# uopStore: an E-commerce Platform with a Peer-to-Peer Infrastructure

by

Sen Wang

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Master of Science degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

**Abstract**

We propose to put forth a platform with a Peer-to-Peer infrastructure for e-commerce. The systems that host e-commerce applications are facing the challenge of withstanding the explosive growth experienced in the realm of e-commerce transactions. Bearing in mind the workload characteristics, the systems for e-commerce applications should be scalable and flexible. For instance, transactions in such systems often occur in dynamic environment, where users in different geographic areas communicate concerning their business activities. Also, a varying number of products are sold through purchasing transactions. Furthermore, the number of different transactions for the same product may vary frequently from time to time. However, currently, such systems are built on centralized architecture in general. Such architecture does not easily meet the demands of the workloads of e-commerce. In this thesis, we developed uopStore, a platform focus on a distributed Peer-to-Peer architecture for e-commerce applications. The architecture is inspired by BestPeer++ [14]. BestPeer++, a large-scale data processing platform, enables data sharing between enterprise systems. The shared data is distributed over the computing nodes in a server farm. These computing nodes are organized with the BATON Peer-to-Peer overlay network, the shared resource is then located with BATON's query (or lookup) operation. We adopted the architecture of BestPeer++ for uopStore, modified its components, and added one more layer to the top, the client layer. The client layer provides the interfaces for the users to access the remote services in the server farm. This way, uopStore is able to support decentralized e-commerce activities as well as deliver elastic data sharing services. The system also has some basic social networking functions in order to benefit from social media trends: the ability to chat with friends, send recommendations to friends, etc. In this thesis, we present the architecture of the proposed system first. We then describe the components of different peer nodes in detail. We further present the distributed operations which are used to realize different e-commerce activities. Furthermore, we discuss the results from experiments. These experiments are used to verify that the distributed operations have been implemented correctly.

**Acknowledgements**

Foremost, I would like to thank my supervisor Dr. Iluju Kiringa and co-supervisor Dr. Tet Yeap for giving me this invaluable opportunity to do research in the Peer-to-Peer area. I could not have achieved my original goal without their continuous support and help. Dr. Iluju Kiringa gave me numerous inspiring ideas and thoughts that have helped me to find the right path towards this thesis. Through his mentoring, I have gained great knowledge of databases and Peer-to-Peer architecture. This knowledge is the key to this research. I appreciate that he has spent many hours patiently helping me to shape my work and to correct my faults as well. Moreover, Dr. Tet Yeap gave me instructions from time to time. With his help, I could define and achieve the goals of this thesis precisely. Their ideas, suggestions and positive attitude have enabled me to finish this thesis.

I would like to acknowledge the support from Dr. Ying Qiao. She has given me a great deal of help in the matter of refining my work. I sincerely appreciate her contribution of time and ideas that have enabled me to successfully complete the thesis.

I really treasure the time that I have spent with my teammates during my attempt to find the correct solutions to my problems. The team accomplishments inspired my work great deal.

I am grateful for the support that I received and for the facilities that I have had access to at the School of Electrical Engineering and Computer Science.

Lastly, I want to thank my parents for their understanding, inspiration and constant support throughout all kinds of situations.

# Table of Contents

# List of Figures

# Chapter 1

# Introduction

Peer-to-Peer systems (or P2P systems) are able to provide data sharing and computing services on a large scale. We envision building a platform with a P2P infrastructure for e-commerce applications to process their transactions. Even though Peer-to-Peer technology has been in existence for over a decade, there is not yet a P2P system that can efficiently and practically support e-commerce activities. Through the duration of this thesis work, we have analyzed the characteristics of one such potential platform and have developed the components in this platform as well. These components provide the basic services for e-commerce applications.

## 1.1 Motivation

E-commerce activities have already become a necessary part of our daily lives, a fact made evident by the thousands of online shopping transactions occurring every second in the world. For instance, the total value of goods sold on eBay in 2010 alone had a worth of 62 billion dollars. That's 2,000 dollars per second [5]. A successful e-commerce service provider has to face the challenge of building a system that processes a large number of transactions and must also be able to also store a large amount of data. To overcome this challenge and to satisfy users by providing high quality services, service providers normally build a client/server system with a number of servers and a network that has an expanded

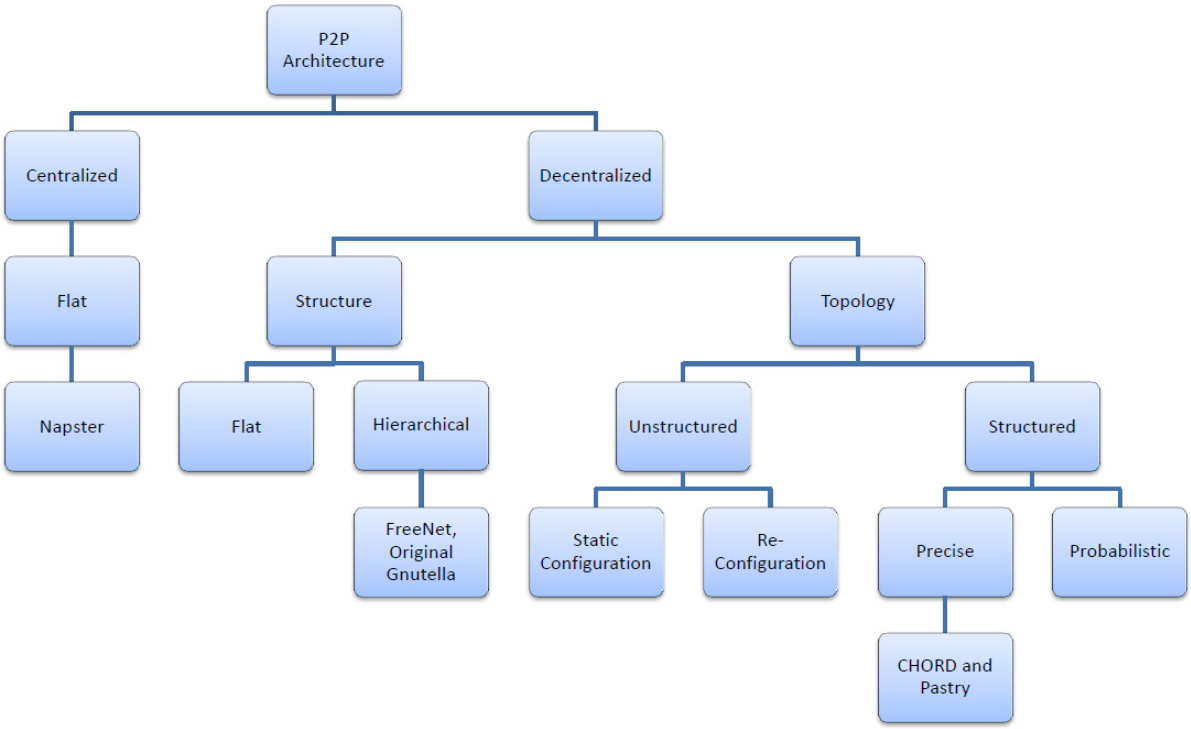



Figure 1.1: Peer-to-Peer topology ( Re-generated from Figure 2.1 in chapter 2 in [42] )

bandwidth. The client/server architecture with central servers has been adopted by most of the e-commerce service providers [34].

Although the client/server architecture supports systems that have a large number of users with lower costs of facility usage, maintenance and security mechanisms, it is difficult for this architecture to meet the requirements of e-commerce applications. The amount of merchandise being sold online is growing at a rapid speed. The diverse geographic orientation of users communicating on such platforms also has a dynamic impact. Products are sold with varying numbers of purchase transactions. Furthermore, the number of purchasing transactions for the same product may vary frequently from time to time. It is difficult for traditional architecture with central servers to handle the demands induced by the explosive growth in purchasing activities.

P2P systems have been used as the infrastructure of resource sharing or file downloading applications. Compared to the traditional client/server architecture, P2P systems can support more users with less cost. In academia, many new techniques have been developed

for P2P technology. Many of them can be potentially used for application in the industry. For instance, Chord [40] has been widely adopted for projects in the industry where a point-to-point network at the application layer is needed.

Computers in a P2P system are organized into a network called an overlay network. Such a network has a type of decentralized and distributed architecture where the nodes act as both suppliers and consumers of resources [6]. Companies and universities have been utilizing this kind of distributed architecture for more than 40 years. The network Net was originally developed in the late 1960s. Usenet [8], a distributed network, was then developed at Duke University and the University of North Carolina for the purpose of exchanging information throughout the Unix community in 1979. Through the development of Usenet, some of the best examples of decentralized control structures were developed on the net. In the structures of these systems, there is no central authority in control. Since the 1990s, networks with Peer-to-Peer structures were being used and further developed by companies such as the Aerospace manufacturer Boeing, the energy company Amerada Hess and the semiconductor chip maker Intel. In 1999, the structure of the Peer-to-Peer network began to draw more attention due to the advanced features of Napster [4]. Napster was originally developed as a pioneering Peer-to-Peer file sharing Internet service for the purpose of downloading audio files, typically, music encoded data in MP3 format. However, the original company ran into legal difficulties due to copyright infringement, and it ceased its operations. Ultimately, Napster was acquired by Roxio. After this, many companies and projects such as Gnutella [3] and FreeNet Project [2] successfully followed the Napster P2P file sharing technique. P2P technology has been continuously evolving in the file sharing arena since then.

There are various architectures developed and designed to organize a P2P network. Figure 1.1 shows the categories of these P2P systems from the perspective of network topology from a design standpoint. P2P systems can be classified as either centralized P2P systems such as Napster or the decentralized P2P system such as the example of FreeNet. In the decentralized category, the design of P2P architecture is recognized from a structure and topology point of view. Structure may be flat or hierarchal, while topology

may be structured or unstructured. The centralized P2P systems have the features that exist both in centralized architecture and decentralized architecture [42]. In this type of P2P system, there is a central node that works as a resource directory. It stores the information of the shared resources on all of the peer nodes. Normally, a user executes a resource searching request on a client node. The client node sends the request to the central node first. The central node locates the host nodes, easily storing the respective information by searching its directory, then proceeding to send a reply to the client node with the collected data. Once the client peer receives the information of these host nodes, it communicates with them directly and accesses the required resource without interference from the central node. However, the central node may easily become overloaded in a system with a large number of peers. Overall, this type of system lacks scalability when handling a large number of requests and lacks robustness in the potential case of a single point of failure at the central node.

Conversely, in a fully decentralized P2P system, peers have equal rights and responsibilities. With the involvement of more peers, both a greater computing capacity and a larger storage space can be provided. However, locating the peers that host specific services or data quickly is a critical and challenging issue. Unstructured P2P systems are decentralized systems with a flat structure. The peer nodes in such systems connect with random neighbours. These nodes are not able to look up resources efficiently due to the limited knowledge they have concerning their neighbours' circumstances, such as the resources location of resources that enable routing lookup requests. The structured P2P systems with hierarchical and structured topology can look up resources more efficiently. Among these, Chord is a popular system, its architecture has been widely used in applications with P2P infrastructures. The hybrid P2P systems combine the advantages of both the centralized and decentralized systems. In these systems, high capacity nodes are selected to act as servers and provide P2P networking services to other nodes. These nodes are called supernodes. In these systems, the resource lookup function can be undergone through either the decentralized or centralized manner of execution.

## 1.2 Objective

In this thesis, we engage in designing an e-commerce platform that builds on peer-to-peer architecture. This e-commerce platform is able to provide users with an experience that includes social networking, online shopping and online selling. Individual user or organizations have the ability to make contributions in providing system computing and storage resources, in contrast to the old model that required all system resources are contributed by one or few service providers.

- We should first select the application which has already been implemented and can be easily built upon. We are targeting P2P architecture that has features that maintain a high level of data locating capability and have ability of processing large scale data. Additionally, the architecture that we seek should have a well implemented fault tolerant mechanism and dynamic load balancing.
- We design the use cases carefully. We need to take into consideration the benefits and positive affects that online shopping and social networking have on each other. Thus, use cases should be designed accordingly with the involvement of social networking and online shopping relationships.
- Even in P2P architecture, our users should be experiencing a comparably fast and convenient social networking function. For example, if the time required to locate a friend in centralized systems is 5s, locating the same friend in our system should take 8s. We should try to eliminate operational delays in the distributed environment.
- We should have a system that supports e-commerce activities. The system should allow user to sell and buy products and be notified of products of potential interest.

## 1.3 Contributions

The traditional model where one service provider is responsible for the entire system is illustrated in Figure 1.2. This System is easily under the risk of overload when faced with

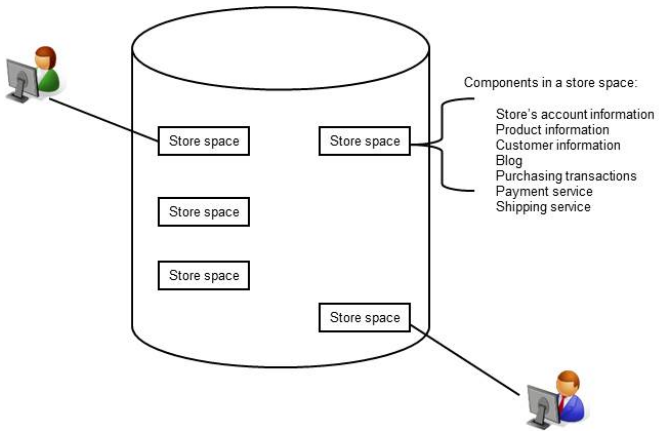



Figure 1.2: Centralized e-commerce model

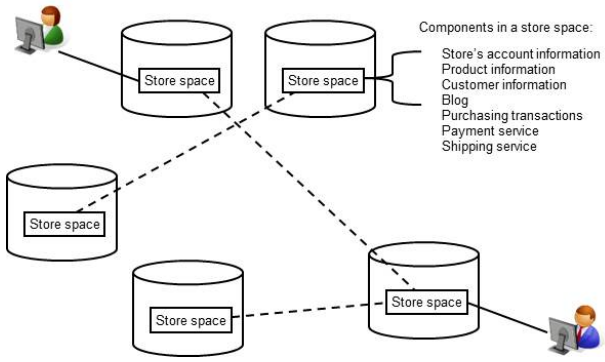



Figure 1.3: Decentralized e-commerce model

large-scale data and large amounts of processing requests per second. Our aim is to deliver the model illustrated in 1.3. In this model, we allow various organizations to contribute their resources to the system. With help from a third party, stronger computing ability and greater storage can be delivered to relieve the system pressure.

After examining numerous P2P applications, such as Chord, PeerDB and OpenDHT. We eventually selected BATON as our base to build on. We developed a platform for e-commerce called uopStore. The design of this system has a structure consisting of three layers. We developed the components that support this architecture as well as those that provide services to E-commerce activities.

BestPeer++ was originally used to support information sharing between different companies, with a focus on providing load-balancing and fault-tolerant services. To achieve that, the Bootstrap peer in BestPeer++ utilizes a large amount of storage space storing meta-data in order to manage the servers network.

However, BestPeer++ is designed as an information sharing platform for companies. We proposed a system with a three-layered structure by adding one more layer to existing 2-layer BestPeer++ structure. The newly added layer is the Client layer. The three layers of our system are: Bootstrap Peer layer, Server Peer layer and Client Peer layer. The Bootstrap peer in Bootstrap Peer layer monitors the status of the server peers. In the Server Peer layer, the server takes the form of the BATON overlay network. In addition to the services of BATON, these server peers also store data and provide services to users and support e-commerce activities. The Client Peer layer was created for the purpose of further support of e-commerce activities. This layer consists of client peers. Primarily, a client peer provides users with the interfaces needed to access the services on a server peer.

In the proposed system, e-commerce data is organized through a unified schema. We propose to use a multi-value function to generate the meta-data of the resources in the system. The generated meta-data is distributed on the servers in the server farm. This shared data is stored in the form of a key/value pair on the servers. The system searches this data by using the BATON lookup operation. By using this operation, we should be able to locate resource efficiently in order to well serve social networking functions and

e-commerce.

We designed and implemented use cases that covers social networking and e-commerce activities. The user is able to manage their account and make friends in our system. Furthermore, we designed and implemented the log in operation that allows users to have complete knowledge to talk to online friends. We also developed and implemented the e-commerce activities. As a customer, a user can search products by using meta-datas stored in server peers and then buying it. Users can also subscribe to stores of their choice and convey their areas of interest to the system. As a seller, the user can manage their store and their products. Sellers can add a new product then have the system forward the information to potential customers automatically.

## 1.4 Organization of the thesis

The following chapters are organized in the following manner. Chapter 2 presents the background knowledge of this work. It includes literature reviews of the research of typical structured P2P overlay networks as well as the data sharing applications based on these overlay networks. It also includes an overview of some popular e-commerce systems. Chapter 3 presents the functional requirements of the proposed system. The requirements are described through cases describing different types of use cases. This includes user management, store management, and product management. Additionally, this chapter presents the three-layered system architecture of the platform and describes the components that implement the services of this platform. Chapter 4 describes the distributed operations provided by this system. These operations satisfy the functional requirements of the system. The results from the experiments are also discussed. These results show that the implementations are correct. Chapter 5 concludes this thesis and discusses future work.

# Chapter 2

# Background and Related Work

The proposed system is constructed with a Peer-to-Peer overlay network to achieve the high scalability, fault-tolerance and robustness. In this chapter, we first present the structures of some typical P2P overlay networks. We then present the architecture of the data sharing platforms built on top of these P2P overlay networks. At the end, we present the functions of some popular e-commerce applications.

## 2.1 Structures of P2P systems

One main function of a P2P system is to locate a node that stores a resource or a data item upon request from the user. This task is processed by the distributed lookup operation of the system. The lookup algorithm of this operation is determined by the structure of the system overlay network. A peer node in an unstructured P2P system always forwards a resource lookup request to all of its neighbours. Such a lookup algorithm produces excessive amounts of traffic on the Internet by the act of message flooding. On the other hand, a peer node in a structured P2P system forwards a lookup message to the neighbours whose IDs match the lookup key to a certain degree. This kind of lookup algorithm performs more efficiently with less network bandwidth cost. In the following sections, we will describe the structures of some Distributed Hash Tables(DHTs) as well as their lookup mechanisms. We have included the BATON Tree in this subsection since it has been used as the P2P

infrastructure in the proposed system.

### 2.1.1 Distributed Hash Table (DHT) for key searching

A P2P Distributed Hash Table (DHT) system integrates a distributed hash table with the computing resources of its peer nodes. The hash table is stored in the storage spaces of these nodes. The nodes construct an overlay network. They forward (or route) the lookup messages in the overlay network. Compared to the unstructured P2P systems, the DHTs, such as Chord, Pastry [37], Tapestry [44] and CAN [35] sufficiently reduce the costs of searching and undergoing communications.

**Chord**: Chord is a widely known overlay network that has a circular topology. In Chord, the IP addresses of the peers as well as the keys of the data are hashed with the consistently one-way hash function. Through this method, a node or a data item is assigned an identifier in the one-dimensional identifier space. The identifier space must be large enough to render the probability of assigning the same identifier to different nodes negligible. The nodes are connected in a ring by the order of their IDs. Each node has a successor and a predecessor. A node stores the data items with the keys in the range between the IDs of its predecessors and itself. In Chord, a node also has connections with the nodes whose IDs are the first beyond the distances of 2n. The structure of Chord is shown in Figure 2.1

Chord has a lookup mechanism that uses the search algorithm of the skip list. Right after receiving a request, a node first compares the key contained in the message with its own ID. It sends a reply to the requester of the query, in the case where the key is located in its data range and the data is stored locally. Otherwise, it forwards the request to a neighbour who is the first successor of the saught after key. In the case where the node discovers that the key is not in its own range and is in fact larger than the IDs of its immediate successor, it identifies that the resource does not exist in the system. Through this manner, Chord guarantees a worst case search result where a request is forwarded with logarithmic hops and the routing table stores a logarithmic number of neighbours. The joining procedure for a new peer also uses this lookup algorithm to locate the predecessor

and the successor of the new peer in the ring.

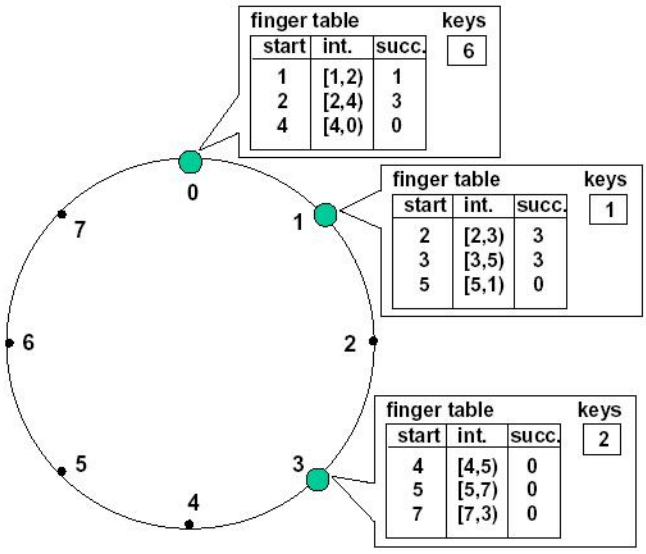



Figure 2.1: Chord Finger Table ( Copied from Figure 3 in [40] )

**CAN** 2.2: It is a scalable, fault-tolerant and self-organizing Distributed Hash Table with a structure built on a virtual d-dimensional Cartesian coordinate space. Each peer is assigned a zone with partitioning. The zone is defined by the ranges. One range is allotted to each dimension. The keys of the data items are also hashed. A key refers to a point in a zone and should be stored on the node that is in charge of the zone. The lookup algorithm for a data item is required to find the node whose zone contains the key of item. In order to locate the node for the item, a node forwards the request to the neighbour whose coordinates are the closest to the requested key. The forwarding process is continued until the node taking charge of the key has been located.

