NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A LIGHTWEIGHT TWIDDLENET PORTAL

by

Antonios Rimikis

March 2008

Thesis Co-Advisors: Gurminder Singh
Thomas Otani

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 2008	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A Lightweight TwiddleNet Portal	5. FUNDING NUMBERS
6. AUTHOR(S) Antonios Rimikis	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release, distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

TwiddleNet is a distributed architecture of personal servers that harnesses the power of the mobile devices, enabling real time information and file sharing of multiple data types from commercial-off-the-shelf platforms.

This thesis involves research in mobile personal members, mobile social networks and media sharing models and develops a TwiddleNet portal running on a smart phone or a PDA so that the entire TwiddleNet system can be run on handheld devices for rapid deployment in emergencies.

14. SUBJECT TERMS Mobile Personal Server, Seamless Mobility, Social Networking Models, Media Sharing Services, Peer-to-Peer Architecture.	15. NUMBER OF PAGES 87
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

A LIGHTWEIGHT TWIDDLENET PORTAL

Antonios M. Rimikis
Lieutenant, Hellenic Navy
B.S., Hellenic Naval Academy, Piraeus, Greece

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2008

Author: Antonios M. Rimikis

Approved by: Dr. Gurminder Singh
Thesis Advisor

Dr. Thomas Otani
Co-Advisor

Dr. Peter J. Denning
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

TwiddleNet is a distributed architecture of personal servers that harnesses the power of the mobile devices, enabling real time information and file sharing of multiple data types from commercial-off-the-shelf platforms.

This thesis involves research in mobile personal members, mobile social networks and media sharing models and develops a TwiddleNet portal running on a smart phone or a PDA so that the entire TwiddleNet system can be run on handheld devices for rapid deployment in emergencies.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank Richard Betancourt for his tremendous amount of effort and contribution to the TwiddleNet portal during our cooperation. In addition, I would like to thank Jonathan Towle, Chris Clotfelter, Rob Myers and Dirk Ableiter for their help and support. Furthermore, I would like to thank my wife Xanthippi for her patience, understanding and continuous support during my studies. Finally, I would like to thank my Professors Gurminder Singh and Thomas Otani for their guidance and valuable help during my whole effort.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

As the new mobile lifestyle becomes ubiquitous, single-network cell phones give way to multi-network, multi-function devices based on multiple integrated radios. Moreover, personal servers that virtualize the hard disk through a wireless connection in cell phones and in various computing devices have started penetrating our lives. Full-scale seamless mobility, which marries Wi-Fi and WiMax technologies with the 2G and 3G cellular networks, is, however, in an early stage of development. Together with the development of middleware platforms, it could make mobile devices capable of detecting and automatically selecting the best wireless network. This could indeed provide amazing services to our personal and professional lives.

In this context of ubiquitous technology and seamless mobility, social networking models become mobile, and give their users amazing access everywhere and at any time. Indeed, cell phone, PDA and smart phone users can now create their own profiles, make friends, participate in chat rooms, create chat rooms, hold private conversations, and share photos, videos and blogs. Some companies even provide wireless services that allow their customers to build their own mobile community.

While these new technologies and services improve and support a variety of multi-user sessions, they are basically designed for personal use. We have all been witnesses to natural disasters such as the Indian Ocean earthquake that caused the Asian Tsunami in December 2004 or hurricane Katrina in Louisiana and Mississippi in August 2005, during which the network infrastructure failed both during and after the disaster. Humanitarian assistance would be much more effective if capabilities afforded by social networks are used in the aftermath of natural disasters such as an earthquake, a flood, a hurricane, or wildfires. This is because such networking can assist humanitarian operations and disaster relief

missions by supporting communication among multiple entities at the same time. Such information can, in turn, improve decision turn-around time and resource allocation.

TwiddleNet is a system that exploits the capabilities of contemporary mobile devices, primarily smart phones. A special social network can be created, not for entertainment or knowledge sharing but for quick and immediate responses in the case of a natural or man-made disaster. The entire system can be run on handheld devices to support a small first-responder team, especially during the first 48 to 72 hours after the strike of a disaster. TwiddleNet uses the multiple communication modalities commercial off-the-shelf devises, such as GSM/CDMA, GPRS/EDGE, Bluetooth, and Wi-Fi, and can operate on several separate network infrastructures.

A. OBJECTIVES

The objective of this thesis is to develop a lightweight portal for a distributed mobile personal servers' network known as TwiddleNet. As mentioned above, TwiddleNet exploits the advanced capabilities and multiple applications of mobile devices to perform real-time content capture and publishing among teams of people. To support this capability, the portal plays a major role by maintaining the members' control and allowing them to download and upload content that a mobile device can support. It does so in a hybrid peer-to-peer fashion.

A key consideration in the development of the TwiddleNet architecture was the type of file sharing model that needs to be chosen. Due to the relatively small number of devices in TwiddleNet schema, we have chosen the Server-Client model, which is used in the first generation of peer-to-peer file sharing networks. In this centralized peer-to-peer model, the portal keeps track of all the files and images in its database. In other words, the portal is the "heart of the TwiddleNet" because it constitutes the gateway to every node and the repository of the shared information. Other design features addressing the portal are the

type of the database in relation to the software platform that supports primarily smart phones and the suitable web server for mobile devices.

B. SCENARIOS

The TwiddleNet application has been primarily designed to exploit the multiple networking modalities available in the current generation of smartphones. TwiddleNet enables well-organized and well-informed teams in the field of humanitarian assistance and disaster relief missions, in real time. Similar setups could also be used for military applications and especially in emergency cases. For a better understanding of the way that TwiddleNet provides its services, we will present two scenarios. The first scenario in the field of humanitarian assistance describes the way in which TwiddleNet can contribute to extinguish a vast forest fire. The second scenario focuses on putting out a fire aboard a war ship.

1. Scenario 1

In the pine forest of Monterey, a small fire has been notified to the local fire department and the first fire engines are arriving at the nearest point. The fire started from the top of the hill where a cell antenna was situated. The speed of the wind is about 50 miles per hour with changeable direction. The local head decides to face the fire with five teams of ten men each. Due to the destruction of the cell antenna, there is no existing communication infrastructure available. Therefore, the teams of fire-fighters set-up their own Wi-Fi networks quickly by using BGAN nodes. Broadband Global Areal Network is a form of satellite Internet and telephony that provides back-haul link and creates a local Wi-Fi cloud. For the current scenario, TwiddleNet can use the Wi-Fi network to enable fire fighters to share content with one another. The two first responders of every team take several pictures of the forehead of the fire, label them and share them on TwiddleNet. TwiddleNet automatically tags the images with date/time, location and author information. Chris, the local head of the fire brigade, is

heading up the command post and he notifies that new pictures are available from the first two responders of every team. Chris assesses these photographs with the current speed of the wind and decides to call for additional help. In half an hour, three helicopters enter the operations and suddenly one of them notices that one of the teams is almost within the fire. The co-pilot takes photographs from the air and shares them onto TwiddleNet with a map and the route that the team must follow, in order to exit from the fire.

2. Scenario 2

Fire in the forward engine room of frigate F-1234 is in progress and the first responder's team is ready to enter into the compartment. Action station has been declared aboard the ship and the Chief Engineer together with the Damage Control Officer, are at the Damage Control Center where they are continuously informed about the status of the fire. The actions initially required have already taken place and the first responders who entered the engine room have found that the fire was caused by fuel leakage of the main forward gas turbine. The distance and the level of the fire let them take a couple of photographs and share them on TwiddleNet, which is still in operation through the access points along side the ship. In the current situation, in which electrical isolation has taken place, these access points operate in emergency mode through local batteries. In the Damage Control Station, the Officers receiving the alerts see the photographs of the fire and immediately give new orders to the fire team. From the information they have received through the walky-talkies from the fire team, they trace the fire and smoke limits in proper diagrams and distribute these every minute again through TwiddleNet, to the heads of every team aboard the ship. In this way, every one aboard the ship is informed about the proper route that must follow to move safely from one compartment to the other.

C. RESEARCH QUESTIONS

1. What is the most effective design for a web-based portal on handheld devices?
2. Which is the best web-server and the database system for a handheld device?
3. Which is the best way to develop a portal, running on a handheld device, while keeping in mind the power and processing limitations of the device?
4. What management tools are needed for a handheld-based portal?
5. How the time of transmission can be reduced for a handheld portal?

D. SCOPE AND METHODOLOGY

The scope of this thesis is to search the different media sharing models and design a portal that can provide the following to distributed personal servers which compose the devices/member of the group: (1) connectivity and access to a common independent network, in which they can share information in real time. (2) a repository for shared files among the members. (3) an alert system that notifies users when there is a new content available in the repository. After the design and the development of the software, we test the prototype using smart phones that run Microsoft Windows Mobile 5.0, sending a various number of images.

E. ORGANIZATION

The chapters in this thesis are organized as follows. Chapter II is a review of previous work related firstly to mobile personal servers and seamless mobility and secondly to mobile social networking and media file sharing services. Chapter III is an overview of the TwiddleNet system. In this chapter, we shall describe the TwiddleNet portal, its concept, architecture and design, and, finally, we shall deal with aspects of its software development. Chapter IV will detail the prototype's performance and will look into measurements, compatibility with

specific access points, encryption issues and restrictions. Chapter V completes this thesis with conclusions and recommendations for future work.

II. BACKGROUND

This section deals with mobile personal servers, content sharing models and distributed networks. Issues specific to TwiddleNet architecture which comprises a proposed mobile personal server, are also examined.

A. MOBILE PERSONAL SERVERS

A mobile personal server is any small, lightweight battery-powered mobile device with capability for processing, data storage and some forms of network connectivity, such as Bluetooth and Wi-Fi. It may not have any standard I/O capabilities such as keyboard or display, and may be accessible from any computing device within the range of the wireless connection [1]. Roy Want, a Principal Engineer of Intel Corporation, gives the following simple explanation about Personal Server:

> A simple way to think about the concept is to ask this question: What makes your PC your PC? The answer is that it is . . . the hard disk, which contains your data; the rest is simply the access device. Using the Personal Server concept, we are virtualizing the hard disk through a wireless connection to whatever computing device is nearby and available [2].

TwiddleNet makes use of the utility and capabilities that these devices give, and composes a social network, which consists of hybrid personal servers. For a better understanding, a little history of personal servers follows.

1. General Issues about Personal Servers

The evolution of personal servers is still at an early stage; no successful commercial product is readily available in the market. The common theme is to carry user's personal data but differences exist from one product to another in how this capability is implemented. All personal server products exploit the continuously increasing computing power and the multiple network connectivity options of cell and smart phones.

Roy Want and his ubiquity group within Intel envisioned the idea of personal server in 2002 [2]. In this early conception, the prototype device was a lightweight computer with high-density data storage capability and was small enough to fit in a pocket. The personal server represents the integration of three technologies. The first is the high density and the small volume storage. The second is the high performance processors with low power such as StrongARM and XScale; and the third is a standardized high bandwidth wireless protocol such as Bluetooth. A personal server allows a user to “carry” the hard disk of a PC to whichever computing device that is available through a wireless connection.

Figure 1. Prototype of an Intel XScale/Linux-based personal server (From: [2])

In 2004, Intel’s new project personal media server extended the initial personal server concept by making it phone-based and using it as a personal media server. Taking advantage of advances in processing, storage and communication technologies, Intel’s project provided ubiquitous access to personal information and applications through the existing fixed infrastructure. The integration of Personal Media Server technology has the potential to add significant value to cell phones, which enables them to be used for a broad range of mobile and wireless computing applications. As storage continues to increase in density, this model of mobile computing could become even more attractive and provide reassurance to users that they would always have their documents and media available while in the move [3].

Figure 2. Prototype of an Intel XScale/Linux-based cell phone with personal server integrated (From: [3])

BluOnyx is a brand of mobile content server, which was introduced in December 2006 by Agere Sytems. BluOnyx is about the size of a credit card and enables mobile users to share and stream music and video through electronic devices ranging from cell phones to PCs, digital cameras, game machines, DSL routers and many more. BluOnyx is a peer-to-peer device that does not require a PC for its operations but can move content to and from a PC using USB or SD cards and wireless connectivity such as Bluetooth and Wi-Fi. The amount of storage ranges from one to forty gigabytes. The whole system consists of 3 subsystems, the storage, the application processing and the connectivity subsystem [4].

Figure 3. Agere's BluOnyx mobile content server (From:[4])

With a BluOnyx server, consumers may be able to do the following easily and quickly:

- Wirelessly access and display personal content carried on the BluOnyx server.
- Create a digital campfire, broadcasting and sharing content with multiple friends, authorized to access the BluOnyx server through their cell phones.
- Stream videos to one or more cell phones.
- Back up pictures, music, video, emails, personal and business documents and images from cell phones, cameras and PCs.
- Enable access to the Internet for cell phones and PDAs that are not broadband enabled and cannot access the Internet on their own.
- Protect content via multiple tiers of authentication and encryption.

