-tree, that together with a top-k document search algorithm facilitates four major tasks in document searches, namely, 1) spatial filtering, 2) textual filtering,3) relevance computation, and 4) document ranking in a fully integrated manner. In addition, IR-tree allows searches to adopt different weights on textual and spatial relevance of documents at the runtime and thus caters for a wide variety of applications. A set of comprehensive experiments over a wide range of scenarios has been conducted and the experiment results demonstrate that IR-tree outperforms the state-of-the art approaches for geographic document searches.

Keywords: Spatial filtering, textual filtering.

Introduction

The firewall is one of the central technologies allowing high level access control to organization networks. Packet matching in firewalls involves matching on many fields from the TCP and IP packet header. At least five fields (protocol number, source and destination IP addresses, and ports) are involved in the decision which rule applies to a given packet. With available bandwidth increasing rapidly, very efficient matching algorithms need to be deployed in modern firewalls to ensure that the firewall. Modern firewalls all use "first match" The firewall rules are numbered from 1 to n, and the firewall applies the policy (e.g., pass or drop) associated with the first rule that matches a given packet. Firewall packet matching is reminiscent of the well studied packet matching problem.

Purpose : However, there are several crucial differences which make the problems quite different. First, unlike firewalls, routers use "longest prefix match" semantics. Therefore, firewalls require their own special algorithms.

Literature survey

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy n company strength. Once these things arr satisfied, then next steps is to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.
We have to analysis the Secure computing…

Data centre Security?

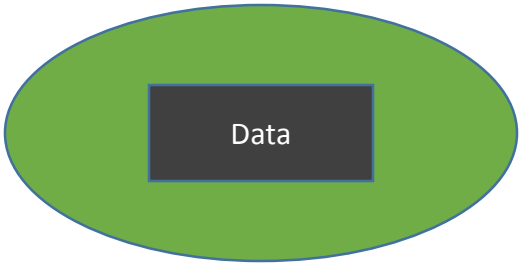
1. Professional Security staff utilizing video surveillance, state of the art intrusion detection systems, and other electronic means.
2. When an employee no longer has a business need to access datacenter his privileges to access datacenter should be immediately revoked.
3. All physical and electronic access to data centers by employees should be logged and audited routinely.
4. Audit tools so that users can easily determine how their data is stored, protected, used, and verify policy enforcement.

Data Location:

When user uses the cloud, user probably won't know exactly where your data is hosted, what country it will be stored in?
Data should be stored and processed only in specific jurisdictions as define by user.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Provider should also make a contractual commitment to obey local privacy requirements on behalf of their customers,
Data-centered policies that are generated when a user provides personal or sensitive information that travels with that information throughout its lifetime to ensure that the information is used only in accordance with the policy



Backups of Data:
1. Data store in database of provider should be redundantly store in multiple physical locations.
2. Data that is generated during running of program on instances is all customer data and therefore provider should not perform backups.
3. Control of Administrator on Databases.

Data Sanitization:
1. Sanitization is the process of removing sensitive information from a storage device.
2. What happens to data stored in a cloud computing environment once it has passed its user's "use by date"
3.What data sanitization practices does the cloud computing service provider propose to implement for redundant and retiring data storage devices as and when these devices are retired or taken out of service.

Network Security:
1. Denial of Service: where servers and networks are brought down by a huge amount of network traffic and users are denied the access to a certain Internet based service.
2. Like DNS Hacking, Routing Table "Poisoning", XDoS attacks
3. QoS Violation: through congestion, delaying or dropping packets, or through resource hacking.
4. Man in the Middle Attack: To overcome it always use SSL
5. IP Spoofing: Spoofing is the creation of TCP/IP packets using somebody else's IP address.
6. Solution: Infrastructure will not permit an instance to send traffic with a source IP or MAC address other than its own.

How secure is encryption Scheme:
Is it possible for all of my data to be fully encrypted?
What algorithms are used?
Who holds, maintains and issues the keys? Problem:
Encryption accidents can make data totally unusable.
Encryption can complicate availability Solution
The cloud provider should provide evidence that encryption schemes were designed and tested by experienced specialists.

Information Security:
1. Security related to the information exchanged between different hosts or between hosts and users.
2. This issues pertaining to secure communication, authentication, and issues concerning single sign on and delegation.
3. Secure communication issues include those security concerns that arise during the communication between two entities.
4. These include confidentiality and integrity issues. Confidentiality indicates that all data sent by users should be accessible to only "legitimate" receivers, and integrity indicates that all data received should only be sent/modified by "legitimate" senders.
5. Solution: public key encryption, X.509 certificates, and the Secure Sockets Layer (SSL) enables secure authentication and communication over computer networks.

System analysis

After analyzing the requirements of the task to be performed, the next step is to analyze the problem and understand its context. The first activity in the phase is studying the existing system and other is to understand the requirements and domain of the new system. Both the activities are equally important, but the first activity serves as a basis of giving the functional specifications and then successful design of the proposed system. Understanding the properties and requirements of a new system is more difficult and requires creative thinking and understanding of existing running system is also difficult, improper understanding of present system can lead diversion from solution.

Analysis Model

The model that is basically being followed is the WATER FALL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If system exists one and modification and addition of new module is needed, analysis of present system can be used as basic model.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model the sequence of activities performed in a software development project are: -

- Requirement Analysis
- Project Planning
- System design
- Detail design
- Coding
- Unit testing
- System integration & testing

Here the linear ordering of these activities is critical. End of the phase and the output of one phase is the input of other phase. The output of each phase is to be consistent with the overall requirement of the system. Some of the qualities of spiral model are also incorporated like after the people concerned with the project review completion of each of the phase the work done. WATER FALL MODEL was being chosen because all requirements were known beforehand and the objective of our software development is the computerization/automation of an already existing manual working system.

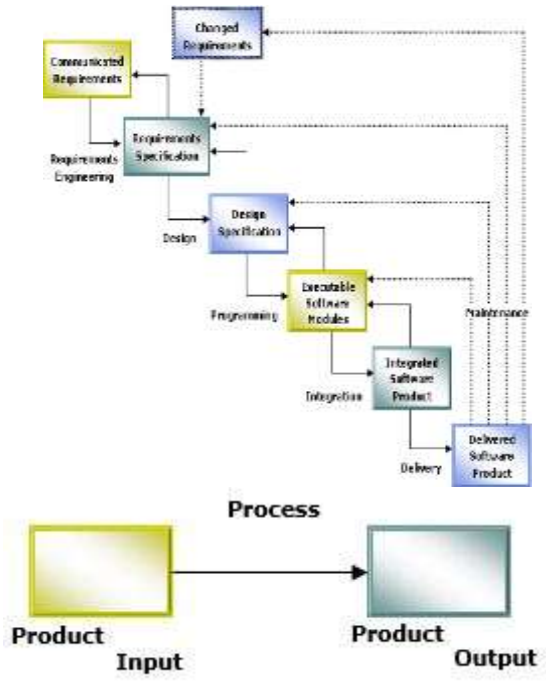


Fig 3.1: Water Fall Model

tudy of the System

Existing System:

In Existing system a PHR system model, there are multiple owners who may encrypt according to their own ways, possibly using different sets of cryptographic keys. Letting each user obtain keys from every owner who's PHR she wants to read would limit the accessibility since patients are not always online. An alternative is to employ a central authority (CA) to do the key management on behalf of all PHR owners, but this requires too much trust on a single authority (i.e., cause the key escrow problem).

Key escrow (also known as a "fair" cryptosystem) is an arrangement in which the keys needed to decrypt encrypted data are held in escrow so that, under certain circumstances, an authorized third party may gain access to those keys. These third parties may include businesses, who may want access to employees' private communications, or governments, who may wish to be able to view the contents of encrypted communications.