**PRR Trees, Pastry and Tapestry**: Pastry and Tapestry have similar routing algorithms. Both of them are built with PRR Trees. As Chord introduced before, Pastry also guarantees a lookup procedure with O(logN) forwarding steps for the worst case scenario at the stable state where there are few nodes joining or leaving. Normally, a peer in Pastry has a unique 128-bit peer identity (called peerID) obtained by hashing its IP address. The peerId is viewed as a number in base 2b. The entry in the cell $(i, j)^{th}$ ID of the routing

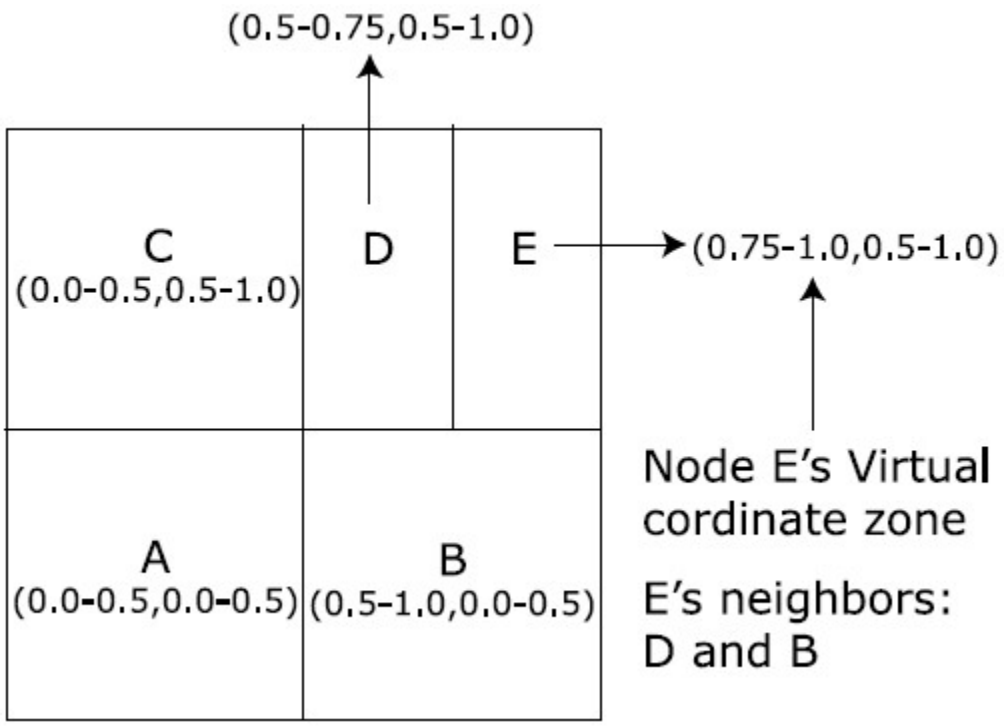



Figure 2.2: CAN Resource Overlay network ( Copied from 3.9 in Chapter 3 in [42] )

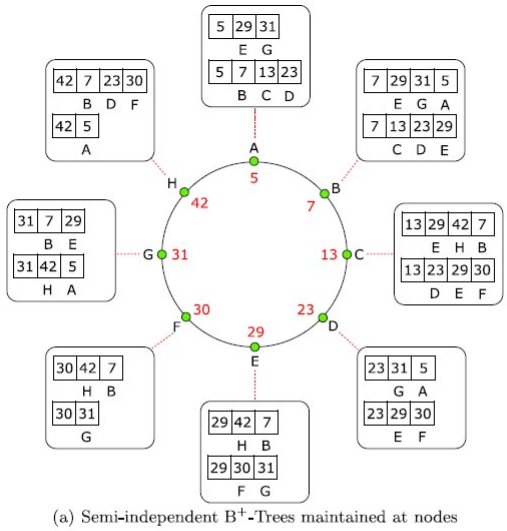
table is the address of the target peer. The target peer agrees with the peerID in the first i bits and with the peerID whose $(i+1)^{th}$ bit is j. By using the routing table, a node can efficiently route a message to a neighbour whose peerID is numerically closest to the key.

### Structured P2P overlay networks for range query

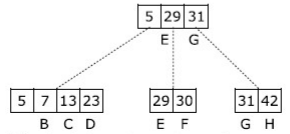
In the Distributed Hash Table(DHT) based systems, peer nodes and data are hashed into the same identifier space by using a hash function such as the secure hash algorithm-1 (SHA-1) [22]. However, hashing does not maintain the order of the data. Therefore, the effort to support range query is a challenge for such kinds of P2P systems. Locality sensitive hashing [20] and locality preserving hashing [9] are proposed to solve this issue.

**SkipGraph** [11] is a tree data structure that is based on Skip List [33]. However, it organizes data by order of multiple doubly linked lists. Therefore, compared to Skip List, Skip Graph is highly resilient and tolerant to the loss of the network connectivity caused by the failure of a large fraction of nodes. In summary, Skip Graph has a randomized and

distributed tree-like index that can equally and efficiently answer query and range queries in a P2P system.




(a) Semi-independent $B^+$-Trees maintained at nodes




(b) The fully independent $B^+$-Tree at node A

Figure 2.3: P-tree structure ( Copied from Fig.3.18 in Chapter 3 in [42] )

**P-tree** [17] is a P2P system that also has a distributed and fault-tolerant index structure. P-tree can be used to answer queries regarding equality, range, prefix match and the nearest one-dimensional neighbour. The architecture of P-tree is shown in Figure 2.3. The nodes in a P-Tree construct a virtually balanced B+ Tree on top of a Chord ring. Each peer node maintains a Chord node as well as a semi-independent B+-Tree. A semi-independent tree is a part of a fully independent B+-Tree. The value stored at the peer is considered as the smallest value in the Chord ring. The semi-independent B+-Tree located on a peer contains all peers in the leftmost root-to-leaf path of the corresponding fully complete B+-Tree.

| | |
|---|---|
| `join(P)` | Join the network |
| `leave(P)` | Leave the network |
| `put(k, v)` | Insert a key-value pair into the network |
| `remove(k, v)` | Delete the value with the key |
| `get(k)` | Retrieve the value with the single key |
| `get(begin, end)` | Retrieve values with the key range |

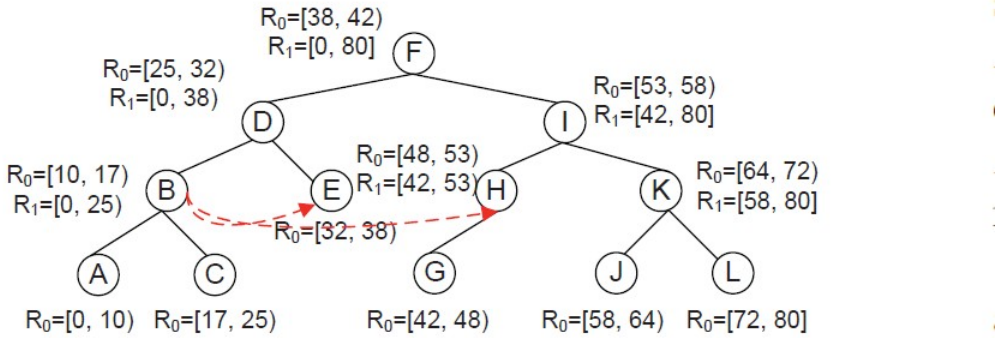



Figure 2.4: BATON Overlay ( Copied from Fig.3 in [14] )

### BATON Tree Structure

The proposed system uses the structure and protocols of BATON Tree to organize its server peers. BATON tree is probably the first P2P system that uses a tree structure for the topology of its overlay network. The structure of BATON is shown in Figure 2.4. The idea of supporting range queries in BATON Tree is inspired by [18].

### Peer Connections in BATON Tree

The peers in the overlay network are organized in a tree structure. Every peer has a uniquely logical position in the tree. A position is identified by a level (starting from 0) and an identifier (or ID) number (starting from 1 to $2^l$ at level l). For instance, at level L, the maximum ID number of the peers is $2^L$. There are four kinds of essential connections in the overlay network. For example, for a peer p, the connections are listed as written below.

- Parent Node Connection: this connection provides a communication link between a peer and its parent node.
- Child Node Connection: still to the parent node connection, this connection provides the physical and logical information of its children.
- Connections in the routing table: the routing table stores the information of each connection, including their physical information and the logical information. The routing table also stores the resource ranges and the information pertaining to children of neighbours at the opposite ends of these connections. Each peer has multiple left or right neighbours. Left (or right) neigbhours are the peers whose IDs are $2^i$ smaller (or bigger) than that of the peer p at the same level. The i exists in a range from 0 to level number L.
- Adjacent node connections: a node also has connections with the left or right adjacent node. These connections dynamically change in keeping with the occurrences of node joining or leaving. The left adjacent node of a peer is always located on the left side while the right adjacent node is located always on the right side. This is the case for a node who have full children who will in turn have adjacent nodes but no children.

**Joining**

A new peer needs to know at least one node in the BATON network overlay for joining to occur. To locate the position of that new node, peer nodes forward the joining request message in the overlay network. In the case that a peer p receives a request to join from q, p will first check if it has full children or not. If this is not the case, p forwards this request to parent. P checks if it has a child; if not, it inserts q as its child. P, forwards this request to one of its adjacent nodes with a higher level node in the tree. If p accepts q as its left child, p splits the range it is responsible range with q; and p updates the left adjacent link, placing q as a left adjacent node. If p is the right child of its parent's, p tells

q to set left adjacent node to k and to set right the adjacent link to q. P then notifies the peers that exist in its routing table that its range has been changed.

### Searching

The exact value search in Baton tree is depicted in Figure 2.5. For example, peer p receives a resource search request with identifier g. P first checks if g is within its range or not. If it is within its range, p processes the query locally. Otherwise, it checks if it has a value larger than the minimum value of one of the peers in its left routing table or smaller than the largest value of one of the peers in its right routing table. If so, p forwards this request to the respective node. Otherwise, p will forward the search message to its left adjacent node if g is smaller than its minimum value; or, p will forward the search message to its right adjacent node if g is larger than its maximum value.

```
Algorithm: search exact(node n, query q, value v)
If ((LowerBound(n)<=v) and (v<=UpperBound(n))
  q is executed at x [1]
Else
  If (UpperBound(n)<v)
    m=TheFarthestNodeSatisfyingCondition
      (LowerBound(m)<=v)
    If (there exists such an m)
      Forward q to m
    Else
      If (RightChild(n)!=null)
        Forward q to RightChild(n)
      Else
        Forward q to RightAdjacentNode(n)
      End If
    End If
  Else
    //A similar process is followed towards the left
  End If
End If
```

Figure 2.5: BATON Exact Value Search Algorithm

## 2.2 P2P Data Sharing Applications

### 2.2.1 Query Processing with Heterogeneous Data Sources

Some data sharing application systems, such as BestPeer++, PeerDB, and Piazza, are constructed and deployed in a Peer-to-Peer system overlay network which is used as their infrastructure. Piazza enables the integration and exchanging of data between heterogeneous data sources. The structures of the data in these sources are defined with different schemas. In Piazza P2P networks, each peer maps its schema onto its neighbors. Peers initiate queries based on their local schema. The query is rewritten according to the schema mapping and sent to neighbours so that it can be answered by other peers. The query roams in the P2P network and continuously sends results back during its life-time. Another P2P system that we have considered is Hyperion [10]; this employs mapping tables to solve the problem of mapping heterogeneous data sources. Two main issues arise from the use of this method: the matter of creating mapping tables and the method by which queries are processed based on mapping tables.

### 2.2.2 BestPeer++

BestPeer++ is a large-scale data processing platform with BATON overlay network (reviewed in Section 2.1.2.2). Using BATON, BestPeer++ locates data efficiently and is also fault tolerant. Furthermore, it supports data sharing between companies by using a mature technique. It has integrated with the Amazon Cloud EC2 to provide the data locating, sharing and processing services to business companies.

As we can see in Figure 2.6, there are two kinds of peers in the overlay network: bootstrap peers and normal peers.

- The Bootstrap peer is the entry point of the whole network system. It has several responsibilities which are listed as follows.

    1. Monitoring and managing normal peers: it is responsible for the auto-fail and

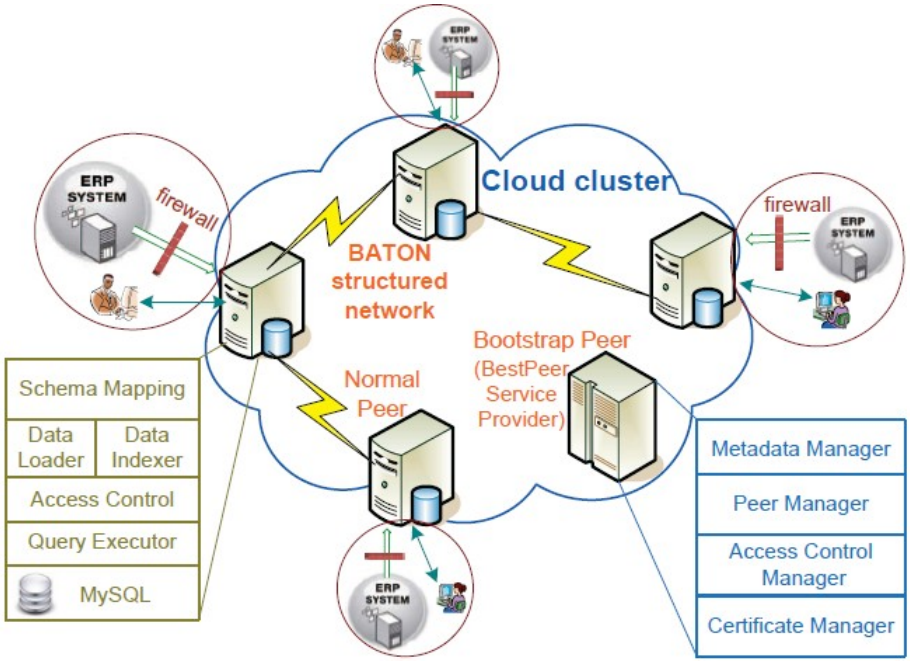



Figure 2.6: BestPeer++ topology ( Copied from Fig.1 in [14] )

auto-scaling function. This function can replace a failing node or upgrade an over-load node in order to guarantee the services of the infrastructure.

2. It must schedule various network management events.
3. It acts as a central repository for the meta-data of the corporate network applications, when the server peers are failing or are overloaded.
4. It is in charge of the PKI data encryption/decryption transmission between normal peers.

- Normal peers may be understood as the instances that manipulate businesses; this means that a normal peer represents a company in this system. Normal peers are organized in the BATON [24] overlay network.

To achieve the auto fail-over and auto-scaling function, the Bootstrap peer periodically collects the status information from normal peers. Once it has realized that a peer has failed, it will perform the fail-over function to create a peer in Cloud. The new peer will then take the position of the failing peer in the overlay network and it will place the failing

peer in the black-list. Similarly, if the bootstrap peer discovers that some peers are over loaded, it will trigger the auto-scaling function to upgrade the capability of the overloaded peer.

In BestPeer++, the data items are stored on normal peers. For a request placed on a data item, BestPeer++ uses the lookup operation of BATON to locate the nodes storing the item. In BATON (shown in Figure 2.4), each normal peer has sufficient knowledge of its neighbours; this includes information about its parents, children and the selected neighbours. Each peer is responsible for a certain range of resource indexes; and, each index points to a resource in the system. During a search process for a query with a key, a user will first invoke the query on its normal local peer. The peer will then check if the requested resource is locally available. If the requested resource does not exist locally, the peer will forward this query to its neighbours whose data indexes a clue for the location of the resource.

BestPeer++ is a system that for the most part provides elastic data sharing services. Additionally, it can handle typical workloads in a corporate application.

### 2.2.3 Piazza

Piazza[21] enables the data integration and exchange between heterogeneous data sources. As shown in Figure 2.7), each peer maps the relational schema of its local data (i.e. local schema) to the schema of its neighbours. For a request, a user first initiates queries based on the local schema of a peer. The queries are rewritten into different messages according to the schema mapped between the local node and its neighbours. These reformulated query messages are then sent to the neighbours. The neighbours either answer the query or forward it further. Through this way, the query roams in the P2P network and continuously sends results back during its lifetime.

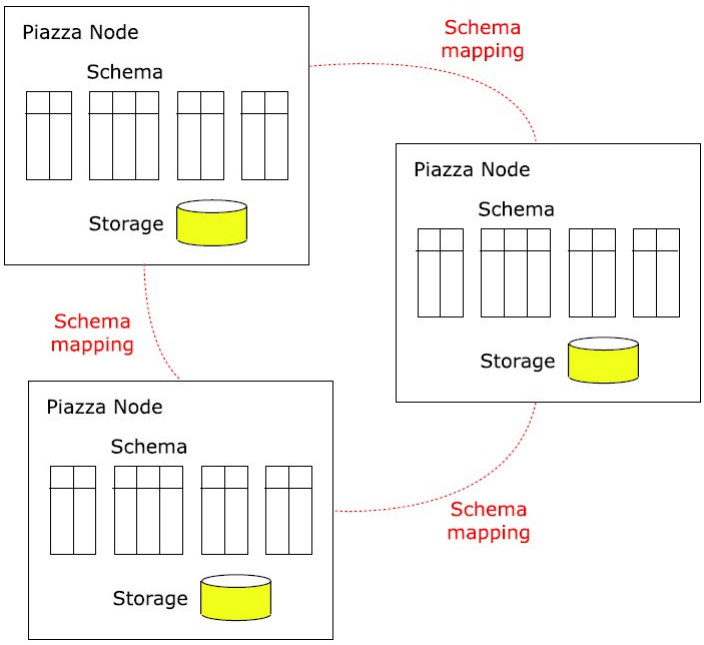



Figure 2.7: Piazza structure ( Copied from Fig.4.16 in [42] )

### 2.2.4 PeerDB

PeerDB [31] is a P2P-based system for the distributed sharing of relational data. Similar to Piazza, PeerDB does not require a global schema. Unlike Piazza, PeerDB does not use schema mapping between peers. Instead, PeerDB employs an Information Retrieval based approach to reformulate queries. In this approach, the relation of a peer (and of each of its columns) is associated with a set of keywords. When a query is put forth over a peer scheme, PeerDB reformulates the query into the schema of other peer by matching the keywords associated with the two schemas. The resulting and reformulated queries may not be semantically meaningful. The system requires user input to decide which queries are to be executed. Consequently, PeerDB is able to route a query message without following a semantic path.

PeerDB system runs on top of BestPeer [30] It asks its users to specify the keywords for the tables and columns at the time of creation. These keywords are descriptive and can be used to find the matching tables for a query during query processing. The basic idea of this solution is to apply an information retrieval method to find tables, whose descriptive

keywords, in addition to the descriptive keywords of the columns, are best matched to queried tables and columns. Based on relevant tables retrieved, corresponding queries can be created for execution.

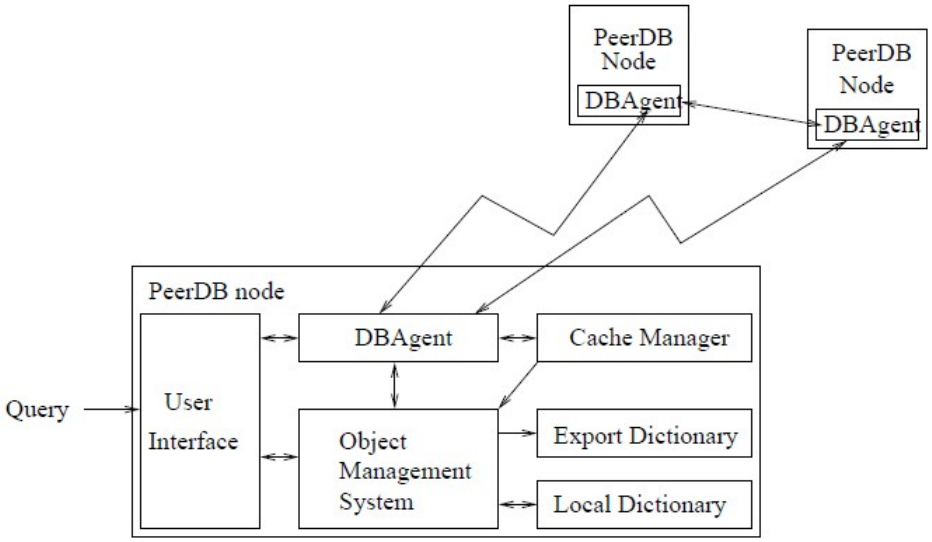



Figure 2.8: PeerDB node architecture ( Copied from Figure.1 in [31] )

PeerDB is probably the first reported platform on a database sharing system that facilitates data sharing without a predetermined shared schema. It combines mobile agents and P2P technologies for the large-scale resource sharing of raw data as well as processed information. However, the underlying assumption of PeerDB is an unstructured overlay network with no indexing mechanism. Data discovery is thus achieved by flooding requests to all nodes. It follows that PeerDB inherits all the disadvantages of Gnutella-like resource discovery systems.