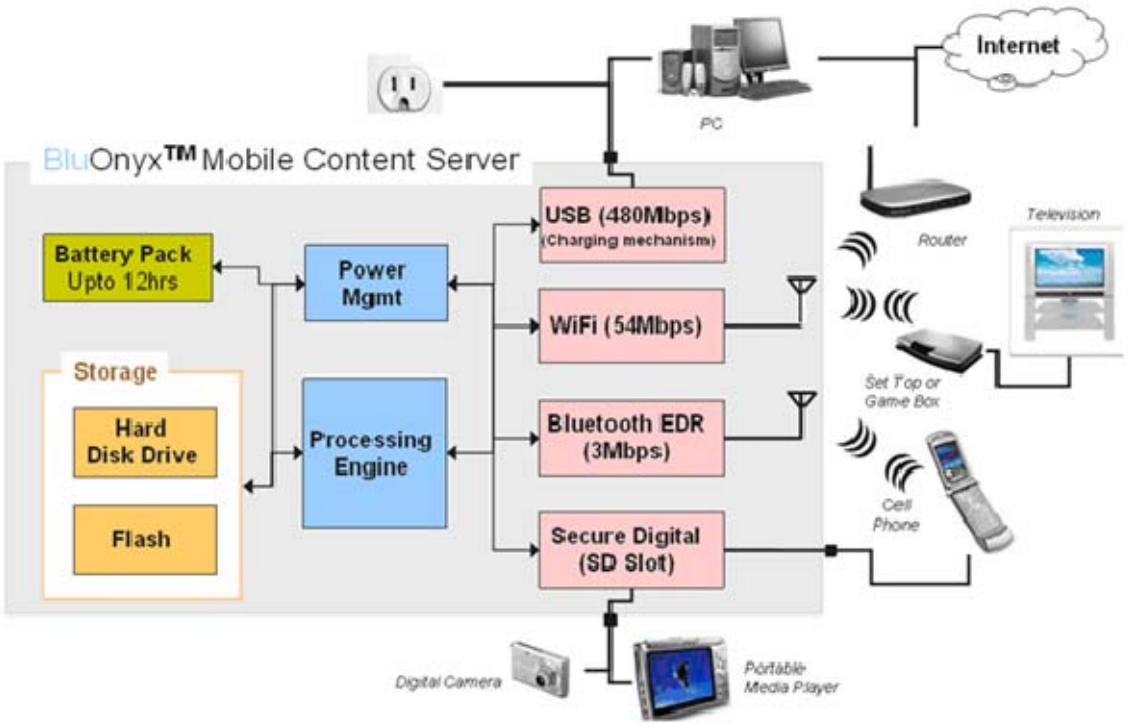

Figure 4. BluOnyx's subsystems: the storage, the application processing and the connectivity subsystem (From:[4])

John Barton, Shumin Zhai and Steve Cousins of IBM proposed, in 2006, a personal server embedded in a cell phone equipped with large digital storage [5]. With this specific model, there is the capability of a PC at work and at home but with the portability of a phone in the intermediate period during traveling. One important issue is the processing power. The design is intended to have one mobile CPU dedicated to personal storage and communications, while other CPUs work with the display and the keyboard. Of course, a cell phone is unlikely to have the same processing power as a PC. However, a phone combined with a television containing one or more processors may offer mobility with performance when needed. Another major issue is power. In the future, computer displays and televisions will be designed in a dual way to allow people to use their large screens and to charge their cell phones while working on them.

2. Seamless Mobility

Seamless mobility is often viewed as the marriage between cellular and Wi-Fi technologies. Seamless mobility is enabled by a set of technologies that allows users to be automatically and uninterruptedly connected and able to communicate anywhere and anytime regardless of devices, networks and locations. It offers an experience continuity of communication without any interruption.

The following scenario may facilitate a better understanding of seamless mobility. As John arrives in town by train, he receives an email on his PDA, using the railway station's wireless network. The email is from his friend Chris who asks John for a coffee at the Moon Deer Café downtown. John then takes the first available taxi to downtown. During the ride, John sends a confirmation email to Chris from his PDA, which switches to a GPRS wireless network as the WLAN connection of the railway station is no longer accessible. Arriving at the coffee shop and after a long conversation with Chris, they both start to download sources for their tomorrow's presentation. Their PDAs are now connected with the high-speed WLAN network of the coffee shop, which enables them to acquire quickly the necessary files for the project. Suddenly John remembers that he has a family obligation and he must leave. While Chris gives him a ride home, John continues to download the files. John's PDA has now switched the connection to the WiMAX network to seamlessly continue the download. When John arrives home, he has already finished with the download and he writes the first lines of the report. As he enters the house, his device detects his higher bandwidth home WLAN and switches to it. He decides to switch his PDA with his desktop machine for a larger monitor and a keyboard. John switches his working environment seamlessly by transferring the files via the personal server. Finally, he finishes the first draft and sends to Chris.

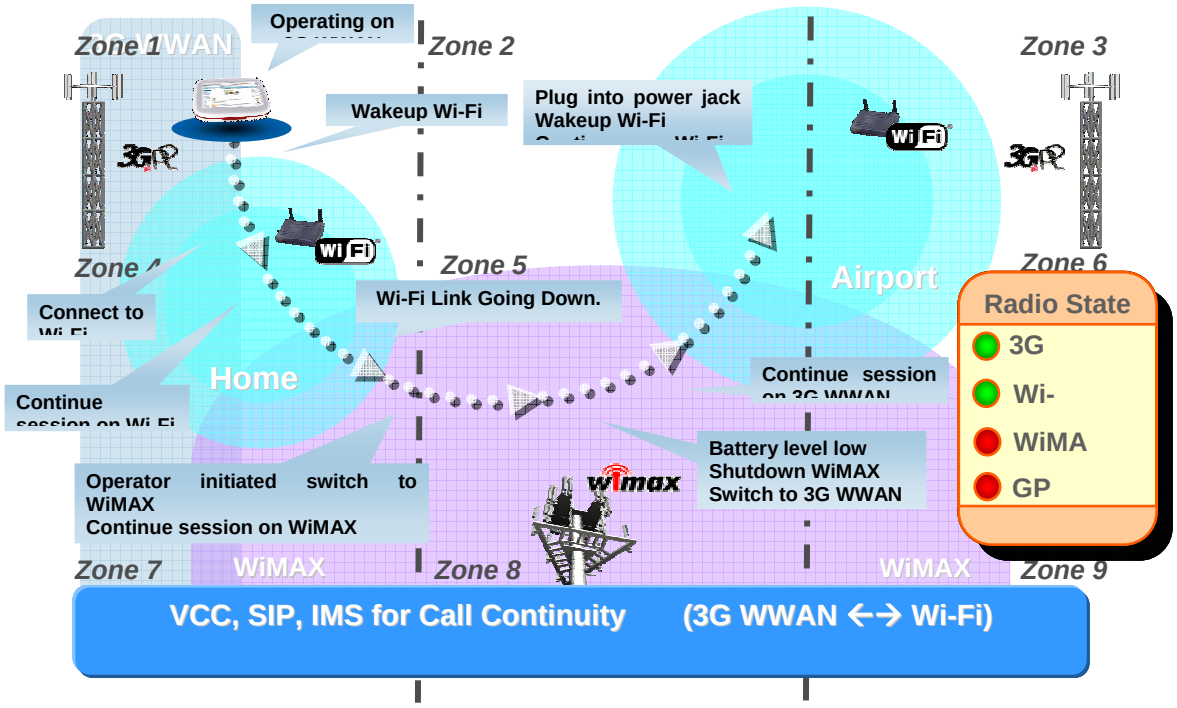

Figure 5. Mixed-Network Client Use Case (From:[6])

To be able to maintain the connection and to experience the broadband multimedia content while crossing the heterogeneous networks seamlessly, a set of technologies must be used. These technologies include but are not limited to GSM, CDMA, UMTS, WLAN and Bluetooth, as well as emerging technologies such as HSDPA, HSUPA, IMS and WiMAX.

To accelerate the development of seamless mobility, IEEE established several work groups to define the standards of handovers between both homogeneous and heterogeneous networks.

IEEE 802.11k and 802.11r are the chosen standards to support seamless mobility in WLAN environment. IEEE 802.11k enables intelligent roaming within the Wi-Fi network by provisioning the algorithm to discover the best available access point. IEEE 802.11r defines the secure and fast transition between access points within the same Extended Service Set.

Mobility issues for WiMAX are dealt with in 802.16e standard, while 3GPP and 3GPP2 define the standard of mobility in cellular networks.

It is the responsibility of IEEE 802.21 standard to maintain uninterrupted sessions while crossing heterogeneous networks, including WLAN, WiMAX and cellular. IEEE 802.21 defines the handover between Layer 2 and Layer 3 of the OSI network model. The standard can support handovers for both mobile and stationary users.

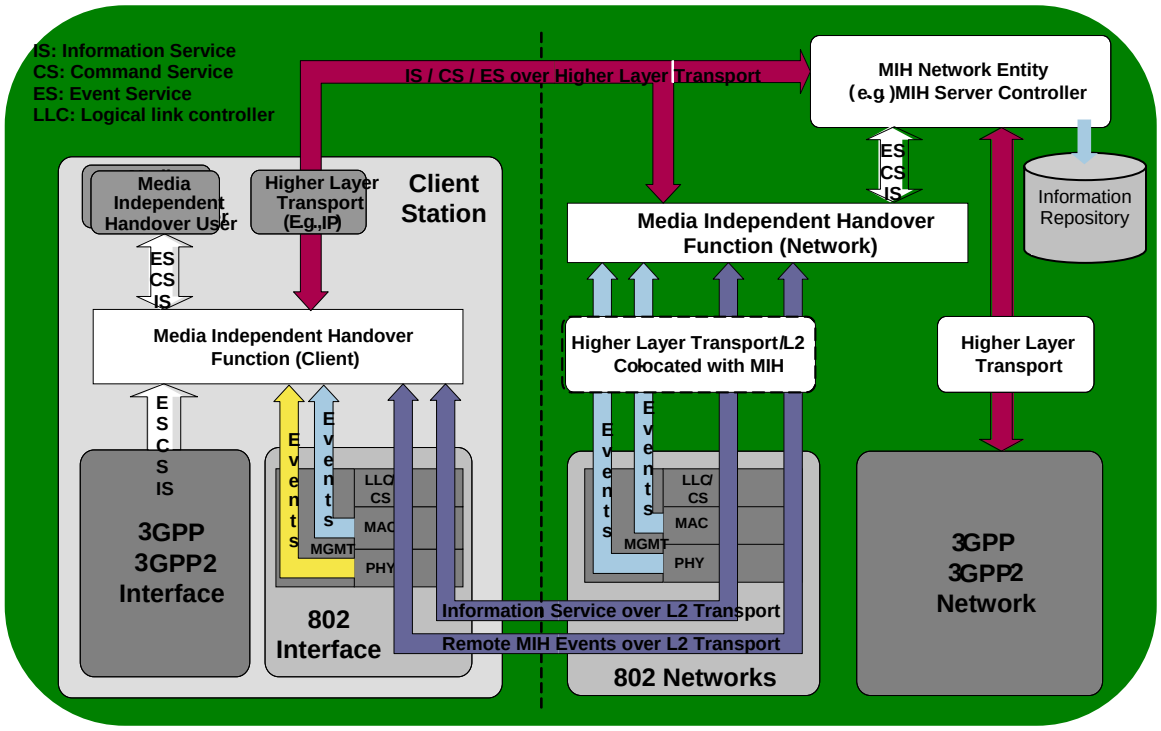

Figure 6. IEEE 802.21 Services (From:[6])

Expanding the concept of seamless mobility, Markus Bylund and Zary Segall proposed three major factors to achieve seamless mobility [7]: network capabilities and coverage; UI and device capabilities; and the importance of true

user experience continuity. According to Bylund and Segall, very small, portable server systems can allow their users to seamlessly access networks retrieve data from embedded sensors, control embedded actuators, perform functions proactively on the user's behalf, and interact with other people's personal systems as people come into proximity. Personal servers provide a variety of new, dynamic modes of operation and interaction depending on the environment and the user's needs; in other words, they are potentially highly suitable for realizing seamless mobility.

Infinity is a middleware framework introduced by IBM in January 2007, which unifies techniques for local and remote device data access, automatic communication channel selection and application deployment across heterogeneous platforms [8]. Infinity enables information sharing and collaboration among mobile devices in a privacy-preserving manner, regardless of operating system, hardware or communication nodes. Stefan Schoenauer, the lead researcher on the Infinity team, noted:

> Devices nowadays speak so many different languages that it is very hard to connect devices and to share information among them. So we have built a piece of software that runs atop all these mobile devices and makes them speak a common language, makes the exchange of information easier and takes security and privacy into account so that you're only sharing information with whom you want.

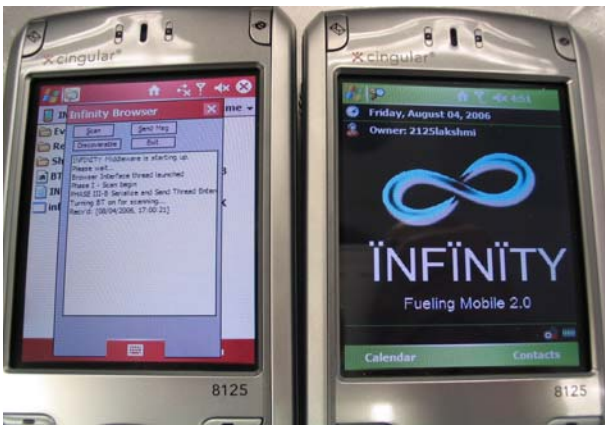

Figure 7. Infinity, Deploying the Evacuation Routing Application (From:[7])

One of the three scenarios that have been described by IBM illustrating the utility of Infinity is the so-called Evacuation Routing. According to this scenario, the employees of an enterprise could evacuate their building in the case of an emergency, following the best exit and bypassing congested areas. The employees must just install a simple application on their cell phones or PDAs that contains a map of the building in which the employee is situated, including the locations of available exits. The evacuation application could query other local devices to determine the location and concentration of evacuees in each exit area. The aggregate query results that return to each mobile device provide current and reliable information about the status of evacuation, and allow each employee to find the most expedient route out of the building.

3. Personal Servers and TwiddleNet

TwiddleNet is a mobile personal server architecture that enables real time sharing of content by exploiting the content capture capability of smart phones. It utilizes the multiple communication modalities available in modern smart phones to provide a fail-safe and rapidly deployable infrastructure. Usually the communication scheme consists of GSM/CDMA, GPRS/EDGE, Wi-Fi or Bluetooth.

