An Open Wireless Mesh Testbed Architecture with Data Collection and Software Distribution Platform[1]

Sachin Lal Shrestha, Jinsung Lee, Anseok Lee, Kyunghan Lee, Junhee Lee and Song Chong
School of Electrical Engineering and Computer Science
Korea Advanced Institute of Science and Technology,
Daejeon 305-701
Email: {sachinls,ljs,anseok,khan,junhee}@netsys.kaist.ac.kr, song@ee.kaist.ac.kr

***Abstract*—A Wireless Mesh Network (WMN) is a fast growing network, which is now a popular technology for providing wireless internet connection to industry as well as community. A WMN is a collection of nodes (usually a computer with one or more wireless Network Interface Cards (NICs)) that are connected to one another with single or multiple hop ad hoc links forming a mesh backbone network. Ad hoc links are popular in mesh connectivity as they are self-configuring and self-healing. In this paper, we discuss WMN design and deployment issues with reference to our WiSEMesh testbed. WiSEMesh has 56 nodes deployed in the campus area providing internet connection for over 1000 users. Each node consists of a small form factor computer with three wireless NICs. We developed the WiSEMesh node software stack that contains unix based operating system, wireless NIC drivers, tools such as DHCP server, NAT etc. WiSEMesh has properties like scalability, open source operating software, heterogeneous hardware support, high performance, globally reachable, and easy to configure network. Hence, it can spread faster as a community wireless network and also provide a flexible platform for testing communication network layers. We also address design issues of a Network Management System (NMS) that can be implemented in any wireless network. Our NMS named WiVi provides platform for software distribution and network configuration with an elaborate data collection and storage facilities. For a large and scalable network like WiSEMesh, a massive database needs to be designed. One of the WiVi design objectives is to provide a robust, flexible and scalable database. WiVi achieves database design goals by implementing a concept "network in database" where network is logically divided into hierarchy of hardware components and each component is treated as an object in the database. The "network in database" implementation guarantees data collection and storage flexibility. By implementing the NMS software distribution platform and the hardware independent WiSEMesh node software stack, our testbed is confirmed to achieve scalability.**

***Index Terms*—Wireless Mesh Network, Testbed, Management, Visualization.**

I. Introduction

A generic Wireless Mesh Network (WMN) is self-organized and self-configured, with the nodes in the network automatically establishing an ad hoc network and maintaining the mesh connectivity. A WMN node contains additional routing functions to support mesh networking. To increase the capacity of mesh networking, a node is usually equipped with multiple wireless interfaces built with either the same or different wireless access technologies. In spite of these differences, mesh and conventional wireless routers are usually built based on a similar hardware platform. In addition to mesh networking among nodes, the gateway/bridge functionalities in mesh routers enable the integration of WMNs with various other networks.

WMNs diversify the capabilities of ad-hoc networks with features such as low up-front cost, easy network maintenance, robustness, reliable service coverage, etc. Therefore, in addition to being widely accepted in the traditional application sectors of ad hoc networks, WMNs are undergoing rapid commercialization in many other application scenarios that incorporate last mile networks such as broadband home networking, enterprise networking, community networking.

Few testbeds are setup in research laboratories and companies [5]-[13] to enhance or re-invent existing protocols for WMNs. However, for a WMN to be all it can be, considerable research efforts are still needed. [1] lists WMN research issues such as physical layer stability, a scalable MAC, better performance metrics for routing protocol, adaptive TCP and network management. To develop and validate solutions to any WMN open issues in a testbed environment with ease and convenience, the testbed should possess the following characteristics:

1) The testbed should support different types of radio technology. Each node should have multiple radios to increase capacity.
2) The testbed should be hybrid network consisting of several heterogeneous networks. For example, a WMN is used as a backhaul network for several sensor networks. Hybrid network is a practical approach that is becoming popular in military and industrial applications.
3) Testbeds require open source codes in all communication protocol layers. A WMN testbed node should consist of freeware software components. Drivers and firmware used should be developed under open source code projects. The testbed should also be accessible or reachable for research purpose from anywhere in the world. If above two criteria are satisfied, then a truly "open WMN testbed" can exist.
4) The testbed should have a reliable management channel

[1]This work was supported by the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (MOST)(R01-2006-000-10753-0).

[1]This study has been supported in part by a grant (Next Generation PC Project) from the Institute of Information Technology Assessment (IITA).

provides means to configure network parameters.

Furthermore, a testbed should also have enough users to emulate real world traffic. A community based testbed can generate real user traffic pattern. Therefore, a testbed should also have the following characteristics to attract a larger user base.

1) A testbed should operate with heterogeneous hardware where nodes are built with already available and ubiquitous components such as PCs. A critical but not mandatory requirement of a node is that a wireless network card should support open source drivers/firmware for research purpose.
2) Usually, in campus, large business complex, and residential area, many unsecured wireless hotspots are freely available. So, a WMN testbed needs to compete with these hotspots. To attract and maintain a large user group, a WMN testbed should have a good quality of service.