### 2.2.5 PeerSoN: a P2P infrastructure for social networks

PeerSoN [12] is a unique solution that provides a fully decentralized P2P social networking service to shake the domination of the centralized social networking services. Its design is motivated by the fact that a centralized platform cannot preserve the privacy of personal information and data. Centralized social networking services are always criticized for the high risk of leaking users' private information and the possibility that data may be leaked.

In PeerSoN, each peer has a unique ID. These peers are organized with the Open-DHT [36] overlay network. The overlay network of PeerSoN is described in Figure 2.9.

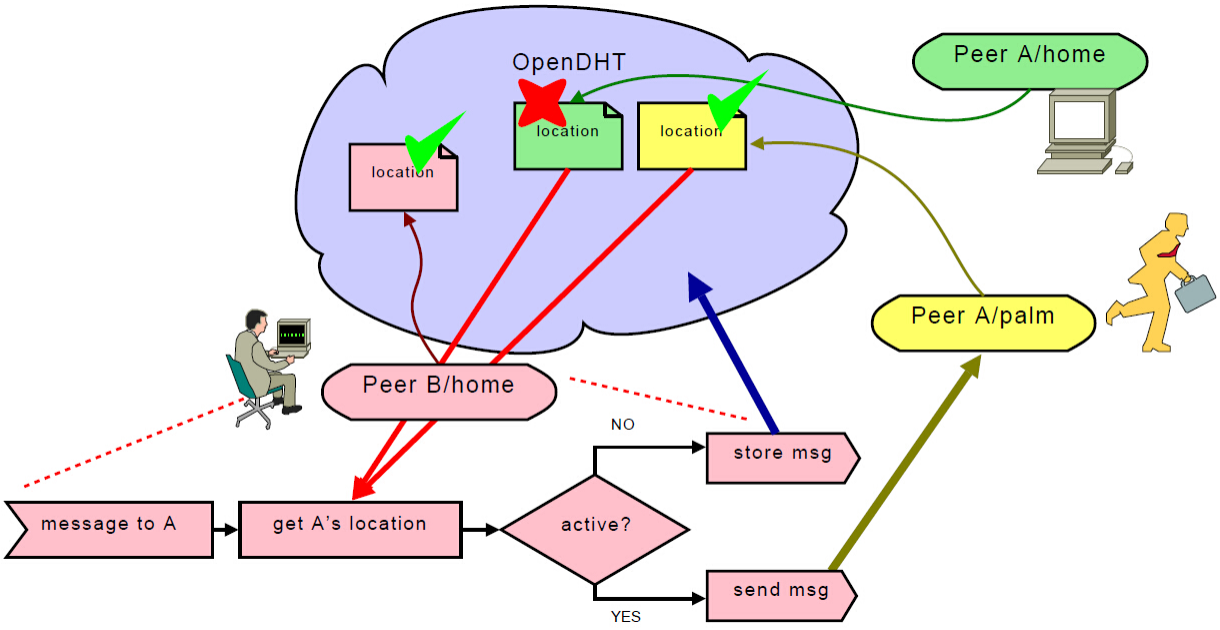



Figure 2.9: PeerSoN architecture ( Copied from Figure 3.1 in [12] )

### 2.2.6 Hyperion

Hyperion solves the problem of mapping data in heterogeneous data sources in a way that in different from Piazza. The architecture of Hyperion is shown in Figure 2.10. Each peer has a mapping table. A mapping table contains a set of data associations between the values of two different peers. These associations can be used to transform the queries between two interacting peers that have different data structures.

The mapping table needs to be built by the experts who know the schema of the data of each peer in a transaction. A mapping table needs to record both the mapping values and the confidence level of experts concerning mapping. The confidence level of experts can be categorized into four different sections.

- Open-open-world (OO-world): this mode is met with the lowest amount of confi-

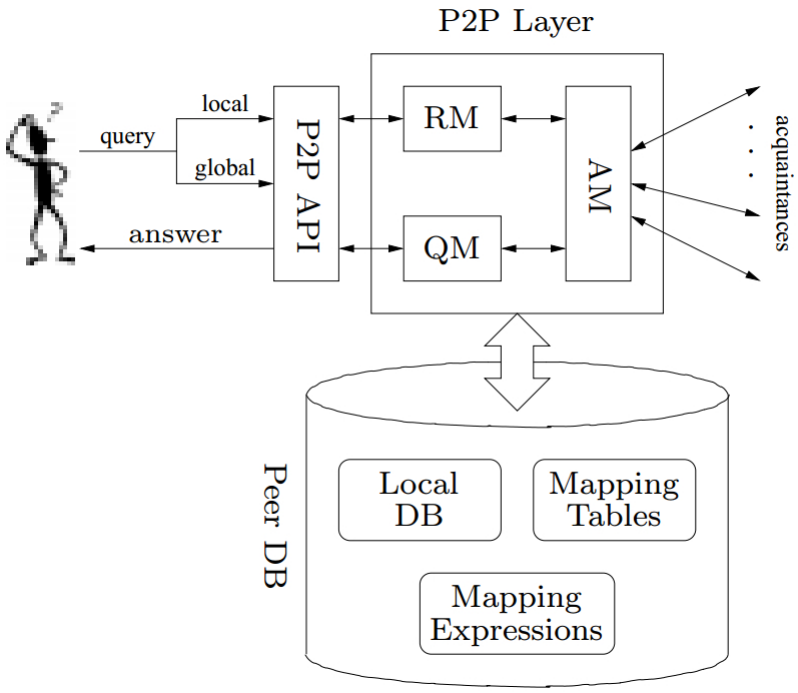



Figure 2.10: Hyperion structure ( Copied from Figure 3 in [10] )

dence. The mapping implies that a value x can be associated with any value y whether x exists in the mapping table or not. This mode is used when the mapping table creator does not have a clue about the mapping at all has absolutely no knowledge of the mapping. Actually, this mode does not involve in practice. In practice, this node is actually not involved.

- Open-close-world (OC-world): this mode implies that if a value x does not exist in the mapping table, there is no associated value y. However, if x exists in the mapping table, x can be associated with any value y. This mode is used when the mapping table creator only knows the domain value of X. As a result, it is not very useful in the query processing.

- Close-open-world (CO-world): this mode implies that if a value x exists in the mapping table, it is associated with an exact value y. However, the mapping table creator does not know the mapping values of the missing values. In general, this mode represents partial knowledge of the domain.

- Close-close-world (CC-world): this mode has the highest level of confidence. It represents the complete knowledge of the domain. If a value x exists in the mapping table, it is associated with an exact value y; however, if x does not exist in the mapping table, there is no associated value y.

No expert can have complete knowledge of the domain especially when the system involves several peer nodes. It is therefore necessary to infer new mapping tables automatically from existing mapping tables and to check the consistency of the mapping tables created by different experts. Essentially, new mapping tables can be inferred by combining existing values in data sources and by using mapping tables as constraints to filter invalid mapping values. On the other hand, the consistency of mapping tables can be verified spontaneously from other mapping tables and the values of related data sources. The algorithm for generating new mapping tables automatically consists of two phases: information gathering and computation.

## 2.3 Auction

Auction refers to a the process of buying and selling goods or services by offering them up for bids, taking bids and by selling them to and accepting them based on the the highest bidder bid. Auctions exited existed thousands of years ago ; and, they are still an important part of today's E-commerce domain activities. There are several common auction models that have been widely adopted, for instance, the English auction model, the Dutch auction model and the Second price auction model [27].

### 2.3.1 English Auction

The English Auction is a type of traditional auction. It has been widely used in the e-commerce system, one example being eBay. At the beginning of an auction, the auctioneer sets a suggested opening bid for an item on sale and sets the deadline for accepting bids. The auctioneer then starts to gather bids. When the deadline arrives, the auctioneer checks

the bids to determine whether the highest price for this item is acceptable. In the case that the auctioneer considers the highest price a proper offer, the auction is finished; and, an E-commerce transaction processing the deal is initiated.

There are many variations for this of this kind type of auction. For example, in some kinds of auctions, the reserved price is not revealed. Bids may, moreover Moreover, bids may be made with signals instead of by being called out. Such signals can be the tugging of an ear or the raising of a bidding paddle. Another variation on of the English Auction is the open-exit auction; here Here the bidders must explicitly announce that they are dropping out of the bidding and they are not allowed are not permitted to re-enter at a later time.

Existing large E-commerce applications maintain their auction mechanisms on large server based infrastructures. At the research level, PeerMart [23] is an auction-based market on a P2P overlay network; it can be used by peers to efficiently trade services. PeerMart provides a scheme that enables double auctions in the overlay network. As we have learnt from Industrial Economics [20], the English auction, the Dutch auction and the second unit price auction are still popular in today's E-Commerce activities.

### 2.3.2 Dutch Auction

The Dutch auction also has a long history and still maintains popularity until today. The procedure of the Dutch auction will follows the steps listed below:

- An auctioneer sets a fixed-time for biding and, at first, an extremely high price for a product on sale. The auctioneer then waits for the bid-peers to bid for the product; and, the price keeps going down periodically at intervals until the auction ends.
- The auctioneer waits for the bid-peers to bid for the product. The auctioneer may adjust the speed at which the price decreases.
- A bid-peer who considers the current price is acceptable bids for the product. The auction is terminated in the event of the bid-peer agreeing to purchase the product

with the price at the respective time interval. The bid-peer is authorized by the auctioneer. Otherwise, the auction will be terminated at the end of the biding time.

There are several existing Dutch auction implementations in e-commerce applications. In [28], the authors proposed to use an agent to adjust the price based on different auction market environments. In [26], the author proposed a strategy that helps the seller selling the product at a maximum price in an auction.

### 2.3.3 Second-price Auction

A Second-price auction is very similar to a Dutch auction. Some financial commentators and some third-party auction sites use the term Dutch auction to refer to Second-price auctions. However, in a Second-price auction, the seller offers more than one identical item for sale, in this case there may be more than one winning bidder. A bidder may bid for all the items or for only some of them by publicly indicating the price for each item that he/she is willing to pay. However, all winning bidders need to pay only the lowest qualifying (successful) bid. If the successful bids are more than the available items, priority goes to the bidders who submit their bids first.

## 2.4 Existing E-commerce Applications

### 2.4.1 eBay

eBay is an American multinational Internet consumer-to-consumer corporation and its head-quarters are located in San Jose, California. It was founded by Pierre Omidyar in 1995; and, it became a notable success story of the dot-com bubble. It is now a multi-billion dollar business with operations localized in over thirty countries. The company manages eBay.com, an online auction and shopping website through which people and businesses buy and sell a broad variety of goods and services worldwide.

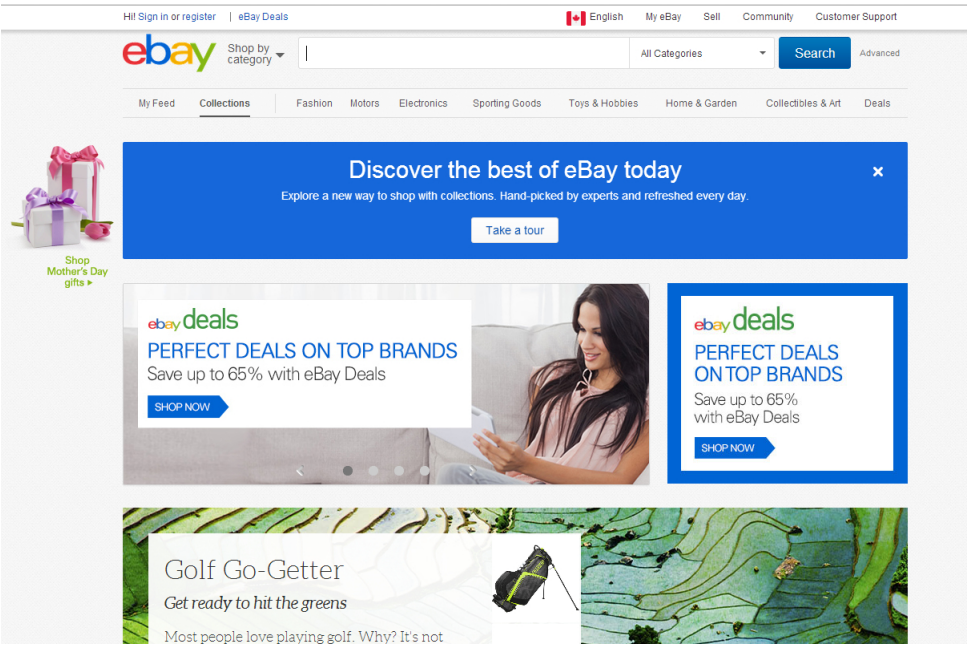



Figure 2.11: ebay.ca website interface

eBay has achieved huge success in the online shopping domain. When it comes to online shopping, eBay dominates in the mind of consumers. There is an auction study that focuses on eBay [39]. We study the auction model, mostly the English auction model from eBay.

eBay is successful because it provides a secure shopping and selling environment by bringing together buyers and sellers worldwide. eBay also has been successful due to the latest technology that is used to carry out transactions.

### 2.4.2 Amazon

Compared to other e-commerce sites, Amazon probably has the most extensive history. It was founded in 1994. The company provided a customer reviewing and grading mechanism for customers to choose and evaluate the sellers selling items in their stores [29]. The Amazon website has attracted a large number of customers. Also, the customers usually pay the lowest prices for products compared to prices from other vendors.

Amazon is the world's largest online retailer. Amazon.com started as an online book-

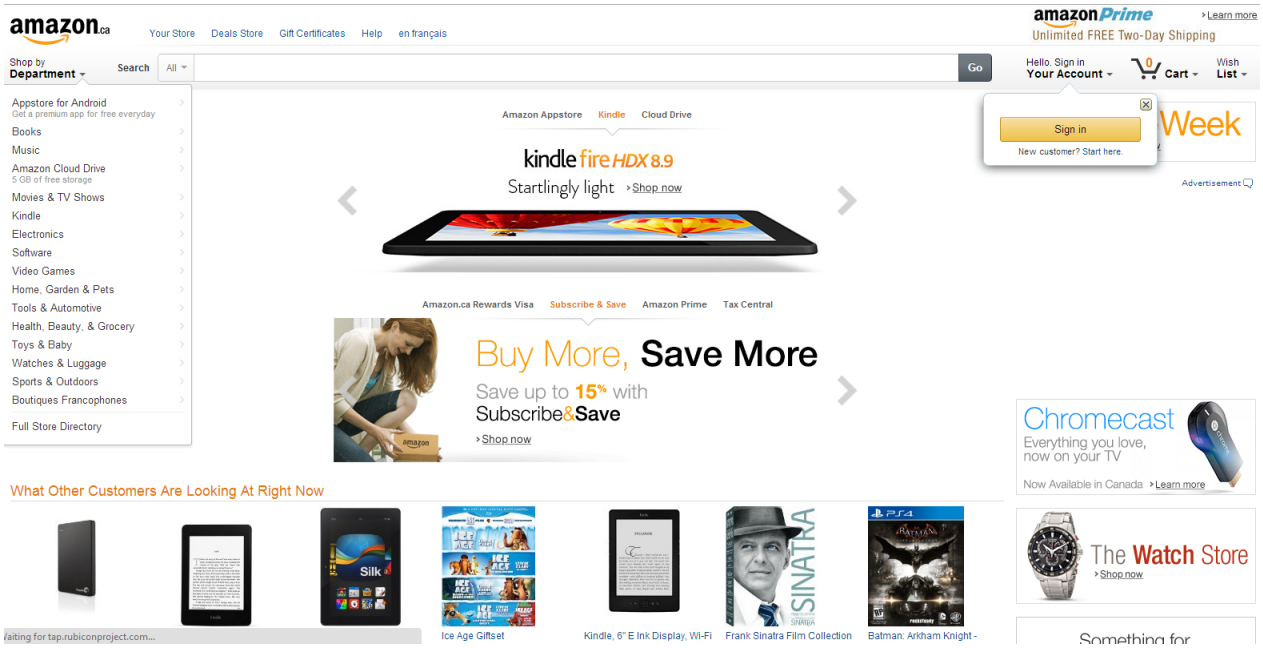



Figure 2.12: Amazon.ca website interface

store; but, it soon diversified by selling DVDs, VHSs, CDs, video and MP3 downloads/streaming, software, video games, electronics, apparel, furniture, food, toys, and jewellery. The company also produces the E-book reader Amazon Kindle and the tablet computer Kindle Fire. Furthermore, it is a major provider of cloud computing services. Amazon and eBay are the two largest online market-places in the world. However, Amazon is a centralized online shopping center. The key concept battle between Amazon and eBay is auction vs. fixed price. However, eBay does not take inventory; and, it prefers to be an intermediary that facilitates C2C commerce. The eBay strategy dominates in its operation of the decentralized e-commerce model. Amazon, however, succeeds at the centralized model due to its own scale, efficiency, and the advanced technology. Its customers consider the price from Amazon hard to beat.

### 2.4.3 Taobao

Similar to eBay and Amazon, Taobao is a Chinese website for online shopping. It has been operating in the People's Republic of China under the Alibaba Group since May

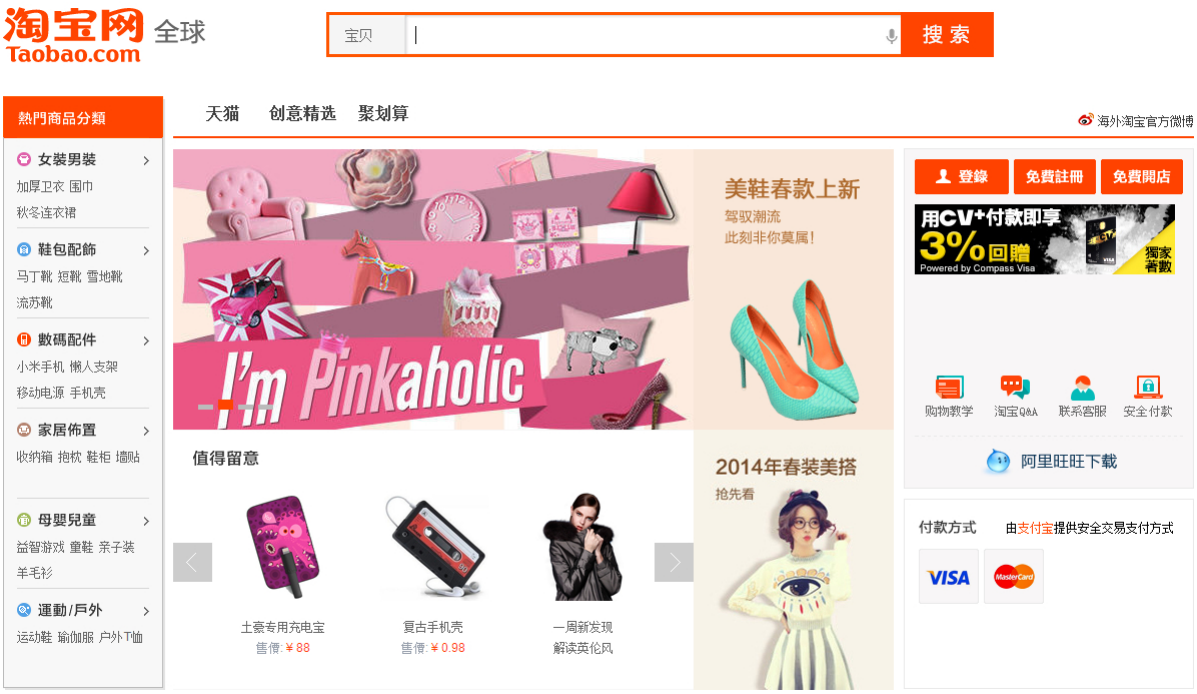



Figure 2.13: Taobao website interface

10, 2003. Taobao Marketplace is a platform that facilitates consumer-to-consumer (C2C) retail. This platform enables small businesses and individual entrepreneurs to open online retail stores for consumers in China, Hong Kong, Macau and Taiwan. In March 2013, with around 760 million products listed, Taobao Marketplace was one of the world's top 10 most visited websites according to Alexa. At the end of March 31, 2013, the combined gross merchandise volume (GMV) of Taobao Marketplace and Tmall.com exceeded 1 trillion yuans. The sellers are able to post new and used goods for sale or for resale on the Taobao Marketplace either with a fixed price or as an auction. While auctions make up a small percentage of the transactions, the overwhelming majority of the products in the Taobao Marketplace take the form of brand new merchandise; and, these products are sold at fixed prices. Buyers can assess sellers' backgrounds available on the site, such as information pertaining to ratings, comments and complaints.