Proposed System

We endeavor to study the patient centric, secure sharing of PHRs stored on semi-trusted servers, and focus on addressing the complicated and challenging key management issues. In order to protect the personal health data stored on a semi-trusted server, we adopt attribute-based encryption (ABE) as the main encryption primitive.

Using ABE, access policies are expressed based on the attributes of users or data, which enables a patient to selectively share her PHR among a set of users by encrypting the file under a set of attributes, without the need to know a complete list of users.

The complexities per encryption, key generation and decryption are only linear with the number of attributes involved.

Input & Output Design

Input Design: Input design is a part of overall system design. The main objective during the input design is as given below:

To produce a cost-effective method of input.

1 To achieve the highest possible level of accuracy.

2 To ensure that the input is acceptable and understood by the user.

Output Design: Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Implementation: Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Interaction Model:

1. Client-driven interventions

Client-driven interventions are the means to protect customers from unreliable services. For example, services that miss deadlines or do not respond at all for a longer time are replaced by other more reliable services in future discovery operations.

2. Provider-driven interventions

Provider-driven interventions are desired and initiated by the service owners to shield themselves from malicious clients. For instance, requests of clients performing a denial of service attack by sending multiple requests in relatively short intervals are blocked (instead of processed) by the service.

System study

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

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.

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.

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.

System specifications

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 512 Mb.

Software Requirements:

- Operating system : Windows XP.
- Coding Language : C#.net
- Data Base : SQL Server 20**05**

Software environment

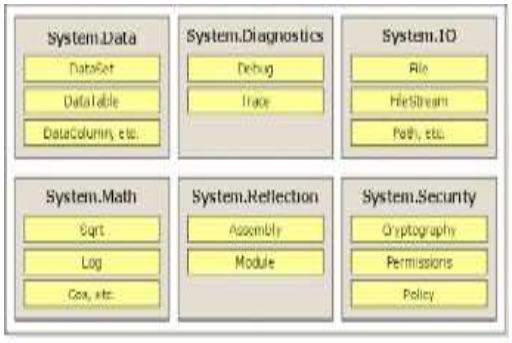
Features of .Net : Microsoft .NET is a set of Microsoft software technologies for rapidly building and integrating XML Web services, Microsoft Windows-based applications, and Web solutions. The .NET Framework is a language-neutral platform for writing programs that can easily and securely interoperate. There's no language barrier with .NET: there are numerous languages available to the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

developer including Managed C++, C#, Visual Basic and Java Script. The .NET framework provides the foundation for components to interact seamlessly, whether locally or remotely on different platforms. It standardizes common data types and communications protocols so that components created in different languages can easily interoperate. ".NET" is also the collective name given to various software components built upon the .NET platform. These will be both products (Visual Studio.NET and Windows.NET Server, for instance) and services (like Passport, .NET My Services, and so on).

The .Net Framework : The .NET Framework has two main parts:
1. The Common Language Runtime (CLR).
2. A hierarchical set of class libraries.
The CLR is described as the "execution engine" of .NET. It provides the environment within which programs run. The most important features are

1.Conversion from a low-level assembler-style language, called Intermediate Language (IL), into code native to the platform being executed on.
2. Memory management, notably including garbage collection.
3. Checking and enforcing security restrictions on the running code.
4. Loading and executing programs, with version control and other such features.
5. The following features of the .NET framework are also worth description:
Managed Code: The code that targets .NET, and which contains certain extra
Information - "metadata" - to describe itself. Whilst both managed and unmanaged code can run in the runtime, only managed code contains the information that allows the CLR to guarantee, for instance, safe execution and interoperability.
Managed Data: With Managed Code comes Managed Data. CLR provides memory allocation and Deal location facilities, and garbage collection. Some .NET languages use Managed Data by default, such as C#, Visual Basic.NET and JScript.NET, whereas others, namely C++, do not. Targeting CLR can, depending on the language you're using, impose certain constraints on the features available. As with managed and unmanaged code, one can have both managed and unmanaged data in .NET applications - data that doesn't get garbage collected but instead is looked after by unmanaged code.
Common Type System: The CLR uses something called the Common Type System (CTS) to strictly enforce type-safety. This ensures that all classes are compatible with each other, by describing types in a common way. CTS define how types work within the runtime, which enables types in one language to interoperate with types in another language, including cross-language exception handling. As well as ensuring that types are only used in appropriate ways, the runtime also ensures that code doesn't attempt to access memory that hasn't been allocated to it.
Common Language Specification : The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.
The .NET Class Framework : We will now discuss about the .NET Class Framework. In conjunction with the CLR, the Microsoft has developed a comprehensive set of framework classes, several of which are shown below:



Since the .NET Class Framework contains literally thousands of types, a set of related types is presented to the developer within a single *namespace.* For example, the *System* namespace (which you should be most familiar with) contains the Object base type, from which all other types ultimately derive. In addition the *System* namespace contains types of integers, characters, strings, exception handling, and console I/O's as well as a bunch of utility types that convert safely between data types, format data types, generate random numbers, and perform various math functions. All applications use types from *System* namespace. To access any platform feature, you need to know which namespace contains the type that exposes the functionality you want. If you want to customize the behavior of any type, you can simply derive your own type from the desired .NET framework type. The .NET Framework relies on the object-oriented nature of the platform to present a consistent programming paradigm to software

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

developers. It also enables you to create your own namespaces containing their own types, which merge seamlessly into the programming paradigm. This greatly simplifies the Software Development. The table below lists some of the general namespaces, with a brief description of what the classes in that namespace is used for:

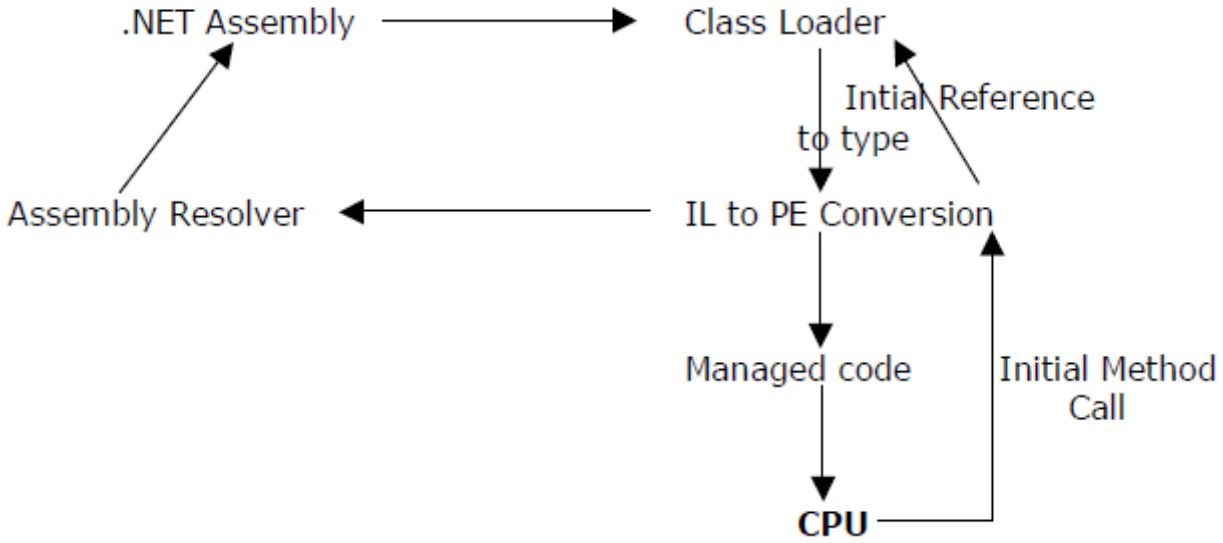
Namespace	Purpose of Class
System	All the basic types used by every application.
System.Collections	Managing collections of objects. Includes the popular collection types such as Stacks, Queues, HashTables etc.
System.Diagnostics	Instrumenting and Debugging your application.
System.Drawing	Manipulating 2D graphics. Typically used for Windows Forms applications and for creating Images that are to appear in a web form.
System.EnterpriseServices	Managing Transactions, queued components, object pooling, just in time activation, security and other features to make use of managed code more efficient on the server.
System.Globalization	National Language Support(NLS), such as string compares, formatting and calendars.
System.IO	Doing Stream I/O, walking directories and files.
System.Management	Managing other computers in the enterprise via WMI.
System.Net	Network Communications.
System.Reflection	Inspecting metadata and late binding of types and their members.
System.Resources	Manipulating external data resources.
System.Runtime.InteropServices	Enabling managed code to access unmanaged OS platform facilities, such as COM components and functions in Win32 DLLs.
System.Runtime.Remoting	Accessing types remotely.
System.Runtime.Serilization	Enabling instances of objects to be persisted and regenerated from a stream.
System.Security	Protecting data and resources.
System.Text	Working with Text in different encodings, like ASCII or Unicode.
System.Threading	Performing asynchronous operations and synchronizing access to resources.
System.Xml	Processing XML Schemas and data.