The elements that compose the TwiddleNet architecture are the portal and the clients. The portal is a unique entity and has a centralized role within the architecture. It performs the function of a central repository for the network's shared metadata. It enables clients to join a TwiddleNet group and to send and receive information of interest. The clients are distributed around the portal in order to operate either as personal content servers or as personal content requesters. They can switch between these roles rather transparently to the user. In the content server mode, the client captures or shares current content and sends the equivalent metadata informing the portal about content availability. In the content requester mode, the client receives updates or requests permission for selected content from the portal in order to receive the content from the client

that possesses the content and is operating in the server mode. The general simplified TwiddleNet infrastructure is shown in the Figure 8:

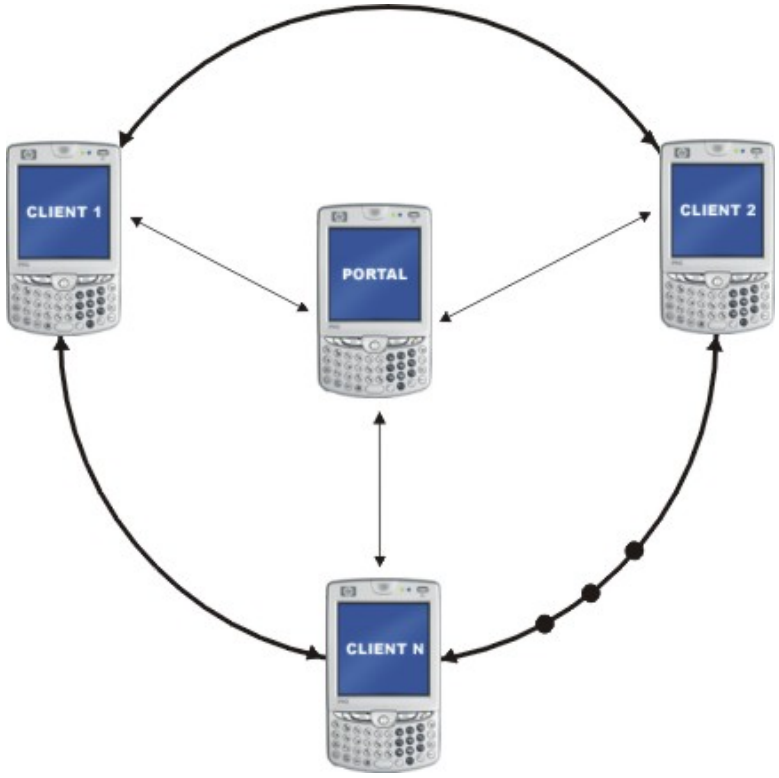

Figure 8. General simplified TwiddleNet Infrastructure

Before we deal with the way in which all these personal servers coordinate and communicate between themselves, we shall deal with some aspects related to the other major idea of the TwiddleNet, i.e. content capture, distribution and dissemination. Some of the aspects that will be described in the second part of this chapter are the current social networking models, the media sharing services and the peer-to-peer architecture.

An attempt to compare the personal servers that compose the TwiddleNet and the personal servers that have already been examined in the previous paragraphs indicates that any comparison is risky. This is because TwiddleNet does not use any of the models that have been examined. Furthermore, it

implements improvised software applications that run on mobile devices such as PDAs and smart phones composing the personal servers of TwiddleNet.

As mentioned earlier in this chapter, Infinity is a middleware framework that enables information sharing and collaboration among mobile devices in a privacy-preserving manner. In general, both Infinity and TwiddleNet focus on enabling access to content on mobile devices. The most significant difference between these two models is that, while TwiddleNet operates on smart phones as a personal server and provides a portal front-end to enable content access, Infinity is a middleware environment that develops general data sharing applications for smart phones.

B. CONTENT SHARING AND DISTRIBUTION

As already mentioned, TwiddleNet is not just a personal server that views smart phones as computing devices with limited screens and no keyboards. Instead, TwiddleNet exploits the content capture capability of smart phones and creates a complete network among personal server mobile devices instead of a stand-alone personal server only.

In order to understand the way in which TwiddleNet manages to compose this entire network, in what follows we shall deal with the basic ideas behind this model. In particular, we will examine the current and future capabilities of the mobile social networking models. Following this, we will present some aspects of Peer-to-Peer architecture and the basic media sharing models. Finally, we will present the content distribution and sharing mechanisms of TwiddleNet.

1. Social Networking Models

A social networking model is an Internet community in which people who register can upload and share their favorite songs, videos, photographs and in general, multiple files. A mobile social networking is a social service where one or more individuals of similar interests can connect and converse with one another using their mobile devices such as cell phones, PDAs and smart

phones. Today, the increasing computer power and miniaturization of mobile devices make such networks mobile.

One of the most popular social networks is My Space with 70 millions visitors who load the site with photos and videos, news about music groups and ideas and thoughts of their likes and dislikes. Another well-known site is Facebook, which initially restricted its memberships to students of Harvard College. It was subsequently expanded to other universities and even to high schools and some large companies. It has been estimated that the site had more 60 million users by the end of 2007. Other social network sites include LinkedIn, which targets professionals, and Xanga, which is a blog-based community site. An estimated number of three hundred sites, including smaller ones such as the StudyBreakers for high school students and the Photobucket for posting images, compose the social network universe [9].

Due to the continuously increasing use of cell and smart phones, especially by teenagers who populate most social networks, these networks have already begun to be mobile. From April 2006 well-known networking groups such as MySpace and Facebook started, with the cooperation of large cell providers and phone companies, to create their new mobile social networks.

Q121 is a free mobile social networking launched in June 2007 by the online marketing firm Traffix. People who register with Q121 can upload their favorite songs, videos and photos onto the site, and then send them to the cell phones of other registered users. Q121 is also counting on amateur and user-generated content to drive use of its network. So far, its 50,000 members, mostly in their teens and early twenties, use it primarily to exchange music files, a large number of which has been recorded and mixed by the users, themselves. Q121 provides an "express signup" page for artists and bands who would like others to hear their songs or use clips as ringtones [9].

Vipera is a media cell phone community which lets every one to browse through the sites content, post blogs, meet people, make comments, and have discussions on members' interesting topics, all this from their cell phones. The members can create a profile with their photo, list of interests and links to their blogs. They can rate members and make comments about their interests. Vipera is a global community and hosts members from all over the world. Vipera is the first interactive media community designed for mobile, inspired by events and generated by opinions [11].

Myubo developed by Tom Horn Enterprise is a universal, video sharing system which uploads, views and shares live and pre-recorded video via mobile 2.5G and 3G networks such as GPRS, EDGE, CDMA, UMTS, web and fixed IP networks.

Sonopia, which was launched on April 2, 2007, aims to fundamentally shift the value proposition in the mobile phone market from the carriers to an affinity model where all parties involved share revenue. Juha Christensen, a seasoned mobile veteran who helped to found Symbian and was the president of Macromedia prior to starting this latest venture, heads the team at Sonopia. Christensen told ETel that the impetus for starting Sonopia was to increase differentiation in the U.S. mobile market. The team quickly realized that providing a service for anyone to create his or her own affinity-branded mobile proposition could be a killer app in the growing MVNO (Mobile Virtual Network Operator) space. Sonopia is a platform that allows for user-generated content and community building among affinity groups. It is very different from the MVNOs that we have seen to date, most of which appeal to the MTV crowd or value-conscious prepaid customers. The ETel blog will be tracking the progress of Sonopia and will have a comprehensive review of the service in the coming weeks [12].

Twango, which was acquired by Nokia in July 2007, is an online media sharing site that supports multiple file types such as photos, video, audio, and documents. It provides users a means of repurposing their media, including

sharing, editing, organizing and categorizing. In addition, Twango saves all the original media and its metadata. Non-members are free to browse the site, while members can upload media of their own. Sign up for a basic account is free and provides 250 megabytes of upload bandwidth per month. Twango users organize their media in containers known as channels. A user's entire media collection is stored in what is known as "My Media." Channels represent a cross-section of "My Media." The same piece of media can appear in multiple channels, giving a different context to said media. For example, a picture of friends at a party could appear in a user's "friends" channel as well as in the user's "party" channel.

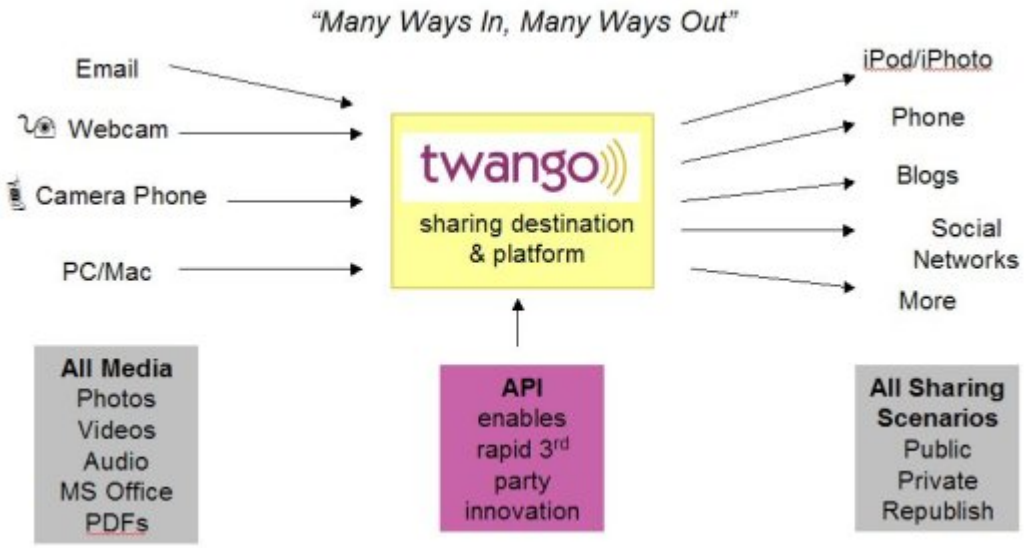

Figure 9. Illustration of Twango Infrastructure (From:[13])

Twango members can subscribe to a channel to receive notifications when new media has been uploaded to a channel or when new comments have been added. Each channel can also be specified as public or private. If it is specified as private, it can only be accessed by invitation. Users can also create open channels that allow other users to upload their own content. Twango Mobile provides an XHTML web site for people with compatible mobile browsers. This mobile site gives access to photos, videos, audios, and other media on Twango,

as well as some interactivity including the ability to add comments and share media to mobile numbers and email addresses [13].

2. Peer-To-Peer Architecture

A peer-to-peer computer network architecture is an architecture in which each node has the same capabilities and either node can initiate a communication session. Peer-to-peer model uses diverse connectivity between participants rather than conventional centralized resources where a relatively low number of servers provide the core value to a service or application. Peer-to-peer networks are typically used for connecting nodes via largely ad hoc connections. Such networks are useful for many purposes. Sharing content files containing audio, video, data or anything in digital format is very common, and so is real time data, such as telephony traffic, which is also passed using Peer-to-Peer technology.

According Stephanos Androutsellis-Theotokis and Diomidis Spinellis:

> Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies with the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failures and accommodating transient populations of nodes while maintaining acceptable connectivity and performance without requiring the intermediation or support of a global centralized server or authority [14].

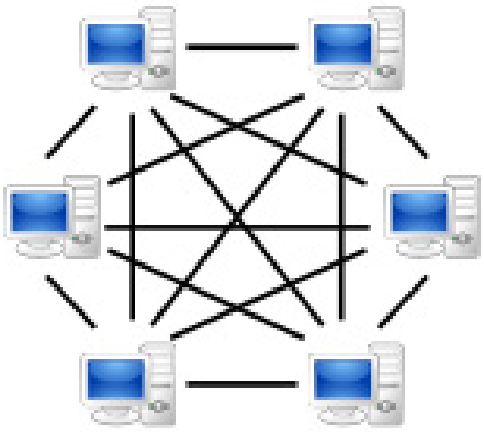

Figure 10. A pure peer-to-peer based Network (From:[15])

One of the greatest strengths of the peer-to-peer architecture is its scalability. Millions of peers may participate in the file-sharing community, with each one functioning as a server and contributing resources to the community. While each peer generates workload by requesting files, each peer also adds service capacity to the system by responding to the requests of other peers. In this way, the limited bandwidth, power and storage of one peer can be massed together to create a more powerful resource. In addition, since files may be shared between peers, individual device storage capacity is conserved.

On the other hand, there are some cases where the highly distributed and decentralized nature of the peer-to-peer architecture creates conditions that are difficult to manage. For instance if a peer who has the unique copy of an important file drops out of the community, this file will be unavailable for the whole community.

Both Peer-to-peer and Client-server network architectures have their advantages and disadvantages. Some advantages that P2P architectures have

over centralized client-server architectures are, the reduced censorship, the increased accessibility of popular content, the reduced susceptibility to single point of failures and increased privacy However, a Client-server configuration is preferable to peer-to-peer, especially in a small business environment where there is an expectation of growth.

In P2P networks, resources are usually distributed among many nodes. For example, even if one or more nodes depart and abandon a downloading file, the remaining nodes should still have the data needed to complete the download. By contrast, under client-server case, if a critical server fails, clients' requests cannot be fulfilled. The client-server paradigm therefore lacks the robustness of a good P2P network.

Traffic congestion on the network has been another issue since the inception of the client-server paradigm. As the number of simultaneous client requests to a given server increases, the server can become severely overloaded. The results are reversed in the case of P2P networks, where the bandwidth increases as more nodes are added. Indeed, in this case the overall bandwidth can be roughly computed as the sum of the bandwidths of every node.

In general, there are two distinct forms of P2P networks, the pure and the hybrid P2P. In pure P2P, all network entities are equal and if any randomly chosen entity is removed, the network will not suffer any negative consequence. In hybrid, P2P there is some functionality, which is still centralized. Some of the most well known hybrid P2P architectures are Napster and Gnutella. Due to their special architecture, we will examine these and other related P2P architectures in the following paragraph, where different media and file sharing models are presented.

3. Media Sharing Services

Media sharing service is the mechanism with which a social network is implemented and organized. Anyone can upload whatever he or she wants to share on the Internet, and others can search for anything and download it once they find it. These services usually follow the peer-to-peer (P2P) model, where the files are stored on and served by personal computers of the users. Most people who engage in file sharing on the Internet either provide files (upload) or receive files (download). From their beginning before 2000 to date, the different schemes of media sharing can be distinguished in four generations [15].