With its structure described in [45], Taobao has already become the largest online shopping market and has gained 70 percent of an on-line market share in China since 2010.

It is a web application that provides a mature and reliable platform for online shopping. Taobao's great success not only places it on par with the world's top companies, but also provides many employment opportunities in China by enabling individuals to open online stores where they can sell items. Meanwhile, due to the large number of goods-related transactions induced by the success of online shopping, the Chinese mail system has also benefitted tremendously.

However, Taobao's platform is still based on the traditional client/server architecture.

### 2.4.4 Shopify

Shopify is a new E-commerce platform that provides its customers with a convenient approach to create creating their own stores and to sell goods selling goods. This approach simplifies the procedure of creating an online store and cuts down the expense as well also cuts down the expenses associated with it

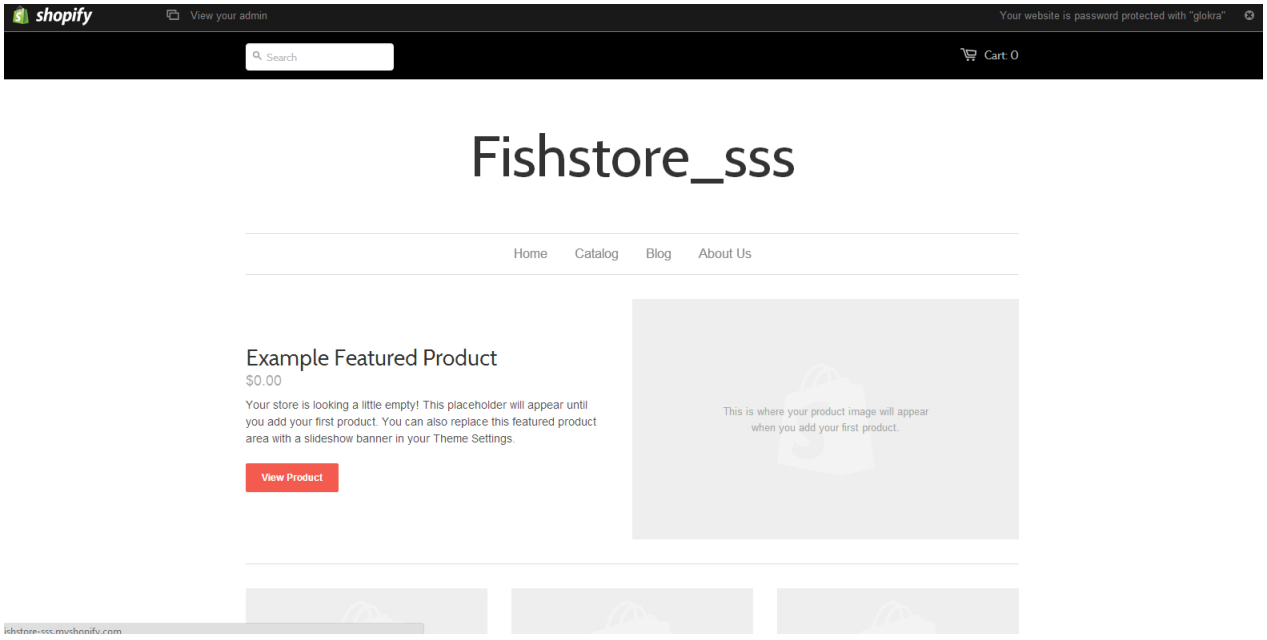



Figure 2.14: Shopify website interface

## 2.5 Summary

There have been a number of P2P overlay networks and data sharing applications based on these overlay networks that have inspired us. For instance, Chord, CAN and SkipGraph have already provided practical ways to build a Peer-to-Peer network architecture with efficient resource locating abilities. We chose BATON as the overlay network infrastructure to organize the server nodes in our system; this is because it has an efficient resource locating and fault-tolerant nature. BestPeer++ provides an application built with the nodes in the BATON overlay network. By using data mapping, BestPeer++ is an impressive platform for data sharing between businesses. Through integration with a number of advanced techniques, BestPeer++ has been developed into a commercial product. The examples of query processing with heterogeneous data sources shows us how to merge with the largest e-commerce service providers in the future.

# Chapter 3

# System Design

In this chapter, we present system requirement and architecture of the uopStore e-commerce platform. The proposed system has a hybrid P2P architecture with a server farm constructed on a Peer-to-Peer overlay network. We start by introducing the structure of the system design. The components of different types of peers are described in detail at the end.

## 3.1 System requirements

### 3.1.1 Supporting social activity

With the rising tremendous success of social networking services, the inclusion of the components of these services in other types of applications, such as online shopping, gaming, or job hunting, has became a trend. For example, Alibaba TradeManager [1] in Taobao, LinkedIn [19] and the Playstation network, all have their methods of facilitating the building of social connections for their users. From the perspective of the social networking services, there is also evidence that suggests e-commerce is trying to merge with these services. Not only do lots of users make advertisements on their homepages in Facebook and on Twiter, but business companies are also interested in spending money on such advertisement. In [13], it made evident that small businesses are also very effective at using

these services for their own businesses.

However, the inclusion of social networking services in e-commerce applications may lead to some issues for users. For example, the users of social networking services may be exposed to advertisements that they do not have any interest in [7]. In [41], the author claimed that the use of the social networking services will not increase the number of e-commerce transactions. They showed that the social network of a buyer has little impact on his decision making. However, a seller can receive higher customer satisfaction from the transactions with his friends. In [25], the authors' study showed that the social network of a buyer has a positive impact on his intention to view items and purchase them. Overall, there is no strong experimental results that show whether the social network significantly benefits transactions in e-commerce.

There are other factors that affect e-commerce activities. For instance, a price war may have an impact that is bigger than the impact of social network [15]. It has been shown that an e-commerce application combined with a social network increases the trust between a buyer and a seller. Trust plays a huge role in the C2C business model. According to [32], the reason that eBay was lost to Taobao in China is because it hardly addresses the trust issue. Increasing the trust between a buyer and a seller ultimately plays an important role in e-commerce activities although it may not have a significant impact on the number of transactions.

The benefits of including a social networking service in an e-commerce application are listed below.

- This helps to develop a good relationship between a buyer and a seller: over a long-term period, a good relationship can lead to a win-win situation for both sides.
- Benefit from user profile: user profiles may provide better information regarding market targeting to a seller. Marketing with user profiles potentially offers an effective approach to providing customized advertisement [43]. However, due to user privacy protection, exposing user profiles for better targeting decisions should be handled carefully.

- Forwarding business advertisements through social networking services: this approach improves the efficiency of advertising [16].
- Supporting basic online shopping activities: a customer should be able to find a product in the system and to buy the product as well.

We have listed the drawbacks of integrating social networking services with e-commerce applications as shown below:

- Challenges present in preventing users from receiving excessive advertisements:

  it is almost impossible for users to avoid useless advertisements on a social network website. We can develop a mechanism to manage the advertisements and to decrease the ratio of useless advertisement promotion. However, a practical advertisement pushing method is still lacking in e-commerce application integrated with social networking services.
- The requirement for a trust model that is more complex than the one present in a traditional e-commerce system: we need to consider the review of friends against random customer reviews.

After considering the drawbacks and advantages mentioned above, we have decided to integrate the basic functions of a social networking service so that the proposed system can provide a better online-shopping experience to customers and sellers.

### 3.1.2 Use cases

The framework of the proposed system contains the components that implement the basic functions of a social networking service. In addition, we need to address the generic functions of an e-commerce application. We should consider that a user will play the role of a customer or a seller. With regards to the cooperation between social network service and e-commerce applications, we consider the functions of advertisement recommendations and store subscriber management.

The requirements can be classified into four categories shown follows:

- Account management: a user should be able to register in the system and to open an account there. The user should be able to log into the system with his username and password and to retrieve the confidential information thereafter.
- Basic social networking service: when a user is on-line, he should be able to identify the on-line statuses of his friends and to chat with them through the instant message service.
- Basic business management service: a user should be able to create his store and to manage it. He should be able to add products to his store and to manage them as well.
- Basic online shopping service: a user should be able to find a product, buy a product in the system and to further follow the progress of store as well.

### 3.1.3 User account management

The account management use cases are illustrated in Figure 3.1


Figure 3.1: Account management use cases

**Log in**

- **Pre-condition**

  A user has an active account in the uopStore system.

- **Story**

  A user logs into the system with the value of the username/password pair; and, thereafter, he receives the information mentioned below:

  - Online friends' physical information (e.g., IP Address and port).
  - Advertisements of potential interest.
  - If a user has already registered for a store in the system, the user should receive the information for the store (e.g., store status, product lists and invoice lists).

- **Post-condition**

  The user is able to use the social networking functions now. Online shopping is also available for the user. The user may create or manage his stores.

**Register**

- **Story**

  A user should be able to join the uopStore by placing a registration request with the user's profile information. The profile should at least contain:

  - A unique username.
  - A password.
  - An e-mail address, for password resetting purposes. A user who want to open a store should fill in the payment information to enable the online payments.

- **Post-condition**

  If the register is successful, the user information will be written into database. The user can log into the system with the value of username/password pair.

**Update account**

- **Pre-condition**

  A user should have an active account in the system already; and, he must have already logged on.

- **Story**

  A user should be able to update his user account by:

  - Resetting the password.
  - Updating the profile information.
  - Updating the email address

- **Post-condition**

  If the update is successful, the user information will be overwritten into database. And user will have a choice to notify the friends immediately or wait the other user needs log in to notify.

### 3.1.4 Business management services

The use cases for the basic business management service are described in Figure 3.2.

**Create a store**

- **Pre-condition**

  A user should have an account in the system already. He also must have the direct deposit information stored in his profile, like the account name in payPal.

- **Story**

  A user should be able to create a new store for operating online business activities. He needs to set the following information in the system:

  - Store name.

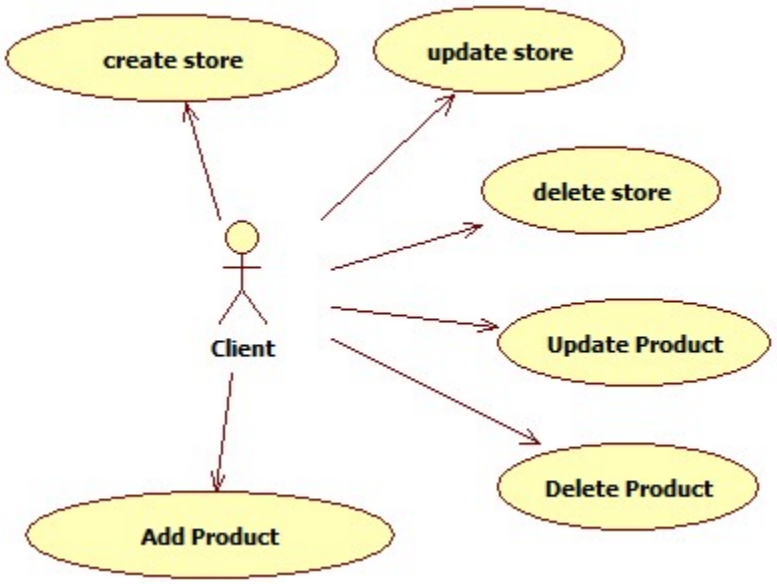



Figure 3.2: Basic business management use cases

– Store description. A user can also put in the product to sell at this time.

- **Post-condition**

  A new store associated with the user will be created and stored in the database. The user can handle store management and product management.

### Update a store

- **Pre-condition**

  A user should have a store associated with his account already. The user must have already logged into the system.

- **Story**

  A user be able to edit the store name and the information contained in the store profile.

- **Post-condition**

  The store in the database will be updated; and, user who owns this store will be able

to chose if he want to notify other user directly.

**Delete a store**

- **Pre-condition**

  A user should have a store associated with this account already. The user must have already logged into the system.

- **Story**

  A user should be able to delete a store owned by him. All invoices should be preserved.

- **Post-condition**

  The store data in the database will still be there, and, only the operator of the system can see it and delete it.

**Add a product**

- **Pre-condition**

  A user should have an account and should have already logged into the system. The user should have a store associated with his account.

- **Story**

  A user should be able to add a product to the store and to sell this product through his store.

- **Post-condition**

  - The product information will be stored locally.
  - The information pertaining to the added product should be published as an advertisement.
  - The advertisement should be forwarded to the subscribers of the store.
  - In the case where the category of this product has not been tagged in the store yet, the category will be automatically added to the list in the category tags of stores.

**Update a product**

- **Pre-condition**

  A user should have an account and already have logged into the system. The user should have the knowledge of the product that needs to be updated.

- **Story**

  The user should be able update the product information, such as price per unit, amount of available items and description etc.

- **Post-condition**

  - The updated product information will be stored locally.
  - A new advertisement will be published and the old advertisement will be replaced.
  - The new advertisement will be published to the store's subscribers.
  - In the case where the category of the updated product has not yet been tagged in the store, the categorty will be automatically added to the list of category tags of the store.

**Delete a product**

- **Pre-condition**

  A user should have an account and have already logged into the system. The user should have the knowledge of the product that needs to be deleted.

- **Story**

  The user deletes the product from the store. The advertisement associated with the product is disabled. In the case where the product is the last one belonging to a certain category in the store, the category is deleted from the list of the category tags. For example, a store contains two categories: car and pen. The pen will be removed from the list of the category tags right after the last product in the pen category has been sold.

- **Post-condition** The product data in database will be erased. Also, the call to unpublish the advertisement will be generated.

### 3.1.5 Online shopping service

We described the use cases of the online shopping service in Figure 3.3.


Figure 3.3: Basic online shopping service use cases

**Search products**

- **Pre-condition**

  A user has an account and has already logged into the system.

- **Story**

  A user should be able to find the advertisement for a product by searching with the product's name and description. After reading the advertisement, the user may invoke a request to retrieve the detailed product information.

- **Post-condition**

  A user will use the advertised information to locate the server responsible for the store of this product. The user would further retrieve the detail information of the product and send a purchase request to the store's server.

**Buy products**

- **Pre-condition**

  A user should have an account and must have already logged into the system. He must already have obtained the product information by using the Search a product function.

- **Story**

  A user should be able to send a valid purchasing request to the system and to complete the transaction of the purchasing activity.

- **Post-condition**

  If the transaction is validated, an invoice will be generated and sent both of the customer and the seller.

**Follow a store**

- **Pre-condition**

  A user should have an account and must have already logged into the system. The user should have knowledge of the store of interest, such as the name or URI.

- **Story**

  A user should be able to send a Subscribe request to the store in order to become a follower of the store.

- **Post-condition**

  The user should be able to receive the advertisements of the store. The subscribers will be stored in the system.

### 3.1.6 Social networking Services

The use cases of the basic social networking service in our system are shown in Figure 3.4

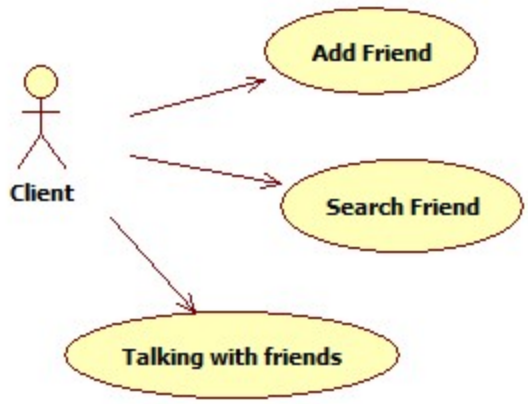



Figure 3.4: Basic social networking service use cases

**Search Friend**

- **Pre-condition**

  A user should have an account and must have already logged into the system.

- **Story**

  A user can find a friend by using a friend's name. He can view the information of the expected friend, including:

  - The public information in the profile;
  - The unique resource index (URI).

- **Post-condition**

  A user is able to retrieve the resource information of the expected friend and is able to further send the Add_Friend request to the server through the URI.

**Add Friend**

- **Pre-condition**

  A user should have an account and must have already logged into the system. He should have the knowledge of URI of the expected friend.

- **Story**

  A user should be able to perform an Add Friend procedure in the system and the friend will be added thereafter.

- **Post-condition**

  A user should be aware of whether his friends are online or not. A user can talk with online friends directly. He can forward advertisements for his product to his friends.

**Talking with friends**

- **Pre-conditions**

  A user should have an account and must have already logged into the system. He should have the friendship relation established with a friend in the system. The user and the friend must be both online.

- **Story**

  A user should be able to send the friend a talking request. A connection between them will be established. They will talk through the connection.

## 3.2 Organizational structure

The architecture of the proposed e-commerce platform has a three-layered structure. In this architecture, the peers at the two bottom layers construct a server farm. The peers at the top run the client software to access the services in the server farm. In this subsection, we first present the structure of the server farm. We then describe the client peers in the top layer.

### 3.2.1 Peer-to-Peer server farm

The architecture of BestPeer++ inspired us to design the architecture of the proposed syste. The system has the client/server architecture with a server farm. As we have

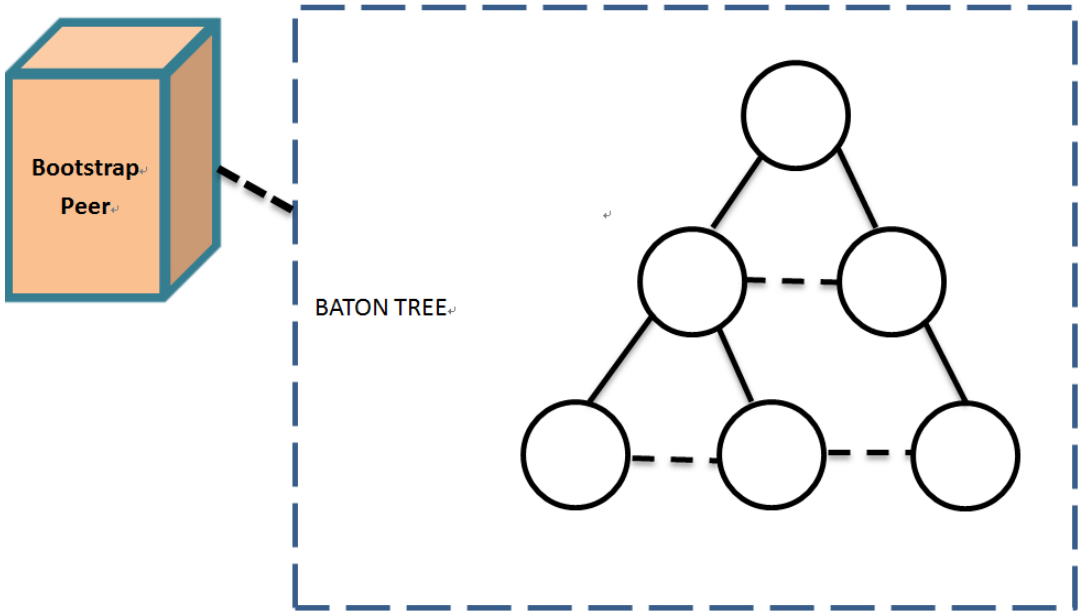



Figure 3.5: BestPeer++ [14] architecture

reviewed in Chapter 2, BestPeer++ has a server farm with a two-layered structure that is shown in Figure 3.5. The top layer contains the Bootstrap peer; and, the bottom layer contains the server peers. The server peers are organized in a BATON tree (reviewed in Chapter 2). This infrastructure allows the server farm to support a range of queries and guarantees the resource lookup service with log(N) hops in a system that has server peers. Additionally, the infrastructure is highly reliable and robust due to the use of the BATON's self-management function.

The Bootstrap peer manages the health of the server peers. It monitors the BATON overlay network during most of the running time. It is responsible for joining, leaving and load-balancing of server peers. For joining to take place, a new server peer has to first send a register request to the Bootstrap peer. The Bootstrap peer appends the new server to the list of the available server peers. In the case that an existing server peer wants to leave the overlay network, it sends a request to leave to the Bootstrap peer and its neighbours in the BATON tree. The Bootstrap peer pings all the peers in the system periodically and collects the statuses of these nodes: e.g., whether they have failed or are over loaded. The server farm is able to provide reliable services with the two-

layered structure; however, the BestPeer++ is an information sharing platform designed for businesses with enterprise network environments rather than for individual users. We thus have added extra components to the system to support personal users.

The architecture of BestPeer++ inspired us to design the client/server architecture with a server farm. As we have reviewed in Chapter 2, the BestPeer++ has a server farm with the two-layer structure shown in Figure 2.4 The first layer contains the bootstrap peer; and, the second one contains the server peers. The server peers are organized in a Baton tree (reviewed in Chapter 2). This infrastructure allows the server farm to support a range of queries and to guarantee the resource lookup service with log(n) hops. Also, the infrastructure is highly reliable and robust due to the use of BATON's self-management function.