Our testbed, Wireless Scalable and Efficient Mesh network (WiSEMesh) deployed at KAIST, has all of the above specified characteristics. It is deployed into two parts: one is in a department office building and another in 6 undergraduate dormitory buildings. These deployments are physically separated and belong to two different IP domains that is connected by a single wireless link; it can be visualized as two mesh island networks connected by a link. The office building testbed is used by many research labs inside and outside of Korea. The undergraduate dormitory testbed focuses on establishing a community WMN for more than 1000 residents. The pretested algorithms and software from the office building testbed are loaded into the community testbed for further tests. This ensures a smooth network functioning at community testbed with virtually no down time; the needed characteristics for user satisfaction.

A Scalable WMN like WiSEMesh grows faster and also poses a significant task of management. The wired network domain has a couple of management frameworks such as Simple Network Management Protocol (SNMP) [2]. The architecture of these frameworks mainly consists of master agent, subagent and management station. A typical wireless mesh testbed like WiSEMESH requires an array of functionality to guarantee a manageable, efficient and scalable network. Therefore, a Network Management System (NMS) for a wireless mesh testbed needs to incorporate features like network configuration, fault management, debugging ability, software distribution, data collection, and visualization. Generally, crucial issues in designing a typical NMS are data transfer methodology between nodes and management station, and network management functions for each node and the management server. [3] and [4] discuss information transfer mechanism of the NMS framework. To date, we have not found any literature on data handling and storage, and control features specific to a WMN NMS.

Considering the WMN nature and the problems faced by the researchers working on WiSEMesh, we developed an NMS named WiVi which is presently implemented in our testbed. The features of WiVi are listed below.

1) Dual mode operation: The management server can operate in central mode or distributed mode where more than one server, scattered throughout the testbed, manage subset of nodes. In either case, management station communicates to the nodes either via Ethernet infrastructure or using the testbed itself. Currently, WiVi is operating in central mode.
2) Network in database: WiVi uses relational database system to store network details such as hardware parameters and topology details. The database also stores parameters from each layered communication protocols such as SINR, Rogue APs, throughput, etc. As a result, WiVi is scalable, i.e. it can handle as many networks as possible regardless of the physical location of the networks as long as there is a connectivity between the management station and the network nodes.
3) WiVi uses Apache server to provide browser based interface to any privileged user. Therefore, a user can monitor, configure as well as manage the networks registered with WiVi anytime from anywhere.
4) A WMN testbed requires frequent software updates or upgrades. For a large testbed, the task of updating is cumbersome and prone to human errors when manually performed. WiVi provides a platform to distribute software to all the nodes with a click of a button. WiVi recognizes different hardware components of the nodes present in a network (or a testbed). Therefore, it is capable of componentwise node configuration. For example, it can configure any wireless Network Interface Cards (NICs) of individual node or group of nodes at a time.
5) WiVi consists of a visualization tool that shows actual location of nodes on a user provided map. It also shows the link throughput and number of users connected to the node. The visualization tool is flexible and customizable.

WiVi NMS is platform independent and therefore supports heterogeneous nodes. It can be implemented with any testbed or commercial WMNs.

Following section provides a brief summary of on going activities in WMN testbed research and management system. Section III provides details on WiSEMesh design. NMS is discussed in Section IV. Section V shows WiSEMesh performance using WiVi. Conclusions are drawn in Section VI.

II. Related Work

In this section, we discuss some testbeds and commercial WMN deployments and briefly discuss some of their unique features. In next section, we introduce the existing WMN management technologies.

A. Testbed

Roofnet [5] at MIT is a pioneer WMN testbed which is spread in a city block and the nodes are hosted by residents and business houses. Each Roofnet node consists of a PC, an

802.11b card and a roof mounted omnidirectional antenna. A user or a volunteer hosting the Roofnet node collects the node with software pre-installed. The software is a stack of tools and protocols namely, a DHCP server, an auto-configuration platform, a routing protocol, and a bit-rate selection MAC algorithm called SampleRate. The operating software in the testbed implementation is open source codes. In this context, our testbed, WiSEMesh is designed after Roofnet where all the network layers are open for changes thus giving high flexibility to researchers. WiSEMesh differs from Roofnet mainly in the testbed realization such as multiple radios per node, multiple management channels, and data collection and dissemination. Yet another WMN implementation based on Roofnet is Berlin Roof Net [6] which uses commercial off the shelf routers with integrated access-points; a feature that adversely effects a fast deployment. A community WMN can only spur and spread quickly when already ubiquitous computing devices such as PCs can be used; a desirable feature that WiSEMesh has. A WMN with commercial incentives is being developed by Microsoft [7]. This testbed consists of nodes with two radios and a 2.5 sublayer called mesh connectivity layer that determines the routing path by selecting the best radio on a node and the best path between two nodes. Even though the software is available in public domain, it is limited for the testing purpose only. A node in the WMN testbed MeshNet [8] at University of California, Santa Barbara, has two wireless devices (Linksys WRT54G) strapped together. One device is a mesh node and the other device is out-of-band management node. Again the deployment suffers from similar drawbacks of Berlin Roof Net. MAP [9] at Purdue University has similar deployment norms to Roofnet. BWN-Mesh [10] is deployed on personal computers and laptops equipped with 802.11b and 802.11g cards located in various rooms on the floor; the testbed consists of mobile mesh node. Carleton University WMN [11] uses a mesh node consisting of an Intel IXP425 series XScale network processor with two 802.11b/g radios. Even though such implementation equips a mesh node with high processing capability, the deployment cost will be expensive. Moreover, the processing capability required by a WMN node with minimum features like self-configuration and DHCP operation can be sustained by a PC. Hyacinth [12] is comparatively small testbed of 10 nodes solely deployed for experimental purposes. Each node has three radios; a similar feature to WiSEMesh. SMesh research [13] at John Hopkins University focuses on a seamless VoIP handoff mesh network; an application which can readily be implemented in WiSEMesh.

