# Design and Development of a Proposed System to Enable Efficient Data Query in Network Settings

**Mugala Raju[1], Ms.G.Sangeetha[2], Prof.Dr.G.Manoj Someswar[3]**

[1]M.Tech.(CSE) from Narasimha Reddy Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India

[2]M.Tech. (CSE), Assistant Professor, Department of CSE, Narasimha Reddy Engineering College, Affiliated to JNTUH, Hyderabad, Telangana, India

[3]B.Tech., M.S.(USA), M.C.A., Ph.D., Principal & Professor, Department Of CSE, Anwar-ul-uloom College of Engineering & Technology, Affiliated to JNTUH, Vikarabad, Telangana, India

**ABSTRACT:**

*This research work addresses the problem of how to enable efficient data query in a Mobile Ad-hoc Social Network (MASON), formed by mobile users who share similar interests and connect with one another by exploiting Bluetooth and/or WiFi connections. The data query in MASONs faces several unique challenges including opportunistic link connectivity, autonomous computing and storage, and unknown or inaccurate data providers. Our goal is to determine an optimal transmission strategy that supports the desired query rate within a delay budget and at the same time minimizes the total communication cost. To this end, we propose a centralized optimization model that offers useful theoretic insights and develop a distributed data query protocol for practical applications. To demonstrate the feasibility and efficiency of the proposed scheme and to gain useful empirical insights, we carry out a testbed experiment by using 25 off-the-shelf Dell Streak tablets for a period of 15 days. Moreover, extensive simulations are carried out to learn the performance trend under various network settings, which are not practical to build and evaluate in laboratories.*

***Keywords:*** Mobile Adhoc Social Network; Delay-Tolerant networks (DTNs); Time-To-Live (TTL); Delegation Query; Delay-tolerant networks; Dynamic Redundancy Control

## INTRODUCTION

Distributed computing is a field of computer science that studies distributed systems. A distributed system is a software system in which components located on networked computers communicate and coordinate their actions by passing messages. The components interact with each other in order to achieve a common goal. There are many alternatives for the message passing mechanism, including RPC-like connectors and message queues. Three significant characteristics of distributed systems are: concurrency of components, lack of a global clock, and independent failure of components. An important goal and challenge of distributed systems is location transparency. Examples of distributed systems vary from SOA-based systems to massively multiplayer online games to peer-to-peer applications.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers, which communicate with each other by message passing.[1]

The word *distributed* in terms such as "distributed system", "distributed programming", and "distributed algorithm" originally referred to computer networks where individual computers were physically distributed within some geographical area. The terms are nowadays used in a much wider sense, even referring to autonomous processes that run on the same physical computer and interact with each other by message passing. While there is no single definition of a distributed system, the following defining properties are commonly used:

- There are several autonomous computational entities, each of which has its own local memory.
- The entities communicate with each other by message passing.

In this article, the computational entities are called *computers* or *nodes.*

A distributed system may have a common goal, such as solving a large computational problem.] Alternatively, each computer may have its own user with individual needs, and the purpose of the distributed system is to coordinate the use of shared resources or provide communication services to the users.[2]

Other typical properties of distributed systems include the following:

- The system has to tolerate failures in individual computers.
- The structure of the system (network topology, network latency, number of computers) is not known in advance, the system may consist of different kinds of computers and network links, and the system may change during the execution of a distributed program.
- Each computer has only a limited, incomplete view of the system. Each computer may know only one part of the input.[3]

Distributed systems are groups of networked computers, which have the same goal for their work. The terms "concurrent computing", "parallel computing", and "distributed computing" have a lot of overlap, and no clear distinction exists between them. The same system may be characterized both as "parallel" and "distributed"; the processors in a typical distributed system run concurrently in parallel. Parallel computing may be seen as a particular tightly coupled form of distributed computing, and distributed computing may be seen as a loosely coupled form of parallel computing. Nevertheless, it is possible to roughly classify concurrent systems as "parallel" or "distributed" using the following criteria:[4]

- In parallel computing, all processors may have access to a shared memory to exchange information between processors.
- In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.

