ITS Malisious Packet Loss

Yuvarani N
M.sc., M.Phil.,(Asst.professor)
M.Phil.(CS), Rathinam college of arts & science
Eachanari, Bharathiar university.

Abstract - We consider the problem of detecting whether a compromised router is maliciously manipulating its stream of packets. In particular, we are concerned with a simple yet effective attack in which a router selectively drops packets destined for some Victim. Unfortunately, it is quite challenging to attribute a missing packet to a malicious action because normal network congestion can produce the same effect. Modern networks routinely drop packets when the load temporarily exceeds their buffering capacities. Previous detection protocols have tried to address this problem with a user-defined threshold: too many dropped packets imply malicious intent. However, this heuristic is fundamentally unsound; setting this threshold is, at best, an art and will certainly create unnecessary false positives or mask highly focused attacks.

I. INTRODUCTION

The Internet is not a safe place. Unsecured hosts canexpect to be compromised within minutes of connectingto the Internet and even well-protected hosts may be crippled with denial-of-service (DoS) attacks. However,while such threats to host systems are widely understood, itis less well appreciated that the network infrastructure itselfis subject to constant attack as well. Indeed, throughcombinations of social engineering and weak passwords, attackers have seized control over thousands of Internet routers . Even more troubling is Mike Lynn's controversial presentation at the 2005 Black Hat Briefings,which demonstrated how Cisco routers can be compromised via simple software vulnerabilities. Once a router hasbeen compromised in such a fashion, an attacker mayinterpose on the traffic stream and manipulate it maliciously to attack others—selectively dropping, modifying,or rerouting packets.

Several researchers have developed distributed protocols to detect such traffic manipulations, typically byvalidating that traffic transmitted by one router is received unmodified by another. However, all of theseschemes—including our own—struggle in interpreting theabsence of traffic. While a packet that has been modified in transit represents clear evidence of tampering, a missingpacket is inherently ambiguous: it may have been explicitly blocked by a compromised router or it mayhave been dropped benignly due to network congestion.In fact, modern routers routinely drop packets due to bursts in traffic that exceed their buffering capacities, andthe widely used Transmission Control Protocol (TCP) is designed to cause such losses as part of its normalcongestion control behavior. Thus, existing traffic validationsystems must inevitably produce false positives for benign events and/or produce false negatives by failing toreport real malicious packet dropping.

In this paper, we develop a compromised routerdetection protocol that dynamically infers the precise number of congestive packet losses that will occur. Oncethe congestion ambiguity is removed, subsequent packetlosses can be safely attributed to malicious actions. Webelieve our protocol is the first to automatically predictcongestion in a systematic manner and that it is necessaryfor making any such network fault detection practical.In the remainder of this paper, we briefly survey therelated background material, evaluate options for inferringcongestion, and then present the assumptions, specification, and a formal description of a protocol that achieves thesegoals. We have evaluated our protocol in a small experimentalnetwork and demonstrate that it is capable ofaccurately resolving extremely small and fine-grainedattacks.

II. SYSTEM STUDY

2.1 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

2.2 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.

2.3 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.

2.4 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.

III. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Network routers occupy a unique role in modern distributed systems. They are responsible for cooperatively shuttling packets amongst themselves in order to provide the illusion of a network with universal point-to-point connectivity. However, this illusion is shattered - as are implicit assumptions of availability, confidentiality, or integrity - when network routers are subverted to act in a malicious fashion. By manipulating, diverting, or dropping packets arriving at a compromised router, an attacker can trivially mount denial-of-service, surveillance, or man-in-the-middle attacks on end host systems.

Consequently, Internet routers have become a choice target for would-be attackers and thousands have been subverted to these ends. In this paper, we specify this problem of detecting routers with incorrect packet forwarding behavior and we explore the design space of protocols that implement such a detector. We further present a concrete protocol that is likely inexpensive enough for practical implementation at scale. Finally, we present a prototype system, called Fatih, that implements this approach on a PC router and describe our experiences with it. We show that Fatih is able to detect and isolate a range of malicious router actions with acceptable overhead and complexity. We believe our work is an important step in being able to tolerate attacks on key network infrastructure components.

3.2 PROPOSED SYSTEM:

We have designed, developed, and implemented a compromised router detection protocol that dynamically infers, based on measured traffic rates and buffer sizes, the number of congestive packet losses that will occur.