### 3.2.2 Client Peer

In our system, users may be individuals. They are the customers and sellers engaging in e-commerce activities. They use low-end devices such as PCs or Smartphones. These devices have a smaller computing capacity and a lower network bandwidth compared to the computing machines used in enterprise network environments. These customers and sellers may not be online according to the 24/7 mode.

In our opinion, users in the system should be considered as the owners of low-end devices; and, these devices are not able to play the role of servers. For instance, such devices can not be online all the time. Users on the e-commerce information sharing platform may lose their data stored on devices that have left the system by being offine. Also, login or logout operations of users are fairly costly. These activities induce the joining/leaving of nodes. In BATON tree, the joining/leaving of a node causes data migration between the node and its neighbours. These activities consume a lot of system resources in the case where they happen frequently. Users log in or out of applications on their personal devices more often than devices on company servers do. Moreover, in the case where products on a server become popular, the number of requests on that server per second can become very great. This would cause the server peer to become overloaded very easily.

We face the same challenges as PeerSoN and Clique did [38]. We have thus added one more layer of peers to represent those devices. We refer to such a peer as a client peer. A client peer runs on a user's device; and, the user logs into the system through the client peer. For each user, the system assigns a server peer to take over the respective data during the user's registration. The server peer is the host of the user. It stored the data of the hosted user, including the user information, store information and product information. The user's client peer communicates with the host server when accessing the services of the system.

Figure 3.6 describes the architecture of the proposed system. Similar to BestPeer++, the proposed system organizes the Bootstrap peer and the server peers as a server farm. A client peer is attached to a server peer by connecting with it. In this manner, the application is divided into different layers by these different peers.

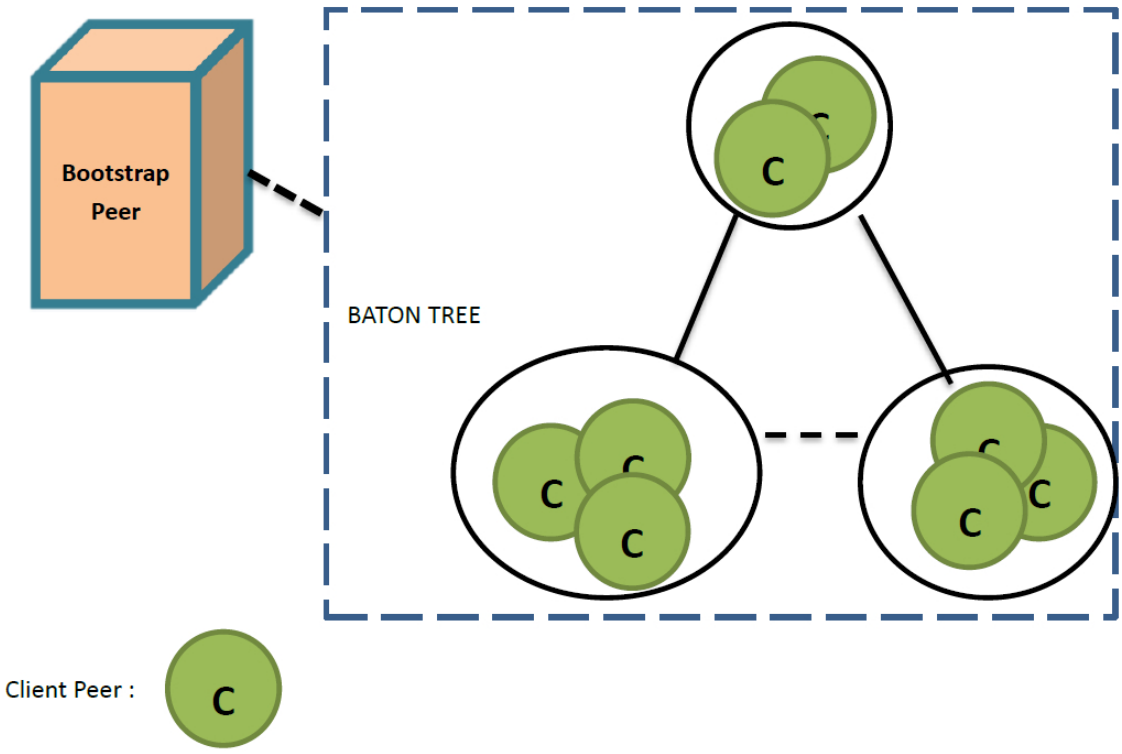



Figure 3.6: uopStore architecture

## 3.3 Components

In this section, we describe the components of the Bootstrap peer, server peers and client peers. We first introduce the components adopted from BestPeer++; and, we then discuss the extra components specifically designed for the proposed system. At the end, we present the components of client peers.

### 3.3.1 Bootstrap Peer

Figure 3.7 shows the main components of the Bootstrap peer in the BestPeer++ system. We reused some parts of these components and did some changes on others. The components of the Bootstrap peer in the proposed system are shown below:

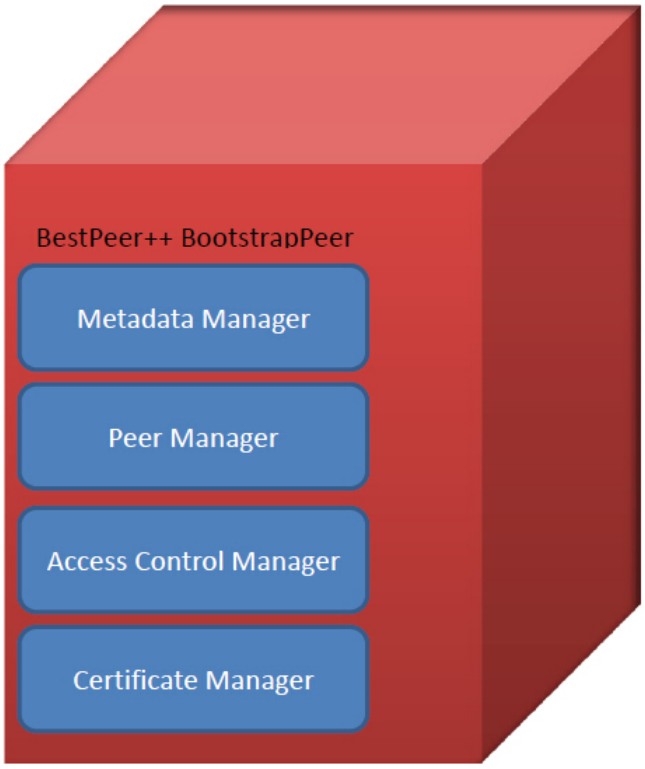



Figure 3.7: Bestpeer++ [14] Bootstrap Peer

- **Meta-data Manager:** : this is used to store the status information of online server peers. The Bootstrap peer continues to record this information in its database.

- **Peer Manager:** this component monitors the health of BATON tree. It is also in charge of the auto-fail and auto-scaling features. Both of these functions are kept in the proposed system. Furthermore, this component is also responsible for authenticating a user's login request sent from a client peer. Since the Bootstrap peer is a login server for all users, this component locates the server peer for the client peer during the login procedure.

- **Access Control Manager:** this is used to configure the policies that grant the access privileges to users based on their roles. We have saved a further examination of this component for future work.

- **Certificate Manager:** this is used as the security mechanism of the system. It is used in the case where this framework is published as a software product. We will presently design this component.

Since the proposed system is an e-commerce platform, we have designed the following components:

- **Advertisement Manager:** it stores the advertisements for popular hot products in the system. In order to build up the knowledge concerning the most popular products, the Bootstrap peer may run a distributed election algorithm on the servers. Or, the advertisements of the new products in a store may be forwarded to the Bootstrap peer by the host server of the store. A user can find these products easily by accessing the information stored in the Bootstrap peer.

- **User Profile Storage:** this stores the information of the user employed when the Bootstrap peer processes a user's login request.

- **Store Profile Storage:** this stores a copy of the information pertaining to each store. This information is used to guarantee the correctness and trust-worthiness of the information.

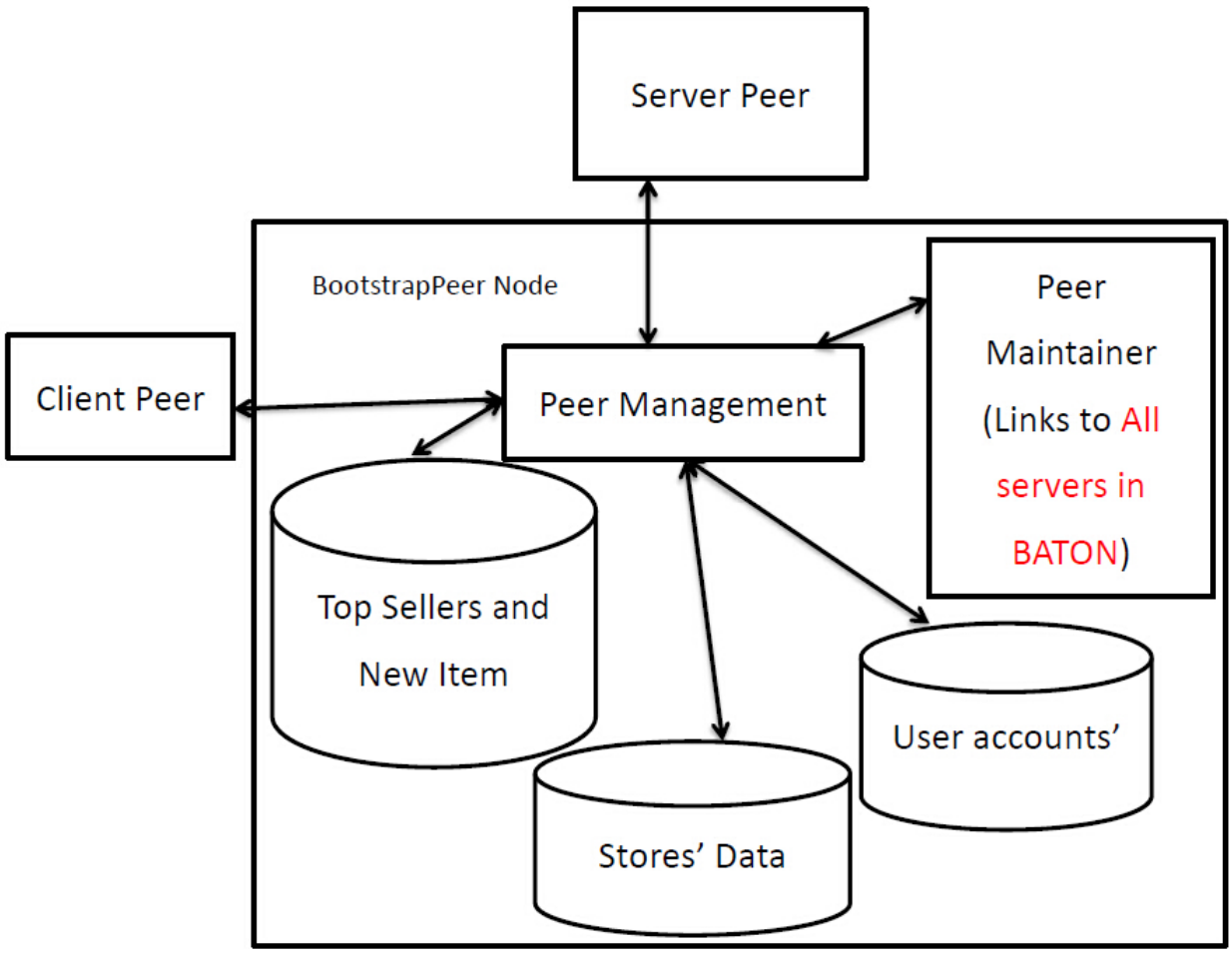



Figure 3.8: uopStore Bootstrap Peer

Figure 3.8 shows the component diagram of the Bootstrap peer. The peer manager in BestPeer++ is still used. However, the meta-data manager is divided into the following components: peer maintainer, top seller's storage, the data stores data and user accounts data storage.

### 3.3.2 Server Peer

In the proposed system, the server peers connect to the BATON overlay network. Also, they store the data for the users and provide services to the client peers of users. The components of the server peers are mainly used for constructing and maintaining the P2P infrastructure and for creating and maintaining the user accounts, in addition to managing their data. In the BATON overlay network, each server has a range of data it is responsible and acts as a host server for those client peers that have an URI (Unique Resource Index) in its range. For instance, a server that is responsible for the indices ranging from 3 to 50 will act as a host server for the client peer whose URI is 15. In BestPeer++, the server peer has components that are listed in Figure 3.9

Some of these components are kept in the proposed system yet have been modified. The following list describes the updates.

- **Schema Mapping:** this has been discontinued because there is no need for data mapping to take place between sources with different schemas.
- **Data Loader and Data Indexer**
  - **Data Loader:** this is used while retrieving data from the database of the client during schema mapping. Since we do not need to perform complex schema mapping and a data retrieval process, this component is simplified as a very generic data retrieving component which is combined with the Data Indexer.
  - **Data Indexer:** this is an essential component. In BATON, each server peer stores the shared data in a certain range. The Data Indexer is used to index the data on the server peer. The local search algorithm uses this component to determine whether or not to forward a resource lookup request.

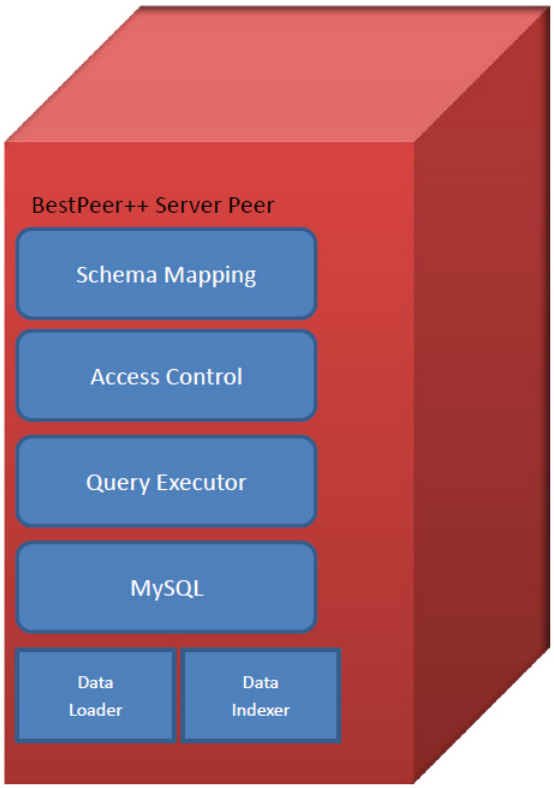



Figure 3.9: Bestpeer++ Server Peer

- **Access Control:** this will be further developed in the case where the system may need to control the access privilege to the data published on the overlay network. For instance, this is applicable in the case where a user only wants to publish the advertisement of a special sale to those users who already have subscribed to his store for more than 10 months.

- **Query Executor:** this is a tool used to retrieve data from databases in the system. In the proposed system, we have kept the component but have used another approach to implement it.

  **Modifications** we have kept this in the design, but have used JPA to implement it instead of pure SQL language.

- **MySQL:** this has been replaced by the Postgresql database due to licensing reason.

In the proposed system, the data of users is stored on server peers. This data presents the information of users, their stores, and their purchasing activities. The system locates a server peer for a user during the user's registration process. This server peer stores the data for this user; and, is the hosting server of the user. A server peer can host multiple users. In addition to storing the data for the stores, the system can publish product information to the overlay network for stores. For example, right after a seller adds a product to his store, the server peer automatically generates an advertisement for the product and further publishes it to the Bootstrap market place as well as on the overlay network. The data of the published information is distributed to the server peers in the overlay network. The server peers process a distributed lookup operation to search for the published information. A server peer uses the Index Management and Index Meta-data databases to process a lookup request locally. The lookup request contains the index value of a queried data item. In addition to the functions mentioned, the server peers handle online shopping transactions and store management operations. The component diagram for server peers is shown in Figure 3.10

- **Client Peer Maintainer:** this stores the information regarding the physical connections with the client peers. The users are logged into the system with these client

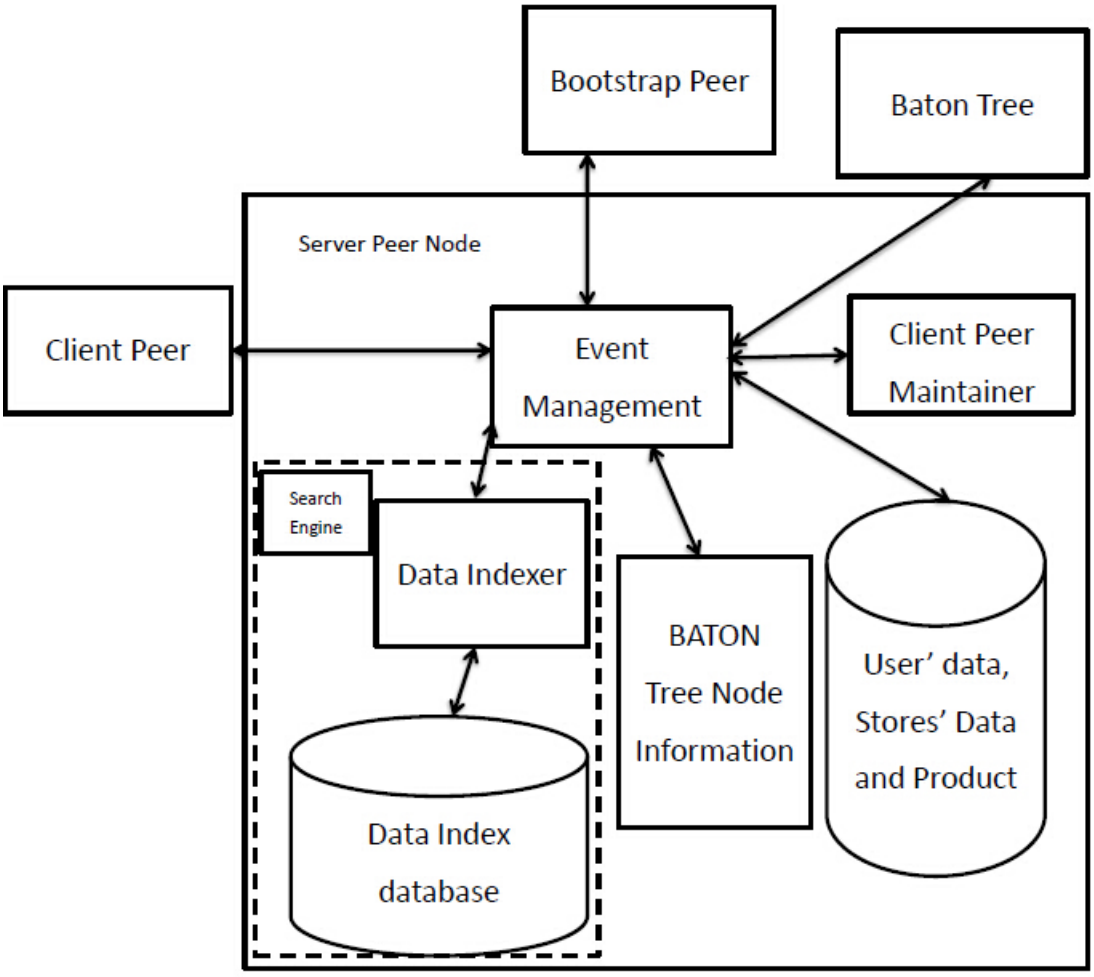



Figure 3.10: uopStore Server Peer

peers. To improve the response speed, some of the data information is cached in the maintainer mechanism so that it can be retrieved at the time of a user's request quickly.

- **Store Data and Product Data:** this saves the data of stores' belonging to the hosted users, including the information of stores and of products.

- **Data Index Database:** this stores the indices of the resource. In this system, there are usually two kinds of indices:

  - Global index helps the system find the meta-data of the resources.
  - Local index is used by the server peer to retrieve detailed data concerning local resources.

  These two kinds of indices are both used during the processing of a resource lookup operation. For instance, in order to find a product, the lookup operation locates the server peer storing the meta-data of the product by using the global indices located on the server peers first. It then searches the meta-data and finds the physical information of the product. Using the physical information, it visits the hosting server of the seller and retrieves the product information there by using the local indices located on that server.

- **Data Indexer**

  A server is responsible for the global indices in a certain range and for all of its local indices. The Data Indexer component on a server peer takes over the global indices for the server. It is used for resource management by a server in two ways:

  - **While a user adds or updates a product**, the server first publishes the product locally. The Data Indexer then generates an advertisement that contains the physical information pf the server as well as the simple product information. Meanwhile, the global index of the product, which takes the form of a key/value pair, is also generated. Thereafter, the server publishes the advertisement in the BATON overlay network according to the global index of the product.

– **While searching a product** the server generates a key for the global index first query of the product. The Data Indexer judges the location of the key immediately according to the data range of the servers. If the key belongs to the server, it returns the meta-data of the product to the user; if this is not the case, it initiate a resource lookup procedure in BATON by forwarding the query to the one neighbor who is the most likely to know the answer to the query.