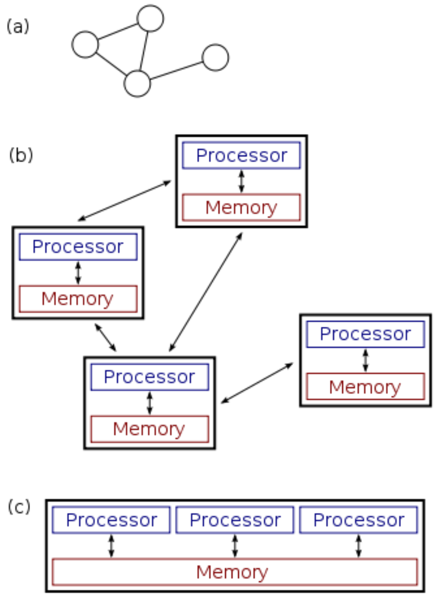

**Figure 1: Difference between distributed and parallel systems**.

The figure on the right illustrates the difference between distributed and parallel systems. Figure (a) is a schematic view of a typical distributed system; as usual, the system is represented as a network topology in which each node is a computer and each line connecting the nodes is a communication link. Figure (b) shows the same distributed system in more detail: each computer has its own local memory, and information can be exchanged only by passing messages from one node to another by using the available communication links. Figure (c) shows a parallel system in which each processor has a direct access to a shared memory.[5]

The situation is further complicated by the traditional uses of the terms parallel and distributed *algorithm* that do not quite match the above definitions of parallel and distributed *systems*; see the section Theoretical foundations below for more detailed discussion. Nevertheless, as a rule of thumb, high-performance parallel computation in a shared-memory multiprocessor uses parallel algorithms while the coordination of a large-scale distributed system uses distributed algorithms.[6]

## LITERATURE SURVEY

Many existing systems for sensor networks rely on state information stored in the nodes for proper operation (e.g., pointers to parent in a spanning tree, routing information, etc). In dynamic environments, such systems must adopt failure recovery mechanisms, which significantly increase the complexity and impact the overall performance. In this paper, we investigate alternative schemes for query processing based on random walk techniques. The robustness of this approach under dynamics follows from the simplicity of the process, which only requires the connectivity of the neighborhood to keep moving.[7] In addition we show that visiting a constant fraction of sensor network, say 80%, using a random walk is efficient in number of messages and sufficient for answering many interesting queries with high quality. Finally, the natural behavior of a random walk, also provide the important properties of load-balancing and scalability.[8]

In this research paper, we consider the problem of searching for a node or an object (i.e., piece of data, file, etc.) in a large network. Applications of this problem include searching for a destination node in a mobile ad hoc network, querying for a piece of desired data in a wireless sensor network, and searching for a shared file in an unstructured peer-to-peer network. We consider the class of controlled flooding search strategies where query/search packets are broadcast and propagated in the network until a preset time-to-live (TTL) value carried in the packet expires. Every unsuccessful search attempt, signified by a timeout at the origin of the search, results in an increased TTL value (i.e., larger search area) and the same process is repeated until the object is found. The primary goal of this study is to find search strategies (i.e., sequences of TTL values) that will minimize the cost of such searches associated with packet transmissions. Assuming that the probability distribution of the object location is not known a priori, we derive search strategies that minimize the search cost in the worst-case, via a performance measure in the form of the competitive ratio between the average search cost of a strategy and that of an omniscient observer.[9] This ratio is shown in prior work to be asymptotically (as the network size grows to infinity) lower bounded by 4 among all deterministic search strategies. In this research paper, we show that by using randomized strategies (i.e., successive TTL values are chosen from certain probability distributions rather than deterministic values), this ratio is asymptotically lower bounded by e. We derive an optimal strategy that achieves this lower bound, and discuss its performance under other criteria. We further introduce a class of randomized strategies that are sub-optimal but potentially more useful in practice.[10]