Once the ambiguity from congestion is removed, subsequent packet losses can be attributed to malicious actions. We have tested our protocol in Emulab and have studied its effectiveness in differentiating attacks from legitimate network behavior.

IV. SOFTWARE ENVIRONMENT

4.1 Java Technology

Java technology is both a programming language and a platform.

4.2The 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
- Portable
- Distributed

- High performance
- Interpreted
- Multithreaded
- Robust
- 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.

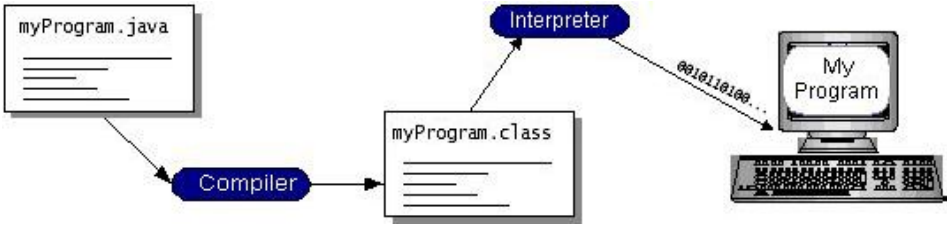


Fig 4.2.1:working process of JAVA

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.

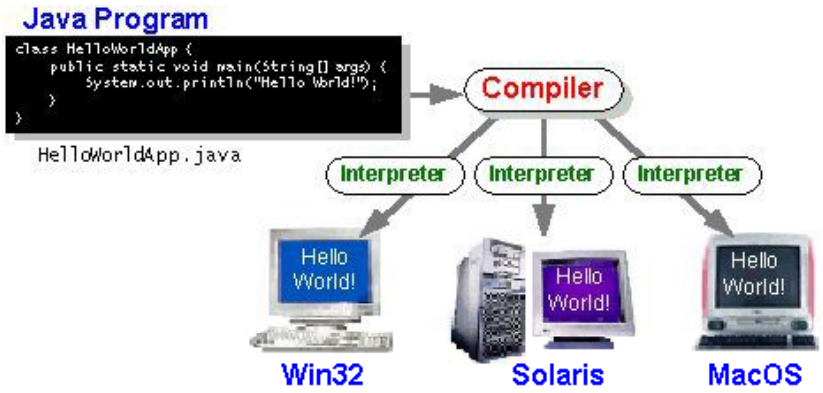


Fig 4.2.2:How to run JAVA

4.3 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.

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, servlets 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.

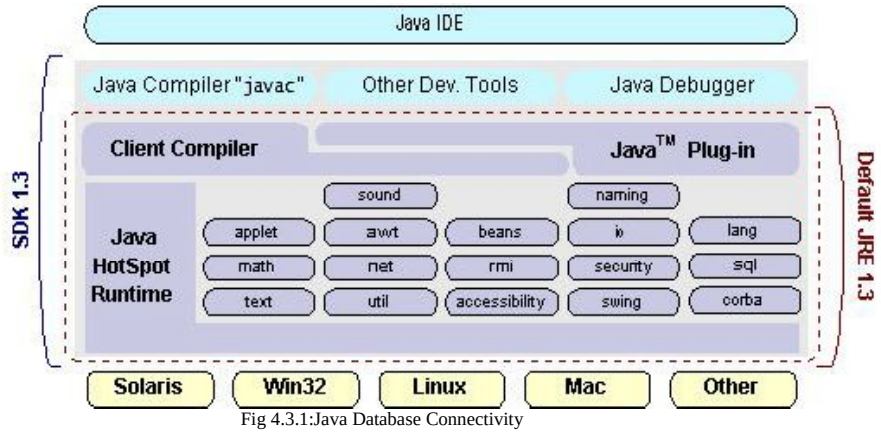


Fig 4.3.1:Java Database Connectivity

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.