Server-Client is the first generation of peer-to-peer file sharing networks that had a centralized file list. In the centralized peer-to-peer model, a user would send a search to the centralized server of what they were looking for. The server would then send back a list of peers who have the data and facilitate the connection for download. Two well-known examples of this architecture were Napster and eDonkey2000.

Decentralization is the second generation where there is not any central index server. In the very beginning, there were many problems due to bottlenecks as the network was growing very quickly. FastTrack was the solution to the problem using “supernodes” to improve scalability. By electing some higher-capacity nodes to be indexing nodes, with lower capacity nodes branching off from them, FastTrack allowed for a network that could scale to a much larger size. Many networks quickly adopted this model and most current peer-to-peer networks implement this design, as it allows for large and efficient networks without central servers. Also included in the second generation were the distributed hash tables (DHTs), which help solve the scalability problem by electing various nodes to index certain hashes and by allowing for fast and efficient searching for any instances of a file on the network.

The best examples of this second generation are Gnutella, Kazaa or eMule with Kademlia, whereby Kazaa has still a central server for logging in. eDonkey2000/Overnet, Gnutella, FastTrack and Ares Galaxy have summed up approximately 10.3 million users.

Anonymous Peer-to-Peer is the third generation where a degree of anonymity is realized by routing traffic through other users' clients. These clients have the function of network nodes. This makes it harder for someone to identify who is downloading or who is offering files. Most of these programs also have strong encryption to resist traffic sniffing. Third-generation networks have not reached mass usage for file sharing. The reason is that the current implementations are too complicated, slow and user-unfriendly because they try to ensure anonymity. However, in countries where very fast fiber-to-the-home Internet access is commonplace, such as Japan, a number of anonymous file-sharing clients have already reached high popularity. Examples of anonymous networks are Rshare, Freenet, I2P, GNUnet and Entropy.

Streams over peer-to-peer, is the forth and last architecture model where apart the traditional file sharing, services send streams instead of files over a P2P network. Someone can then hear radio and watch television without any server involved, because the streaming media is distributed over a P2P network. It is important that instead of a treelike network structure, a swarming technology known from BitTorrent, is used. Best examples are Peercast, Cybersky and demo TV.

4. Content Distribution and Sharing in TwiddleNet

The existence of portal in TwiddleNet makes it a hybrid peer-to-peer architecture and makes it belong to the first generation of peer-to-peer file sharing networks, which is the Server-Client model. As in Napster, TwiddleNet has a centralized portal that keeps track of all the files and images in its database. In other words, the portal is the "heart of TwiddleNet" organism because it constitutes the gateway to every node and the repository of the

shared information. After that, the clients using the pure peer-to-peer method download their favorite file in the passive “searching case,” or the brand new file in the active “alerting case,” from the node that created or shared it. The following diagram shows the above hybrid peer-to-peer architecture.

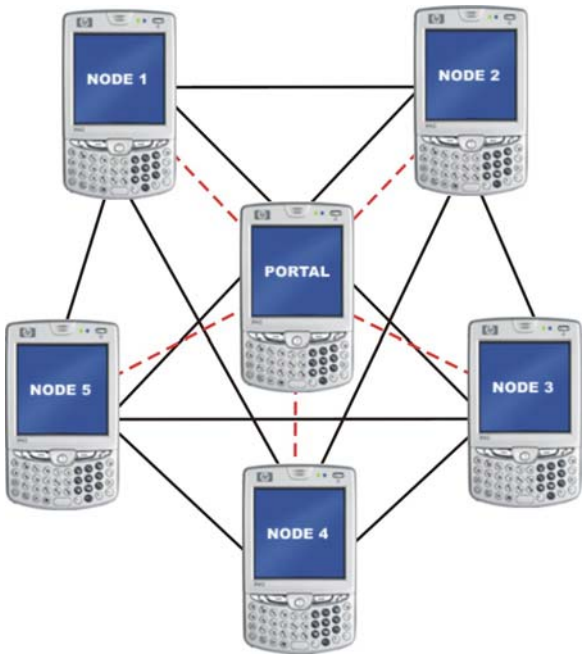

Figure 11. The hybrid peer-to-peer architecture in TwiddleNet

While the sharing mechanism in Napster and TwiddleNet is almost the same, there are differences in the way that TwiddleNet implements the content distribution.

In TwiddleNet, there are two different methods for a user to receive and view shared information. In the first one, the user or the member of the group receives automatically an alert for new information, when this information becomes available. In the second method, a member may just browse the portal to find specific information. The next step in both cases is the user to download

the info from the user that generated it. For a better understanding of content distribution and file sharing in TwiddleNet, we will describe the above two different methods in detail.

In the first method, every member has requested from the portal to be notified every time that a new document becomes available. The whole process may be described in three steps. In the first step, a new content related to the generated file is posted by a member to the portal, using pull technique. In the second step, the portal examines if the content matches the member's request. If the result is positive, the portal adds the content in its database and pushes an automated string alert to every member except the sender. The alert contains the title, type, author and the data and time of the new document. The portal sends to the author's device a confirmation message that the content has already been added in portal's database. Finally, in the third step, if the user chooses to view the document, he or she will pull it from the server by placing an HTTP GET request to the URL in the alert. The above entire process is shown step-by-step in the diagram below.

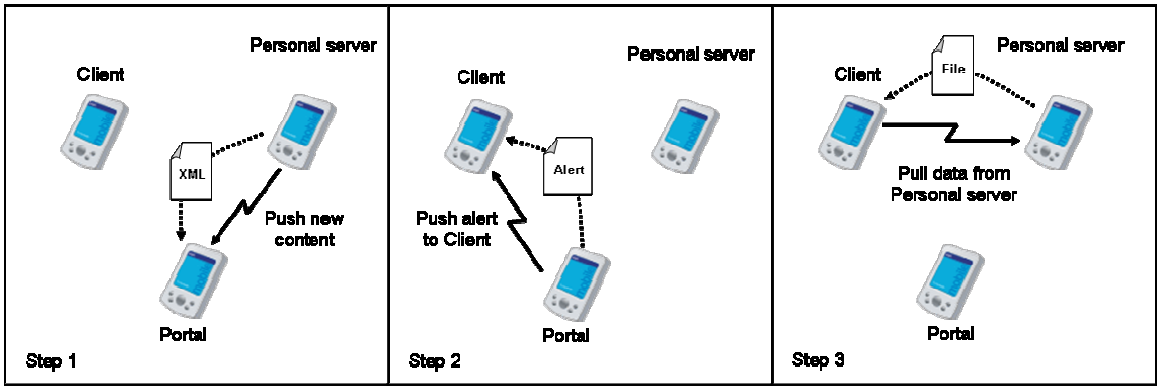

Figure 12. The connection procedures step by step in the Active Scenario of TwiddleNet

In the second method, a member can view content once a document has been posted in the portal. The process may be described in two steps. In the first step, the client initiates the process by searching the portal's URL. The user may search by author, title or date. By clicking, the search button pulls from the portal an XML with a list of documents and their corresponding tags. These tags give the title, author, summary, the date and the time of creation, the date and the time of update, mission, priority, type, link, phonenum and id code. In the second step the user by clicking the link provided initiates the HTTP session by sending a GET request that contains the URL for the specified document. The server device then returns an HTTP POST message including the document. The above entire process is shown step-by-step in the diagram below.

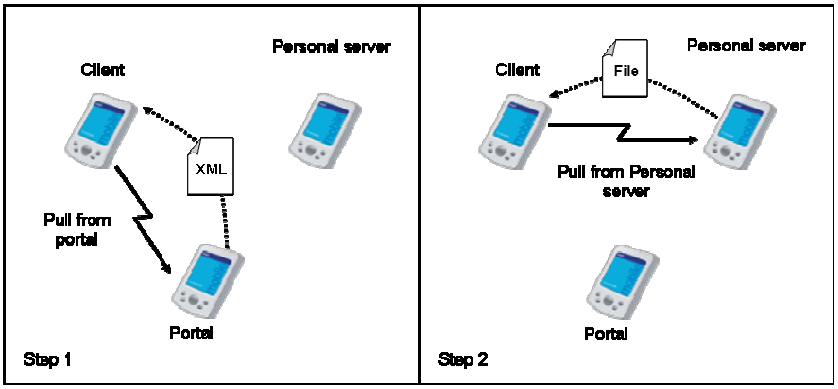

Figure 13. The connection procedures step by step in the Passive Scenario of TwiddleNet.

THIS PAGE INTENTIONALLY LEFT BLANK

III. TWIDDLENET ARCHITECTURE

This chapter will introduce the TwiddleNet system and describe its portal in detail. In so doing, it will focus on the portal's concept, architecture, and software development, as well as on special issues such as database operation and web server selection.

A. CONCEPT AND GENERAL ARCHITECTURE

TwiddleNet is a mobile social networking model comprising mobile personal servers that enable real time sharing, exploiting the content capture capability of mobile devices particularly those of smart phones. TwiddleNet treats smart phones as personal servers and by applying the first file sharing model technique, provides a centralized portal that connects and exploits these powerful and multipurpose mobile devices. These devices can work either as personal content servers sending or sharing files, or as content requesters receiving these files. In the case of content servers, the devices capture new content or share existing contents in their memory. Then they generate their corresponding tags and send them to the portal where they are stored in a database. Now the portal informs the rest of the devices which maybe in content requesting mode, that there is a new content available. On the other hand, in the second case of content requesters, the devices receive the file from the device that created or shared it through two different procedures. In the first procedure, they receive information about a new available content in the portal immediately through an alert message. In the second procedure, they search the portal database to find a suitable existing content.

For a better understanding of TwiddleNet logic, we refer below to the basic modules of the device, whether it operates as a content server or as a content requester and of the portal. Figure 14 shows the basic modules of TwiddleNet

software application i.e. the XML Generator, the Sender, the Server and the Browser. The content generator belongs to the mobile device application and is not one of TwiddleNet features.

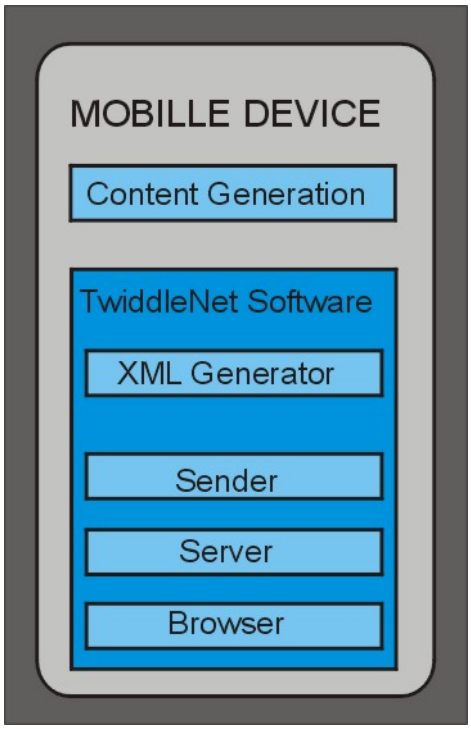

Figure 14. Basic Modules of TwiddleNet Client Mobile Device

Figure 15 illustrates the main portal features. A detailed description of each of these features will follow in the latter part of this chapter when the portal's architecture is presented.

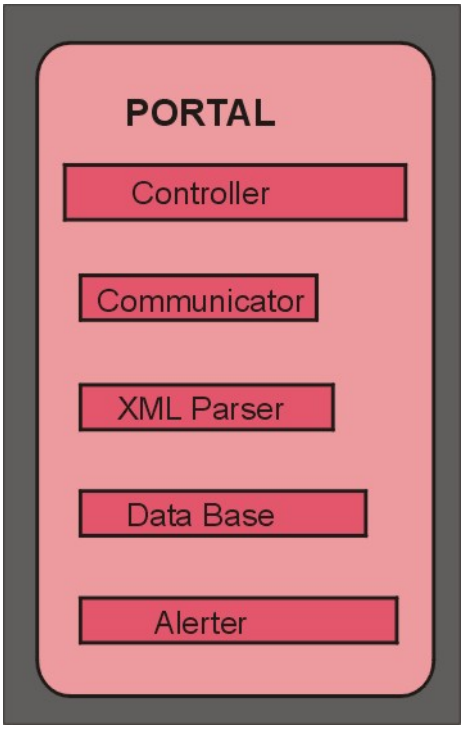

Figure 15. Basic Modules of TwiddleNet Portal

We will now describe in detail two different cases that indicate the exact procedure in the active and passive mode of TwiddleNet. Jon and Chris are two firefighters that participate in the extinction of a fire in a building. Suddenly Jon finds out that there is a fuel pipeline in the second floor and in order to give Chris the exact position, he decides to take a picture and send it through TwiddleNet. As Jon takes the photo with his device, which in this case operates as a content server, the Content Generator creates a JPEG file, as shown in Figure 16.