The advances in computer and wireless communication technologies have led to an increasing interest in ad hoc networks which are temporarily constructed by only mobile hosts. In ad hoc networks, since mobile hosts move freely, disconnections occur frequently, and this causes frequent network division. Consequently, data accessibility in ad hoc networks is lower than that in the conventional fixed networks. We propose three replica allocation methods to improve data accessibility by replicating data items on mobile hosts. In these three methods, we take into account the access frequency from mobile hosts to each data item and the status of the network connection. We also show the results of simulation experiments regarding the performance evaluation of our proposed methods.[11]

In delay-tolerant networks (DTNs), information search is a significant topic that has yet to be widely investigated. Although social-based approaches can be used to address the problem, most existing schemes employ the multihop paradigm and leave out the severe resource constraint in DTNs.[12] In this research paper, we experimentally explore several realistic data sets and then reveal that users' one-hop neighbors can cover most range of the whole network in a reasonable time period, which lays a solid fundamental for two-hop information search schemes. Therefore, we propose DelQue (delegation query), which is a novel two-hop delegation query scheme integratedly considering query and response to save network energy in terms of the number of involved relays.[13] In DelQue, we exploit the social utility of each neighbor to represent its capability to query interesting information and then colocate with the source to respond. Furthermore, we also present a spatio-temporal prediction method of user mobility to compute neighbors' utility. Such a lightweight forecasting technique only requires network users to maintain two parameters, making it suitable for a resource-scarce mobile setting. Extensive realistic trace-driven simulations show that DelQue allows for the maintenance of a very high and steady information query ratio with extremely low energy cost and, meanwhile, achieves comparable or shorter delays compared with some existing schemes.[14]

This research paper focuses on distributed data query in intermittently connected passive RFID networks, which are characterized by extraordinarily limited communication capacity and asynchronous and opportunistic communication links. To address such unique challenges, we propose a distributed data query framework that clusters RFID readers and establishes a 0-1 Knapsack model based on dynamic packet appraisal to enable highly efficient data transmission. We implement a prototype by using Alien RFID gears and carry out experiments that involve 52 volunteers for 14 days to evaluate the proposed data query framework.[15]

## 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 DESIGN

### SYSTEM ARCHITECTURE:

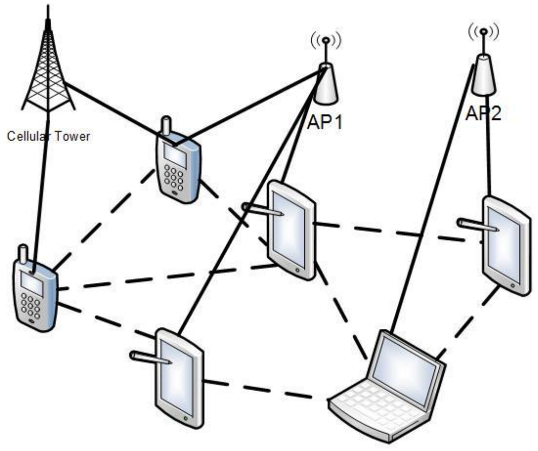

**Figure 2 : System Architecture**

### DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

### UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

### GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

## USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

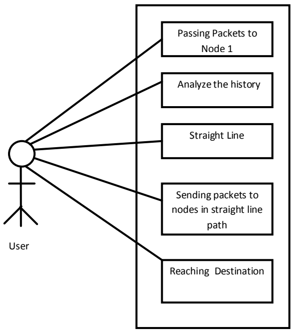
Passing Packets to Node 1
Analyze the history
Straight Line
Sending packets to nodes in straight line path
Reaching Destination
User