4.4 Networking

TCP/IP stack

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

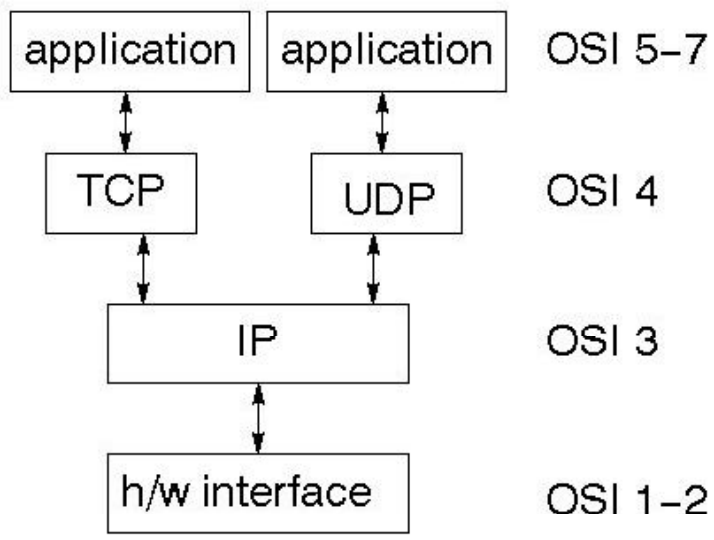


Fig 4.4.1:Format of TCP/IP

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.

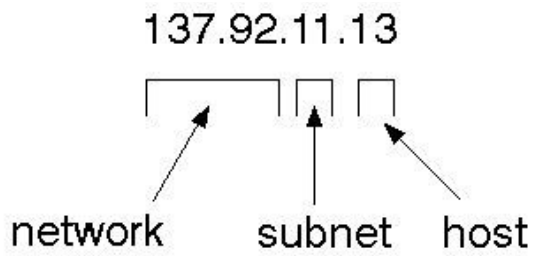
Subnet address

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.

V. 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.

5.1 MODULES

1. Network Model
2. Threat Model
3. Distributed Detection
4. Traffic Validation Correctness

5.2 Module Description

5.2.1 Network Model

We consider a network to consist of individual homogeneous routers interconnected via directional point-to point links. This model is an intentional simplification of real networks (e.g., it does not include broadcast channels or independently failing network interfaces) but is sufficiently general to encompass such details if necessary. Unlike our earlier work, we assume that the bandwidth, the delay of each link, and the queue limit for each interface are all known publicly.

5.2.2 Threat Model

As explained in Section 1, this paper focuses solely on data plane attacks (control plane attacks can be addressed by other protocols with appropriate threat models such as, Moreover, for simplicity, we examine only attacks that involve packet dropping. However, our approach is easily extended to address other attacks such as packet modification or reordering similar to our previous work. The protocol we develop validates traffic whose source and sink routers are uncompromised. A router can be traffic faulty by maliciously dropping packets and protocol faulty by not following the rules of the detection protocol. We say that a compromised router r is traffic faulty with respect to a path segment during. If contains r and, during the period of time, r maliciously drops or

misroutes packets that flow through. A router can drop packets without being faulty, as long as the packets are dropped because the corresponding output interface is congested.

5.2.3 Distributed Detection

Since the behavior of the queue is deterministic, the traffic validation mechanisms detect traffic faulty routers whenever the actual behavior of the queue deviates from the predicted behavior. However, a faulty router can also be protocol faulty; it can behave arbitrarily with respect to the protocol,
by dropping or altering the control messages , We mask the effect of protocol faulty routers using distributed detection.

5.2.4Traffic Validation Correctness

Any failure of detecting malicious attack by TV results in a false negative, and any misdetection of legitimate behavior by TV results in a false positive.

Within the given system model, the example TV predicate in Section 5.1 is correct. However, the system model is still simplistic. In a real router, packets may be legitimately dropped due to reasons other than congestion: for example, errors in hardware, software or memory, and transient link errors. Classifying these as arising from a router being compromised might be a problem, especially if they are infrequent enough that they would be best ignored rather than warranting repairs the router or link.

VI. 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.

6.1 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.
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.

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.

6.2 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.

6.3 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.

6.4 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.