In addition to the general namespace the .Net Class Framework offers namespaces whose types are used for building specific application types. The table below lists some of the application specific namespaces:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Namespace	Purpose of Types
System.Web.Services	Building web services
System.Web.UI	Building web forms.
System.Windows.Forms	Building Windows GUI applications.
System.ServiceProcess	Building a windows service controllable by Service Control Manager.

Just-In-Time Compilation (JIT)
The MSIL is the language that all of the .NET languages compile down to. After they are in this intermediate language, a process called Just-In-Time (JIT) compilation occurs when resources are used from your application at runtime. JIT allows "parts" of your application to execute when they are needed, which means that if something is never needed, it will never compile down to the native code. By using the JIT, the CLR can cache code that is used more than once and reuse it for subsequent calls, without going through the compilation process again.
The figure below shows ***the JIT Process***:



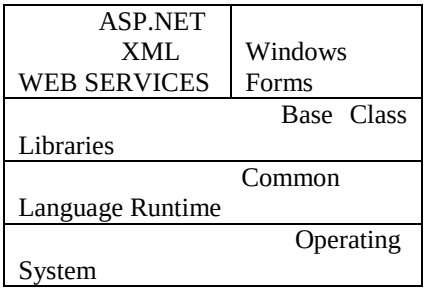
JIT Compilation Process: The JIT process enables a secure environment by making certain assumptions:
• Type references are compatible with the type being referenced.
• Operations are invoked on an object only if they are within the execution parameters for that object.
• Identities within the application are accurate. By following these rules, the managed execution can guarantee that code being executed is type safe; the execution will only take place in memory that it is allowed to access. This is possible by the verification process that occurs when the MSIL is Converted into CPU-specific code. During this verification, the code is examined to ensure that it is not corrupt, it is type safe, and the code does not interfere with existing security policies that are in place on the system.

Common Language Specification : The CLR provides built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

The Class Library : .NET provides a single-rooted hierarchy of classes, containing over 7000 types. The root of the namespace is called System; this contains basic types like Byte, Double, Boolean, and String, as well as Object. All objects derive from System. Object. As well as objects, there are value types. Value types can be allocated on the stack, which can provide useful flexibility. There are also efficient means of

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

converting value types to object types if and when necessary. The set of classes is pretty comprehensive, providing collections, file, screen, and network I/O,



threading, and so on, as well as XML and database connectivity. The class library is subdivided into a number of sets (or namespaces), each providing distinct areas of functionality, with dependencies between the namespaces kept to a minimum.

Languages Supported By .Net : The multi-language capability of the .NET Framework and Visual Studio .NET enables developers to use their existing programming skills to build all types of applications and XML Web services. The .NET framework supports new versions of Microsoft's old favorites Visual Basic and C++ (as VB.NET and Managed C++), but there are also a number of new additions to the family. Visual Basic .NET has been updated to include many new and improved language features that make it a powerful object-oriented programming language. These features include inheritance, interfaces, and overloading, among others. Visual Basic also now supports structured exception handling, custom attributes and also supports multi-threading. Visual Basic .NET is also CLS compliant, which means that any CLS-compliant language can use the classes, objects, and components you create in Visual Basic .NET.

Managed Extensions for C++ and attributed programming are just some of the enhancements made to the C++ language. Managed Extensions simplify the task of migrating existing C++ applications to the new .NET Framework. C# is Microsoft's new language. It's a C-style language that is essentially "C++ for Rapid Application Development". Unlike other languages, its specification is just the grammar of the language. It has no standard library of its own, and instead has been designed with the intention of using the .NET libraries as its own. Microsoft Visual J# .NET provides the easiest transition for Java-language developers into the world of XML Web Services and dramatically improves the interoperability of Java-language programs with existing software written in a variety of other programming languages. Active State has created Visual Perl and Visual Python, which enable .NET-aware applications to be built in either Perl or Python. Both products can be integrated into the Visual Studio .NET environment. Visual Perl includes support for Active State's Perl Dev Kit.

Other languages for which .NET compilers are available include

- FORTRAN
- COBOL
- Eiffel

C#.NET is also compliant with CLS (Common Language Specification) and supports structured exception handling. CLS is set of rules and constructs that are supported by the CLR (Common Language Runtime). CLR is the runtime environment provided by the .NET Framework; it manages the execution of the code and also makes the development process easier by providing services.C#.NET is a CLS-compliant language. Any objects, classes, or components that created in C#.NET can be used in any other CLS-compliant language. In addition, we can use objects, classes, and components created in other CLS-compliant languages in C#.NET .The use of CLS ensures complete interoperability among applications, regardless of the languages used to create the application.

Constructors and Destructors Constructors are used to initialize objects, whereas destructors are used to destroy them. In other words, destructors are used to release the resources allocated to the object. In C#.NET the sub finalize procedure is available. The sub finalize procedure is used to complete the tasks that must be performed when an object is destroyed. The sub finalize procedure is called automatically when an object is destroyed. In addition, the sub finalize procedure can be called only from the class it belongs to or from derived classes.

Garbage Collection : Garbage Collection is another new feature in C#.NET. The .NET Framework monitors allocated resources, such as objects and variables. In addition, the .NET Framework automatically releases memory for reuse by destroying objects that are no longer in use. In C#.NET, the garbage collector checks for the objects that are not currently in use by applications. When the garbage collector comes across an object that is marked for garbage collection, it releases the memory occupied by the object.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Overloading: Overloading is another feature in C#. Overloading enables us to define multiple procedures with the same name, where each procedure has a different set of arguments. Besides using overloading for procedures, we can use it for constructors and properties in a class.

Multithreading: C#.NET also supports multithreading. An application that supports multithreading can handle multiple tasks simultaneously, we can use multithreading to decrease the time taken by an application to respond to user interaction.

Structured Exception Handling: C#.NET supports structured handling, which enables us to detect and remove errors at runtime. In C#.NET, we need to use Try…Catch…Finally statements to create exception handlers. Using Try…Catch…Finally statements, we can create robust and effective exception handlers to improve the performance of our application.

Objectives of .Net Frame work :The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet.

1. To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
2. To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
3. Eliminates the performance problems. There are different types of application, such as Windows-based applications and Web-based applications.