Figure 16. The Image generated from the smart phone

XML Generator then creates the corresponding XML file, as shown in Figure 17, and through the sender module, it is sent to the portal.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <feed xmlns:t="http://www.cs4137.com">
  - <action action="add">
    - <entry>
        <title when="predefined" how="userdefined"
          authority="mandatory" dataType="0">GAS LINE</title>
        <id when="predefined" how="userdefined"
          authority="mandatory" dataType="0">id</id>
        <updated when="onSend" how="automatic"
          selected="true" authority="mandatory"
          dataType="2">2008-02-04T16:54:03Z</updated>
      - <author>
          <name when="predefined" how="userdefined"
            authority="mandatory" dataType="0">JON</name>
          <email when="predefined" how="userdefined"
            authority="optional" dataType="1" />
          <url when="predefined" how="userdefined"
            authority="optional" dataType="4" />
        </author>
        <summary when="onGen" how="userdefined"
          authority="optional" dataType="0" />
        <t:created when="onGen" how="automatic"
          selected="true" authority="optional"
          dataType="2">2008-02-04T16:53:44Z</t:created>
        <t:fileUpdated when="onGen" how="automatic"
          selected="true" authority="optional"
          dataType="2">2008-02-04T16:53:48Z</t:fileUpdated>
```

```
        <t:extension when="onGen" how="automatic"
          selected="true" authority="optional"
          dataType="6">.jpg</t:extension>
        <t:missionNumber when="predefined" how="userdefined"
          authority="optional"
          dataType="0">928</t:missionNumber>
        <t:priority when="onGen" how="userdefined"
          authority="optional" dataType="6" />
        <t:kw when="onGen" how="userdefined"
          authority="optional" dataType="0" />
        <t:idcode when="onSend" how="automatic"
          selected="true" authority="mandatory"
          dataType="6">00-09-2D-FD-BA-10-00-00</t:idcode>
        <t:phone when="onSend" how="automatic"
          selected="false" authority="optional" dataType="7" />
        <t:title when="onGen" how="automatic" selected="true"
          authority="optional"
          dataType="0">jon0230.jpg</t:title>
        <t:length when="onGen" how="automatic" selected="true"
          authority="optional" dataType="0">4140</t:length>
        <t:ipaddress when="onSend" how="automatic"
          selected="true" authority="optional"
          dataType="0">10.2.0.40</t:ipaddress>
        <t:image when="onGen" how="automatic" selected="false"
          authority="optional" dataType="0" />
        <link when="onSend" how="automatic" selected="true"
          authority="mandatory" dataType="5" rel="enclosure"
          title="jon0230.jpg" length="4140"
          href="http://10.2.0.40/tnet/shared/jon0230.jpg" />
      </entry>
    </action>
  </feed>
```

Figure 17. XML File coresponing to the image in Figure 16

The Communicator module receives the XML file, extracts the data and builds up a “request” object that will be sent to the Controller. This request goes now to the XML Parser where it retrieves all content metadata and sends them back to the Controller. Then these metadata go to database where they are stored and a feedback of this action returns to Controller. The Controller now implements two different procedures. In the first, it sends the metadata to the

Alerter to create the corresponding alert message (Figure 18), which through the Controller will go to Chris' device, which in this case operates as content requester.

Figure 18. Alert message corresponding to the XML file in Figure 17

In the second procedure, the Controller creates and sends to the Content Server a confirmation message that the metadata of the image have been added in the portal's database, as we can see in Figure 19.

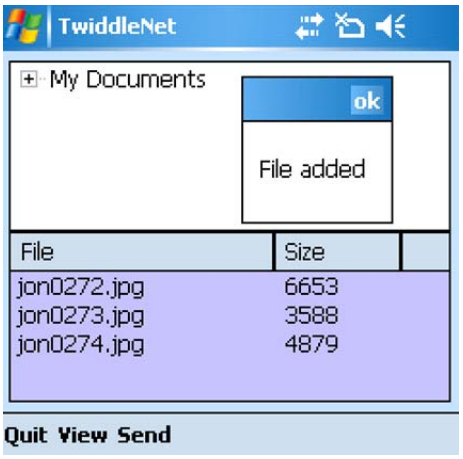

Figure 19. The confirmation message that the Portal sends to the Client

The Server receives the alert and shows this on the screen. Chris chooses to view the feed and automatically his device's browser will place an HTTP GET request to the URL in the alert, pulling the image from Jon's device through his server module. The full, step-by-step procedure for the active scenario is illustrated in Figure 20.

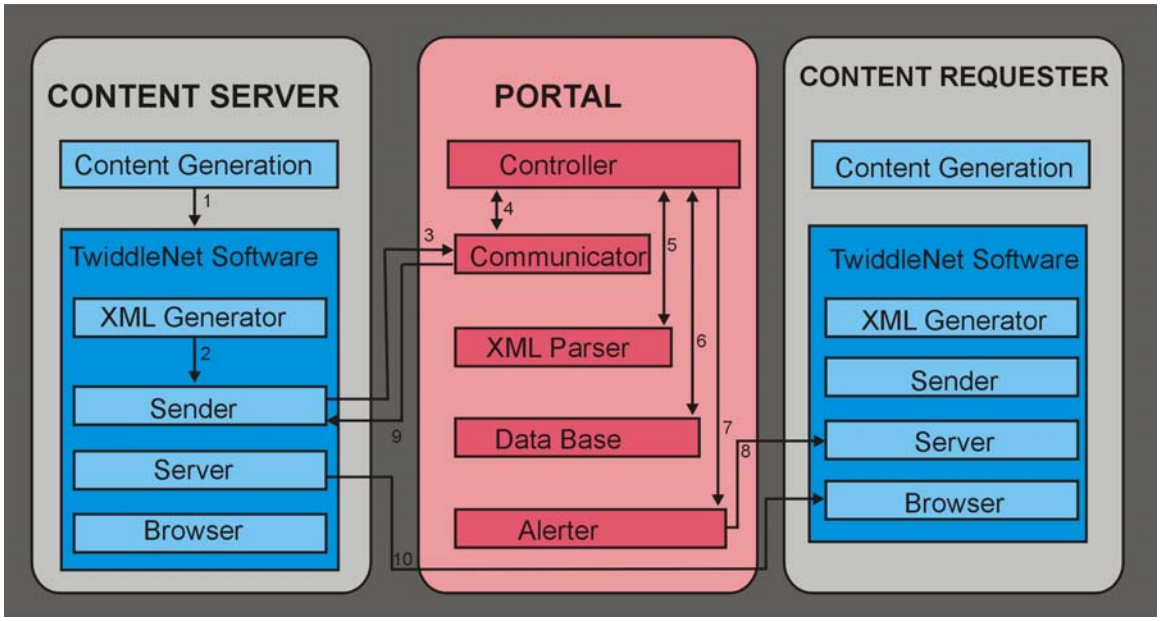

Figure 20. Active Scenario: The total procedure in detail

Several hours later and after a successful operation Jon and Chris are discussing the day's difficult operation. Chris applies now the passive mode of TwiddleNet trying to see all the photos that Jon sent to him during the operation. Chris initiates the process by browsing to the portal's URL. The URL provides a search function where the user can search via keywords such as title, author, date, and other options as shown in Figure 21.

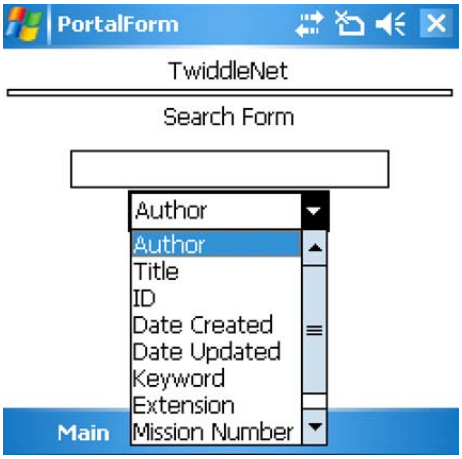

Figure 21. Keyword Search in TwiddleNet

Chris searches by author and sets the name Jon in the corresponding field. Now Chris' Content Requester will send through Sender module the request to the portal. Retrieving the database the portal will send to the server module of Chris Device the search results as a list of documents with their corresponding tagged information as shown in Figure 22.

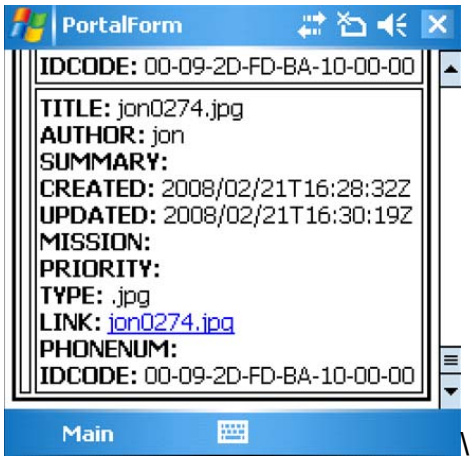

Figure 22. List of documents and their corresponding tag information

Chris can view any document by simply clicking the link provided. Then the Browser initiates the HTTP session by sending a GET request that contains the URL for the specified document. Through the server, Jon's device returns an

HTTP POST message including the document. For the retrieving of the remaining images, Chris repeats the same procedure. The full, step-by-step procedure for the passive scenario is illustrated in Figure 23.

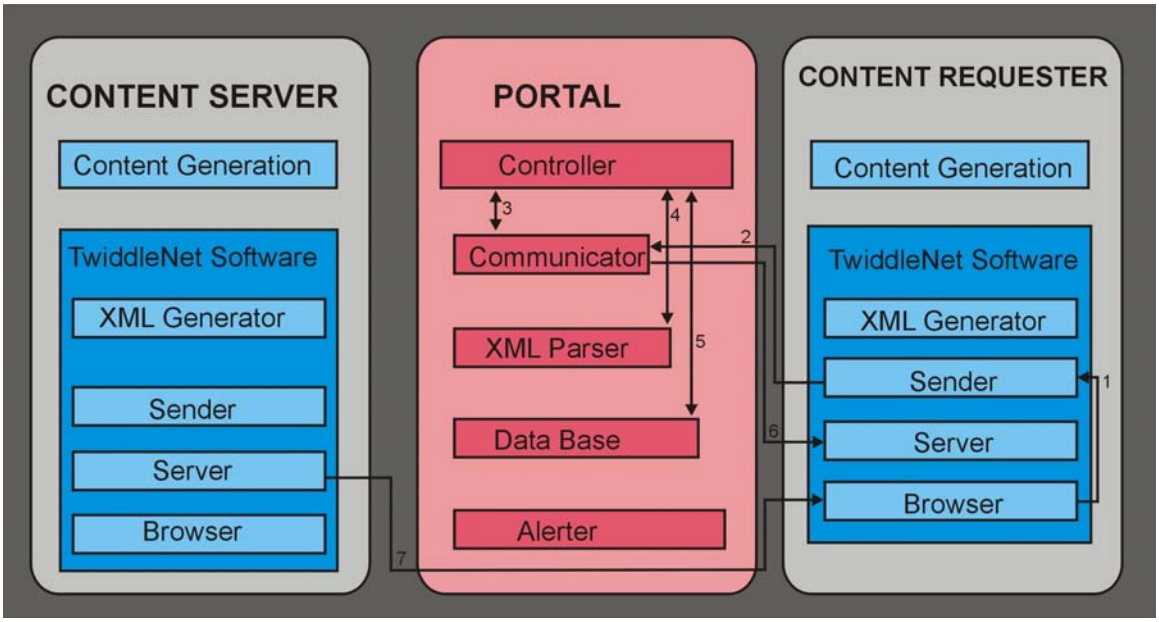

Figure 23. Passive Scenario: The total procedure in detail

B. TWIDDLENET PORTAL

The portal is the center point of the TwiddleNet architecture composing the gateway to this relatively small social networking model. In this section, we shall deal with the concept and the architecture of the portal. We will also examine the design and the software development, which is intended to run on numerous hardware platforms ranging from a high-end server and desktop or laptop to PDAs and smart phones.

1. Concept

The portal within the TwiddleNet architecture has several tasks and responsibilities in order to compose a robust and operational social network for humanitarian assistance and disaster relief.

The principal task of the portal is to connect the mobile personal servers, which compose the members of TwiddleNet. Every time a member wants to share or create a new image, he or she uploads this image onto the portal. The portal then informs the other members about the availability of this new entry, sending alert messages with the main characteristics of this entry, such as the type of the file, sender time and date. Moreover, a member has the ability to perform a search on all available contents from all registered TwiddleNet members, and then to download the specific file from its creator by instantly obtaining the corresponding Uniform Recourse Locator (URL) from the portal.

The portal also plays the role of a central repository for the feeds or the so-called metadata generated by TwiddleNet members. Using a database, the portal populates and stores the feeds of the corresponding files that members share or create. In this data base, for every feed there a twenty three different detailed description tags, i.e. author, link-title, id-code, title, id, summary, extension, mission-number, priority, link-href, link-length, updated-year, updated-month, updated-day, updated-time, updated-timezone, created-year, created-month, created-day, created-time, created-timezone, phone, and content.