B. WMN Management Technology

Non of the leading WMN research groups mentioned in Section II-A have implemented a data collection and software distribution platform. MIT Roofnet and Berlin Roof Net have software update platform which can only work provided that the original software is already installed in the nodes. Berlin Roof Net uses infection based distribution method that uses TFTP protocol for file transfer. In an unreliable channel condition, the update can take several minutes to several hours. This can disrupt the service causing dissatisfaction among users. Therefore, the transfer needs to be fast as well as secure. TFTP is not a secure means of data transfer compared to Secure Copy (SCP) that the WIVI NMS adopts. Firetide [14], BelAir [15] and Mesh Dynamics [16] are total WMN solution providers; the NMS are tailor made for their mesh products and thus can't be implemented for any other network. WIVI NMS is hardware platform independent; it can be incorporated with any wireless network. The NMS developed by the above mentioned vendors can accumulate already defined network parameters whereas WIVI NMS can be customized to collect any user defined network parameters from any node. Furthermore, WIVI NMS captures per interface per node parameters unlike the NMS tools mentioned before. Map views available in Firetide HotView Pro, Mesh Dynamics NMS and BelAir NMS show static locations of the nodes, i.e. a user has to configure the approximate location of the nodes on to the given map but in WIVI NMS the node location is obtained using Global Positioning System (GPS) at the node and is dynamically displayed on the map.

In the sections to follow, the paper explains how both the testbed implementation and the NMS were deployed and developed respectively to include the benefits and to overcome the shortcomings of the previous and ongoing work discussed in this section.

III. WiSEMesh Design

A WMN emerged as a next generation wireless network due to its economic, easy to build and faster deployment characteristics. A WiSEMesh node can be built using any computing platform, for example small form factor computers to desktop computers. WiSEMesh can have different types of computing hardware with at least two radios. WiSEMesh design exploits the fact that every household and certainly the offices have computers; this encouraged us to develop portable software that can run in any hardware platform. At present, our software is running on small form factor machine which resembles Apple MAC mini. We have deployed 16 of such machines in an unplanned way in our four story office building and 40 in a planned fashion in the 6 undergraduate buildings. Fig. 1 shows our deployment using satellite imagery from Google Earth.

A. Hardware

Each WiSEMesh node consists of a PC, three 802.11b/g wireless NICs along with three omnidirectional antennas. We use the small form factor machine that has Intel Celeron mobile processor 1.5GHz (bus speed 400MHz), 2.5" HDD 40G (5400rpm), two USB 2.0 ports and one 10/100 Ethernet port. A USB 2.0 HUB is used to connect two version 4 Linksys 802.11b/g cards and one version 1 Linksys 802.11b/g card (Fig. 2).

Each node is equipped with three 802.11b/g card (WUSB54G) based on the Intersil Prism 54 chip with USB interface. WUSB54G device has the feature of RT2500USB WLAN system of Ralink Technology [17]. The card has two

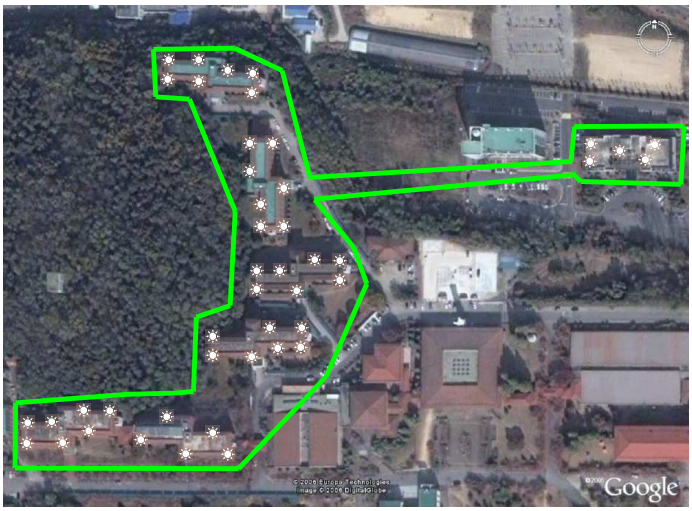

Fig. 1. Deployment of two island testbeds. The left enclosed contour is undergraduate dormitories and right enclosed contour is 4 story office building.