Structure of a .NET Application

DLL Hell : DLLs gave developers the ability to create function libraries and programs that could be shared with more than one application. Windows itself was based on DLLs. While the advantages of shared code modules expanded developer opportunities, it also introduced the problem of updates, revisions, and usage. If one program relied on a specific version of a DLL, and another program upgraded that same DLL, the first program quite often stopped working. Microsoft added to the problem with upgrades of some system DLLs, like comctl.dll, the library used to get file, font, color and printing dialog boxes. If things weren't bad enough with version clashes, if you wanted to uninstall an application, you could easily delete a DLL that was still being used by another program. Recognizing the problem, Microsoft incorporated the ability to track usage of DLLs with the Registry starting formally with Windows 95, and allowed only one version of a DLL to run in memory at a time. Adding yet another complication, when a new application was installed that used an existing DLL, it would increment a usage counter. On uninstall, the counter would be decremented and if no application was using the DLL, it could be deleted. That was, in theory. Over the history of Windows, the method of tracking of DLL usage was changed by Microsoft several times, as well as the problem of rogue installations that didn't play by the rules--the result was called "DLL HELL", and the user was the victim. Solving DLL hell is one thing that the .NET Framework and the CLR targeted. Under the .NET Framework, you can now have multiple versions of a DLL running concurrently. This allows developers to ship a version that works with their program and not worry about stepping on another program. The way .NET does this is to discontinue using the registry to tie DLLs to applications and by introducing the concept of an assembly. On the .NET Platform, if you want to install an application in the clients place all you have to do is use XCopy which copies all the necessary program files to a directory on the client's computer. And while uninstalling all you have to do is just deleting the directory containing the application and your application is uninstalled.

Metadata: An Assembly is a logical DLL and consists of one or more scripts, DLLs, or executables, and a manifest (a collection of metadata in XML format describing how assembly elements relate). Metadata stored within the Assembly, is Microsoft's solution to the registry problem. On the .NET Platform programs are compiled into .NET PE (Portable Executable) files. The header section of every .NET PE file contains a special new section for Metadata (This means Metadata for every PE files is contained within the PE file itself thus abolishing the need for any separate registry entries). Metadata is nothing but a description of every namespace, class, method, property etc. contained within the PE file. Through Metadata you can discover all the classes and their members contained within the PE file. Metadata describes every type and member defined in your code in a Multilanguage form. Metadata stores the following information:

- Description of the assembly
- Identity (name, version, culture, public key).
- The types that are exported.
- Other assemblies that this assembly depends on.
- Security permissions needed to run

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Description of types

- Name, visibility, base class, and interfaces implemented.
- Members (methods, fields, properties, events, nested types)

Attributes

- Additional descriptive elements that modify types and members

Advantages of Metadata:

Self describing files: CLR modules and assemblies are self-describing. Module's metadata contains everything needed to interact with another module. Metadata automatically provides the functionality of Interface Definition Language (IDL) in COM, allowing you to use one file for both definition and implementation. Runtime modules and assemblies do not even require registration with the operating system. As a result, the descriptions used by the runtime always reflect the actual code in your compiled file, which increases application reliability.

Language Interoperability and easier component-based design:

Metadata provides all the information required about compiled code for you to inherit a class from a PE file written in a different language. You can create an instance of any class written in\ any managed language (any language that targets the Common Language Runtime) without worrying about explicit marshaling or using custom interoperability code.

Attributes:

The .NET Framework allows you to declare specific kinds of metadata, called attributes, in your compiled file. Attributes can be found throughout the .NET Framework and are used to control in more detail how your program behaves at run time. Additionally, you can emit your own custom metadata into .NET Framework files through user-defined custom attributes.

Assembly

Assemblies are the building blocks of .NET Framework applications; they form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions. An assembly is a collection of types and resources that are built to work together and form a logical unit of functionality. An assembly provides the common language runtime with the information it needs to be aware of type implementations. To the runtime, a type does not exist outside the context of an assembly.

An assembly does the following functions:

- It contains the code that the runtime executes.
- It forms a security boundary. An assembly is the unit at which permissions are requested and granted.
- It forms a type boundary. Every type's identity includes the name of the assembly at which it resides.
- It forms a reference scope boundary. The assembly's manifest contains assembly metadata that is used for resolving types and satisfying resource requests. It specifies the types and resources that are exposed outside the assembly.
- It forms a version boundary. The assembly is the smallest version able unit in the common language runtime; all types and resources in the same assembly are versioned as a unit.
- It forms a deployment unit. When an application starts, only the assemblies the application initially calls must be present. Other assemblies, such as localization resources or assemblies containing utility classes, can be retrieved on demand. This allows applications to be kept simple and thin when first downloaded.
- It is a unit where side-by-side execution is supported.

Contents of an Assembly

• Assembly Manifest
• Assembly Name
• Version Information
• Types
• Locale
• Cryptographic Hash
• Security Permissions

Assembly Manifest

Every assembly, whether static or dynamic, contains a collection of data that describes how the elements in the assembly relate to each other. The assembly manifest contains this assembly metadata. An assembly manifest contains the following details:

• **Identity**. An assembly's identity consists of three parts: a name, a version number, and an optional culture.

• **File list**. A manifest includes a list of all files that make up the assembly.

•**Referenced assemblies**. Dependencies between assemblies are stored in the calling assembly's manifest. The dependency information includes a version number, which is used at run time to ensure that the correct version of the dependency is loaded.

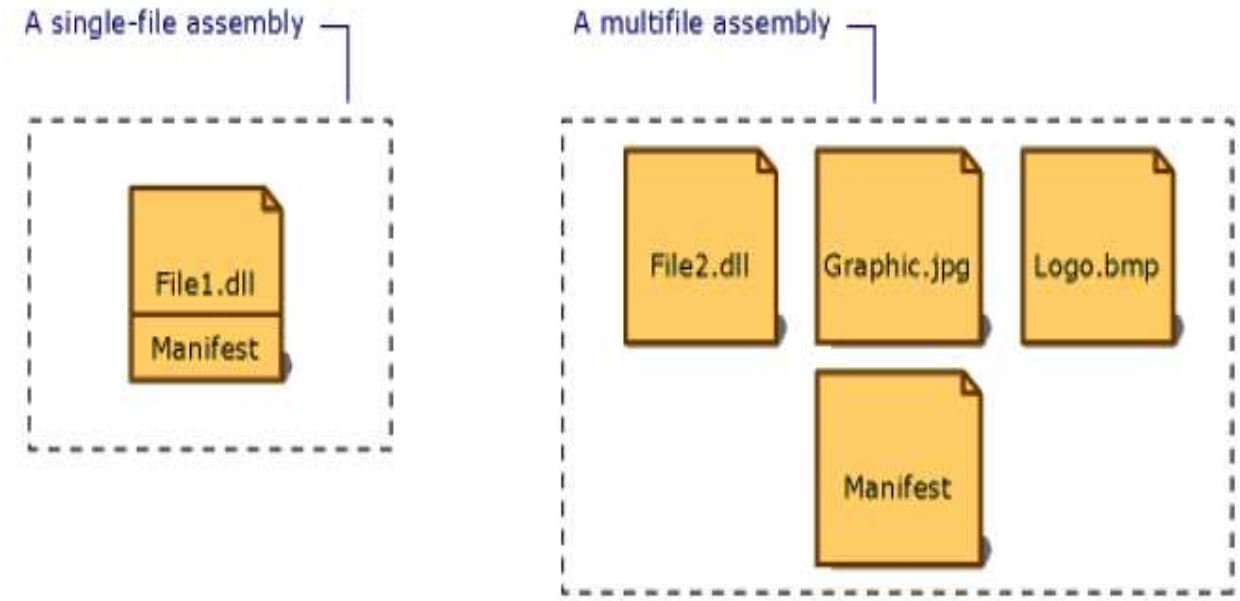
•**Exported types and resources**. The visibility options available to types and resources include "visible only within my assembly" and "visible to callers outside my assembly."

