**1. INTRODUCTION**

This paper investigates the problem of finding optimal paths in single-source single-destination accumulative multihop networks. We consider a single source that communicates to a single destination assisted by several relays through multiple hops. At each hop, only one node transmits, while all the other nodes receive the transmitted signal, and store it after processing/decoding and mixing it with the signals received in previous hops. That is, we consider that terminals make use of advanced energy accumulation transmission/reception techniques, such as maximal ratio combining reception of repetition codes, or information accumulation with rateless codes. Accumulative techniques increase communication reliability, reduce energy consumption, and decrease latency. We investigate the properties that a routing metric must satisfy in these accumulative networks to guarantee that optimal paths can be computed with Dijkstra’s algorithm. We model the problem of routing in accumulative multi-hop networks, as the problem of routing in a hypergraph. We show that optimality properties in a traditional multi-hop network (monotonicity and isotonicity) are no longer useful and derive a new set of sufficient conditions for optimality. We illustrate these results by studying the minimum energy routing problem in static accumulative multi-hop networks for different forwarding strategies at relays.

* 1. **ORGANIZATION PROFILE**

**1.2 SYSTEM SPECIFICATION**

**1.2.1 HARDWARE CONFIGURATION**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* Ram : 4 GB.

**1.1.2 SOFTWARE SPECIFICATION**

* Operating System : Windows 7
* Front End : PHP
* Back End : MY-SQL
* Document : MS-Office

**1.1.3 SOFTWARE DESCRIPTION**

**About PHP:**

**PHP** is a general-purpose server-side scripting language originally designed for web development to produce dynamic web pages. It is among one of the first developed server-side scripting languages that is embedded into a HTML source document, rather than calling an external file to process data. Ultimately, the code is interpreted by a web server with a PHP processor module which generates the resulting web page. It also has evolved to include a command-line interface capability and can be used in standalone graphical applications. PHP can be deployed on most web servers and also as a standalone shell on almost every operating system and platform free of charge. A competitor to Microsoft's Active Server Pages (ASP) server-side script engine and similar languages, PHP is installed on more than 20 million websites and 1 million web servers.

## PHP stands for PHP: Hypertext Pre-processor.

* As we said before it is a server-side scripting language.
* [PHP is free](http://php.net/index.php) and is an open source software product.
* The PHP scripts are executed on the server.
* PHP supports many databases (MySQL, Sybase, Oracle and many others.)
* PHP runs on different platforms (Unix, Linux, Windows.)
* PHP is compatible with almost all web-servers used today (Apache, IIS, etc.)
* A PHP file can contain plain text, HTML tags and scripts.
* The PHP files can have one of the following extensions: php, php3 or phtml.

## What do you need?

Before you can follow the coming PHP tutorials you need to have the following:

* Access to a web server (like Apache)
* PHP and MySQL should be installed on the web server.

The easiest way is to find a cheap web hosting plan that support PHP and MySQL (most of the web-hosting providers do.) For a couple of bucks a month you can get a virtual host. The advantage is that everything is already installed and that you can share your program (that you are going to write) with the rest of the world.

If you don’t want to spend money, you can also install everything on your local machine. The easiest way is to install WAMP if you have a Windows machine and LAMP if you have a Linux machine. ([WAMP](http://www.wampserver.com/en/) = Windows Apache MySQL PHP and LAMP = Linux Apache MySQL PHP.) These packages install everything you need. Read the install manual of these packages for instructions. After installation you should have access to http://localhost in your browser.

That is all for this PHP language introduction tutorial. We hope that you will learn something from the coming PHP tutorials and as always we hope that you will let us know if something is wrong by leaving us a comment.

Before we can start coding you first need to know some basics. In this PHP language introduction tutorial you’ll find some descriptions of words you should know before you can start coding. Of course we will also look at where you can download PHP, MySQL and Apache.

PHP is probably the most popular scripting language on the web. It is used to enhance web pages. With PHP, you can do things like create username and password login pages, check details from a form, create forums, picture galleries, surveys, and a whole lot more. If you've come across a web page that ends in PHP, then the author has written some programming code to liven up the plain, old HTML.

PHP is known as a server-sided language. That's because the PHP doesn't get executed on your computer, but on the computer you requested the page from. The results are then handed over to you, and displayed in your browser. Other scripting languages you may have heard of are ASP, Python and Perl. (You don't need to know any of these to make a start on PHP. In fact, these tutorials assume that you have no programming experience at all.)

The most popular explanation of just what PHP stands for is "Hypertext Pre-processor". But that would make it HPP, surely? An alternative explanation is that the initials come from the earliest version of the program, which was called Personal Home Page Tools. At least you get the letters "PHP" in the right order!

But PHP is so popular that if you're looking for a career in the web design/web scripting industry then you just have to know it! In these tutorials, we'll get you up and running. And, hopefully, it will be a lot easier than you think.