Fig. 2. Bottom right: WiSEMesh node deployment in the office building. Top left: Sensor network node.

chips, first, RT2526 chip used for transceiver chip, second, RT2571 chip used for 802.11 b/g MAC/BBP (Base Band Processor). After the modifications, we have used Ralink Technology RT2500 Device Driver, which is an open source driver, for RT2571 chipset. The radios transmit 14 dBm power at 54Mbps OFDM, and operate with RTS/CTS disabled. Each WUSB54G card is equipped with a 2dBi omnidirectional antenna. The cards use an IBSS (or ad hoc) mode, in that nodes communicate directly without access points.

High gain antennas are deployed for the corner nodes of each undergraduate building for connectivity with other nodes on the opposite side of the building. The 7dBi omni-directional external antenna has a 3dB vertical beam width of 20 degrees. The antenna is attached to its node with a cable which introduces 2dB of attenuation. In Fig. 1, the two island testbeds are connected to each other using two 13dBi directional Yagi antennas located on the 3rd floor of the office building and the 5th floor of the nearest undergraduate dormitory building with a line of sight (LOS).

B. Software Package and Networking Protocols

Each WiSEMesh node runs identical software consisting of Linux OS, OLSR routing protocol, a DHCP server, etc. Most nodes that are deployed have software pre-installed and some nodes are configured by transferring the image of the pre-configured software package using systemimager over the management channel. Systemimager is a software that does automated installs (makes clones). We also upgrade all the nodes' software over management channels i.e. WiSEMesh or Ethernet connection. From the user's perspective, the node acts like an access point: the user connects a PC or laptop to the node's wireless interface, and the node automatically configures the user's computer via DHCP, listing the node itself as the default IP router.

1) Addressing: The mesh gateway needs both Ethernet and wireless interface configuration. Ethernet interface connects to the Internet and is assigned a public IP address. NAT is a part of software package that translates the private IP address to public in mesh gateway node. NAT in the nodes translates the user IP address to mesh network IPs. DHCP server is needed at each node to accept connection from users. The user tries to associate with some nodes. At the time, the node assigns a private IP address using DHCP. All the nodes including mesh gateway need to configure ARP proxy to find out WiSEMesh node's address that is located multihop away from itself. The user traffic travels via multiple nodes arriving at the mesh gateway by OLSR routing protocol. At the gateway, the traffic is routed out and in using NAT. A user can also communicate with another user in the WiSEMesh.

2) Routing: WiSEMesh uses OLSR routing algorithm. This routing algorithm is developed for mobile ad hoc networks. It operates as a table driven, proactive protocol, i.e., exchanges topology information with other nodes of the network regularly. Each node selects a set of its neighbor nodes as "multipoint relays" (MPR). In OLSR, only nodes, selected as such MPRs, are responsible for forwarding control traffic, intended for diffusion into the entire network. Nodes which have been selected as MPRs by some neighbor node(s) announce this information periodically in their control messages. Thereby a node announces to the network, that it has reachability to the nodes which have selected it as an MPR.

C. Network

In office testbed deployment, the nodes are placed on the wall near the ceiling of each floor. On average each floor has 5 nodes; total of 16 nodes are deployed in the office building. The three 802.11b/g cards are placed near the nodes.

The undergraduate dormitory testbed nodes are placed near the dorm room windows on the fifth floor. The nodes are kept indoor while the cards are extended outside the window supported by a long hollow rectangular pipe. The antennas outside the window can cover a side of the building. At the corner of the buildings one of the 802.11b/g cards has external antenna in place of the vendor supplied 2dBi omnidirectional antenna. The high gain antenna placed at each corner of the building establishes an LOS with one another and links

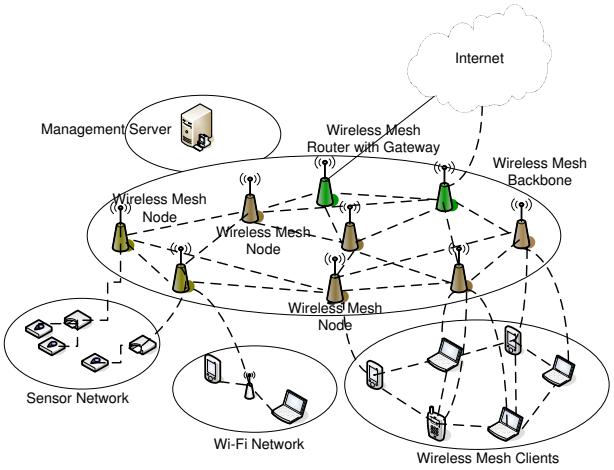

Fig. 3. WiSEMesh Architecture.

both sides of buildings. The strategy for node placement in undergraduate dormitories has two main objectives; first, to provide enough throughput to all the users regardless of the distance from the nodes, and second, to provide a well connected mesh topology. Repeated field experiments revealed the possible optimum sites for the node placement. The planned node placement efficiently provides internet connectivity to over 1000 residents.

WiSEMesh is a hierarchical network, where two wireless interfaces, operating with WEP encryption for security, in each node form mesh backbone while remaining interface provides wireless service to users. Our testbed hierarchy is shown in Fig. 3. WiSEMesh can also be used for backhaul traffic from sensor network testbed deployed in the office building.