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

•**Permission requests**. The permission requests for an assembly are grouped into three sets:
1) Those required for the assembly to run,
2) Those that are desired but the assembly will still have some functionality even if they aren't granted, and
3) Those that the author never wants the assembly to be granted.

In general, if you have an application comprising of an assembly named Assem.exe and a module named Mod.dll. Then the assembly manifest stored within the PE Assem.exe will not only contain metadata about the classes, methods etc. contained within the Assem.exe file but it will also contain references to the classes, methods etc, exported in the Mod.dll file. While the module Mod.dll will only contain metadata describing itself.

The following diagram shows the different ways the manifest can be stored:



For an assembly with one associated file, the manifest is incorporated into the PE file to form a single-file assembly. You can create a multi file assembly with a standalone manifest file or with the manifest incorporated into one of the PE files in the assembly.

The Assembly Manifest performs the following functions:

- Enumerates the files that make up the assembly.
- Governs how references to the assembly's types and resources map to the files that contain their declarations and implementations.
- Enumerates other assemblies on which the assembly depends.
- Provides a level of indirection between consumers of the assembly and the assembly's implementation details.
- Renders the assembly self-describing.

Microsoft Intermediate Language (MSIL):

When compiling to managed code, the compiler translates your source code into Microsoft intermediate language (MSIL), which is a CPU-independent set of instructions that can be efficiently converted to native code. MSIL includes instructions for loading, storing, initializing, and calling methods on objects, as well as instructions for arithmetic and logical operations, control flow, direct memory access, exception handling, and other operations. Before code can be executed, MSIL must be converted to CPU-specific code by a just in time (JIT) compiler. Because the runtime supplies one or more JIT compilers, for each computer architecture it supports, the same set of MSIL can be JIT-compiled and executed on any supported architecture.

When a compiler produces MSIL, it also produces metadata. The MSIL and metadata are contained in a portable executable (PE file) that is based on and extends the published Microsoft PE and Common Object File Format (COFF) used historically for executable content. This file format, which accommodates MSIL or native code as well as metadata, enables the operating system to recognize common language runtime images. The presence of metadata in the file along with the MSIL enables your code to describe itself, which means that there is no

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

need for type libraries or Interface Definition Language (IDL). The runtime locates and extracts the metadata from the file as needed during execution.

Features of SQL-Server

The OLAP Services feature available in SQL Server version 7.0 is now called SQL Server 2000 Analysis Services. The term OLAP Services has been replaced with the term Analysis Services. Analysis Services also includes a new data mining component. The Repository component available in SQL Server version 7.0 is now called Microsoft SQL Server 2000 Meta Data Services. References to the component now use the term Meta Data Services. The term repository is used only in reference to the repository engine within Meta Data Services

SQL-SERVER database consist of six type of objects, They are,

1. TABLE
2. QUERY
3. FORM
4. REPORT
5. MACRO

Table: A database is a collection of data about a specific topic.

View of Table: We can work with a table in two types,

1. Design View
2. Datasheet View

Design View: To build or modify the structure of a table we work in the table design view. We can specify what kind of data will be hold.

Datasheet View: To add, edit or analyses the data itself we work in tables datasheet view mode.

Query: A query is a question that has to be asked the data. Access gathers data that answers the question from one or more table. The data that make up the answer is either dynaset (if you edit it) or a snapshot (it cannot be edited).Each time we run query, we get latest information in the dynaset. Access either displays the dynaset or snapshot for us to view or perform an action on it, such as deleting or updating.

Ajax: ASP.NET Ajax marks Microsoft's foray into the ever-growing Ajax framework market. Simply put, this new environment for building Web applications puts Ajax at the front and center of the .NET Framework.

Implementation

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

Modules

1. Firewall Splitting and Matching: In order to test the build time, data structure size and search speed behavior, we generated rule-bases of sizes from 1000 to 20000 and built the GEM data structure using two approaches: 2-part heuristic splitting and 3-part heuristic splitting, as described .it shows the data structure size of the unsplit, 2- part splitting, and 3-part splitting approaches it shows that both splitting heuristics are very effective in reducing the data structure size. In earlier simulations we verified that the firewall's matching speed is largely unaffected by the distribution of port numbers (both linear search and GEM). There is an extensive literature dealing with router packet matching, usually called "packet classification", Thus we believe that GEM may be a good candidate for use in firewall matching engines.

2. Encryption module: Allows trusted users to access sensitive information while traversing untrusted networks, it is highly useful for users. The services and users are limited in their tunnel traffic.

3. Protection and Detection mode: Easy testing of new rules in a live environment without disrupting the current security policy is supported. Rule sets are applied by deploying them in Protection mode to enforce secure behavior, permit or deny traffic and seal web application parameters against modification. Rule sets are tested by deploying them in Detection mode to evaluate them against traffic and log actions without enforcing them.

4. Random Rule Simulation module: On one hand, these early simulations showed us that the search itself was indeed very fast: a single packet match took around 1μsec, since it only required 4 executions of a binary search in memory. On the other hand, we learned that the data structure size grew rapidly—and that the order of fields had little or no effect on this size. The problem was that since the ranges in the rules were chosen uniformly, almost every pair of ranges (in every dimension) had a non-empty intersection. All these intersections produced a very fragmented space subdivision, and effectively exhibited the worst-case behavior in the data structure size. We concluded that a more realistic rule model is needed.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

System design

Introduction: Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software. The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

E – R Diagrams

- The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
- The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.
- The set of primary components that are identified by the ERD are

  Data object

  Relationships

  Attributes

  Various types of indicators

The primary purpose of the ERD is to represent data objects and their relationships.

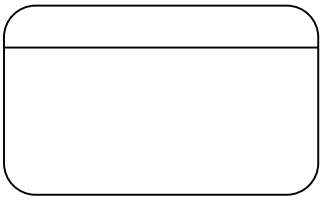
Data Flow Diagrams: A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD. The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process. Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design. A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD Symbols:

In the DFD, there are four symbols

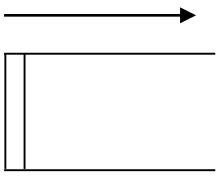
1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852



Process that transform the data flow

Source or Destination of data



Data flow

Data Store

Constructing a DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

Silent Features of DFD's

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

Types of Data Flow Diagrams

1. Current Physical
2. Current Logical
3. New Logical
4. New Physical

Current Physical:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

Current Logical: The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

New Logical:This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

New Physical: The new physical represents only the physical implementation of the new system.

Rules Governing the DFD's

Process

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.
3) A process has a verb phrase label.

Data Store

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3) A data store has a noun phrase label.

Source or Sink

The origin and /or destination of data.