**Figure 3: Use Case Diagram**

## CLASS DIAGRAM:

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

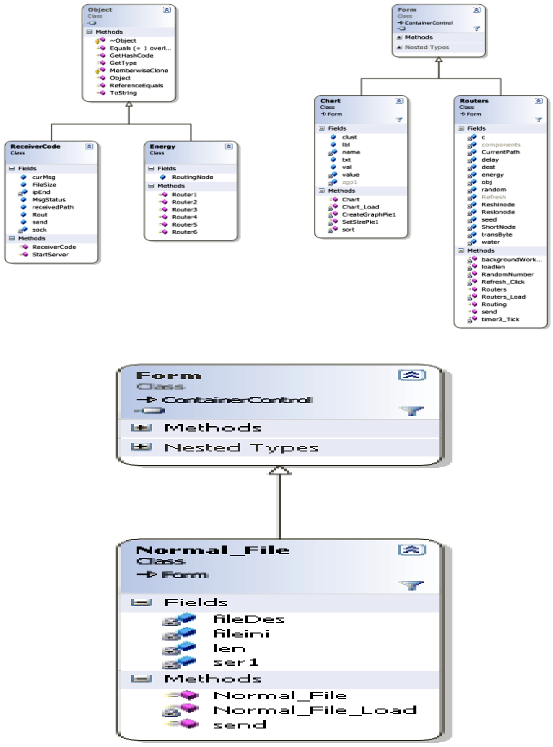

**Figure 4: Class Diagram**

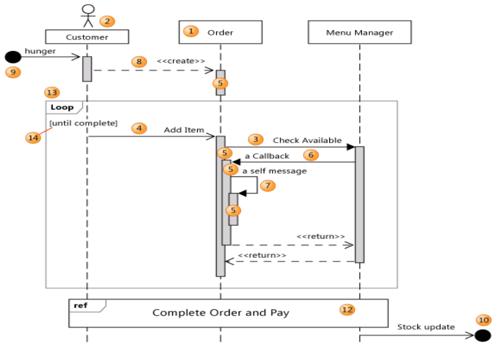

**Figure 5: Sequence Diagram**

## SEQUENCE DIAGRAM:

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

## COLLABORATION DIAGRAM:

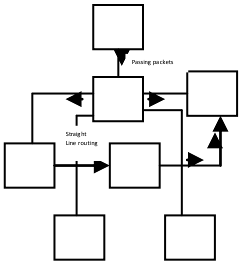

**Figure 6: Collaboration Diagram**

## 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.

**SYSTEM ANALYSIS**

**EXISTING SYSTEM:**

An autonomous social network formed by mobile users who share similar interests and connect with one another by exploiting the Bluetooth and/or WiFi connections of their mobile phones or portable tablets is called MASON. An individual MASON is incomparable with online social networks in terms of the population of participants, the number of social connections and the amount of social media. MASONs gain significant value by serving as a supplement and augment to online social networks and by effectively supporting local community-based ad-hoc social networking. it helps discover and update social links that are not captured by online social networks and allows a user to query localized data such as local knowledge, contacts and expertise, surrounding news and photos, or other information that people usually cannot or do not bother to report to online websites but may temporarily keep on their portable devices or generate upon a request. A query is created by a query issuer. It is delivered by the network toward the nodes that can successfully provide an answer (i.e., data providers). If a data provider receives the query, it sends the query reply to the query issuer.

**DISADVANTAGES OF EXISTING SYSTEM:**

- Opportunistic link connectivity
- Autonomous computing and storage:
- Unknown or inaccurate expertise:

**PROPOSED SYSTEM:**

We propose a centralized optimization model that offers useful theoretic insights and develop a distributed data query protocol for practical applications. Based on the insights gained from the analysis on MASON, a distributed data query protocol is proposed, aiming to enable highly efficient ad hoc query under practical MASON settings. A distributed protocol for the data query in MASONs is based on two key techniques. First, it employs "reachable expertise" as the routing metric to guide the transmission of query requests. Second, it exploits the redundancy in query transmission. Redundancy is not considered in the analysis due to its intractability, but can effectively improve the query delivery rate in practice if it is properly controlled.

**ADVANTAGES OF PROPOSED SYSTEM:**

- The feasibility and efficiency of the data query protocol is increased
- The proposed system provides facilities to gain useful empirical insights,.
- Minimized total communication cost.

**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.

## IMPLEMENTATION

### MODULES:

- System Model
- Protocol Design
- Routing with Dynamic Redundancy Control
- Data query

### MODULES DESCRIPTION:

### System Model

The communication in MASONs depends on nodal meeting events. The time interval between two consecutive meeting events between two nodes.