IV. WiVi Network Management System

WiVi NMS alleviates system administrators and researchers from rather painstaking tasks of collecting data from several nodes distributed in a large area and updating each node with new algorithms. In the sections to follow we discuss the software architecture and database structure of WiVi but next we elaborate our data collection scheme.

A. Data Collection

A database of a scalable NMS needs to accommodate most if not all parameters associated with all layered communication protocols. Moreover, the amount of data to be collected is massive. Therefore, for a systematic data collection network parameters are grouped under layered communication protocols.

- Physical Layer: The wireless network capacity is dictated by channel conditions. Since a WMN node must be placed in a close proximity to other WMN nodes to establish a connected mesh network, the interference within a WMN due its own nodes is significant. To understand the nature of the interference, we have captured Signal to Interference Ratio (SIR). Another parameter that describes channel quality is Signal to Noise Ratio (SNR) for which WiVi identifies the rogue APs. These parameters provide a true nature of the interference in a WMN.
- MAC Layer: The data rate determined by MAC based on link quality for each wireless NIC is recorded with WiVi. The total data transmitted and received by the NICs are also stored in WiVi database.
- Transport Layer: TCP throughput between any two nodes are collected. Since a node contains multiple wireless NICs, the throughput data is only collected for the NICs that form a mesh backbone link.
- Upper Layer: Using the ARP and the DHCP client lists number of connected users of each node are stored in the WiVi database. WiVi is designed to track the location of the nodes. Usually, the nodes are fixed to a location in a typical WMN. But, with evolving wireless technology, a mobile WMN node is very realistic in near future. Other reason to have GPS coordinates of each node is to realize the exact location of the node. When a node location is visualized with reference to a satellite map, it can provide information on surrounding terrain (and assist in determining channel quality).
- System Level: In a temporary deployment of WMN, nodes are usually battery powered. Power is a critical resource in such deployement. Hence, WiVi also collects power level of each node. Among other parameters captured, the overall processor utilization and memory utilization show possible congestion at the node. WiVi can also monitor resources utilized by an individual process running on the node.

WiSEMesh has three radios per node where one is dedicated to client network while other two are used for mesh backbone connectivity. All the NICs have their own physical environment and MAC protocol properties. An aggregate data collection from each node can't verify the performance of each wireless network individually. Therefore, WiVi captures data from the components such as the three wireless NICs and the PC separately.

B. Software Architecture

The WiVi software architecture is shown in Fig. 4. The architecture has a management server and a WMN Node Object. The first one is located in a central server and the second in each WMN node; both software entities can be connected to each other via an enterprise Ethernet backbone or through a mesh backbone.

Management server uses web server (e.g. Apache) that provides user interfaces for Management and Configuration Module (MCM), Software Update and Distribution Module (SUDiM), WiVi Tool Configuration Module (TCM), and Data Translation and Visualization Module (DaTViM). DaTViM dynamically creates and manipulates image files and streams them directly to a browser. DaTViM provides realtime visualization of network status along with several captured parameters in graphical and tabular formats. It also allows users

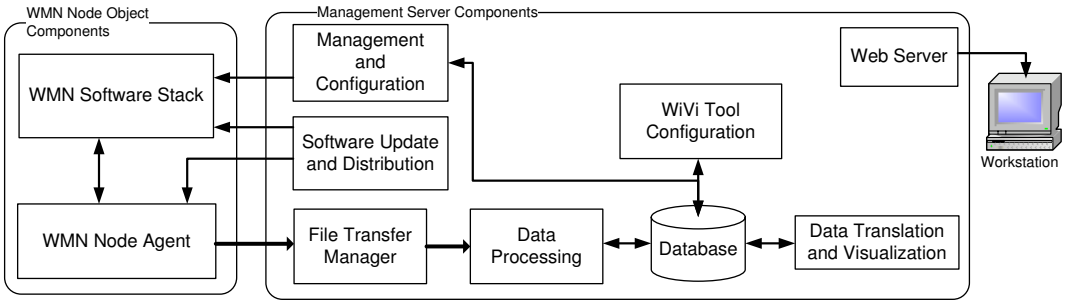

Fig. 4. WiVi Software Architecture.

to view current status of different networks, get performance chart of each node, and monitor physical and MAC layer behaviors of a wireless NIC.

Using TCM, a privileged user can set reference data (discussed in Section IV-C) that are used for visualizing current network status graphically. For instance, ranges of connected client numbers to a node can be represented with various color codes. The user can register new networks with WiVi using TCM. TCM allows the user to provide a satellite or aerial image of the location where the network is deployed.

Both MCM and SUDiM use secure shell library (libssh2) which provides access to resources (shell, remote exec, tunneling, and file transfer) on a remote machine using a secure cryptographic transport. These modules appear as a transparent middle tier that provides users with command line interface to remote nodes. SUDiM manages and distributes software updates to all or selected nodes of a network based on the user preference. The updates can be uploaded to selective nodes of a network with a single click. This can effectively partition a network into numerous subnetworks. Therefore, a large network can host several smaller networks that can host different controlled experiments. Furthermore, the network can be partitioned by configuring each smaller network with different IP domains.