- **BATON Tree Node Information** : it stores the information of the physical connection with the Bootstrap peer and it stores the information of its neighbours in the routing table of BATON. Additionally, it also stores the left child, the left adjacent node, the right child and right adjacent node.

### 3.3.3 Client Peer

A client peer is a light weighted application that runs on a user's device: e.g., PC or Smartphone. While running, a client peer has a user interface that directly interacts with a user. There is no database on a client peer. The data of the client peer (e.g., the cached physical information) is stored in the server farm.

Figure 3.11 shows the components on a client peer. These components are described in the list on the following

- **User Interface:** this component basically is responsible for interacting with the user and for submitting the user request to the system.

- **Online Shopping Management:** this consists of three important aspects of an e-commerce application. These aspects are shopping cart, the advertisement center and product suggestion.

- **Personal Store Management:** this is a component that interacts with the server peer and the user interface. It is responsible for implementing the user operation on user's store. It adds or edits product information in a user's store.

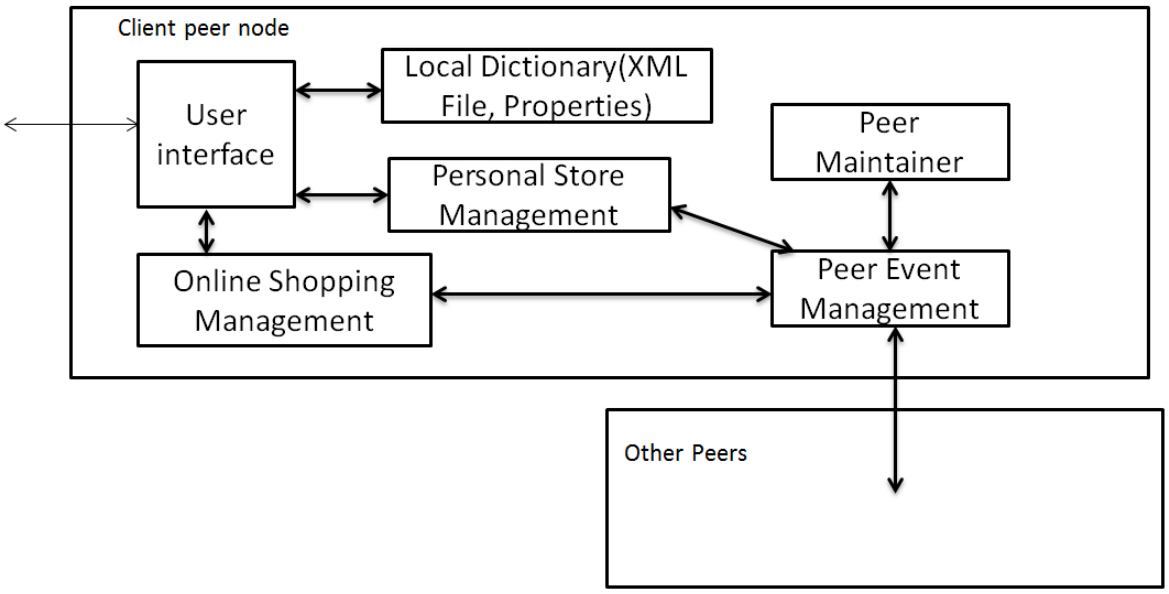



Figure 3.11: uopStore Client Peer

- **Peer Maintainer:** this stores the information of the client peer, such as information regarding the store of the user and those stores that the user has subscribed to. It also stores the information of the physical connections of friends' client peers. Figure 3.12 shows connections with those client peers after a user logs in through client peer A. The unnecessary involvement of the server peer and Bootstrap peer will be cut down minimal.

- **Personal Store Management:** this is a component that exists between a server peer and the user interface. It is responsible for accessing the services on the server peer. A user uses these services to manage his store: e.g., to add or edit products.

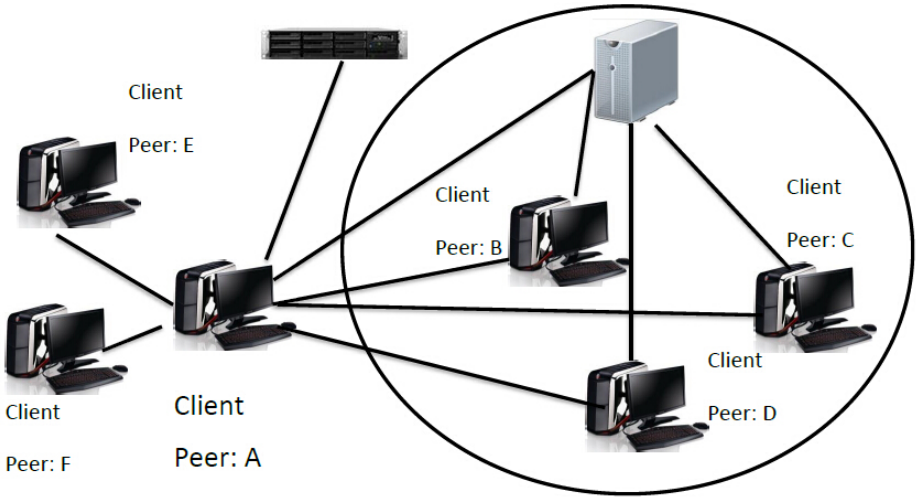



Figure 3.12: The connections between client peers

# Chapter 4

# Distributed Operations

After determining the architecture of the system, we designed the method of implementing of the distributed operations. These operations realize the services specified by the system's functional requirements (analyzed in Chapter 3). We conducted some experiments to verify the correctness of these implementations. In this chapter, we present the activity diagrams of these operations as well as their manner of implementations. Included with in this system of implementation are the messages sent between components on different peers and the algorithms that process these messages. We will also present the results collected from the experiments at the end.

## 4.1 Distributed Operations

The e-commerce platform provides its users with the services of user login, resource searching, resource publishing and product auctioning. These services are realized by the distributed operations.

### 4.1.1 User login operation

A user's login operation is implemented by the components on the Bootstrap peer and on the server peer together. In this operation, the user is authenticated first. The client

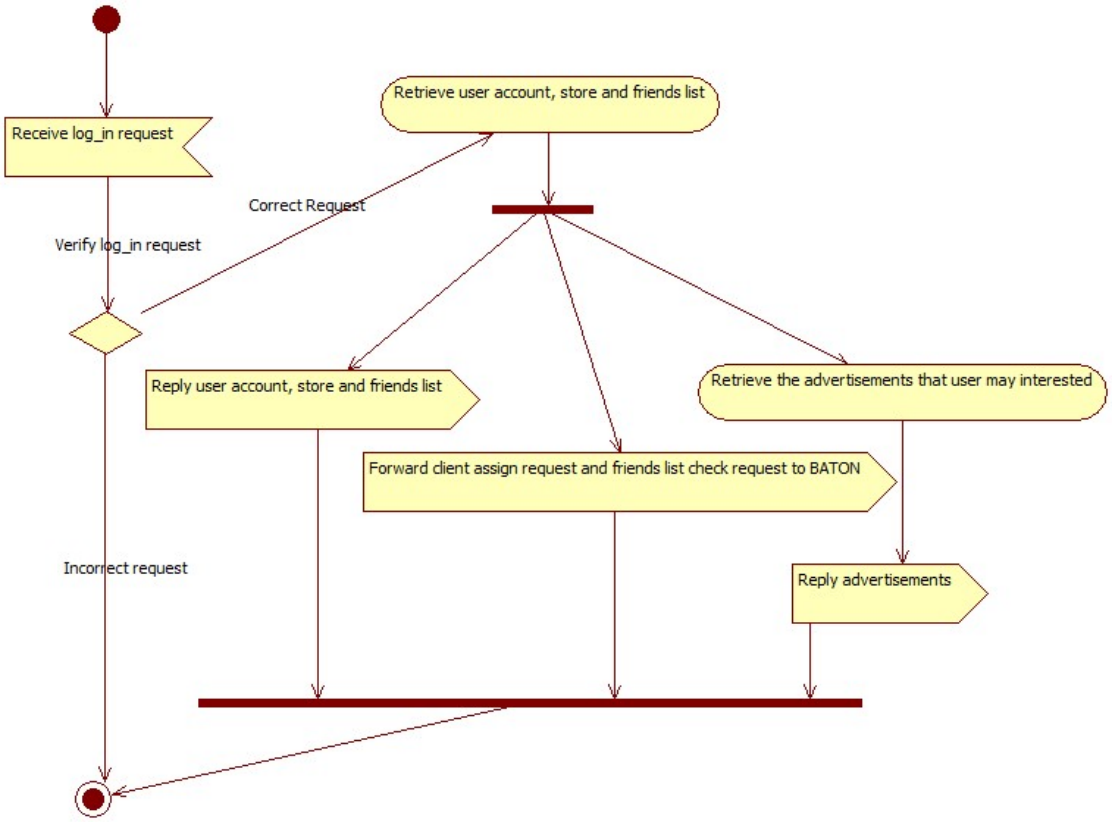



Figure 4.1: The activity diagram for the processing of a user login operation on the Bootstrap peer

peer then sets up connections with the Bootstrap peer and with its host server peer. By using these connections, the client peer is able to access the services of the system without searching further in the BATON for the host server during every request. For example, the user may manage the data of his store by sending requests from the client peer to the hosted server directly. For a user, his client peer also maintains the connections with the client peers of those friends online. Through these connections, the user can easily share resources or exchange messages with his friends without the involvement of other peers.

In a login operation, a client peer also collects the data necessary for future operations. For example, a user may access the data in his user account frequently. Without storing the user data locally, processing each request by finding this data and by retrieving it would be costly. For instance, the user sends the request to the host server first. The host server peer may retrieve the data from its database or cache; and, it may then issue a synchronous or asynchronous callback function so that the data is passed to the client peer. This kind of data retrieval would waste a lot of resources, such as network bandwidth and computing processing power. In order to reduce such requests, the client peer stores the necessary data locally; this stored data is included in the following list: account information, friends' information, advertisements of potential interest, online friends' physical information, and the store information. After the user logs in, he can begin to chat with friends, view advertisement of potentially interest or manage his store directly.

The procedure for the login operation in the Bootstrap peer is described in Figure 4.1.

1. The procedure starts when a login request from a client peer is received. The Bootstrap peer performs user authentication first with the stored username/password pairs. The user is authenticated in the case where the received pair matches the stored pair. Otherwise, the Bootstrap peer denies the user by sending a login error message; and, the Users' Login Operation stops thereafter. For an authenticated user, the Bootstrap peer sends the data pertaining to his account information, the store information (in the case where the user has already opened a store), and the friend list information; in other words, this data is sent to the client peer.

2. For the authenticated user, the Bootstrap peer THEN sends his login request to a

server peer in BATON. To finish the procedure itself, the Bootstrap peer sends the advertisements which the user may be interested in back to the users's client peer. At this moment, the user obtains information pertaining the account, store, friend list and the advertisements. The server peer receiving the login request from the Bootstrap peer continues the login operation. The activity diagram for the procedure on a server peer is illustrated in Figure 4.2. The server peer checks the user's resource index. In the case where the resource index is within the range of the shared data, the server sends the notification to join back to the user's client peer. Otherwise, the server peer initiates a resource searching operation (described in the next subsection); the purpose of this is to find the host server for the user in BATON. This step is repeated until the host server is located.

3. The host server peer updates the statuses of online friends. It goes through the online friends check list and chooses the friends by their indexes.

4. The server peer then sends the friends' physical information back to the client peer in the case where his friend is online. It then removes these friends from the list.

5. If the list is not empty, the server peer divides the rest of the list into based on their indices and packages them into different requests. The server peer then sends those requests to its neighbours accordingly. The server receiving a request continues steps 4 and 5 until the list is empty.

After receiving the request from the Bootstrap peer and the server peers, the client peer sets up the connections as shown in Figure 3.12.

### 4.1.2 Resource searching operation

We have three main kinds of resources in our system: user, product and store. A resource is identified by a unique resource index (URI). A resource searching operation is used to locate a resource upon a users' request. The uopStore can locate the host server of a resource by looking for the URI of the resource in BATON. The URI of a resource, such

as a store or user, can be searched according to the resources' name in the Bootstrap peer. A user may obtain a username or a store name through the services of the system; for instance, in commercial advertisements, through the username or store name may be obtained friends' references, chatting to friends, online searching, or even from daily life offline.

Obtaining the resource of a specific product is different. This is because there is no URI related to a product name. Unlike a store name or a user name, a product name may exist in different servers because multiple stores may sell items of the same product. For instance, iphone 5s is a product that is known to users. A user may search an iphone 5 product by its name. However, from the perspective of this product, there could be several products that have iphone 5s as product name in the system: an example of this is iphone 5 available in Futureshop or on Bestbuy. Thus, the product name is a non-null attribute of a product and many products may share the same product name.

The activity diagram for a product search operation is described in Figure 4.2. A product does not have its product name as the URI; but, a published advertisement may use the name of a product as a resource index. By submitting a search request that contains the resource index of a product, a user may receive a list of advertisements that contain the same or similar names of products. Users may further retrieve the information regarding their products of interest according to the following information: ( sever IP address, server port and products) in advertisements. The user then can send a search request to the host server of the store from his client peer. The host server thereafter, responds with the detailed information of the product.

### 4.1.3 Product publishing operation

This operation is used to publish the information of a new product in a store. The information of a product describes many aspects of the product. The data of this information may be contained in a large-sized message. There are incurred costs on time and network bandwidth when transmitting such a message on the network links of the Internet. Moreover, a consumer may only read the key portion of the information. The system provides

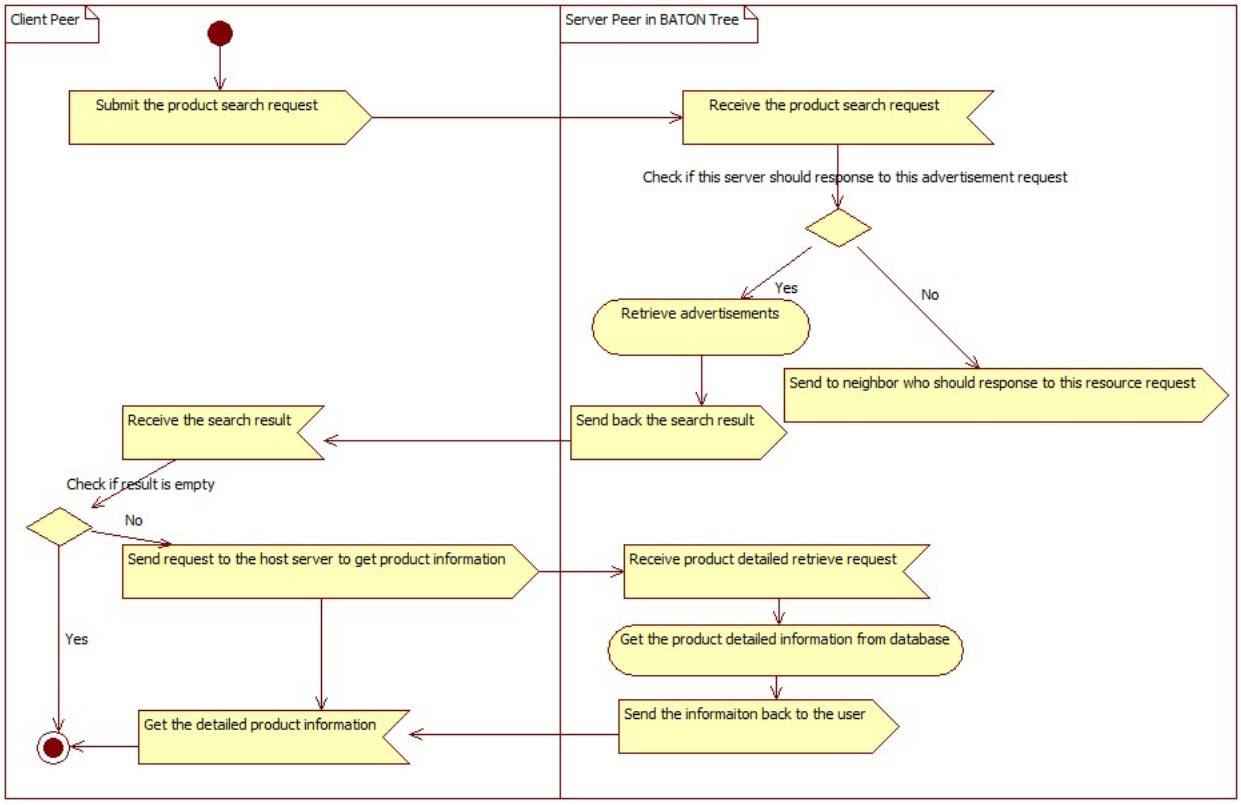



Figure 4.2: The activity diagram for a product search operation

users with a search function for product advertisements. An advertisement contains the key information for a product and the physical information of the host server of the store as well. The consumer may further access the host server to retrieve the detailed information of the product.

A product publishing operation is used to publish advertisements of a product in the system. The advertisement is stored on the peers in the BATON server farm and searched through the product search operation described in the last subsection. An advertisement is generated first for a newly added product to a store. The operation then publishes the advertisement in the system. When a seller publishes an advertisement, he wants to notify as many interested users as possible. Because a user is a customer, he may not want to be exposed to many useless advertisements. Thus, we consider the first targets of the advertisements as the active subscribers of the store. These subscribers are online at the moment. In the system, a market of new products is provided for users to browse the advertisements of products. The market is hosted on the Bootstrap peer. The market stores the advertisements published by stores during the publishing operation of a new product. Only advertisements in a certain time period are kept.

The activity diagram for this operation is shown in Figure 4.3. It describes the following steps. To add a product, a user must fill the product information in the GUI on a client peer. The client peer sends the product information to the host server of the store. The host server stores the product information locally and generates an advertisement based on this information. The server now invokes a product publishing operation. It firstly sends the advertisement to the Bootstrap peer as a new advertisement. Then, the server peer goes into the advertisement pushing mode. The server forwards the advertisement with a list of subscribers, for each server who is responsible for a range that covers a potion of the subscriber: eg, these servers in the routing table. The server THEN receives this advertisement notifies the online subscribers within its range; and, it then forwards the advertisement with the remaining subscribers to its own neighbour accordingly.

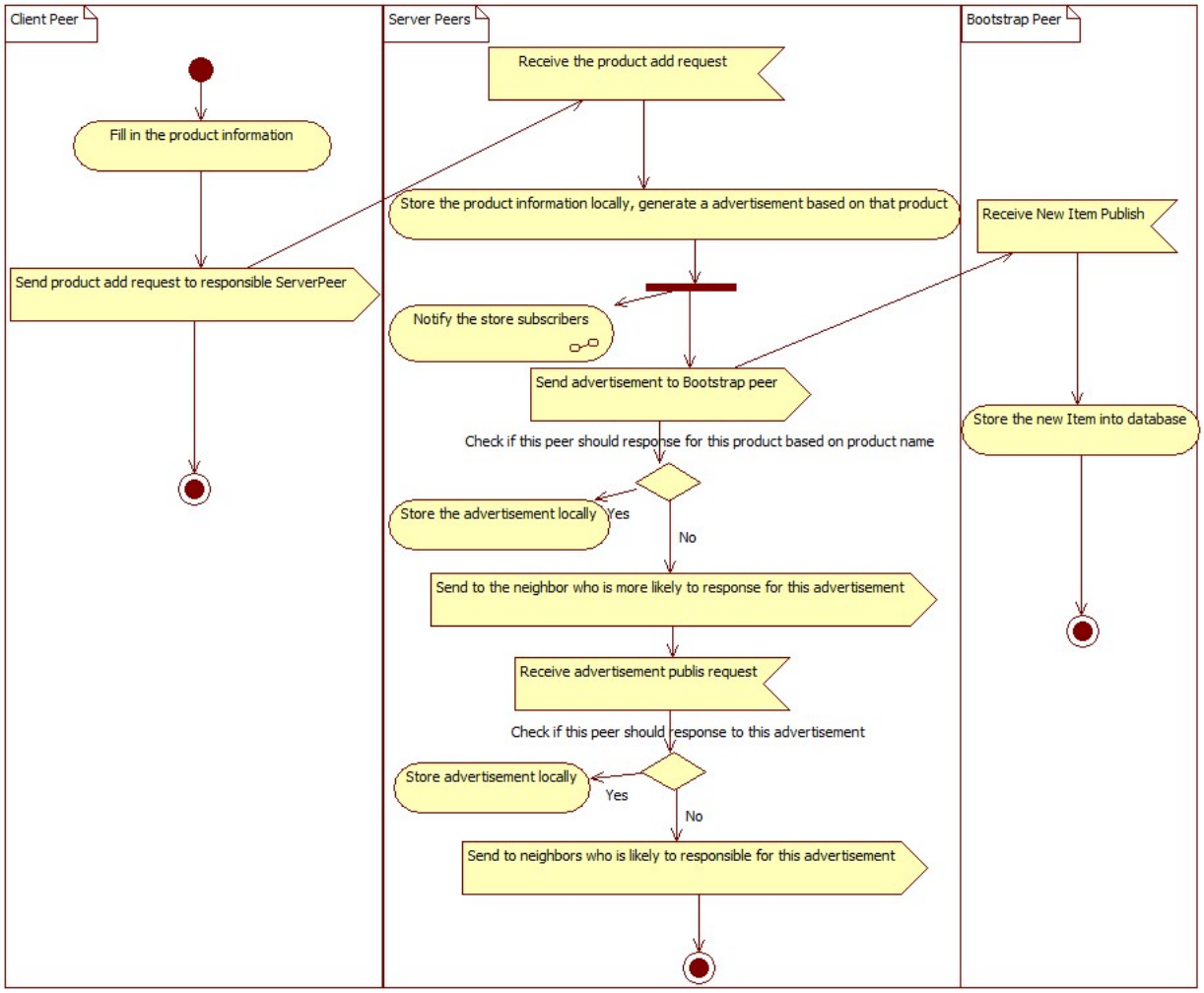



Figure 4.3: Activity diagram for a product publish operation

### 4.1.4 Product auctioning operation

The system provides its users with an on-line auction service. The operation of the service realizes the procedure of a basic English auction as shown in Figure 4.4. A user can place any of his products in his store for an auction. The host server of the user and his store hosts the on-line auction. The product of an auction is implemented as an extension of the product. An auction operation starts right after a user submits an Auction Creation request to the host peer on his client peer. The auction may start at a future time or immediately; and, it may end at a pre-configured future time. The host peer generates an advertisement for the auction and publishes the advertisement in the system.