**Course Files**

To follow along with these tutorials, you'll need our Home and Learn Course files. These can be downloaded from the following address:

The PHP programming language is a server-side HTML embedded scripting language. Let‘s depict the sentence. The PHP language runs on the server-side. This means that the execution (read starting) of the scripts are done on the server where the web-site is hosted. HTML embedded means that you can use PHP statements (read a piece of PHP code) from within an HTML code. PHP files are returned to the browser as plain HTML.

The last piece of the sentence – scripting language – is a little harder to explain, but we will give it a go. A scripting language is a form of programming language that is usually interpreted rather than compiled. In [**programming languages such C**](http://www.codingunit.com/category/c-tutorials) or [**C++**](http://www.codingunit.com/category/cplusplus-tutorials) you compile the program (permanently) into an executable file, before you can execute the program. A program that is written in a scripting language, is interpreted one command at a time by a command interpreter (Command interpreter is in most cases an executable written in another language (for instance C/C++) than the scripting language.) Some other examples of scripting languages are Perl, Phyton, Java and Ruby.

**A last note:** scripting languages are also often used by applications as control or configure language. An example: Firefox is written in C/C++ and can be controlled with JavaScript.

## Things You Should Know

Because PHP is an HTML embedded language you should have a basic understanding of HTML/XHTML and (a little) JavaScript. In the coming tutorials we will explain the PHP code, but we won’t explain the HTML code (at least not in great detail.)

**Architecture of PHP**

The mysqlnd replication and load balancing plugin is implemented as a PHP extension. It is written in C and operates under the hood of PHP. During the startup of the PHP interpreter, in the module init phase of the PHP engine, it gets registered as a mysqlnd plugin to replace selected mysqlnd C methods.

At PHP runtime, it inspects queries sent from mysqlnd (PHP) to the MySQL server. If a query is recognized as read-only, it will be sent to one of the configured slave servers. Statements are considered read-only if they either start with SELECT, the SQL hint /\*ms=slave\*/ or a slave had been chosen for running the previous query, and the query started with the SQL hint /\*ms=last\_used\*/. In all other cases, the query will be sent to the MySQL replication master server.

For better portability, applications should use the MYSQLND\_MS\_MASTER\_SWITCH, MYSQLND\_MS\_SLAVE\_SWITCH, and MYSQLND\_MS\_LAST\_USED\_SWITCH predefined mysqlnd\_ms constants, instead of their literal values, such as /\*ms=slave\*/.

The plugin handles the opening and closing of database connections to both master and slave servers. From an application point of view, there continues to be only one connection handle. However, internally, this one public connection handle represents a pool of network connections that are managed by the plugin. The plugin proxies queries to the master server, and to the slaves using multiple connections.

Database connections have a state consisting of, for example, transaction status, transaction settings, character set settings, and temporary tables. The plugin will try to maintain the same state among all internal connections, whenever this can be done in an automatic and transparent way. In cases where it is not easily possible to maintain state among all connections, such as when using BEGIN TRANSACTION, the plugin leaves it to the user to handle.

**History of PHP**

PHP Tools, FI, Construction Kit, and PHP/FI

PHP as it's known today is actually the successor to a product named PHP/FI. Created in 1994 by Rasmus Lerdorf, the very first incarnation of PHP was a simple set of Common Gateway Interface (CGI) binaries written in the C programming language. Originally used for tracking visits to his online resume, he named the suite of scripts "Personal Home Page Tools," more frequently referenced as "PHP Tools." Over time, more functionality was desired, and Rasmus rewrote PHP Tools, producing a much larger and richer implementation. This new model was capable of database interaction and more, providing a framework upon which users could develop simple dynamic web applications such as guestbooks. In June of 1995, Rasmus » released the source code for PHP Tools to the public, which allowed developers to use it as they saw fit. This also permitted - and encouraged - users to provide fixes for bugs in the code, and to generally improve upon it.

In September of that year, Rasmus expanded upon PHP and - for a short time - actually dropped the PHP name. Now referring to the tools as FI (short for "Forms Interpreter"), the new implementation included some of the basic functionality of PHP as we know it today. It had Perl-like variables, automatic interpretation of form variables, and HTML embedded syntax. The syntax itself was similar to that of Perl, albeit much more limited, simple, and somewhat inconsistent. In fact, to embed the code into an HTML file, developers had to use HTML comments. Though this method was not entirely well-received, FI continued to enjoy growth and acceptance as a CGI tool --- but still not quite as a language. However, this began to change the following month; in October, 1995, Rasmus released a complete rewrite of the code. Bringing back the PHP name, it was now (briefly) named "Personal Home Page Construction Kit," and was the first release to boast what was, at the time, considered an advanced scripting interface. The language was deliberately designed to resemble C in structure, making it an easy adoption for developers familiar with C, Perl, and similar languages. Having been thus far limited to UNIX and POSIX-compliant systems, the potential for a Windows NT implementation was being explored.