MCM has a capability to configure any hardware components of a node. A user can set various wireless NIC parameters such as transmission power, channel (frequency), sensitivity threshold, bit rate, RTS/CTS threshold, fragmentation threshold, and power saving scheme. MCM also configures IP address, subnet mask and broadcast address. MCM can also configure selected wireless NICs of selected nodes. The configuration changes are also reflected in the database. Since the network can have different types of hardware, MCM reports back any unsupported and unrecognized wireless and system configuration requests. Another feature of MCM is fault management. WiVi provides several performance metrics that provide client side network congestion, mesh backbone link congestion, node resources utilization, etc. With the help of these metrics, a user can identify possible performance degradation and locate the source of the fault.

The WMN node has two components: WMN Software Stack (SoS) and WMN Node Agent (NoA). Both of these

```
- <Mesh_Node>
  - <Characteristics>
      <Name>Daejeon</Name>
      <Longitude>127.3612555</Longitude>
      <Latitude>36.37546944</Latitude>
      <XYcoordinate>Y</XYcoordinate>
      <Power_level/>
    - <Processor_util>
        <Overall>22%</Overall>
      - <Process>
          <Processid>1242</Processid>
          <Processid_util>6%</Processid_util>
        </Process>
      </Processor_util>
    + <Memory_util>
    </Characteristics>
  + <Interface>
  - <Interface>
      <Name>rausb1</Name>
      <Num_of_rague_AP>2</Num_of_rague_AP>
      <Essid>WIMESH-DAEJEON</Essid>
      <Received_bytes>1942442704</Received_bytes>
      <Transmitted_bytes>28656318</Transmitted_bytes>
      <Frequency>2.412</Frequency>
      <Channel>1</Channel>
      <Maximum_bit_rate>11</Maximum_bit_rate>
      <Tx_power>20</Tx_power>
      <Num_of_rague_AP>12</Num_of_rague_AP>
      <Num_of_client>0</Num_of_client>
      <Receiving_rate>4060 (bytes/sec)</Receiving_rate>
      <Transmitting_rate>106 (bytes/sec)</Transmitting_rate>
    </Interface>
  - <IP_of_neighbor>
      192.168.254.33
      <Throughput>5.1721 (Mbps)</Throughput>
    </IP_of_neighbor>
  + <IP_of_neighbor>
  </Mesh_Node>
```

Fig. 5. A node agent generated data file in XML format.

components are updated by SUDiM and SoS is configured by MCM. SoS consists of Ubuntu (a unix based operating system), wireless NIC drivers, and pre-configured network setting. NoA is a script file, written in Python, that runs as a cronjob. NoA generates a data file (a sample is given in Fig. 5) that contains WMN node characteristics, interface parameters, and mesh backbone link throughput. The attribute values represent the field for which the data is captured for. The management server uses the attribute specific protocol for handling the data. More detailed discussion is given in Section IV-B2.

The network parameters available in an NoA data file are captured using software tools already available in SoS. This insures a light-weight WMN Node Object. A user can add or remove any number of parameters from the NoA data file by updating the NoA script file at the users' computer and upload to the management server which will distribute the file to all the nodes requested by the user.

In the following two sections we will discuss the data collection methods at each WMN node followed by data processing methods at the server and data dissemination into WiVi database.

1) Node Agent: As discussed earlier, the network parameters captured by NoA by utilizing software tools that are already part of the operating system or SoS. Our design objective for NoA is to construct a simple script which can run in any system with minimum setup. For example, a Linux box usually has DHCP server, NAT, ARP, ftp server, nuttcp, perf, etc. NoA uses these tools to capture network parameter and sorts them into a standard format that is agreed and understood by the Management Server. A part of NoA algorithm is provided in Fig. 6.

Fig. 6 shows part of the data collection process where number of clients and throughput between two nodes are measured. DHCP keeps a record of IP and MAC addresses of the clients connected; the list shows a total of online and past clients. The algorithm cross references DHCP table with ARP table and obtains the list of valid live clients connected to the node. Another parameter obtained is the raw TCP (or UDP) network layer throughput for which a client/server mode "nuttcp" is used. In this mode, a server is first started on one node and then a client may either transmit data to or receive data from the server node. All the information provided by "nuttcp" is reported by the client node, including the information from the server node, thus providing a full snapshot of both the transmitter and receiver ends of the data transfer. The obtained values are then written into the data file and sent to the management server. The data collection process repeats periodically with user specified time gap.

2) Server side Data Management: Server side data management is part of management server where bulk of the data processing and storage is done. The server receives data files from all the nodes in regular time interval. When a new file arrives, it overwrites the older one provided that the existing file has been marked as processed. File synchronizer (Fig. 7) is handled by File Transfer Manager Module (FiTMaM) shown in Fig. 4. File synchronizer keeps a pointer to the data files from a node in FIFO buffer. Once the file is processed, the pointer is deleted from the buffer. All the data files are kept in file repository.