The operation has more steps than the product publishing operation. A user fills in the product information for the auction on his client peer. The client peer then invokes an auction operation by sending an auction that request contains the product information to the host server. Similar to a product publishing operation, the host server generates an advertisement for the auction and publishes the advertisement in the system afterwards. The target receivers of the advertisement are the on-line subscribers of the store.

The host server processes these bids by following the steps of an English auction. The server starts the auction processing at the time specified by the user. The subscribers can bid on the products of the auction during the specified period of time. The server receives the bids from the bid peers of the auction. The server always updates the price of the product with the highest price at that moment. The auction finishes when the auction time specified by the user runs out. The server processes a purchasing transaction with the winner and generates an invoice for this transaction. The server forwards the invoice to the customer and the seller at the end.

## 4.2 Operation Implementation

In this section, we present the implementations that realize the operations in the section above. These implementations include the communication messages between the components on different peers and the algorithms that process these messages there. We describe

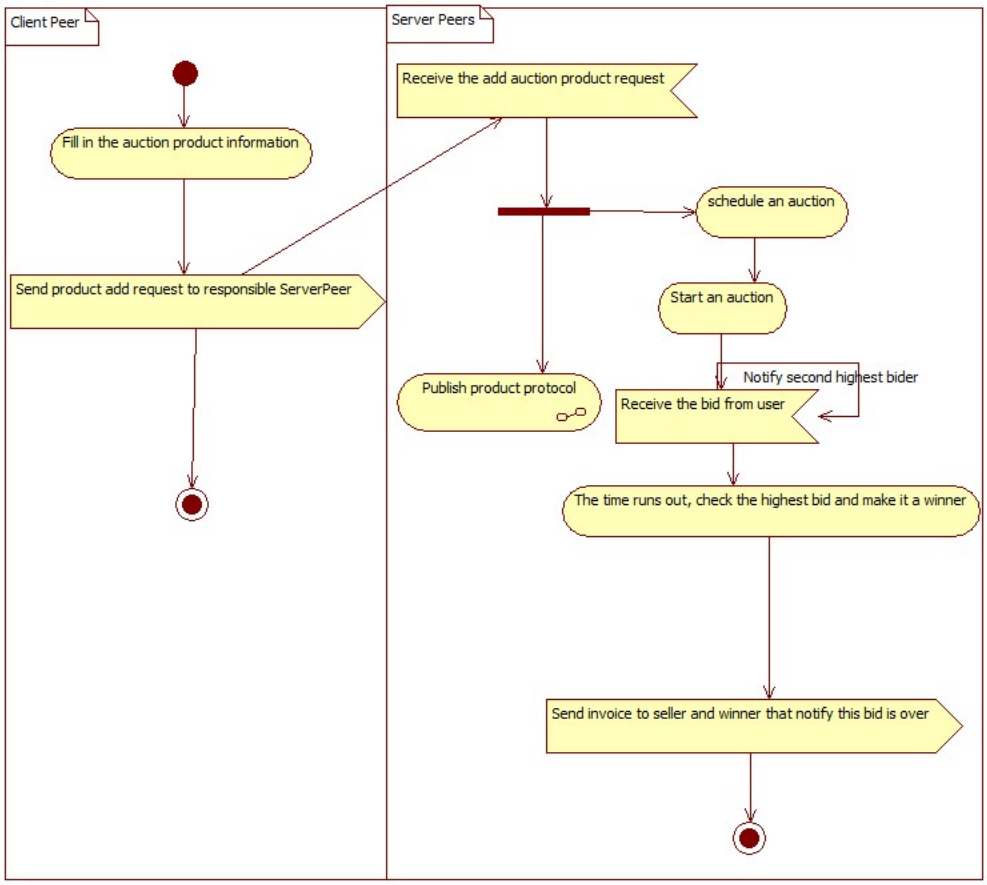



Figure 4.4: The activity diagram for a product auctioning operation

each operation with the sequence diagrams and algorithms.

### 4.2.1 User login operation

The sequence diagram of a user login operation is depicted in Figure 4.5. A user login request is processed by the Bootstrap peer as well as by server peers. In the Bootstrap

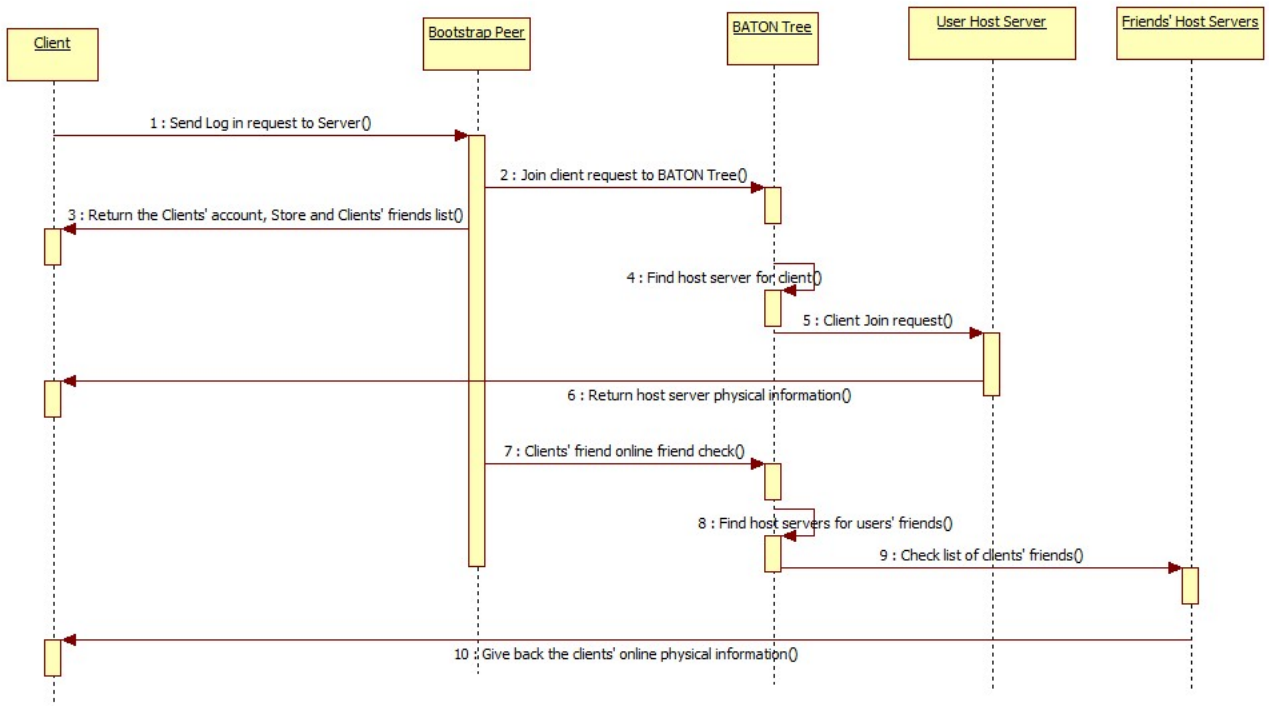



Figure 4.5: Sequence diagram for user login operation

peer, the login listener processes the Client Login request first. It sends a reply the users' client peer with data containing the information pertaining to the user account and friends in the case these the user is authenticated. The listener then selects the advertisements in which the user might be interested and sends these advertisements back to the client peer. After that, the Bootstrap peer sends the Client Join request as well as the OnlineFriends Check request to a random online server.

The server peer receiving the Client Join request and OnlineFriends Check request processes them locally. The algorithms for processing these two requests are shown in

Figure 4.6. During the processing of the Client Join request, the server peer locates the host server of the user by calling up the lookup function of BATON. The server peer verifies whether it is the host server of the user first. For a Client Join request from a user that is not on the server, the server redirects the Client Join request to another server; this is done according to the resource location identifier of the user. Otherwise, the server peer sends its physical information to the client peer.

The algorithm on a server peer for processing the OnlineFriends Check request is used to update the user's online status at the friends' client peers. These friends are hosted on the server peer. The request contains a list of friends ordered according to the lexical order of their usernames. The server identifies the local friends from the friend list. A local friend has the URI in the range of the data on the host server. The server notifies the login's client peer of the on-line statuses of these local friends. Furthermore, the server notifies each local friend of the on-line status of the user. The notification includes the physical information of the client peers, such as the IP address, the Port number, etc. The server removes the local friends from the OnlineFriends Check request, and then, it forwards the request to the next server peer. The next server is selected by lookup algorithm of BATON; and, the key for the lookup is the username of the first friend in the friend list.

### 4.2.2 Resource searching operation

There are two kinds of resource search operations in our system. The first one is used for searching resources that have unique attribute values: for example, a user may be searched by may be searched by his username or a store by its store name. These resources are identified by their URIs. The sequence diagram for such types of searches is shown in Figure 4.8. At the beginning of the operation, a client peer retrieves the URI of the resource stored in the Bootstrap peer. It then invokes a resource lookup operation in BATON by sending a resource lookup request to its host server. The host server sends a reply to the client peer that contains the information of the resource.

The second kind of search is performed to locate the host server of a resource whose name are not unique in the system: on example of this is a product. The sequence diagram

```
Algorithm: Clientloginlistener ( loginMessage msg)
    Account a←getAccountById(msg.getId())
    If ( a.getpassword equals msg.getPassword() )
        List< ClientBaseInfo> friends ← a.getFriends()
        this.getClientPeerMaintainer().put(a.getURI(), a.getPhysicalInfo() )
        List<Advertisement> ads ← getAdvertisementForClient(a)
        Reply a with account a, List<Advertisements> ads and List<ClientBaseInfo> friend.
        If (this.getOnlineServer().size > 0)
            Server s ← getRandomServer()
            Send userJoin(a) to s
            Send friendOnlineCheck(friends) to s
              End If
          Else
        // Only Bootstrap is online which means login failure
        //A similar process is followed towards the left
        Else
            //Password error login failure
      End If
End If
```

Figure 4.6: The algorithm for processing user login request on the Bootstrap peer

in Figure 4.9 illustrates the activity of such an operation. The server peer process a Product Search request with the algorithm shown in Figure 4.2. The product name acts as the resource index in this request. The search operation finds a list of advertisements with the same product name. The server storing these advertisements sends a reply to the client peer with the result. Afterwards, according to an advertisement of the product the user may connect directly with the host server peer of the product; and, the user may have access to further retrieve detail information there.

### 4.2.3 Product publishing operation

The product publishing operation is an essential part of the store management service. The implementation of the operation publishes the advertisement of a newly added item to the system. It also sends the advertisement to the subscribers of the store. These may be understood as the potential buyers of the product. This operation benefits the users on advertising their sales with less manual effort required.

Figure 4.11 shows the sequence diagram of the product publishing operation. For client

```
Algorithm: ClientJoinlistener ( Account a)
    URI res←a.getURIRes()
    TreeNode node ← this.getTreeNodeInformation()
    If(node.getResponsibleRange.isInRange(res))
        this.getClientMaintainer.put(res, a)
        Send this.getPhysicalInfo() to client
        Else
            Server s ← msgDispatchRoute( treeNode, res)
            If(s != null)
                Forward Account a to s
        EndIf
    End

Algorithm: ClientOnlineCheckListener( List<URI> resvalues, PhysicalInfo requester)
TreeNode node ← this.getTreeNodeInformation()
List<URI> remains ← resvalues
List<PhysicalInfo> localOnlineFriends
For each value in resvalues
    If( node.getResponsibleRange().isInRange(value) )
        remains.remove(value)
        If( this.getOnlinePeer(value) !=null )
            Send online notify to this.getOnlinePeer(value)
            localOnlineFriends.add(this.getOnlinePeer(value))
            EndIf
    EndIf
Send requester with localOnlineFriends
Map<PhysicalInfo, List<URI>> resultsSet = msgsPackageDispatch(node, remains)
For each key k in resultSet
    Send List<URI> to k
    End For
End
```

Figure 4.7: The algorithm for process the Client Join and OnlineFriends Check request in Server Peer

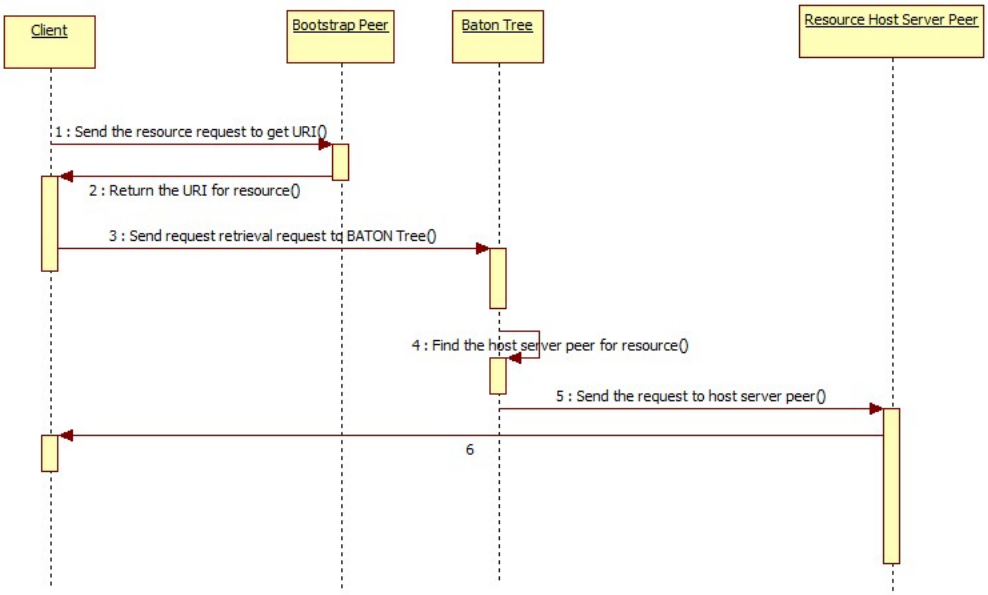



Figure 4.8: Sequence diagram for Resource Search operation

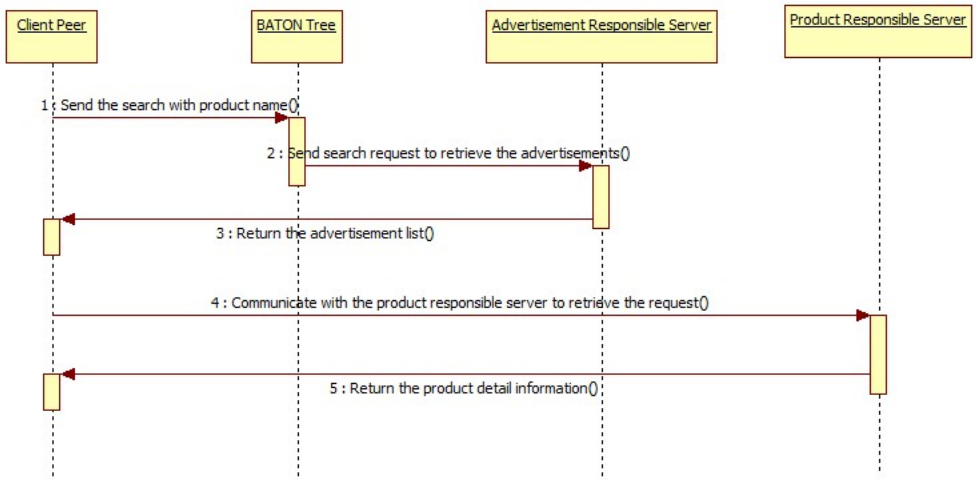



Figure 4.9: Sequence diagram for Product Search operation

```
Alogrithm: ProductSearch (String ProductName, Account requester)
TreeNode node ← this.getTreeNodeInformation()
If (node.getResponsibleGlobalIndexRange().isInRange(ProductName))
    List<Advetisement> ads ← getAdvertisement(ProductName)
    Reply requester with ads
    Else
        Server s ← msgDispatchRoute( treeNode, ProductName )
        If(s != null)
                Forward ProductName and requester to s
        EndIf
    EndIf
End
```

Figure 4.10: The algorithm for processing product searching request on a server peer

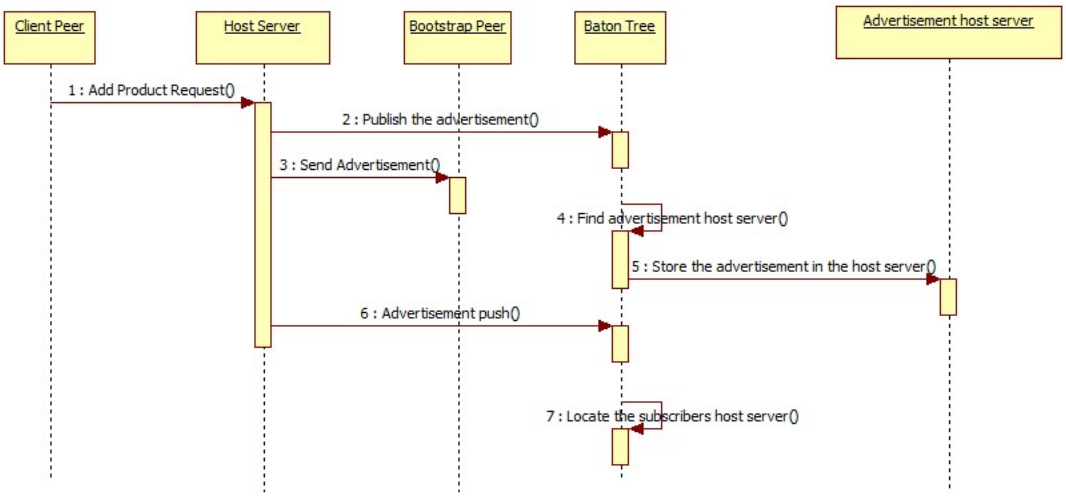



Figure 4.11: Sequence diagram for Publish Product operation

peer Add Product request, the host sever peer must retrieve the information of the hosted store first; and, it then will initiate calls the AddProduct function to add a product to a user's store. An advertisement is generated by the Advertisement generator. The server then invokes a resource publishing operation in BATON to publish the advertisement. The server further notifies the subscribers of the store by pushing the advertisement to their client peers. The advertisement contains the store name. The algorithm that processes the Product Publish request on a server peer is shown in Figure 4.12.

```
Algorithm: ProductPublish ( Product p, Account a )
Store s ← getStoreByAccount(a)
s.addProduct(p)
Advertisement ad ← toAdvertisement(p)
AdvertisementPublish(ad)
List<Membership> members = s.getMemberships()
AdPush(ad, members)
End

Algorithm: AdvertisementPublish(Advertisement ad)
TreeNode node ← this.getTreeNodeInformation()
If (node.getResponsibleGlobalIndexRange().isInRange(ad.getProductName()))
    publishAdvertisement(ad)
    Else
        Server s ← msgDispatchRoute( treeNode, ad.getProductName )
        If(s != null)
                Forward Advertisement ad to s
        EndIf
    EndIf
End
```

Figure 4.12: The algorithm for product publishing request on a server peer

### 4.2.4 Product auctioning operation

The implementation of a product auctioning operation starts in a manner similar to the act of adding a product to a store. After the advertisement of the auction has been published to the system, the host server peer receives the bids from the client peers during the period of the auction. The sequence diagram of the operation is shown in Figure 4.13.