The code got another complete makeover, and in April of 1996, combining the names of past releases, Rasmus introduced PHP/FI. This second-generation implementation began to truly evolve PHP from a suite of tools into a programming language in its own right. It included built-in support for DBM, mSQL, and Postgres95 databases, cookies, user-defined function support, and much more. That June, PHP/FI was given a version 2.0 status. An interesting fact about this, however, is that there was only one single full version of PHP 2.0. When it finally graduated from beta status in November, 1997, the underlying parsing engine was already being entirely rewritten.

Though it lived a short development life, it continued to enjoy a growing popularity in still-young world of web development. In 1997 and 1998, PHP/FI had a cult of several thousand users around the world. A Netcraft survey as of May, 1998, indicated that nearly 60,000 domains domains reported having headers containing "PHP," indicating that the host server did indeed have it installed. This number equated to approximately 1% of all domains on the Internet at the time. Despite these impressive figures, the maturation of PHP/FI was doomed to limitations; while there were several minor contributors, it was still primarily developed by an individual.

**Features of PHP:**

# i) HTTP authentication with PHP

It is possible to use the header () function to send an "Authentication Required" message to the client browser causing it to pop up a Username/Password input window. Once the user has filled in a username and a password, the URL containing the PHP script will be called again with the predefined variables PHP\_AUTH\_USER, PHP\_AUTH\_PW, and AUTH\_TYPE set to the user name, password and authentication type respectively. These predefined variables are found in the $\_SERVER and $HTTP\_SERVER\_VARS arrays. Both "Basic" and "Digest" (since PHP 5.1.0) authentication methods are supported. See the header () function for more information.

# ii) Cookies

PHP transparently supports HTTP cookies. Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the set cookie () or setrawcookie() function. Cookies are part of the HTTP header, so set cookie () must be called before any output is sent to the browser. This is the same limitation that header () has. You can use the output buffering functions to delay the script output until you have decided whether or not to set any cookies or send any headers. Any cookies sent to you from the client will automatically be included into a $\_COOKIE auto-global array if variables order contains "C". If you wish to assign multiple values to a single cookie, just add [] to the cookie name.

## iii) New features

PHP 5.3.0 offers a wide range of new features:

* Support for namespaces has been added.
* Support for Late Static Bindings has been added.
* Support for jump labels (limited goto) has been added.
* Support for native Closures (Lambda/Anonymous functions) has been added.
* There are two new magic methods, \_\_callStatic and \_\_invoke.
* Nowdoc syntax is now supported, similar to Heredoc syntax, but with single quotes.
* It is now possible to use Heredocs to initialize static variables and class properties/constants.
* Heredocs may now be declared using double quotes, complementing the Nowdoc syntax.
* Constants can now be declared outside a class using the const keyword.
* The ternary operator now has a shorthand form: ?:.
* The HTTP stream wrapper now considers all status codes from 200 to 399 to be successful.
* Dynamic access to static methods is now possible.
* Exceptions can now be nested.
* A garbage collector for circular references has been added, and is enabled by default.
* The mail() function now supports logging of sent email. (Note: This only applies to email sent through this function.)

**Advantages of PHP:**

* PHP (Hypertext Pre-Processor) is a server-side web programming language that is widely used for web development.  
  However, here are many languages which are used for web development or web programming. But among all of them PHP is the most popular web scripting language. So, let us find out why PHP is widely used for web development…
* PHP language has its roots in C and C++. PHP syntax is most similar to C and C++ language syntax. So, programmers find it easy to learn and manipulate.
* MySQL is used with PHP as back-end tool. MySQL is the popular online database and can be interfaced very well with PHP. Therefore, PHP and MySQL are excellent choice for webmasters looking to automate their web sites.
* PHP can run on both UNIX and Windows servers.
* PHP also has powerful output buffering that further increases over the output flow. PHP internally rearranges the buffer so that headers come before contents.
* PHP is dynamic. PHP works in combination of HTML to display dynamic elements on the page. PHP only parses code within its delimiters, such as. Anything outside its delimiters is sent directly to the output and not parsed by PHP.
* PHP can be used with a large number of relational database management systems, runs on all of the most popular web servers and is available for many different operating systems.
* PHP5 a fully object oriented language and its platform independence and speed on Linux server helps to build large and complex web applications.
* So, in general PHP is cheap, secure, fast and reliable for developing web applications.

**About MySQL:**

**MySQL** is a relational database management system (RDBMS)[]](http://en.wikipedia.org/wiki/MySQL#cite_note-1) that runs as a server providing multi-user access to a number of databases.  The SQL phrase stands for Structured Query Language.Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google  , Facebook, and Twitter.

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout it's history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software — including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com.

The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius, who had worked together since the 1980's. More historical information on MySQL.

**2. SYSTEM STUDY**

A detailed study to determine whether, to what extent, and how automatic data processing equipment should be used it usually includes an analysis of the existing system and the design of the new system, including the development of system specifications which provide a basis for the selection of equipment.