Given the intermittent network setting, we assume that the communication capacity is ruled by nodal meeting opportunities.

In other words, we assume the communication delay is dominated by nodal meeting intervals. When two nodes meet, the channel bandwidth is sufficient for them to exchange data packets with negligible delay

### Protocol Design

In this research paper, we introduce a distributed protocol for the data query in MASONs. It is based on two key techniques. First, as motivated by our analytic and optimization model discussed above, it employs "reachable expertise" as the routing metric to guide the transmission of query requests. Second, it exploits the redundancy in query transmission. Redundancy is not considered in the analysis due to its intractability, but can effectively improve the query delivery rate in practice if it is properly controlled.

### Routing with Dynamic Redundancy Control

Based on the routing metric, i.e., reachable expertise, we now introduce the routing algorithm. The delivery of a query is guided by the aggregated reachable expertise, where the query is generally forwarded from the node with a lower aggregated reachable expertise to the node with a higher one. In contrast to the conventional store-and-forward data transmission where a single copy of data is transmitted across the network, redundancy is often employed in opportunistic networks. While redundancy is not addressed in the analysis due to its intractability, it is important in practice to achieve the desired query delivery rate. Generally speaking, the higher the redundancy, the higher probability the query is answered successfully. However, redundancy must be properly controlled as excessive redundancy may exhaust network capacity and thus degrade the performance.

### Data query

Multiple copies of a query request may exist in the network, but we assume a node receives and forwards the same request only once. A naive approach is to create a fixed amount of redundancy for each query. For example, a predetermined number of copies of the query can be created and distributed to other nodes in the network. This approach, however, is often inefficient, because the effectiveness of redundancy depends on the nodes that receive, carry and forward the query. In an extreme case, all redundant copies of the query may be transmitted and carried by the nodes that have little chance to meet the node(s) that can answer the query and thus become ineffective. As a matter of fact, the effectiveness of redundancy highly depends on the reachable expertise of the nodes that carry the redundant copies. To this end, we introduce a parameter to dynamically reflect the "effective redundancy".

## RESULTS & CONCLUSION

This work addresses the problem of how to enable efficient data query in a Mobile Ad-hoc Social Network (MASON), formed by mobile users who share similar interests and connect with one another by exploiting Bluetooth and/or WiFi connections. The data query in MASONs faces several unique challenges including opportunistic link connectivity, autonomous computing and storage, and unknown or inaccurate data providers. Our goal is to determine an optimal transmission strategy that supports the desired query rate within a delay budget and at the same time minimizes the total communication cost. To this end, we propose a centralized optimization model that offers useful theoretic insights and develop a distributed data query protocol for practical applications. To demonstrate the feasibility and efficiency of the proposed scheme and to gain useful empirical insights, we carry out a testbed experiment by using 25 off-the-shelf Dell Streak tablets for a period of 15 days. Moreover, extensive simulations are carried out to learn the performance trend under various network settings, which are not practical to build and evaluate in laboratories.

## REFERENCES

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[2] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD Conference*, pages 439–450, 2000.

[3] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *STOC*, pages 503–513, 1990.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Crypto*, pages 1–15, 1996.

[5] A. Ben-David, N. Nisan, and B. Pinkas. FairplayMP - A system for secure multi-party computation. In *CCS*, pages 257–266, 2008.

[6] J.C. Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Crypto*, pages 251–260, 1986.

[7] J. Brickell and V. Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *ASIACRYPT*, pages 236–252, 2005.

[8] D.W.L. Cheung, J. Han, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. A fast distributed algorithm for mining association rules. In *PDIS*, pages 31– 42, 1996.

[9] D.W.L Cheung, V.T.Y. Ng, A.W.C. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. *IEEE Trans. Knowl. Data Eng.*, 8(6):911–922, 1996.

[10] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[11] A.V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *KDD*, pages 217–228, 2002. [12] R. Fagin, M. Naor, and P. Winkler. Comparing Information Without Leaking It. *Communications of the ACM*, 39:77–85, 1996.

[13] M. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.

[14] M.J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, pages 1–19, 2004.

[15] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.