1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

Data Flow

1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The later is usually indicated

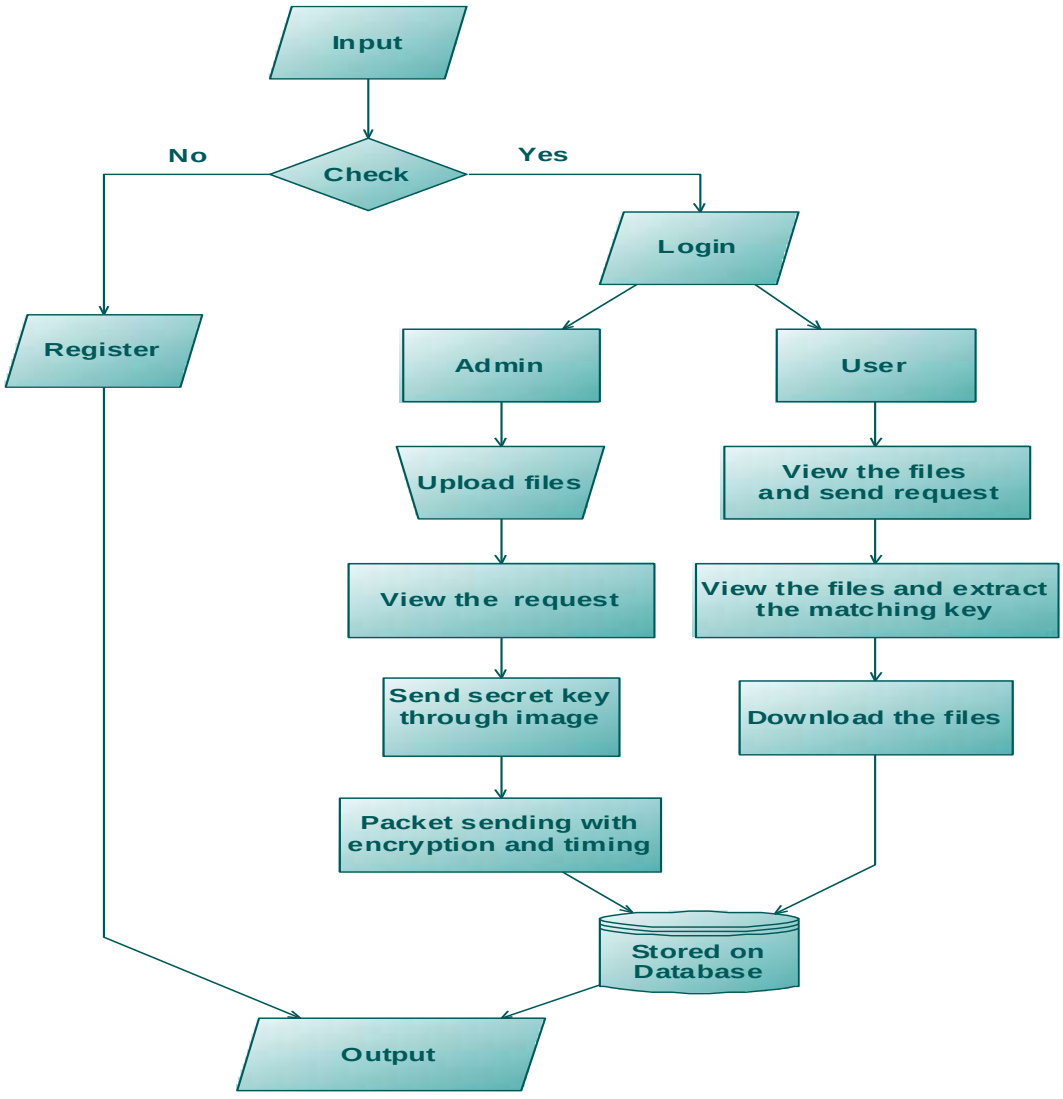
ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

however by two separate arrows since these happen at different type.

2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be atleast one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

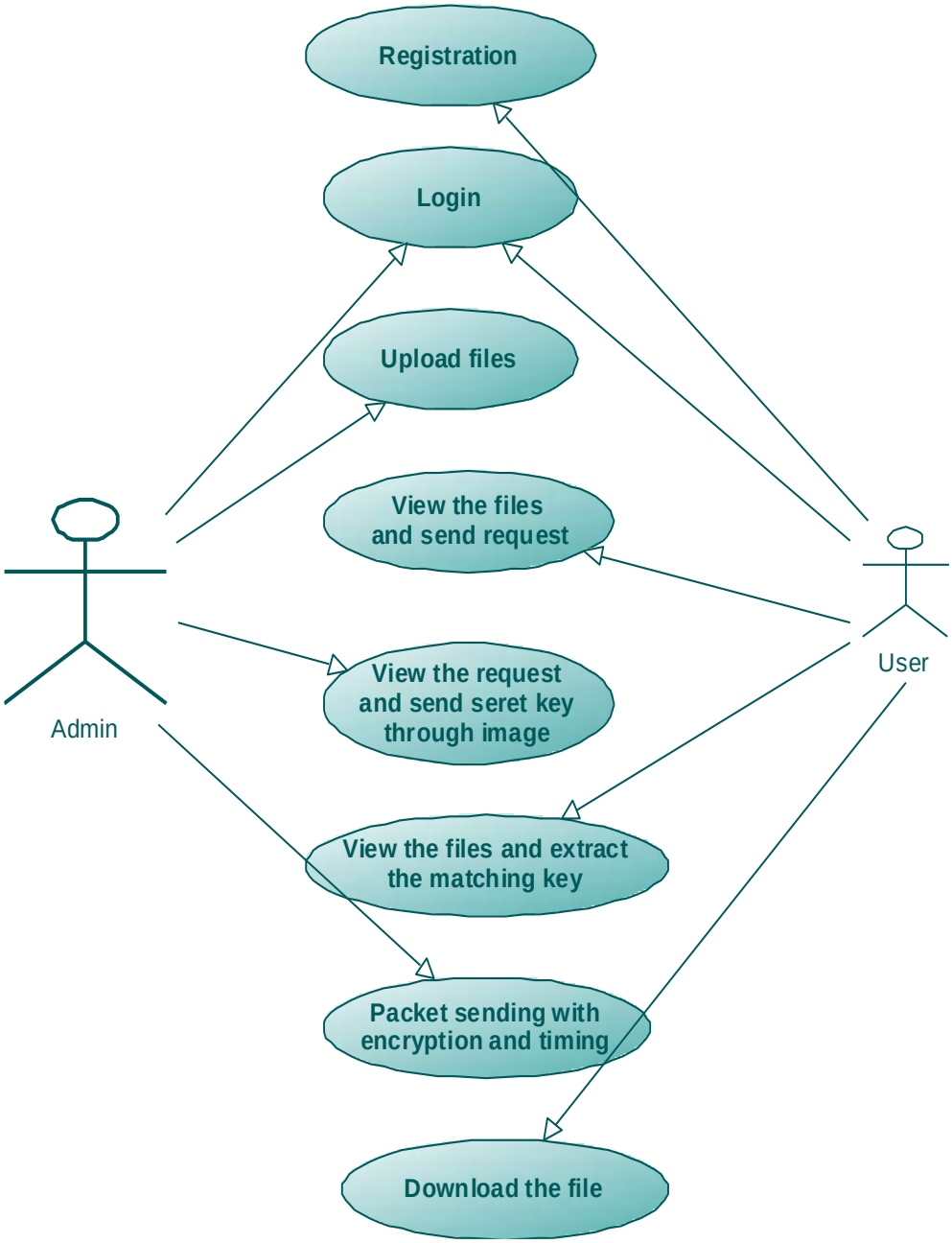
A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

Dataflow Diagram:



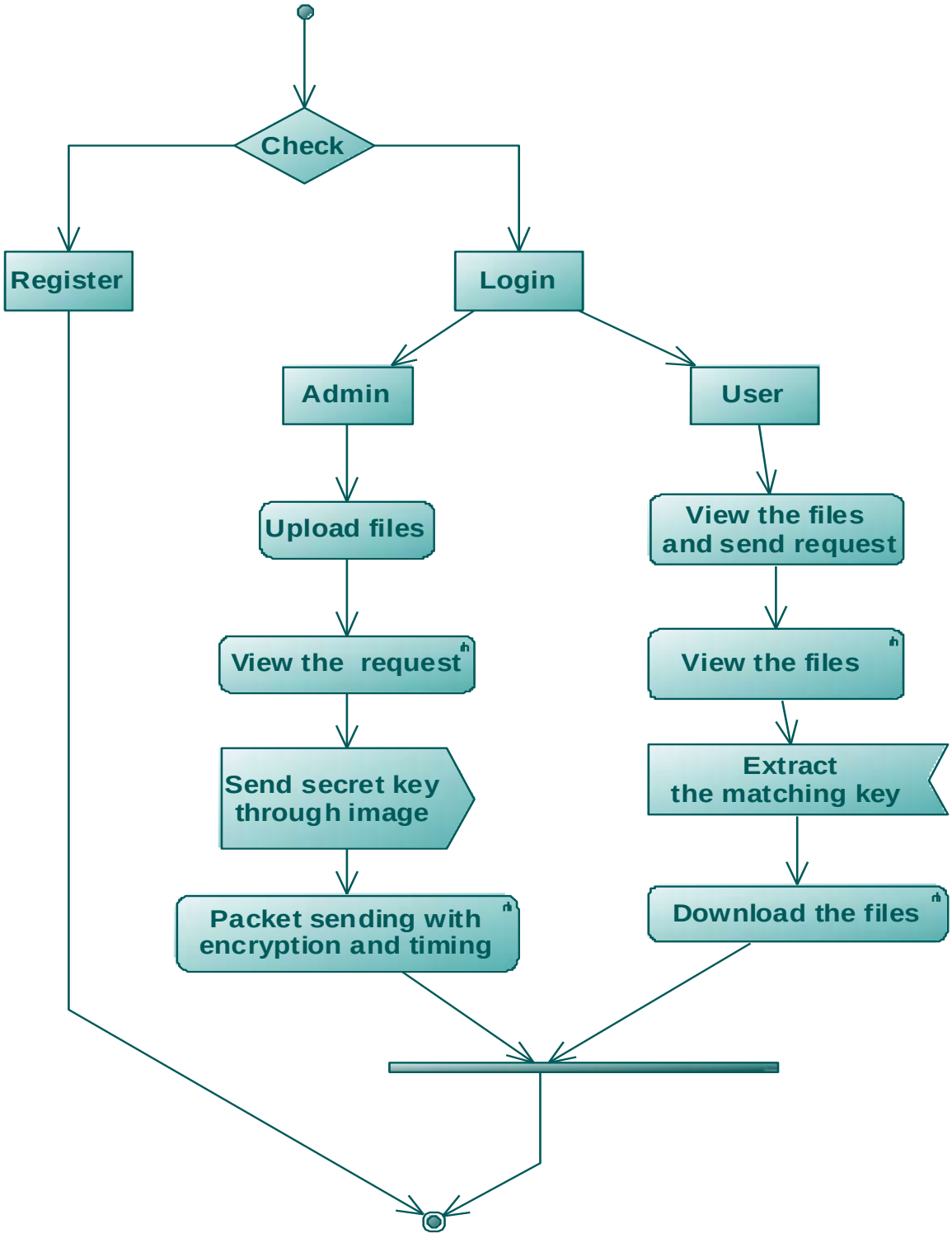
8.4 Uml Diagrams:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852



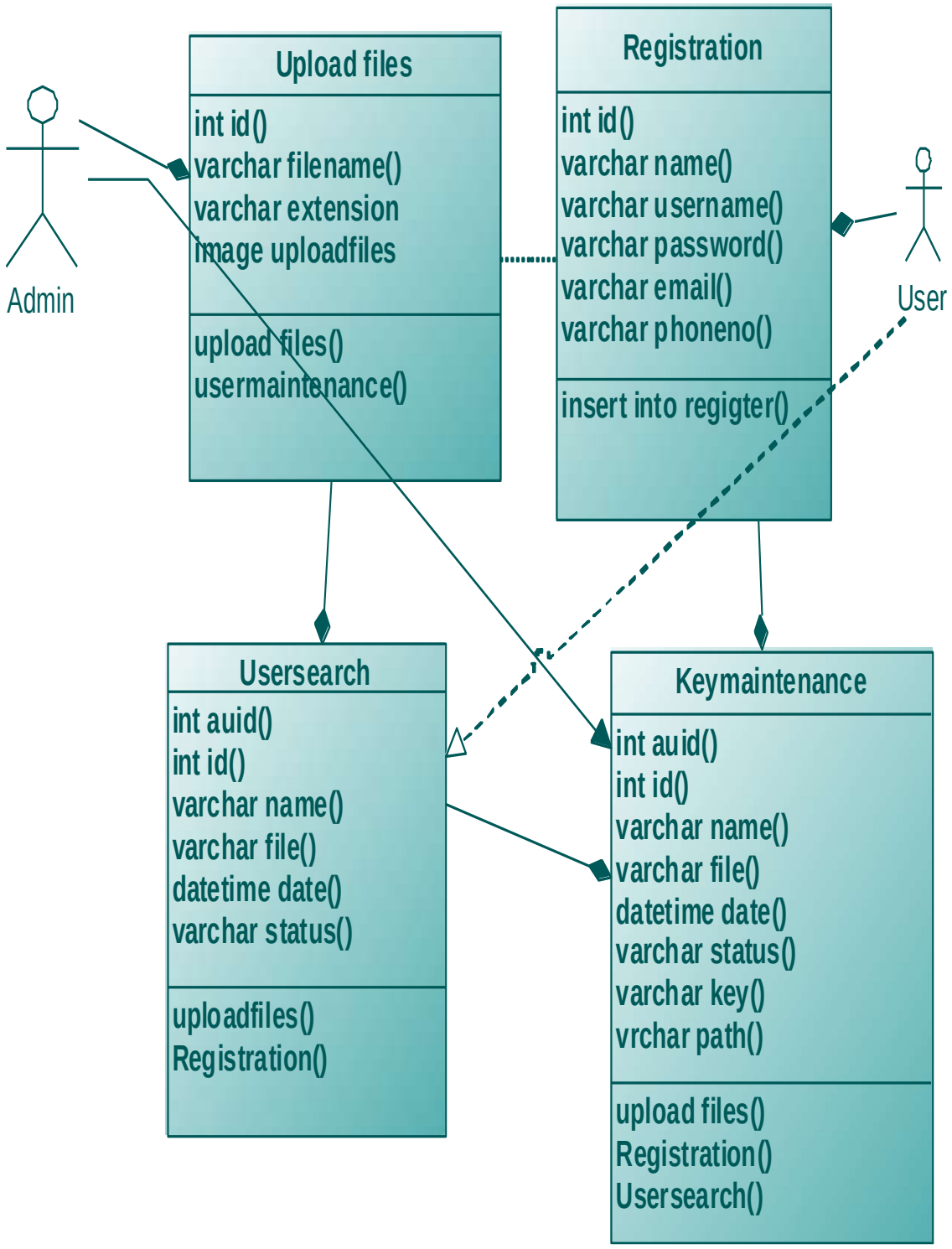
8.4.1. Use Case Daigram:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852