**2.1. EXISTING STUDY**

The problem of routing in accumulative multi-hop (AM) communication networks, in which we are instead interested here, is however far from being understood today. In the simplest accumulative multi-hop network, a single source communicates to a single destination assisted by several relay nodes that can accumulate the received energy/information from previous relay transmissions.

In practice, there are two main accumulation mechanisms at relays: energy and mutual-information accumulation. Energy accumulation can be performed at the receiving nodes, e.g., through spacetime coding or repetition coding.

Mutual-information accumulation can be realized using rateless codes e.g. fountain or raptor codes. Accumulation mechanisms are considered in current and next generation standards since they increase communication reliability and reduce energy consumption

**2.1.1 DRAWBACKS**

* More Energy utilized
* Latency is High.
* The transmission delay.
* Multiple energy leakages while transmissions.

**2.2 PROPOSED SYSTEM**

The work presented here builds, mainly, on top of the works conducted show that the AM network communication routing problem cannot be represented using graphs, and thus, the optimality conditions derived in existing for routing over graphs cannot be invoked.

We instead show that, in general, the AM routing problem needs to be modeled using hypergraphs. We then find new conditions to guarantee the optimality of Dijkstra’s algorithm in hypergraphs. These conditions are only sufficient but not necessary. Equipped with these optimality conditions, we discuss the optimality of Dijkstra’s algorithm for the minimum energy routing problem in static AM networks. To that end, we focus mainly on decode-and-forward (DF) based relaying strategies.

DF relay nodes decode the source message completely by accumulating energy, or information from all previous transmissions. This routing problem has been previously addressed, we already know that finding the optimal transmission order for these networks is an NP-complete problem.

Our approach here consist instead on identifying particular DF AM network situations for which the routing problem can be represented either using graphs that satisfy Dijkstra’s optimality conditions, or using hypergraphs that satisfy the new optimality conditions found here.

**2.2.1 FEATURES**

* The proposed DF EAM algorithm improves the RPAR algorithm by more than 5% and the SP algorithm by more than 25% for networks with more than 5 nodes
* Finds the optimal path in such networks, and presented sufficient conditions for the optimality.

**2.3 FEASIBILITY STUDY**

All projects are feasible given unlimited users and infinite time. Unfortunately the development is more likely to be plagued by a scarcity resources and difficult delivery date. It is both necessary and prudent to evaluate the feasibility of an object, no such limitation imposed that is not feasible as all the resources are easily available and time given was sufficient. Feasibility study was made according to four key considerations

**TECHNICAL FEASIBILITY**

Technical feasibility centres on the exiting computer and to what extent it can support the proposed addition. This involves financial considerations to accommodate technical enhancement. If the budget is a serious constraint, the project is judged as not feasible.

**BEHAVIORAL FEASIBILITY**

An estimate should be made of how strong a reaction the user staff is likely to have toward the development of computerized system. It is common knowledge the computer installation have something to do understandable that the introduction of a candidate system requires special effort to educate, sell and train the staff on new ways of considering business.

**OPERATIONAL FEASIBILITY**

Operational feasibility is the method used to evaluate whether the proposed system is user friendly. They must be easy to handle, maintain and modify. The main objectives is that they should be enough flexible.

**ECONOMICAL FEASIBILITY**

Economic analysis is the most frequently used method for evaluating the effectiveness of a system more commonly known as cost/benefit analysis, the procedures is made to determine the benefits and savings that are expected from system and compare them with costs. The result of comparison for found out and changed if needed. This is an ongoing effort that improves in accuracy at each phase of the system life cycle. Feasibility is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. Feasibility and risk analysis are related in many ways. If project risk is great, the feasibility of providing quality is reduced.

**3. SYSTEM DESIGN AND DEVELOPMENT**

**3.1 FILE DESIGN**

There are numerous types of files used for storing data needed for processing, reference or back up. The main common types of processing files include

* Master,
* Transaction,
* Reference,
* Backup, Report and
* Sort.

**MASTER FILE**

A master file is the main that contains relatively permanent records about particular items or entries. For example a customer file will contain details of a customer such as customer ID, name and contact address.

**TRANSACTION FILE**

A transaction file is used to hold data during transaction processing. The file is later used to update the master file and audit daily, weekly or monthly transactions. For example in a busy supermarket, daily sales are recorded on a transaction file and later used to update the stock file. The file is also used by the management to check on the daily or periodic transactions.

**REFERENCE FILE**

A reference file is mainly used for reference or look-up purposes. Look-up information is that information that is stored in a separate file but is required during processing. For example, in a point of sale terminal, the item code entered either manually or using a barcode reader looks up the item description and price from a reference file stored on a storage device.

**BACKUP FILE**

A backup files is used to hold copies (backups) of data or information from the computers fixed storage (hard disk). Since a file held on the hard disk may be corrupted, lost or changed accidentally, it is necessary to keep copies of the recently updated files. Incase of the hard disk failure, a backup file can be used to reconstruct the original file.