2. Architecture and Design

Based on the concept stated earlier and the implementation details, the TwiddleNet Portal was broken down into several components. This should allow programmers to easily write new versions of desired components and simply plug those components into the overall system with no overall system recompiling. This should significantly decrease development time as well as improve portability of the TwiddleNet project as a whole. Figure 24 schematically shows the components and their relative positions in the TwiddleNet Portal architecture.

.

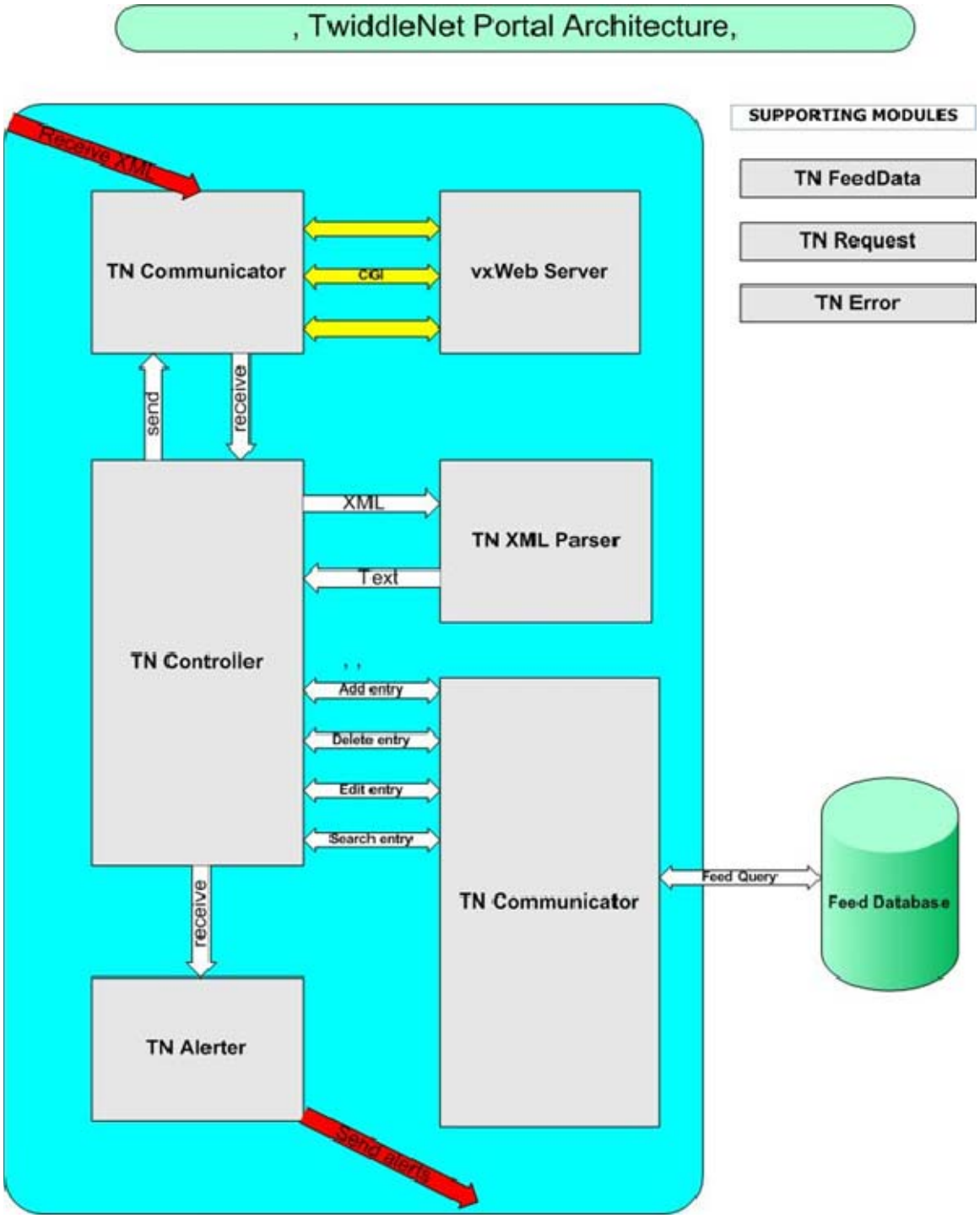

Figure 24. TwiddleNet Portal Architecture

In this architecture, five basic modules compose the portal's operation. Of course, there are various secondary and auxiliary modules, which for simplicity reasons are not illustrated, in the above figure. In what follows we will describe the basic operation of each of the six modules.

The Controller is the logic module of the program and as the name implies, constitutes the module that controls and coordinates the principal features. Depending on the requests received from the TN Communicator, the TN Controller will perform either an active or a passive procedure. In the former case with the cooperation of the Parser and Connector modules, Controller will perform registration of a new feed and alert messages creation through the Alerter module. In the latter case, again with the cooperation of the Communicator, Parser and Connector, Controller will perform a search update content data.

The Communicator module is responsible for providing the interface for the Main Program to interact with the network. It extracts the data and builds up a request object that the Main Program can then use and retrieve the desired client information. It must be mentioned that current working implementation is running as CGI program through a suitable web server. The TN Communicator is the only module that must be changed or modified in order to implement the portal as a standalone service.

The XML Parser module handles the parsing of the XML file, which it receives from the Controller. Its basic operation is to extract all the feed entry data from this file, to store them to text document as content meta-data and finally to send this document back to the Controller. If one wants to change the way in which the content meta-data feeds are built or if one desires a more efficient way of handling the XML file, the XML Parser is the proper module for this.

The Alerter module is solely responsible to send alert messages to all the members and so on, every time there is a new file for sharing. Due to the fact,

that, the alert message may be sent through any possible network interface which may be size-constrained, such as SMS, the message will contain a short, simple message stating that new content is available and the IP address where that content could be reached. An example of this alert message can be illustrated in Figure 18.

The Database Connector module is basically responsible for storing and retrieving users' content metadata to and from the database. The database can be implemented as a simple file or through a full-blown enterprise level database. The Database Connector also handles the search queries that users submit to the portal in order to browse or search the database for a specific file. Any desired changes or modification to the search capabilities or data storage can be accomplished in this module with no changes required on other TwiddleNet Portal modules.

3. Software Development

Software implementation for the whole TwiddleNet system is based on the Windows Mobile Operating System. This choice is based basically on the available hardware devices in the wireless lab while readily available software to help implement this and the remaining parts of the system run on Windows Mobile 5.0. The second key factor was based on overall Windows PDA market share as of the first quarter of 2007. For this reason, the development of the Portal was based on the C# .Net language. This fact allowed for ease application transfer from a desktop/laptop to a small mobile device which could be accomplished through a simply device sync.

The TwiddleNet Portal makes use of a simplified web interface for client interaction, such as uploading content meta-data and performing content search. This allows making use of pre-existing web applications to handle the network communications and easily call the TwiddleNet Portal application through the use the Common Gateway Interface or CGI standard. The current implementation of the TwiddleNet Portal is running as a Common Gate

Interface CGI program through a web server. In our case the Portal runs in the smart phone, the web server is the vxWeb server by Cambridge Computer Corporation. Future implementations of TwiddleNet could be run as a standalone service. The only TwiddleNet Portal component that would be required to be changed or modified would be the Communicator Module. This would provide a clean area for developers to implement their own protocol, if desired, while maintaining portability and ease of maintenance.

C. DATABASE OPERATION

One of the most basic operations of the TwiddleNet portal is to store and collect the XML files, which the members send every time they want to share a file. Furthermore, the TwiddleNet portal stores the users with their related personal characteristics. For the implementation of this task, there is one database into the portal architecture. It is ideal to give a definition of the database before we examine the type or other characteristic of that database.

A database is a structure that contains a collection of information that is organized in a manner to be accessed, managed and updated. For every database, there is structural description, which is related to the type of facts or data held in it. This structure is known as a schema and describes the objects that are represented in the database, and the relationships among them. There is a plethora of different database models. The most common model is the relational model. The relational database is a tabular database in which information is reorganized and accessed in a number of different ways. The relational model of data permits the database to obtain a consistent, logical representation of information. The relation is the main construct that represents data in this relational model. Every relation consists of a relation schema and a relation instance. The schema specifies the name and the domain of each field or column, while the instance is a set of rows or records.

In a relational database design, a unique or primary key is a candidate key to uniquely identify each row in a table. A unique or primary key

comprises a single column or set of columns. A unique key must uniquely identify all possible rows that exist in a table and not only the currently existing rows. A primary key is a special case of unique keys. The major difference is that for unique keys the implicit NOT NULL constraint is not automatically enforced, while for primary keys it is. Thus, the values in unique key columns may or may not be NULL. Another difference is that primary keys must be defined using another syntax.

In our portal database, we have one table that keeps the track of the feeds. This table is consisted of twenty-three different columns, as is illustrated in the table 1. Its unique key comprises a set of four different columns. These columns are the author, the link-title, the id-code, and the phone. In order to describe in detail the database schema, we present a filled table and the corresponding with this XML file.