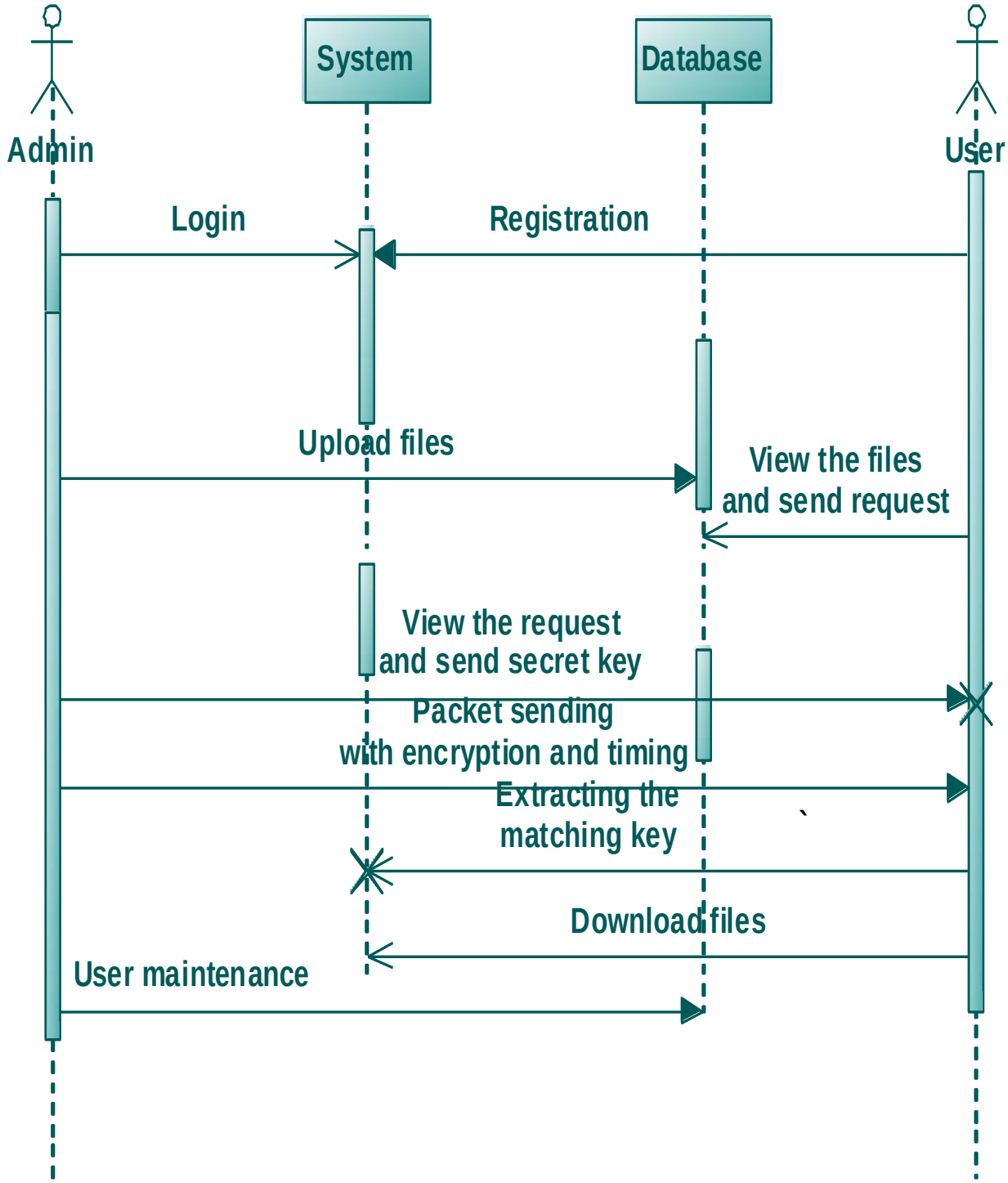
8.4.2 Activity Diagram:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852



8.4.3 Class Diagram:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852



8.4.4 Sequence Diagram:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

Input and output design

INPUT DESIGN: The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

Objectives: 1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

Output Design: A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

Functional Requirements:

Output Design: Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization.
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly with

Output Definition

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

For Example

- Will decimal points need to be inserted
- Should leading zeros be suppressed.

Output Media: In the next stage it is to be decided that which medium is the most appropriate for the output. The main considerations when deciding about the output media are:

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

- The suitability for the device to the particular application.
- The need for a hard copy.
- The response time required.
- The location of the users
- The software and hardware available.

Keeping in view the above description the project is to have outputs mainly coming under the category of internal outputs. The main outputs desired according to the requirement specification are**:** The outputs were needed to be generated as a hot copy and as well as queries to be viewed on the screen. Keeping in view these outputs, the format for the output is taken from the outputs, which are currently being obtained after manual processing. The standard printer is to be used as output media for hard copies.

Input Design :Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

Input Stages: The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

Input Types: It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

Input Media: At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

Error Avoidance: At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded upto the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

Error Detection: Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data. **Data Validation:** Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary. The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with pop up menus.

User Interface Design: It is essential to consult the system users and discuss their needs while designing the user interface:

User Interface Systems Can Be Broadly Classified As:

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

User Initiated Interfaces: User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

Computer-Initiated Interfaces: The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

Error Message Design: The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed. This application must be able to produce output at different modules for different inputs.

Performance Requirements: Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

System testing

SYSTEM 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.

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.

Functional Test:

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 centered 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.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

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

System Test:

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.

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.

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.

Unit Testing: Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

10.3 Integration Testing: Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing: User *Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.*

Test Results:

All the test cases mentioned above passed Successfully. No defects encountered.

Conclusion

We have seen that the GEM algorithm is an efficient and practical algorithm for firewall packet matching. We implemented it successfully, and tested its packet-matching speeds. GEM's matching speed is far better than the naive linear search, and it is able to increase the throughput by an order of magnitude. On rule-bases generated according to realistic statistics, GEM's space complexity is well within the capabilities of modern hardware. Thus we believe that GEM may be a good candidate for use in firewall matching engines.

References

1. User Interfaces in C#: Windows Forms and Custom Controls by Matthew MacDonald.
2. Applied Microsoft® .NET Framework Programming (Pro-Developer) by Jeffrey Richter.
3. Practical .Net2 and C#2: Harness the Platform, the Language, and the Framework by Patrick Smacchia.
4. Data Communications and Networking, by Behrouz A Forouzan.
5. Computer Networking: A Top-Down Approach, by James F. Kurose.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

6. Operating System Concepts, by Abraham Silberschatz.
7. Amichai-Hamburger, Y., Fine, A., & Goldstein, A. (2004). The impact of Internet interactivity and need for closure on consumer preference. Computers in Human Behavior, 20, 103-117.
8. Balabanis, G., Reynolds, N., & Simintiras, A. (2006). Bases of e-store loyalty: Perceived switching barriers and satisfaction. Journal of Business Research, 59, 214-224.
9. F. Baboescu, S. Singh, and G. Varghese, "Packet classification for core routers: Is there an alternative to cams," in Proc. IEEE INFOCOM, 2003
10. F. Baboescu and G. Varghese, "Scalable packet classification," in Proc. ACM SIGCOMM, 2001, pp. 199–210.
11. N. Bar-Yosef and A. Wool, "Remote algorithmic complexity attacks against randomized hash tables," in Proc. International Conference on Security and Cryptography (SECRYPT), Barcelona, Spain, Jul. 2007, pp. 117–124.
12. M. M. Buddhikot, S. Suri, and M. Waldvogel, "Space decomposition techniques for fast Layer-4 switching," in Protocols for High Speed Networks IV, Aug. 1999, pp. 25–41.
13. W. R. Cheswick, S. M. Bellovin, and A. Rubin, Firewalls and Internet Security: Repelling the Wily Hacker, 2nd ed. Addison-Wesley, 2003.
14. M. Christiansen and E. Fleury, "Using interval decision diagrams for packet filtering," 2002, http://www.cs.auc.dk/_fleury/publications.html.
15. E. Cohen and C. Lund, "Packet classification in large ISPs: Design and evaluation of decision tree classifiers," in Proc. ACM SIGMETRICS. New York, NY, USA: ACM Press, 2005, pp. 73–84.
16. S. Crosby and D. Wallach, "Denial of service via algorithmic complexity attacks," in Proceedings of the 12th USENIX Security Symposium, August 2003, pp. 29–44.
17. M. de Berg, M. van Kreveld, and M. Overmars, Computational Geometry: Algorithms and Applications, 2nd ed. Springer-Verlag, 2000.