VII. SYSTEM DESIGN

DATA FLOW DIAGRAM:

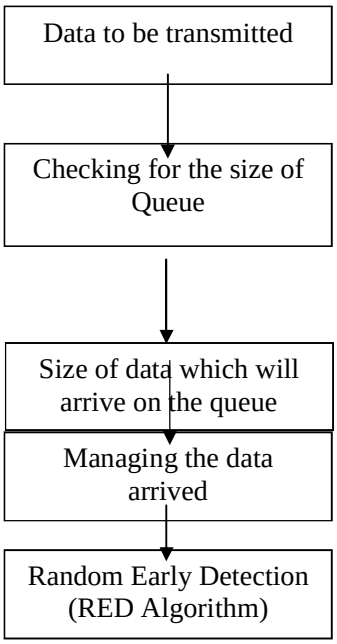


Fig 7.1 data flow diagram

ARCHITECTURE DIAGRAM:

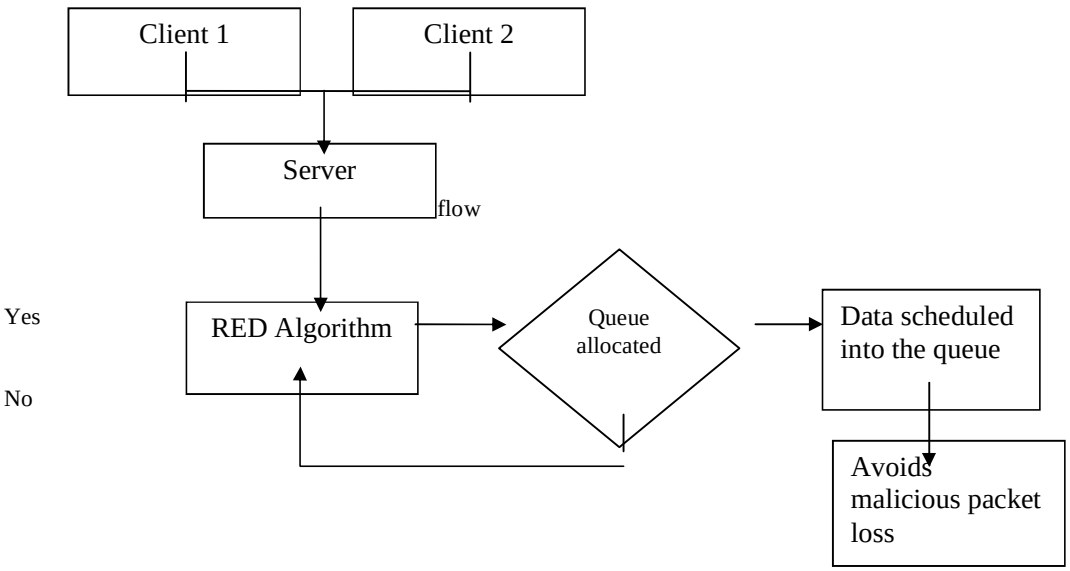


Fig 7.2 architecture diagram

USE CASE DIAGRAM:

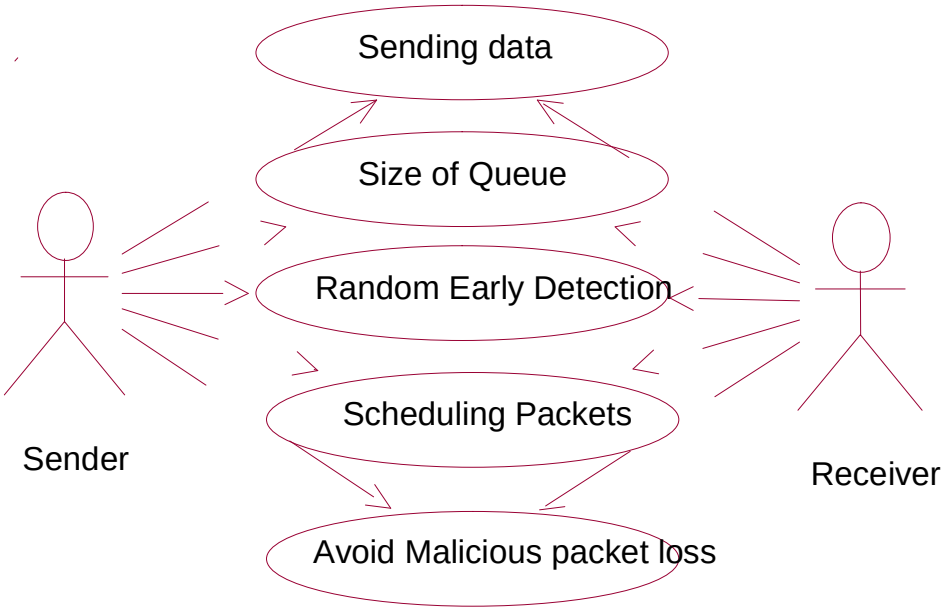


Fig 7.3 use case diagram

SEQUENCE DIAGRAM:

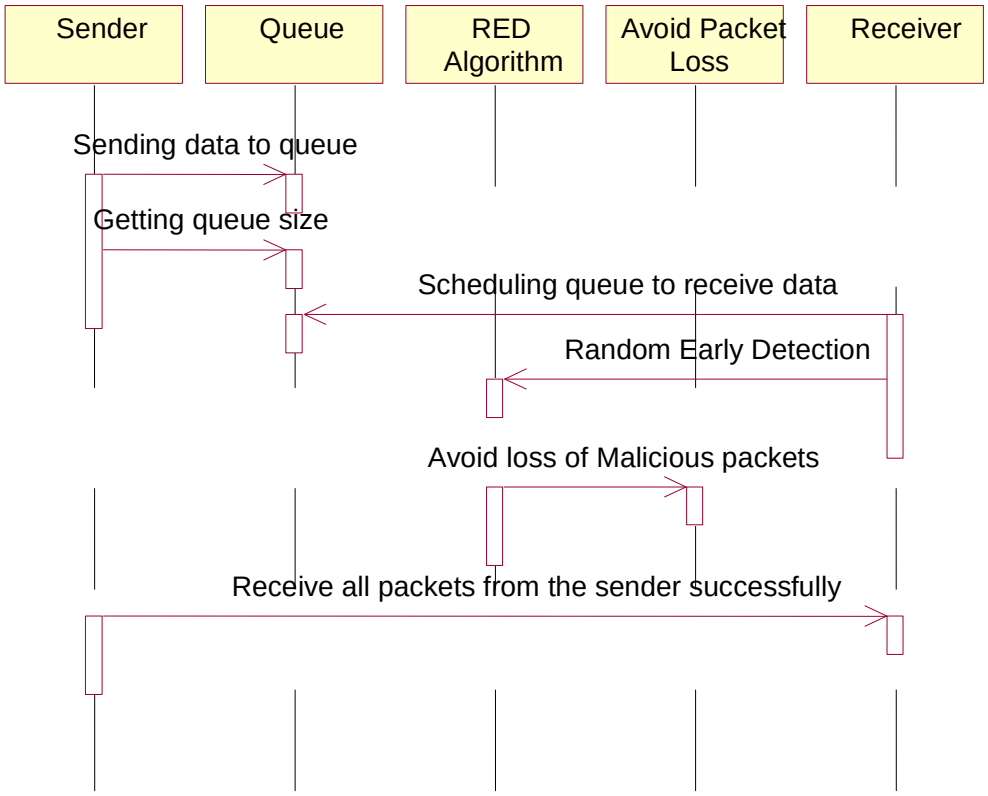


Fig 7.4 sequence diagram

COMPONENT DIAGRAM:

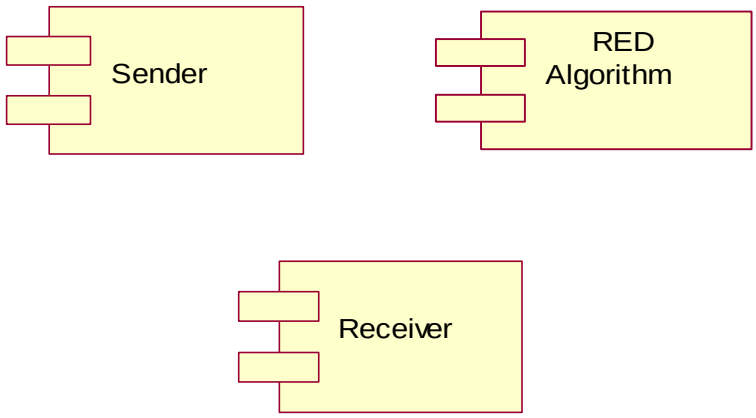


Fig 7.5 component diagram

COLLABORATION DIAGRAM:

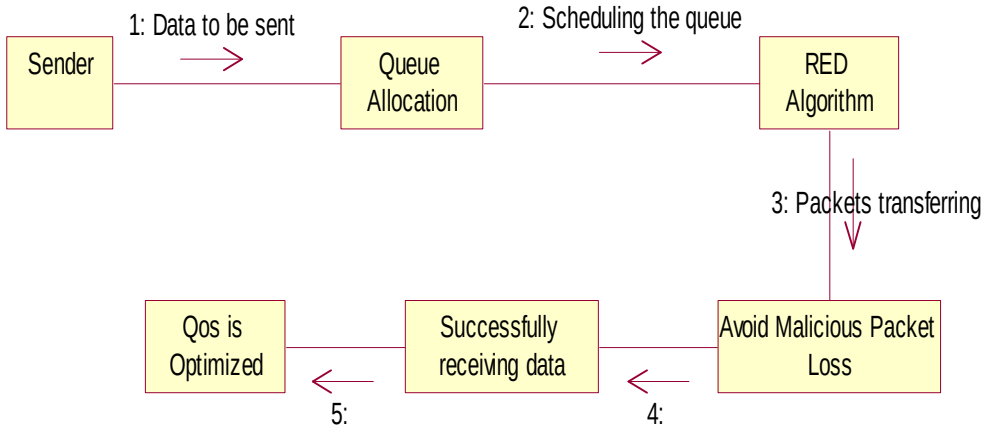


Fig 7.6 collaboration diagram

VIII. SYSTEM REQUIREMENT

8.1 HARDWARE REQUIREMENT

- System : Pentium IV 2.4 GHz.
- Hard Disk : 40 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 256 Mb.

8.2 SOFTWARE REQUIREMENT

- Operating system : - Windows XP Professional.
- Coding Language : - Java5.
- Tool Used : - Eclipse.

IX. CONCLUSION

We study spatial query processing in wireless broadcast environments. A central server transmits the data along with some indexing information. The clients process their queries locally, by accessing the broadcast channel. In this setting, our target is to reduce the power consumption and access latency at the client side. We propose an on-air indexing method that uses a regular grid to store and transmit the data objects. We design algorithms for snapshot and continuous queries, over static or dynamic data. To the best of our knowledge, this is the first study on air indexing that

1) Addresses continuous queries and
2) Considers moving data objects. We demonstrate the efficiency of our algorithm through an extensive experimental comparison with the current state-of-the-art frameworks for snapshot queries and with the constant re-computation technique for continuous queries. A challenging problem is to devise cost models for continuous monitoring of spatial queries in wireless broadcast environments. Such models could reveal the best technique given the problem settings, help fine-tune several system parameters (e.g., grid size), and potentially lead to better algorithms. Another interesting direction for future work is to study different types of spatial queries, such as reverse nearest neighbors, and to extend our framework to process their snapshot and continuous versions.

REFERENCES

[1] N. Shah, Understanding Network Processors. Master's thesis, Univ. of California, Sept. 2001.
[2] W. Feghali, B. Burres, G. Wolrich, and D. Carrigan, "Security: Adding Protection to the Network via the Network Processor,"Intel Technology J., vol. 6, pp. 40-49, Aug. 2002.
[3] L.A. Sanchez, W.C. Milliken, A.C. Snoeren, F. Tchakountio, C.E. Jones, S.T. Kent, C. Partridge, and W.T. Strayer "Hardware Support for a Hash-Based IP Traceback," Proc.Second DARPA Information Survivability Conf. and Exposition(DISCEX II '01), pp. 146-152, 2001.
[4] D.L. Mills, Network Time Protocol (Version 3) Specification, Implementation, RFC 1305, IETF, Mar. 1992.
[5] H.F. Wedde, J.A. Lind, and G. Segbert, "Achieving Internal Synchronization Accuracy of 30 ms under Message DelaysVarying More than 3 msec," Proc. 24th IFAC/IFIP Workshop Real-Time Programming (WRTP), 1999.
[6] B. White et al., "An Integrated Experimental Environment for Distributed Systems and Networks," Proc. Fifth Symp. OperatingSystem Design and Implementation (OSDI '02), pp. 255-270,Dec. 2002.
[7] Emulab—Network Emulation Testbed, http://www.emulab.net,2006.
[8] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," IEEE/ACM Trans. Networking(TON '93), vol. 1, no. 4, pp. 397-413, 1993.
[9] C.V. Hollot, V. Misra, D.F. Towsley, and W. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows," Proc. INFOCOM '01, pp. 1726-1734, Apr. 2001.
[10] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active Queue Management," IEEE Network, vol. 15, no. 3, pp. 48-53, 2001.
[11] S. Floyd, "TCP and Explicit Congestion Notification," ACM Computer Comm. Rev., vol. 24, no. 5, pp. 10-23, 1994.