**REPORT FILE**

Used to store relatively permanent records extracted from the master file or generated after processing. For example you may obtain a stock levels report generated from an inventory system while a copy of the report will be stored in the report file.

**SORT FILE**

It stores data which is arranged in a particular order. Used mainly where data is to be processed sequentially. In sequential processing, data or records are first sorted and held on a magnetic tape before updating the master file.

**3.2 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

**3.3 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.

**3.4 DATABASE DESIGN**

The development of database systems is not a major breakthrough in computer technology; rather it is a logical development in the methods used by computers to access and manipulate data stored in various parts of computer systems. The overall objective in the development of database technology has been to treat data as an organizational resource and as an integrated whole. Database management systems allow the data to the protected and organized separately from other resource. Defining the term ‘database’ is an integrated collection of items. Computer base systems which support a centralized data management function can support a database. If the data management software can provide centralized access to the data from the programs, possible to treat data as a separate resource.

Usually the centralized data management software is the software is a package called a database management system. Within it will be found components which exist as separate software packages, such as a complex file handler, an information retrieval system, or a comprehensive report generator. The organization of data in a database aims to achieve three major objectives such as data integration, data integrity and data independence.

**3.5 SYSTEM DEVELOPMENT**

Systems development is the process of defining, designing, testing, and implementing a new software application or program. It could include the internal development of customized systems, the creation of database systems, or the acquisition of third party developed software. Written standards and procedures must guide all information systems processing functions. The organization’s management must define and implement standards and adopt an appropriate system development life cycle methodology governing the process of developing, acquiring, implementing, and maintaining computerized information systems and related technology.

**3.5.1 DESCRIPTION OF MODULES**

**MODULES:**

* **SERVER CLIENT MODULE**
* **OPPORTUNISTIC DATA FORWARDING**
* **TREE-BASED ROUTING**
* **DESIGN OF PSR PROTOCOL**
* **PERFORMANCE EVALUTION**

**MODULE DISCRIPTION:**

**SERVER CLIENT MODULE:**

Client-server computing or networking is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients. Often clients and servers operate over a computer network on separate hardware. A server machine is a high-performance host that is running one or more server programs which share its resources with clients. A client also shares any of its resources; Clients therefore initiate communication sessions with servers which await (listen to) incoming requests.

**OPPORTUNISTIC DATA FORWARDING:**

Opportunistic data forwarding represents a promising solution to utilize the broadcast nature of wireless communication links. Opportunistic data forwarding refers to a way in which data packets are handled in a multihop wireless network. Unlike traditional IP forwarding, where an intermediate node looks up a forwarding table for a dedicated next hop, opportunistic data forwarding allows potentially multiple downstream nodes to act on the broadcast data packet. One of the initial works on opportunistic data forwarding is selective diversity forwarding a transmitter picks the best forwarder from multiple receivers, which successfully received its data, and explicitly requests the selected node to forward the data. However, its overhead needs to be significantly reduced before it can be implemented in practical networks.

**TREE BASED ROUTING:**

We put forward a tree-based routing protocol, i.e., PSR, which is inspired by the PFA and the WRP. Its routing overhead per time unit per node is on the order of the number of the nodes in the network as with DSDV, but each node has the full-path information to reach all other nodes. For it to have a very small footprint, PSR’s route messaging is designed to be very concise. First, it uses only one type of message, i.e., the periodic route update, both to exchange routing information and as hello beacon messages. Second, rather than packaging a set of discrete tree edges in the routing messages, we package a converted binary tree to reduce the size of the payload by about a half. Third, we interleave full-dump messages with differential updates so that, in relatively stable networks, the differential updates are much shorter than the full-dump messages.

**DESIGN OF PSR PROTOCOL:**

The PSR protocol proposed in this paper uses tree-based routing as in PFA and WRP. To make our PSR more suitable for the MANETs, we adopt a combined route update strategy that takes advantage of both “event-driven” and “timer-driven” approaches. Specifically, nodes would hold their broadcast after receiving a route update for a period of time. If more updates have been received in this window, all updates are consolidated before triggering one broadcast. The period of the update cycle is an important parameter in PSR. Furthermore, we go an extra mile to reduce its routing overhead. First, we interleave fulldump and differential updates to strike the balance between efficient and robust network operation. Second, we package affected links into forests to avoid duplicating nodes in the data structure. Finally, to further reduce the size of differential update messages, each node tries to minimize the alteration of the routing tree that it maintains as the network changes its structure.

**PERFORMANCE EVALUTION:**

We study the performance of PSR using computer simulation with php. We compare PSR against DSDV which are three fundamentally different routing protocols in MANETs, with varying network densities and node mobility rates. We measure the data transportation capacity of these protocols supporting the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP) with different data flow deployment characteristics.