Data processing and trigger functions process an unprocessed data file. These functions are executed by Data Processing Module (DPM) of Management Server (Fig. 4). A data processing function validates the data file, parses it and performs data manipulation such as aggregating the data transmitted and received by wireless NICs, taking average of throughputs, etc. Meanwhile, based on the attributes to be

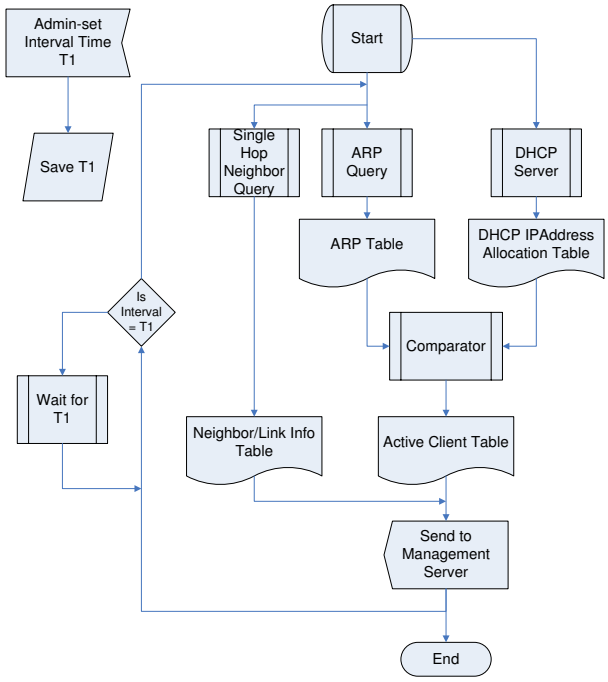

Fig. 6. Node agent data collection algorithm.

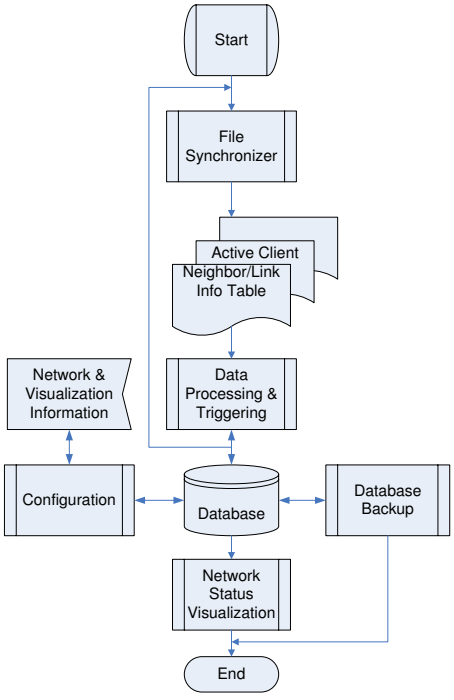

Fig. 7. Management server data processing algorithm.

updated, the trigger function updates specific field of specific table in the database. For instance, the aggregate property of data processing function sums "Transmitted_bytes" of each "Interface" shown in Fig. 5 and provides the trigger function with the attribute, i.e. field name and its value, i.e. the sum. The trigger function looks up the attribute to find a specific field of specific table (in this case, field: txbytes, table: node_details, refer Section IV-C) in the database and updates the field with the value. Different attributes have different database fields assigned to them. Moreover, some attributes' values are updated to their respective fields while some are inserted. These rules or protocols are written in trigger function for each attribute.

In a large network, several management servers are employed and each is assigned a group of nodes. Database backup initiates transfer of local database stored in each management server to central database. This process is initiated by users.

C. Database Design

A repository that stores a network related information has complex design issues where a lack of in depth analysis can cause data loss in a long run. The problem becomes significant for systems like WMN where the network is usually unplanned and the network size grows faster. Adding further challenge to the task of database design, the data to be collected from a WMN is large and not fixed. To overcome the difficulties, the database should be scalable and robust.

WiVi database designing is based on a relational database management system. The database schema is shown in Fig. 8. It is based on a scalable design where tables or objects record data from the WMN node components that are involved in wireless networking. The network components are categorized into hierarchy of hardware components where the hardware like wireless NIC and a PC makes up the fundamental elements of the network and these elements combine together to form nodes and then the nodes compose the network. Three distinct data types emerge. We differentiate these data types into master data and transaction data.

Reference data used solely for visualization purpose, categorizes master and transaction data found in the database. Volumes of reference data are much lower than what is involved in master data and reference data changes more slowly than master data. WiVi has two tables with reference data (*wivi_link* and *wivi_node*) to assign color codes to a user defined "data rate range" and "connected client number range" respectively.

Master data is the data without which no transactions are possible. It describes the things that interact when a transaction occurs. For instance, master data that represents networks and nodes must be present before the transaction is fired to transmit a burst of traffic between nodes of a network.

Transaction data is the business documents that are created using the master data like data rate between two nodes, resource utilization of nodes, data received and transmitted via an interface, etc. Transactional Data can change very often and is not constant.