```
              <entry>
///         <TNet:action>add</TNet:action>
///         <title>one.txt</title>
///         <id>http://www.TwiddleNet.com</id>
///         <updated>2007-03-05T13:23:00Z</updated>
///         <author>
///             <name>Alpha</name>
///         </author>
///         <summary>one</summary>
///         <TNet:created>2006-12-03T19:05:51Z</TNet:created>
///         <TNet:extension>.txt</TNet:extension>
///         <TNet:missionNumber>12345</TNet:missionNumber>
///         <TNet:priority>1</TNet:priority>
///         <link rel="enclosure" title="one.txt" length="27"
///         href="http://192.168.55.101/TNet_Shared/one.txt" />
///     </entry>
      /// </remarks>
```

Figure 25. XML File coresponing to the database as it is illustrated in Table 1.

A/A	FEED ENTRIES	METADATA
1.	Entry_title	One.txt
2.	Entry_id	http://www.TwiddleNet.com
3.	Entry_author	Alpha
4.	Entry_summary	1
5.	Entry_extension	txt
6.	Entry_missionnumber	12345
7.	Entry_priority	1
8.	Entry_link_title	One.txt
9.	Entry_link_href	http://192.168.55.101/TNet_Shared/one.txt
10.	Entry_link_length	27
11.	Entry_updated_year	2007
12.	Entry_update_month	3
13.	Entry_update_day	5
14.	Entry_update_tme	13:23:00
15.	Entry_update_timezone	Z
16.	Entry_created_year	2006
17.	Entry_created_month	12
18.	Entry_created_day	3
19.	Entry_created_tme	19:05:51
20.	Entry_created_timezone	Z
21.	Entry_phone	
22.	Entry_idcode	
23.	Entry_content	

Table 1. The 23 columns of the database table for the corresponding XML file in Figure 25

D. WEB SERVER SELECTION

In contrast to Client's device that uses its personal server, the portal uses a suitable web Server for mobile devices. A web server is a computer program that is designed to accept HTTP requests, which are known as web browsers and to serve them HTTP responses along with optional data contents. These contents are usually web pages such as HTML or XML documents and linked objects such as text, images, videos, etc. As was mentioned earlier, the TwiddleNet Portal makes use of a simplified web interface for client interaction, i.e. uploading content meta-data and performing content searches. This allowed us to make use of pre-existing web applications to handle the network communications and easily call the TwiddleNet Portal application through the use the Common Gateway Interface or CGI standard.

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers. A CGI program is any program designed to accept and return data that conforms to the CGI specification. CGI programs are the most common way for Web servers to interact dynamically with users. Many HTML pages that contain forms, for example, use a CGI program to process the form's data once it is submitted. Furthermore, the use of CGI is a server-side solution because the processing occurs on the Web server.

While there is a plethora of available web servers for laptops or desktops, very few are suitable for mobile devices. After research, we decided that the best web server that supports PDAs or smart phones running Windows Pocket PC 2003 or Windows Mobile 5.0 is the vxWeb server by Cambridge Computer Corporation. VxWeb is a complete multi-threaded web server for Windows CE-based devices, and provides standards-based web server functionality and is compliant with RFCs 2068 and 2069. It is compliant with HTTP/0.9, HTTP/1.0 and HTTP/1.1, completely configurable and compatible with all web browsers. Furthermore, it provides high performance operation and complete logging and tracing capability. VxWeb operates on all Handheld Pro HPCs devices and

Pocket PCs. As almost all the web servers provides Common Gateway Interface support, that constitutes the greatest factor for the portal since the whole executable TwiddleNet code runs on it.

While there are many advantages with CGI programming such as, that it is an ultimate platform technology, the language independency and its very simple interface, there are many disadvantages. The most important disadvantage of CGI programming is that every time it is requested, the script has to be evaluated and then to be executed. As we will see in the next chapter during the experimentation and operation of TwiddleNet in a real scenario, this repeated procedure creates significant delays in Portal operation.

IV. IMPLEMENTATION / EXPERIMENTATION

As already mentioned the main purpose of this thesis is the development of a TwiddleNet portal running on a smartphone or a PDA, which can allow the entire TwiddleNet system to run on handheld devices for rapid deployment in emergencies. This chapter deals with the technical characteristics of a mobile portal, the experimentation and operation in real exercise as well as the restrictions and vulnerabilities of its operation.

A. PORTAL TECHNICAL CHARACTERISTICS

A mobile device that can implement the TwiddleNet portal must be capable of the following:

1. Connecting wirelessly to the network
2. Running C# programs, and
3. Hosting a web server

A device that supports all of the stated features is the mobile messenger iPAQ hw 6945 of Hewlett-Packard Development Company, as it is shown in Figure 26.

Figure 26. The Mobile Messenger (Hewlett-Packard iPAQ hw 6945)

This mobile messenger is a smartphone with many capabilities and ideal characteristics for our purpose. Its lightweight of only 6.2 oz and its dimensions of 2.8 x 0.7 x 4.6 inches make for a very good tool for TwiddleNet missions. Its Lithium-ion battery with 1200 mAh energy gives approximately three hours in the case of continuous operation. Moreover, the fact that the battery is removable and rechargeable gives to the user an option to carry some extra batteries that they will continue the action in excess of three hours. Although in the current stage we use Wi-Fi for the operation of the TwiddleNet, the use of Bluetooth or cellular networks would also be useful as a main or auxiliary network type. The above device has Bluetooth technology, Wi-Fi with IEEE 802.11b and GPRS, EDGE and GSM quad band support.

Its processor, Intel XScale PXA270-416MHz is capable enough to support the operation of the device as a portal. The operating system, Microsoft Windows Mobile V 5.0, is the most appropriate for portal software written in C#, one of the two languages that .NET Compact Framework environment supports. Its ROM with 128 MB and its total user available memory with 64 MB RAM and 45 MB Flash memory are adequate for the current implementation.

The final goal is the capability to support one reliable web server, which is compatible with these mobile devices. Using the vxWeb server of Cambridge Computer Corporation, there is an opportunity to use its CGI bin and to convert in this way the above smartphone to a minimal and flexible TwiddleNet Portal. The following table details the main basic characteristics of the device.

GENERAL SPECIFICATIONS	
Manufacturer	Hewlett-Packard
Product Description	iPAQ hw 6945 Mobile Messenger
Product Type	Smart phone
Operating System	Microsoft Windows Mobile 5.0 Phone Edition
Processor	Intel XScale PXA270 416 MHz
Band	GSM 850/900/1800/1900
Wireless Connectivity	IrDA, Bluetooth, IEEE 802.11b
Battery	Lithium ion
Approximate Dimensions (in)	2.8 x 0.7 x 4.6
Weight (oz)	6.2
ROM	128 MB
RAM	64 MB
Total User Available Memory	109 MB
User Available Memory	RAM: 64 MB / Flash: 45 MB

Table 2. The General specifications of HP iPAQ hw 6945

In order to convert the above device to the mobile portal we need to copy the database to the device and to install the database server and the server's management tools. Furthermore, we need to create and put the portal's search page in HTML form into the vxWeb Sever. Having inserted the executable TwiddleNet portal code in the CGI bin of the server, as shown in Figure 27, we are ready to operate the TwiddleNet.

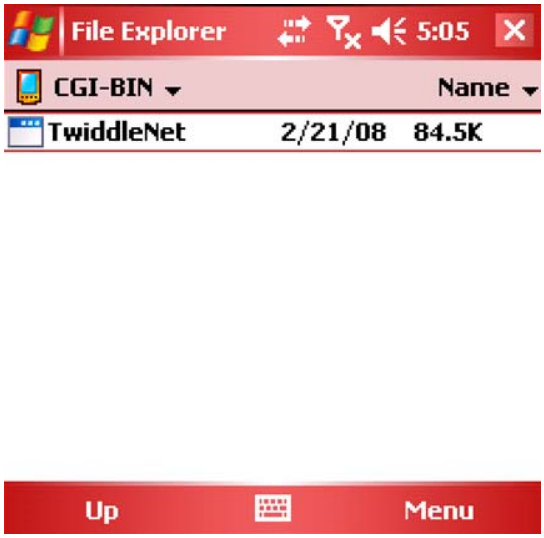

Figure 27. Hewlett-Packard iPAQ hw 6945

B. EXPERIMENTATION IN THE WIRELESS LAB

For the implementation of this experiment, we use four clients and our lightweight portal. The names of the clients are member1, member2, member3 and member4. For technical reasons and availability issues, the portal and clients run into the same type of device (hw 6945). The experiment takes place in the wireless lab and for the system's operation; we use the local wireless Wi-Fi network "TwiddleNet2." The specific network uses the IEEE 802.11b with operational frequency 2.4 GHz and maximum data rate 11Mbits/sec. Its range fluctuates between 38 and 140 meters. The set–up procedure includes the detection of the IP address for every device and the registration to the code. We execute the program and we put the executable file in the CGI bin of the web server that is running in the portal device. Finally, we insert in the portal files, a new empty database and we are ready to start our experimentation.

The screen that every client can see before the transmission of any file is illustrated in the Figure 28. As we can notice, the lower blue field is empty.

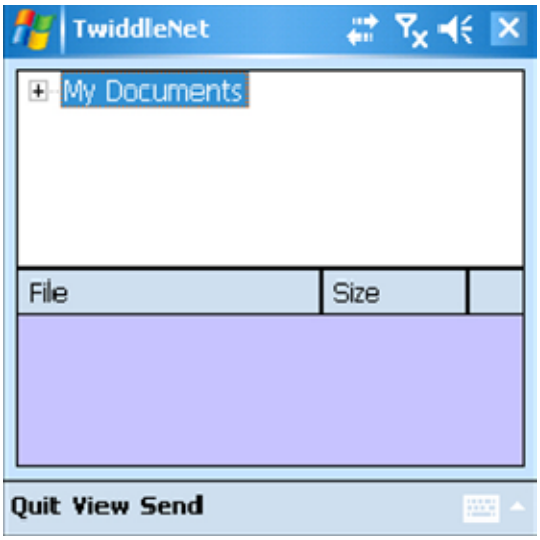

Figure 28. Client's screen before any transmission

In contrast with the Apache web server in the laptop, the vxWeb server has three different options for the control of the transmission. The first gives information to the user about the transfer rate the number of connections, and the received and transmitted bytes. The second shows the connections detailing the number socket and the IP address of the device that sent the data. Finally the third option gives information about the file requests. The initial screen of the portal before any transmission is made is shown in Figure 29

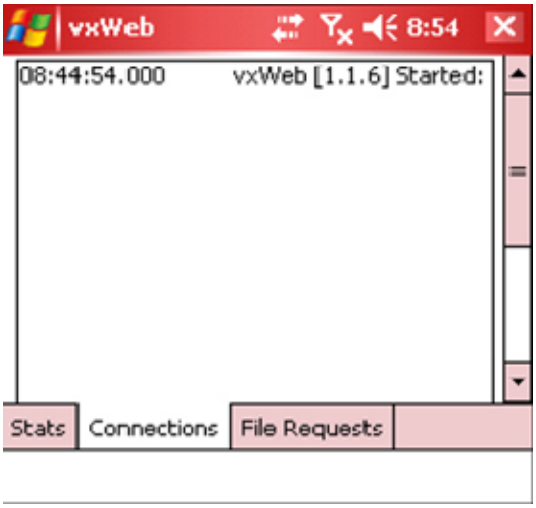

Figure 29. Portal's connections screen before any transmission

Member 1 takes the device and captures a photograph. Instantly, based in the configuration we set, the portal receives the corresponding to this photograph XML file, sending alerts to the other clients, as shown in Figure 30.

Figure 30. Alert message from “member1” Client

If the user chooses to view the photograph, we establish a peer-to-peer connection between the content server (member 1) and the content requester (member 2, 3, or 4) in the specific case. If we assume that member 3 wants to see the photo, it can initiate one HTTP session by sending a GET request to member 1. The GET request contains the URL for the specified photograph. Member 1 returns an HTTP post message including the photograph. In the case we do not want to view the photograph we choose No. As we can see now in the member screen, there is the photograph file, that member 1 sent to the network is there and has been stored in the database (Figure 31).

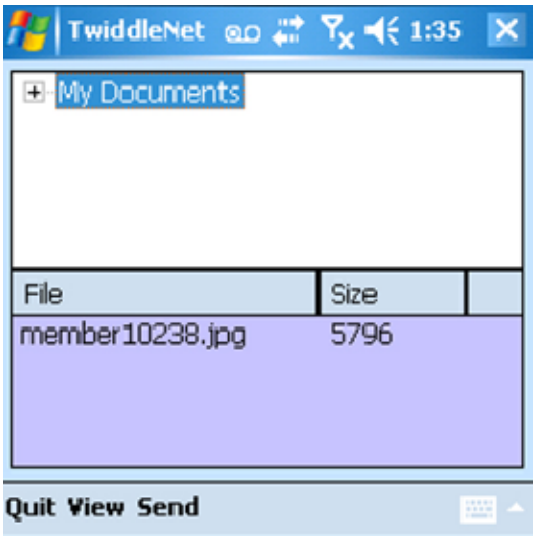

Figure 31. The Client's screen with the information that the XML file of the corresponding file "member10238" is already stored into portal's database.

We repeat now the same procedure two times for every member until the XML files in the portal's database become eight. From the statistics form of the portal, we can notice the transfer rate, the number of connections, the received bytes from the XML files and the sent bytes from the string alerts, as illustrated in Figure 32.

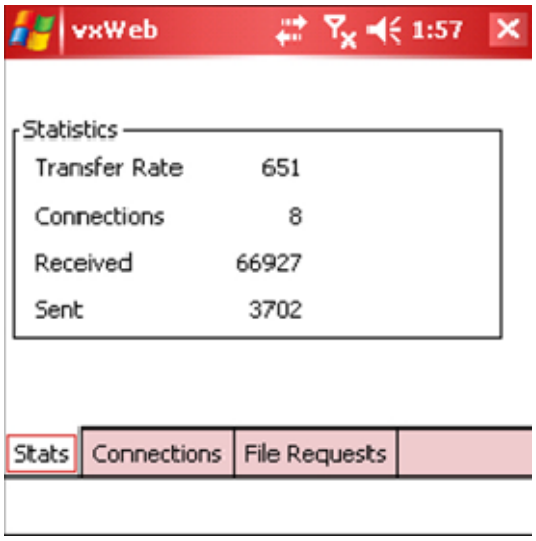

Figure 32. Portal's statistics table

Given that this connections table is very interesting but divided into two parts due to the small screen of a smart phone, we will present them in the following table 3. Additionally, we will insert the necessary transmission time from the time that the member sent the photograph until the alerts arrive at all members.

Accepted time	Accepted socket	Sending device IP address	Transmission time (sec)
12:56:40	5	10.2.0.38	14.1
13:42:34	9	10.2.0.39	17.6
13:43:33	13	10.2.0.40	18.2
13:44:23	17	10.2.0.60	18.6
13:45:09	21	10.2.0.38	15.1
13:45:59	25	10.2.0.39	14.1
13:46:54	29	10.2.0.40	17.0
13:47:34	33	10.2.0.60	18.9

Table 3. The basic features in the portal's connections table

The average time of the above eight transmissions is 16.7 seconds. In case, for instance, member 4 wants to see a file that has already declined, there is the option of the passive procedure of the TwiddleNet. According to this procedure, member 4 can make a search into the portal files by author, title, id, created date, updated date, keyword, extension, mission number or priority. Member 4 initiates this process by browsing to the portal's URL and then decides to see the files of "member 2."

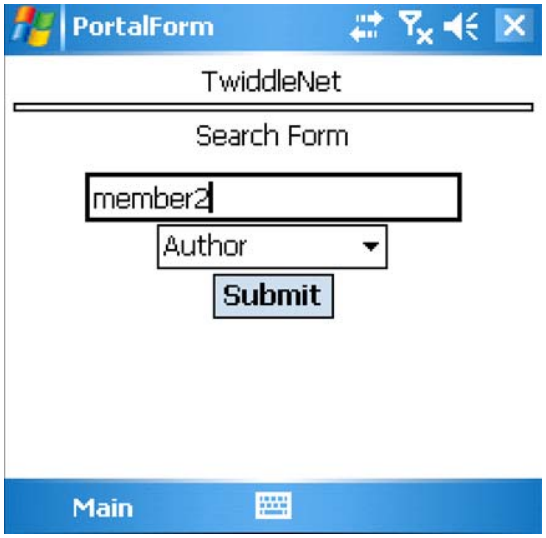

Figure 33. Portal's URL home page

The portal will then provide member 4 with a list of the requested documents and their corresponding tagged information. Part of this list we can see in Figure 34.

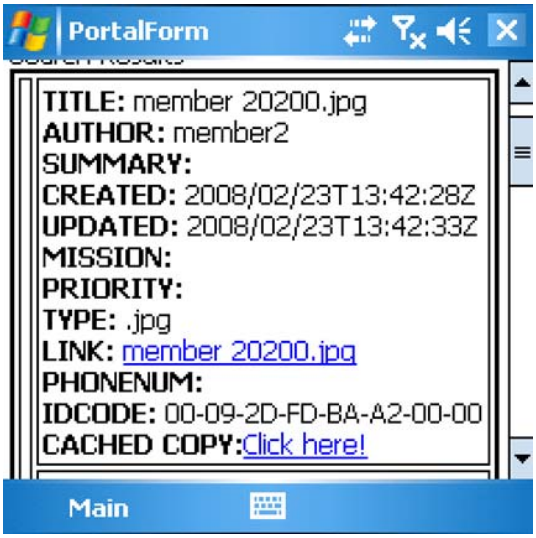

Figure 34. Part of the list that portal sends to the interested Client

Now member 4 can choose the preferable file by clicking the provided link. Like the active procedure, this establishes a peer-to-peer connection between member 2 and member 4. Member 2 initiates one HTTP session by sending a

GET request to member 2. The GET request contains the URL for the specified photograph. Member 2 returns an HTTP post message including the photograph.

The last procedure that a member can try is to delete a file from the database. Let us say that member 3 wants to delete the file, member 40045.jpg from its file. From the time that a member wants to delete a file, he or she must delete the corresponding file from the portal's database. With tap and stay to the specific file, a menu opened and member 3 can choose the delete option, as shown in Figure 35.

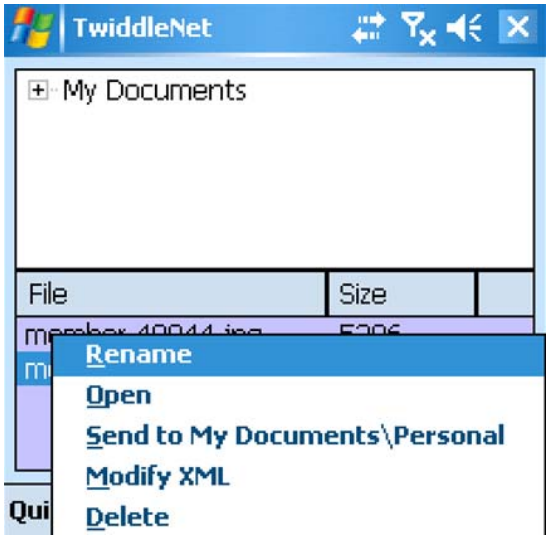

Figure 35. The Users option to handle a file

By clicking the delete option, member 3 sends the message to the portal, which finds and deletes the corresponding to jpg, XML file. The portal then sends to member 3 a deleted message for confirmation, as shown in Figure 36.

Figure 36. Confirmation for the file's deletion from the database

In order to compare the performance of the portal running in the smartphone with the corresponding of the portal in the laptop we repeated the same three procedures with the same number of clients through sending images. We must at this point indicate that the laptop that we used for our experiment is a Dell Latitude 110L with an operating system of Microsoft Windows XP version 2002 Service Pack 2 and processor Intel Celeron 1.30 GHz at 760 MB.

In the first active procedure, we sent the same number of files, with an average time of 6.9 sec. If we compare this time with the corresponding in the first case, which it was16.7 seconds, we find out that it is almost the half. In the second passive procedure, the time was only 1 second both for the search and deletion cases. The corresponding time with the portal in the smartphone for the search case was average 17 seconds, while for the deletion case was average 12 seconds.

C. RESTRICTIONS AND VULNERABILITIES IN A REAL EXERCISE

In order to find out the real capabilities of TwiddleNet we decided to test our networking system during exercise FEX II that took place at Camp Roberts in California on Jan. 16. The above exercise was planned in the field of

Cooperative Operations and Applied Science & Technology Studies (COASTS) which is a U.S.- international field experimentation program that tests advanced commercial-off-the-shelf [COTS] command and control, communications, computers and intelligence [C4I] systems of systems to provide real-time shared situational awareness for multi-national, tactical and remote decision makers. The program's goal is to integrate COTS technologies in innovative ways to generate advantages for search and rescue operations, peacekeepers and disaster response personnel.