Our tests show that the overhead of PSR is indeed only a fraction of that of the baseline protocols. Nevertheless, as it provides global routing information at such a small cost, PSR offers similar or even better data delivery performance. Here, we first describe how the experiment scenarios are configured and what measurements are collected. Then, we present and interpret the data collected from networks with heavy TCP flows and from those with light UDP streams.

We compare the performance of PSR with that of DSDV. The reasons that we select this baseline protocols that are different in nature are as follows. On one hand, OLSR and DSDV are both proactive routing protocols, and PSR is also in this category. PSR sits in the middle ground, where each node maintains a spanning tree of the network. Furthermore, DSDV is a well-accepted reactive source routing scheme, and as with PSR, it support source routing, which does not require other nodes to maintain forwarding lookup tables. All three baseline protocols are configured and tested out of the box of php.

**4. SYSTEM TESTING AND IMPLEMENTATION**

### 4.1 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.

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.

**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.

# 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.

### 4.2 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. CONCLUSION**

This paper has been motivated by the need to support opportunistic data forwarding in MANETs. To generalize the milestone work of ExOR for it to function in such networks, we needed a PSR protocol. Such a protocol should provide more topology information than DVs but must have significantly smaller overhead than LS routing protocols; even the MPR technique in OLSR would not suffice. Thus, we put forward a tree-based routing protocol, i.e., PSR, which is inspired by the PFA and the WRP.

PSR’s route messaging is designed to be very concise. First, it uses only one type of message, i.e., the periodic route update, both to exchange routing information and as hello beacon messages. Second, rather than packaging a set of discrete tree edges in the routing messages, we package a converted binary tree to reduce the size of the payload by about a half. Third, we interleave full-dump messages with differential updates so that, in relatively stable networks, the differential updates are much shorter than the full-dump messages.

Further reduce the size of the differential updates, when a node maintains its routing tree as the network changes, it tries to minimize alteration of the tree next hop of the path to drop the packet. This is fundamentally different from traditional IP forwarding in proactive routing with more built-in adaptivity, where the routing information maintained at nodes closer to the destination is often more updated than the source node.

**FUTURE ENHANCEMENT**

In future recommendation we could explore the same protocol for bigger networking environment. The further research could be done on network security

**BIBLIOGRAPHY**

**Book Reference:**

1. GILLMORE, W.J. ‘Beginning PHP and MYSQL’, Après publication, Fourth Edition.
2. JanatValade ‘PHP AND MYSQL FOR DUMMIES’, Stenge Publication, 5th Edition.
3. Justin Davis ‘PHP 5 IN PRACTICE’, New Edition.
4. Kilgarriff, A. (2007) ‘Googleology Is Bad Science’, Computational Linguistics, Vol.33, pp.147-151
5. Luke willing and Laura Thomson ‘PHP AND MYSQL WEB DEVELOPMENT’, 6th Edition.
6. Steven Holznore ‘PHP THE COMPLETE REFERENCE’, 8th Edition.
7. Steve Schering ‘MYSQL BIBLE’, 8th Edition.
8. Steve Schering ‘PHP AND MYSQL BIBLE’, 4th Edition.
9. VikramVaswi ‘PHP: a Beginners Guide’, First Edition.

**Website References**

1. [www.codango.com/php (password](http://www.codango.com/php%20%20%20(password) condition and form design).
2. www.w3schools.com (MySQL insert into and update forms).
3. swww.happycoadings.com (source code for JavaScript’s).
4. www.sourcecodeonline.com (menu system development).
5. www.codeonline.com (examples of codes).
6. www.netbeanstutorial.org (net beans set up instructions).

**APPENDIX**

**A. DATA FLOW DIAGRAM**

A picture is worth a thousand words. A Data Flow Diagram (DFD) is traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or combination of both.

It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

It is usually beginning with a context diagram as the level 0 of DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing particular function really depending on the complexity that function.

**DFD SYMBOLS:**

In the DFD, there are four symbols

Process that transforms data flow.