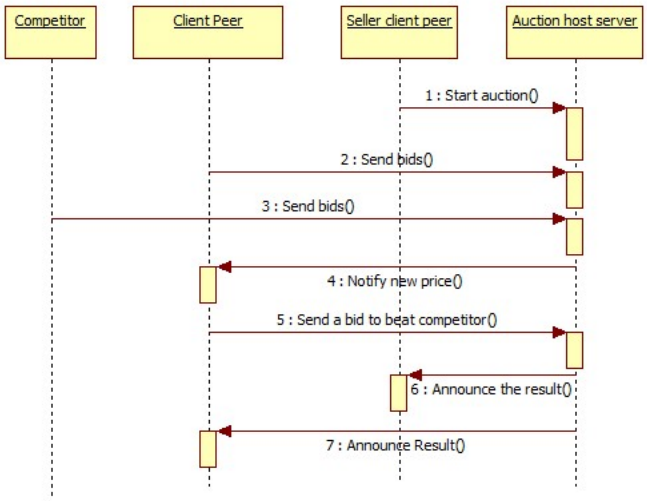



Figure 4.13: Sequence diagram for product auctioning operation

**Alogrithm: OnStart(AuctionItem** it**)**
*this*.**getAuctionMarket.put(***it*.**getURI(), *it*)**

**Alogrithm: OnRunning(AuctionItem** *it*, **Bid** *bid*, **Account** *a*)
*this*. **getAuctionMarket.get(***it*.**getURI()).addBid(***bid, a*)

**Alogrithm: OnComletion()**
**Bid** *b* ← **getTheMaximumBids(** *it* **)**
**Account** *customer* ← **getTheMaximumBidUser(** *it* **)**
**Account** *seller* ← **findClient(***it*.**getSellerName())**
**Invoice** *invoice* ← **processTransaction(***it, Bid, a*)
**Forward** *invoice* **to** *customer* **and** *seller*

Figure 4.14: The algorithm for processing an auction on a server peer

A host server handles the progress of an online auction by using a timer. As shown in Figure 4.14., for an auction, the host server adds the auction to the auction marketplace when the onStart timer fires. The auction item is identified by the URI assigned by the host server. During the auction, the host server processes the bid requests received from other client peers with the onRunning function. When the auction period is finished and the OnCompletion timer has fire, the OnCompletion() function is initiated to handle the purchasing transaction and to finish the auction.

## 4.3 Results from experiments

We conducted some experiments to verify the correctness of our implementation. We mainly verified that the operations of the uopStore can be correctly processed in a large scale system. These operations include the user login, product searching, and product publishing. The product auctioning operation is not included since its process is very similar to the product publishing operation. For instance, a product auctioning operation first generates and publishes an advertisement. Afterwards, messages for bids are directly sent to the host server from participating clients.

We implemented a prototype, and, then implemented an experimental structure by making some changes to prototype. To ensure the behavior of the systems will be same in the real network. We only modified the way peers store and notify each other. Peers had in our prototype notified each other by socket and stored by IP. Peers were modified to store other peers by addresses in RAM and notify through the RAM address. The difference found in the real system is, in cache, the transaction can be triggered and completed very fast while the failure rate is low. The whole experiment runs in the simulation environment on my computer. The configuration of my computer is as shown in 4.15. It has 4 CPUs and 16GB of RAM.

We simulated an environment for only running these experiments. The environment mimics a server farm constructed with a BATON infrastructure. There are only server peers in the server farm. The simulator keeps the components on server peers, with the

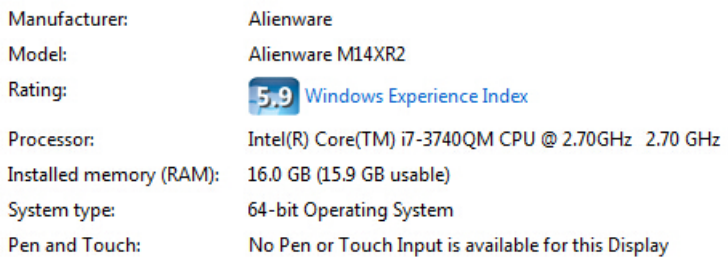



Figure 4.15: System experimental

exception of databases, which are required for processing the distributed operations on server peers.

We conducted three experiments, one for each operation. In each experiment, the number of nodes in the scenarios were configured from 100 to 1100 with an incremental increase of 100. In each scenario, the number of messages created for an operation was recorded. At the end of each scenario, the average number of messages is calculated. In this section, we provide the results collected from these experiments. The experimental results can predict whether the burden upon the system will increase rapidly as the system becomes more popular. We consider each server peer at least responsible for 10 users. With simulate up to 1100 servers we thus can simulate the system that hosts up to at least 11,000 users,we thus can simulate the a system that hosts up to at least 11,000 users. , and, if that happens meansIn the event of this occurring, it would imply the systems is already being very popular, and we can know allowing us to foresee what the kind of pressure will the system be faced may face. And the fact In fact, the regular server can host 1000 users easily.

### Product publishing operation

In this experiment, we assumed that every store has 500 subscribers. Among these 500 subscribers, 50 subscribers are online. We configured the environment to set the number of server peers from 100 to 800 peers in different scenarios. We measured the average number of messages created for a Product Publishing operation. Such an operation includes

two stages. One stage involves publishing of advertisements in BATON. The number of messages in this stage is $O(\log_2 N)$, which is determined by the lookup operation of BATON. Another stage is the pushing stage in which the advertisement is pushed to the client peers of online friends. We are interested in the measurements of the later stage. In the pushing stage, the host server sends the messages to all of its neighbours. Each message contains a list with a number of friends. The friends in a list are assigned to a neighbour according to BATON query algorithm. The neighbours should be able to further resolve the friends in the list.

Figure 4.16 shows the number of messages at the pushing stage of an Pproduct Publishing operation. The BATON Tree messages indicate that the message has been sent between two server peers. The system messages include the total number of messages in the system. This covers the messages in the BATON tree as well as those between a server peer and a client peer. In this figure, the number of messages coincide with $O((\log_2 N)^2)$. This indicates that the pushing algorithm can resolve the friends list quickly. Also, the number of messages among the server peers grows slowly with the increase in the number of servers. The system sends extra messages notifying online friends of the advertisements.

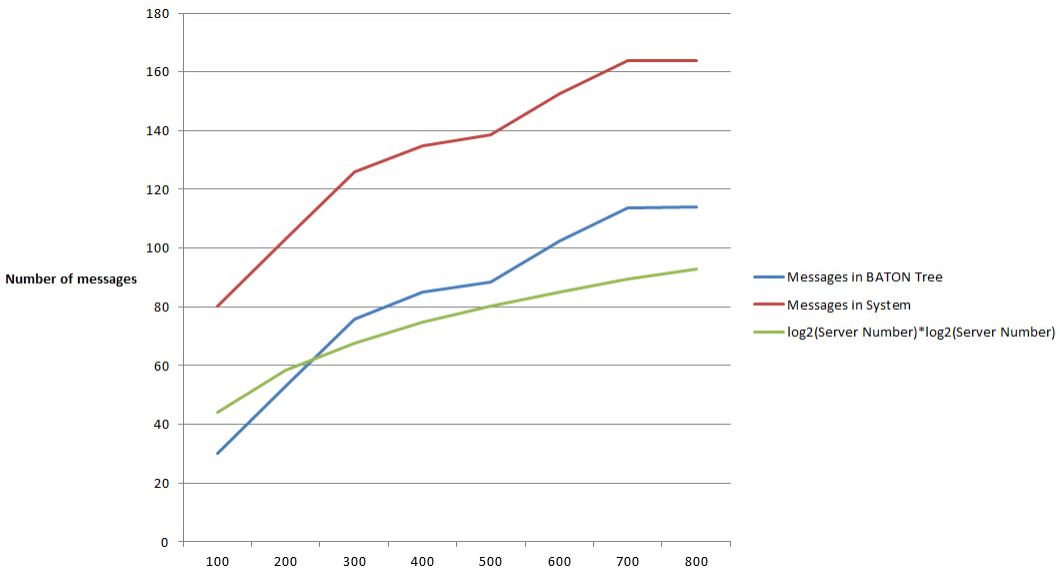



Figure 4.16: The average number of messages in publish product operation

### Product searching operation

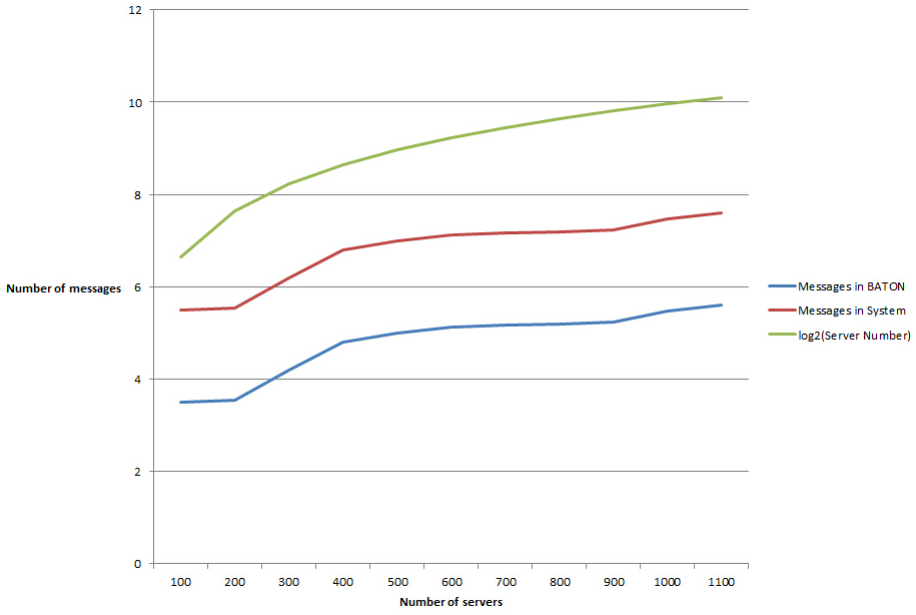



Figure 4.17: The average number of messages in advertisement searching

A product search operation is used mainly to search the advertisement of a product. The major cost of this operation is caused by the BATON lookup algorithm. Figure 4.17 shows the number of messages created in such an operation. We can see that an operation can be solved with a small number of messages (e.g., Figure 4.17 illustrates that in a BATON tree with 100 servers). Also, the number of messages increases by a small amount with the increase in the number of servers. For instance, in an environment with 11000 servers, the average number of messages is around 5. Therefore, we conclude that the messages occurring in the overlay network for the purpose of the product searching operation are manageable.

### User login operation

We also perform experiments on the user login operations in an environment with 800 servers or less. In this experiment, we assumed that every client has 100 friends and that 20 of them are online. As Figure 4.18 suggests, compared to the product searching

operation, the user login operation generates more messages. This is because, in a user login operation, the host server of the user performs a status update request in BATON for his online friends to update the status. However, the user can perform account and store management operations, and communicate with online friends who have already been notified, before the system messages have been received by user. Thus, the user login operation performance is still decent, even with the cost of the messages factored in.

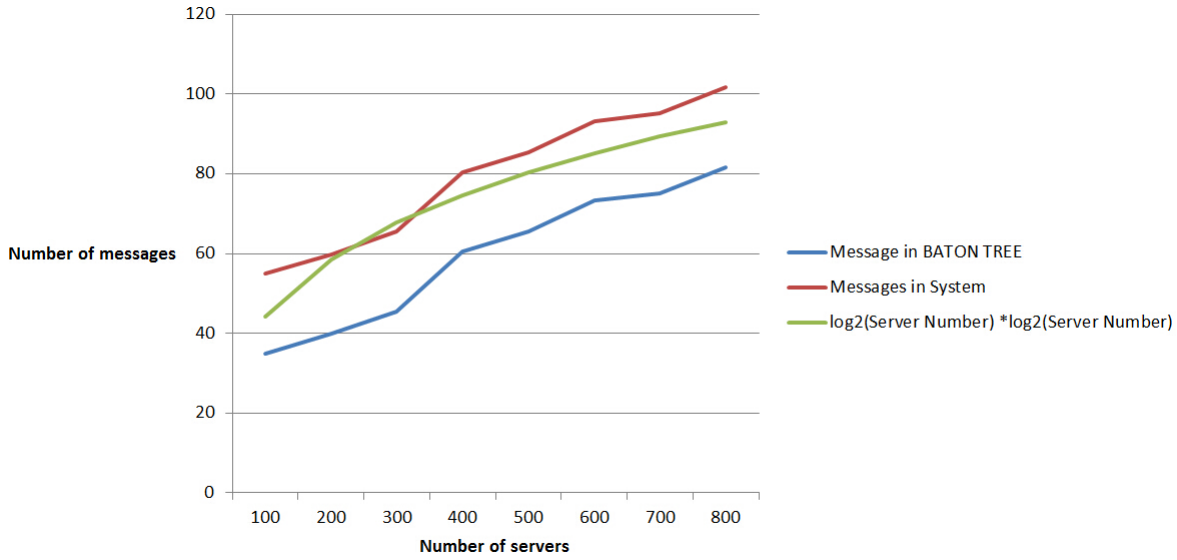



Figure 4.18: The average number of messages in user login operation

# Chapter 5

# Conclusion and Future Work

## 5.1 Conclusion

This thesis proposed an e-commerce platform with a P2P infrastructure. We studied a variety kinds of P2P system structures and the data sharing applications based on these P2P systems. The design of the proposed system was inspired by BestPeer++. We developed the basic e-commerce and social networking functions for this platform.

We specified the functional requirements of the system by analyzing the use case scenarios of an e-commerce application. According to the functional requirements, we designed the components on different peers: the Bootstrap peer, Server peer, and Client peer. We further analyzed the distributed operations and their implementations. These operations realize the use case scenarios of the platform.

The proposed system has a P2P distributed architecture. It can provide the e-commerce applications with a running environment that is scalable and fault tolerant. The results from the experiments show that the message costs do not increase significantly with the increase in the number of server peers. The number of messages created for an individual operation is quite manageable compared to the number created for an unstructured P2P infrastructure.

There are limitations to my work. First, the system communication is enabled by

using socket which cannot be transferred directly in the real world system. Second, I have only implemented the English auction model for the system, while there are several other auctions that can also be implemented. Third, I haven't implemented access control and trust models for this system, despite their known importance to e-commerce. However, judging by the prototype that we designed, we can have great aspirations for the future of this application.

## 5.2 Future Work

We designed and developed a prototype for an e-commerce platform. Future work can be categorized into three categories: server resource management, trust models and fault tolerant services. In the context of server resource management, we need to develop a policy to determine the number of resources a server should host and the load balancing policy that should be used when selecting a resource that must be moved from one server to another. We also need to consider the server capability in the load balancing policy. Since a trust model is crucial for e-commerce, we need to develop a good trust model for the P2P infrastructure; this will provide users with a better environment for shopping. The system should have such a fault tolerant mechanism to handle the failure of servers in the server farm. The mechanism can recover the data lost in the possible event of peer nodes leaving. In addition, the system needs a backup policy to specify the backup approach used for replication.

# References


[1] Alibaba trade manager software review. http://importexportbook.devhub.com/blog/412288-alibaba-trade-manager-software-review/. Accessed: 2014-09-25.

[2] Freenet project. http://www.gnu.org/philosophy/gnutella.html. Accessed: 2014-09-25.

[3] Gnutella project. http://www.gnu.org/philosophy/gnutella.html. Accessed: 2014-09-25.

[4] Napster wiki. http://en.wikipedia.org/wiki/napster. http://en.wikipedia.org/wiki/Napster. Accessed: 2014-09-21.

[5] New wso2 case study describes how ebay uses open source wso2 esb to process over 1 billion transactions per day. http://wso2.com/about/news/new-wso2-case-study-describes-how-ebay-uses-open-source-wso2-esb-to-process-over-1-billion-transactions-per-day. Accessed: 2014-09-25.

[6] Peer-to-peer wiki. http://en.wikipedia.org/wiki/Peer-to-peer. Accessed: 2014-09-25.

[7] Social media ads: Fairly annoying, rather useless, somewhat effective. http://www.marketingcharts.com/wp/online/social-media-ads-fairly-annoying-rather-useless-somewhat-effective-25213/. Accessed: 2014-09-25.

[8] Usenet wiki. http://en.wikipedia.org/wiki/Usenet. Accessed: 2014-09-25.

[9] A. Andrzejak and Z. Xu. Scalable, efficient range queries for grid information services. In *Peer-to-Peer Computing*, pages 33–40, 2002.

[10] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The hyperion project: from data integration to data coordination. pages 53–58, 2003.

[11] J. Aspnes and G. Shah. Skip graphs. 2007.

[12] S. Buchegger, D. Schiberg, L.-H. Vu, and A. Datta. Peerson: P2p social networking: early experiences and insights. In *SNS*, pages 46–52, 2009.

[13] D. Carmichael and D. Cleave. How effective is social media advertising? a study of facebook social advertisements. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 226–229, Dec 2012.

[14] G. Chen, T. Hu, D. Jiang, P. Lu, K.-L. Tan, H. T. Vo, and S. Wu. Bestpeer++: A peer-to-peer based large-scale data processing platform. In *ICDE*, pages 582–593, 2012.

[15] C.-H. Chiu, T.-M. Choi, and D. Li. Price wall or war: The pricing strategies for retailers. pages 331–343, 2009.

[16] W. S. Chow and L. S. Chan. Social network, social trust and shared goals in organizational knowledge sharing. pages 458–465, 2008.

[17] A. Crainiceanu, P. Linga, J. Gehrke, and J. Shanmugasundaram. Querying peer-to-peer networks using p-trees. In *WebDB*, pages 25–30, 2004.

[18] A. Datta, M. Hauswirth, R. John, R. Schmidt, and K. Aberer. Range queries in trie-structured overlays. In *Peer-to-Peer Computing*, pages 57–66, 2005.

[19] J. Guillory and J. T. Hancock. The effect of linkedin on deception in resumes. pages 135–140, 2012.

[20] A. Gupta, D. Agrawal, and A. E. Abbadi. Approximate range selection queries in peer-to-peer systems. In *CIDR*, 2003.

[21] A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: data management infrastructure for semantic web applications. In *WWW*, pages 556–567, 2003.

[22] H. Handschuh, L. R. Knudsen, and M. J. B. Robshaw. Analysis of sha-1 in encryption mode. In *CT-RSA*, pages 70–83, 2001.

[23] D. Hausheer and B. Stiller. Peermart: Decentralized auctions for bandwidth trading on demand. 2007.

[24] H. V. Jagadish, B. C. Ooi, and Q. H. Vu. Baton: A balanced tree structure for peer-to-peer networks. In *VLDB*, pages 661–672, 2005.

[25] K.-Y. Kwahk and X. Ge. The effects of social media on e-commerce: A perspective of social impact theory. In *HICSS*, pages 1814–1823, 2012.

[26] Z. Li and C.-C. Kuo. Revenue-maximizing dutch auctions with discrete bid levels. pages 721–729, 2011.

[27] D. Lucking-Reiley. Auctions on the internet: What's being auctioned, and how? *Journal of Industrial Economics*, 48:227–252, 1999.

[28] S. Mabu, D. Yu, C. Yue, and K. Hirasawa. No time limit and time limit model of multiple round dutch auction based on genetic network programming. pages 3–12, 2011.

[29] S. M. Mudambi and D. Schuff. What makes a helpful online review? a study of customer reviews on amazon.com. pages 185–200, 2010.

[30] W. S. Ng, B. C. Ooi, and K.-L. Tan. Bestpeer: A self-configurable peer-to-peer system. In *ICDE*, page 272, 2002.

[31] W. S. Ng, B. C. Ooi, K.-L. Tan, and A. Zhou. Peerdb: A p2p-based system for distributed data sharing. In *ICDE*, pages 633–644, 2003.

[32] C. X. Ou and R. M. Davison. Technical opinion: Why ebay lost to taobao in china: the global advantage. pages 145–148, 2009.

[33] W. Pugh. Skip lists: A probabilistic alternative to balanced trees. In *WADS*, pages 437–449, 1989.

[34] S. Pujari. Network planning: Centralized vs. decentralized data processing computer networking. http://www.yourar(ticlelibrary.com/networks/network-planning-centralized-vs-decentralized-data-processing-computer-networking/10447). Accessed: 2014-09-23.

[35] S. Ratnasamy, P. Francis, M. Handley, R. M. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[36] S. C. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. Opendht: a public dht service and its uses. In *SIGCOMM*, pages 73–84, 2005.

[37] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[38] O. Schneider. Trust-aware social networking: A distributed storage system based on social trust and geographic proximity. FU Berlin, Berlin, Germany, January, 22, 2009.

[39] J. Song and J. Baker. An integrated model exploring sellers' strategies in ebay auctions. pages 165–187, 2007.

[40] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, pages 149–160, 2001.

[41] G. Swamynathan, C. Wilson, B. Boe, K. C. Almeroth, and B. Y. Zhao. Do social networks improve e-commerce?: a study on social marketplaces. In *WOSN*, pages 1–6, 2008.

[42] L. M. O. B. C. Vu, Quang Hieu. *Peer-to-Peer Computing*.

[43] X. Zhang. Building personalized recommendation system in e-commerce using association rule-based mining and classification. In *SNPD (3)*, pages 853–857, 2007.

[44] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. pages 41–53, 2004.

[45] G. Zhao, N. Zhang, J. Li, and Z. Liu. Structure of the network for taobao. In *ICEE*, pages 361–364, 2010.