During this exercise, both portal and clients ran into the same device hw 6945. We used one of the base local wireless Wi-Fi networks, "Vivato NPS." The specific network uses the IEEE 802.11 b with operational frequency 2.4 GHz and maximum data rate 11Mbits/sec. After the set-up of the devices, we trained four officers to manipulate the devices through the scenario. According this particular scenario, they had to run in the area that an earthquake occurred and to take photographs of the disaster.

In the lab, we had evaluated that the time, needed for the whole of the four devices to send and receive was approximately one (1) minute. We thus requested every officer to take an image almost every minute, so that the portal can finish with the first transmission cycle and then start with another. During this scenario, we discovered the following operational and design problems.

1. Operational Issues
    a. The first malfunction was that every time one member wanted to see an image the image was downloaded but after that, the whole system was frozen. The procedure to setup again the device into operation was complicated and time-consuming for one real scenario like this.
    b. Even in the case that everyone had skipped the images, there were cases where one XML file could not be sent. Every time that this malfunction occurred, the member that was carrying the portal had to handle this error in order for the transmission to be continued.
    c. Another important restriction was that every member should wait until the end of the previous cycle before it could

continue with the next one. If we apply multithreading for every sharing procedure, then everyone will start immediately without waiting the previous one to be completed.

2. Design Issues

   a. The fact that the portal is a CGI program means that every time the client sends form data to the server, a new CGI process gets started. This is the reason for the time delay we had every time we sent or shared a file, when we searched in the portal's database, or when we wanted to find and delete or modify a file. To eliminate and avoid this problem, the best solution is to design and implement a standalone server that runs continuously in the background.

   b. Every time that a member was disconnected or his device accidently was turned off the portal was unable to receive the XML file to store this to the database and to send the alerts. To eliminate this problem, we must modify the code and create multithreads for the alerts.

At the end of the mission, we gave to these officers a questionnaire with questions about the operational and practical experience of TwiddleNet. From these input we received valuable assistance for the avoidance of the malfunctions and the general improvement of our system. The most important proposals and thoughts are the following:

1. While the stylus is not a problem for the manipulation of the device especially for the people that have experience with smartphones and PDAs, perhaps it is not ideal in the case of humanitarian missions. The specific missions take place under very difficult and extreme conditions. Furthermore, the heavy equipment of these crews or their special clothing in the case of firefighters reduces the easy handling of the device.
2. The fact that the user must tap with the stylus the confirmation message for the addition of the XML file in the database is not so convenient and practical. Perhaps a warning message is more proper in the case that the file, cannot not be stored from a portal database.
3. During a real situation, the users do not always need to take alerts for every image or information. One proposed solution is at first the

image evaluation, from the user, to categories. At second, the implementation of an optional choice, for the alerts' transmission only in the case of emergency. The TwiddleNet Mobile Command Post, a design that it is already in progress, is a terminal program that will receive the rest of the alerts for evaluation and storage.

4. Finally, a standard procedure for the allowing access to the system of a user is necessary. The first reason is that every user could login safely to the system with a user name and a password. The second reason is that we can so eliminate the complicated and time-consuming procedure to set up the devices before the operation.

The participation in this exercise was a valuable experience for the elimination of the problems and the improvement of TwiddleNet. While many functions have been implemented, some must be modified and many other must be designed and implemented for a robust and flexible networking system, which will be tailor made to a real humanitarian mission.

V. SUMMARY AND CONCLUSIONS

TwiddleNet is a mobile social networking model made of mobile personal servers that enable real time sharing and exploit the content capture capability of contemporary mobile devices. By applying the first file sharing model technique, TwiddleNet connects and exploits these powerful and multipurpose mobile devices while using a centralized portal.

In this thesis, we examined the concept, the types and the operations of personal servers. We also dealt with aspects of seamless mobility and the capabilities, which such mobility will provide to ordinary mobile devices. We then presented the mobile social networking system. Increasing computer power and the miniaturization of mobile devices make such networks quite mobile, leading to a revolution in contemporary communication and data services. We then briefly presented the four different generations of media sharing systems giving a detailed description of peer-to-peer architecture. In implementing the portal on the smartphone, we decided that the first generation was ideal for file sharing due to restricted storage capabilities of the device. This architecture allows the portal to store only the metadata, while the original files are distributed in the members' devices.

The vxWeb server by Cambridge Computer Corporation was the multi-threaded web server, which we used for the implementation of the portal. Having in mind the ultimate platform technology, the language independency and the choice for one very simple interface, we decided to design a Common Gateway Interface (CGI) program. The code was divided into five different basic modules allowing programmers to easily write new versions of desired components and simply plug those components into the overall system with no overall system recompiling. This should significantly decrease development time as well as improve portability of the TwiddleNet project as a whole. For the implementation

of the storage in the portal, we chose a relational database model that permits a consistent and logical representation of information.

The mobile messenger iPAQ hw 6945 from Hewlett-Packard Development Company was the best available device in the wireless lab that could satisfy our criteria for the implementation of the portal into a lightweight mobile device. The entire system can be run now on handheld devices for rapid deployment in emergencies. Two different possible scenarios were presented for a better understanding of the way that TwiddleNet should contribute to the release of a disaster, either in the humanitarian, or the military field. We tested the whole system in the wireless lab searching the system's operation in different procedures and comparing its attitude with the corresponding implementation of the portal in standard platforms. Finally, in undertaking a real scenario during exercise FEX II, we pointed out some flaws that influence its expected operation. With the valuable feedback from this exercise, we propose some ideas for future work.

A. FUTURE WORK

TwiddleNet started as an early prototype and at present, is a system that operates at a satisfactory level. Although many functions have already been implemented, others must be designed and added or modified to improve the overall performance and operation. In what follows, we present and discuss some thoughts and ideas regarding the modification and improvements of the existing functions.

1. The Portal as a Stand Alone Program

TwiddleNet portal is implemented as a Common Gateway Interface (CGI) that the web server provides. While CGI helps quick implementation, there are a few disadvantages of the CGI programming. The most important of those, which affects the overall performance of TwiddleNet, is the significant transmission delay, when the portal is running on the mobile device. The reason is that every

time the client sends form data to the server, a new CGI process gets started. During this process the script has to be evaluated and then to be executed. Furthermore, in the case of a mobile device where the processing power is much smaller than that in the lap or desktops, this delay is almost doubled.

The best solution to eliminate this problem is to design and implement a standalone server that runs continuously in the background. As it has already been stated, the TN Communicator is the only module that needs to be changed in order to achieve this objective. One obvious example inside the TwiddleNet architecture is the Server in the Client program, which is responsible for serving the original file every time the Content Requester wants so the transmission time in the specific case is less than one (1) second.

2. TwiddleNet Mobile Command Post

Although, one of the major tasks of the portal is to collect all the XML files of the content that clients share there is no device or database that collects the corresponding files. As already mentioned, the TwiddleNet Mobile Command Post (MCP) is a prototype design that it is already in progress, and substitutes a terminal program that will receive all of the alerts as well as the corresponding files, for evaluation and storage. The MCP operation will be similar to the operation of the Clients with the only difference that MCP will receive in addition the XML file so as, to extract the necessary information for all content. The procedure is shown in detail in Figure 37.

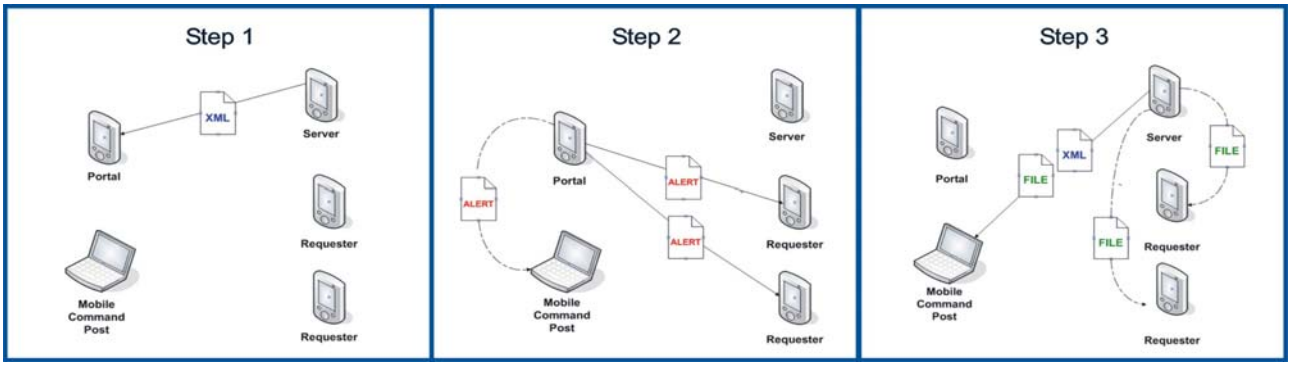

Figure 37. Mobile Command Post operational procedure

MCP using an appropriate web browser will have the capability to display images and files with different ways depending on the situation. Mobile Command Post, except the file collection and storage will create a great tool to decision makers for the best exploitation and elaboration of the contents.

3. Content Exploitation and Scalability

Currently, TwiddleNet portal sends an alert to each member every time new content is available. While this situation is ideal for the members' information, it is arguably not so good during a real life situation where the members of the team do not have the time to receive and notice all the information. As it is already arranged, the user could characterize with priority one (1) the emergency information that the user believes that every member must receive. Then, with a minor modification in the code, the portal could send alerts to the team members only in the case of such emergency messages. The portal will send alerts only to the MCP in the case of information with priority two (2) or less. In this way, with the MCP assistance, we have an ideal exploitation and scaling of information without time waste or contents loss.

4. Access and Registration Procedures

Although access to a social network like TwiddleNet is a fundamental and basic procedure, the current version lacks this function. Since the portal is the gateway for all these clients who want to share information, access into the system is a most important function.

The design and the implementation of an access control procedure for the team members of TwiddleNet is an essential future work. In this way the user will have the ability to log in or out as needed. In the case that the member uses the network for first time he or she will have the obligation to register with a username and a password. With such procedure every user will be independent without the need of a coordinator to set up the system, as is currently the case. Furthermore, the huge amount of time needed for the set up procedure will be eliminated.

5. Encryption and Security Procedures

Except for the basic encryption that the carrier network provides, there is no other security protection built in TwiddleNet. Due to the humanitarian character of TwiddleNet and the importance of the information that is provided security, measures must be introduced to reduce the risk of unauthorized access. Authentication, integrity and confidentiality are some of the areas that need to be addressed. Security must be applied not only to the portal and its database but also to personal servers. Personal Server protection is fundamental because files are transmitted between the members and without the basic encryption their content, is exposed to malicious activities or organizations.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Gurminder Singh, “Personal Mobile Server.” [http://www.nps.navy.mil/cs/singh/CMDC/PersonalServer/PersonalMobileServer.htm] Mar 08.

[2] Roy Want, “Intel ‘Personal Sever’ research: mobile computing in the palm of your hand” [http://linuxdevices.com/articles/AT5772921353.html] Mar 08.

[3] Roy Want, “Personal Media Server” [http://www.intel.com/research/exploratory/personal_server.htm] Mar 08.

[4] Agere Systems, “Connectivity Zone Products for the week of December 18, 2006” [http://www.analogzone.com/iop_1218.htm] Mar 08.

[5] John J. Barton, Shumin Zhai, and Steve B. Cousins B (2006). Mobile Phones Will Become The Primary Personal Computing Device. Proc. 7th IEEE Workshop on Mobile Computing Systems & Applications.

[6] IEEE 802.21, Overview of Standard for Media Independent Handover Services by Vivek Gupta, Michael G. Williams, D. J. Johnston, Stephen McCann, Phil Barber, Yoshihiro Ohba, San Diego July 18 2006 [http://www.3g4g.co.uk/Other/WiLan/802_21/802_21-IEEE-Tutorial.ppt] Mar 08.

[7] Markus Bylund, and Zary Segall (2004). “Towards seamless mobility with personal servers.” Info: The Journal of Policy, Regulation and Strategy for Telecommunications, 6(3): 172-179, 2004.

[8] Alvin Cheung, Tyrone Grandison, Christopher Johnson, Stefan Schonauer (2007) Infinity: A Generic Platform for Application Development and Information Sharing on Mobile Devices, Sixth International ACM Workshop on Data Engineering for Wireless and Mobile Access (MobiDE) 2007, Beijing, China, June 2007.

[9] Knowledge @ Wharton, “My Space, Face book, and Other Social Networking Sites: Hot Today Hot Tomorrow?” 3 May 2006 [http://knowledge.wharton.upenn.edu/article.cfm?articleid=1463R] Mar 08

[10] Technology Review, “A New Platform for Social Computing: Cell Phones,” 3 July 2006 [http://www.technologyreview.com/read_article.aspx?ch=specialsections&sc=telecom&id=17079] Mar 08.

[11] Vipera.com, “A Cell Phone Community” [http://www.killerstartups.com/Web20/vipera--A-Cell-Phone-Community/] Mar 08.

[12] Aaron Huslage, “Sonopia Reinvents the MVNO” [http://www.oreillynet.com/pub/a/etel/2007/04/04/an-introduction-to-sonopia.html] Mar 08.

[13] MacManus Richard “Nokia Acquires Media Sharing Startup Twango” [http://www.readwriteweb.com/archives/nokia_acquires_twango_media_sharing.php] Mar 08.

[14] Stefanos Androutselis-Theotokis, Diomidis Spinellis “A Survey of Peer-to-Peer Content Distribution Technologies’ [http://www.spinellis.gr/pubs/jrnl/2004-ACMCS-p2p/html/AS04.html] Mar 08.

[15] Wikipedia, “File Sharing” [http://en.wikipedia.org/wiki/File_sharing] Mar 08.

[16] Wikipedia, “Peer-to-Peer” [http://en.wikipedia.org/wiki/Peer-to-peer] Mar 08.

[17] Christopher T. Clotfelter, Jonathon E. Towle (2007) TwiddleNet: Metadata Tagging and Data Dissemination in Mobile Device Networks.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California