Source or Destination of data

Data flow

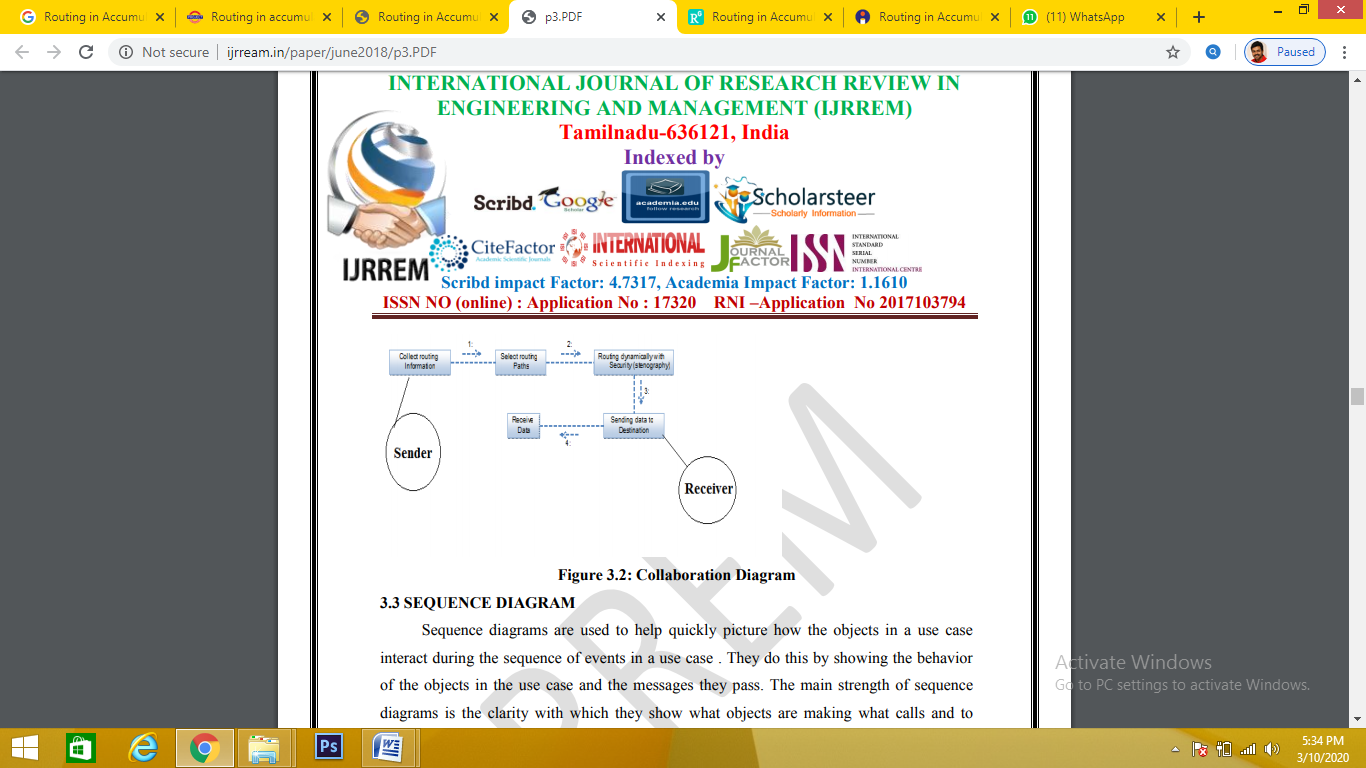
Data Store

**DATA FLOW DIAGRAM**

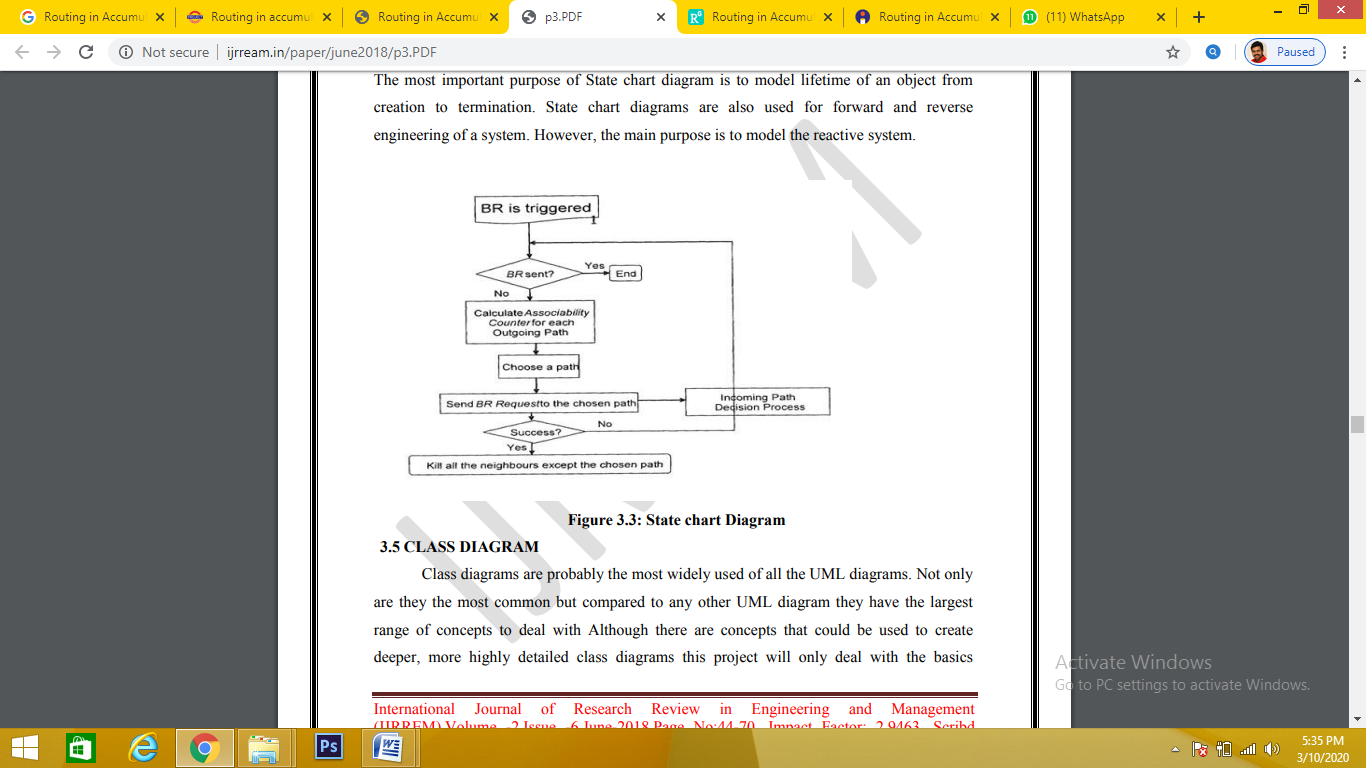
**USECASE DIAGRAM**

****

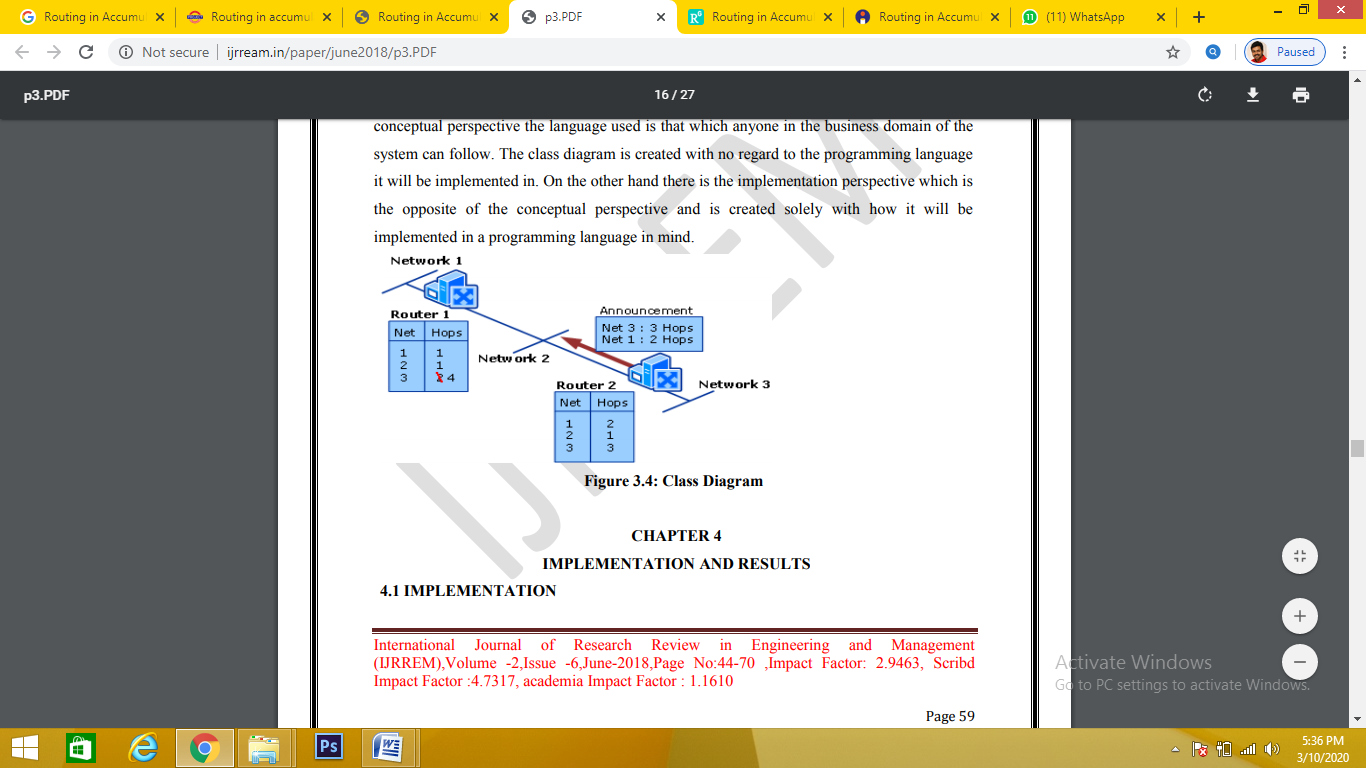
**COLLABORATION DIAGRAM**

****

**SEQUENCE DIAGRAM**

****

**CLASS DIAGRAM**

****

**B. TABLE STRUCTURE**

**C. SAMPLE CODING**

**D. SAMPLE INPUT**

**E. SAMPLE OUTPUT**