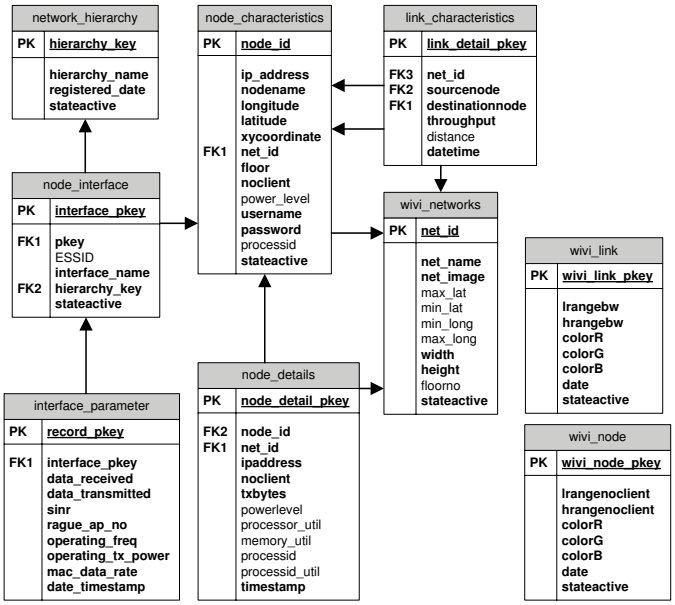

Fig. 8. WiVi Database Schema.

As discussed earlier, a network is composed of a lot of hardware. *wivi_networks* is a master data table (or master table) that stores multiple networks that may have been deployed at different locations. This table stores physical attributes of a network such as location, topology (spread in an area or deployed in a multi-story building). WiVi uses a browser based view where the location of the network is calibrated in x-y coordinates such that users can see the exact location of the network against a background image provided by the user himself. The image is usually an aerial map obtained from GPS tools such as Google Earth.

The network is further divided into nodes and logical networks. In WMN, there are two types of logical networks: a mesh backbone network and several client side networks. *network_hierarchy* keeps record of the logical networks. The node information such as the network id, physical location (longitude, latitude and altitude) is stored in node specific master table, *node_characteristic*. This table also stores the number of connected clients to the node and its power level, which are transaction data. These parameters are updated in this table for a faster visualization process where the search query will take less time since the number of records in the table is less than transaction table.

Each node is composed of wireless NICs and a PC. Another master table, the *node_interface* table stores NIC information of each node such as number of wireless NICs connected and network configuration information of each NIC. The NIC configuration information is ESSID, interface name (e.g. wlan0) and the serviced logical network referred from *network_hierarchy*.

There are three transaction tables: *node_details*, *link_characteristics* and *interface_parameters*. Each transaction table keeps a record of the interaction between master data.

node_details stores per node snap shot values such as number of clients connected, aggregate transmission rate, power level, resource utilization of the system, etc. *link_characteristics* table records throughput between two nodes at a given time. Likewise, *interface_parameters* stores per wireless NIC information of each node. Some of the recorded information are received and transmitted data rates, SINR, number of rogue AP, MAC data rate, etc. The transaction tables keep time stamp for each transaction entry.

V. Performance Measurement and WiVi Data Visualization

In this section, we discuss the performance of WiSEMesh in terms of network reliability and throughput. WiVi performance is measured with database query time. The section ends with WiVi visualization and data arrangement discussions.

Network reliability is the availability of end to end functionality for clients. It is also the ability to experience failures or systematic attacks, without impacting customers or operations. As the network grows, weaknesses in the network infrastructure become clearer. Failures not only affect current voice and data use but could also limit emerging wireless applications such as high-bandwidth Internet access and emergency network. Moreover, to achieve scalable WiSEMesh, the network must be reliable. A reliable network means client satisfaction, that helps a speedy growth of the network.

$$normalized\ reliability = \frac{number\ of\ stable\ links\ per\ day}{total\ number\ of\ links\ per\ day} \quad (1)$$

Eq. 1 is used to measure the reliability of WiSEMesh. The number of stable links per day is the total number of links that stay connected (persistent) throughout the day.

$$normalized\ througput = \frac{average\ of\ link\ throughput}{maximum\ average\ throughput\ of\ a\ link} \quad (2)$$

Eq. 2 measures the network capacity in a day. The numerator of Eq. 2 represents the average of mean throughput of all the links of the network. The denominator is the maximum mean value of all the links. The *link_characteristics* table in WiVi database stores the mesh backbone link status from which we can obtain the link activity and the link throughput.

Fig. 9 shows WiSEMesh reliability for a 30 day period. On average more than 90% of the link is stable every day. The reasons for a link failure are hardware components instability and degradation of the link quality. The WiSEMesh capacity is measured at 5Mbps per link. The normalized WiSEMesh backbone throughput doesn't depend on link stability, due to well connected mesh backbone network. A node is connected via more than one link to the network. Hence, WiSEMesh is highly stable and reliable network with multiple pathways in case of link failure.

Critical processes of WiVi are management server data processing and data visualization. Data processing is a key issue at management server as a large amount of data files are sent to it from the nodes connected. As the network size grows, the data files also increase. Therefore, a faster data processing is required at the server side. Another critical process is data translation and visualization. Fig. 10 shows a growth of query time with an increase in the number of nodes. The server can update the database for a network of 40 nodes in under 120 milliseconds. Usually a node is set to send the file every one minute. This means the database can be updated for a large number of nodes within a minute. The data processing module executes 4 SELECT, 3 INSERT and 1 UPDATE queries for each data file.

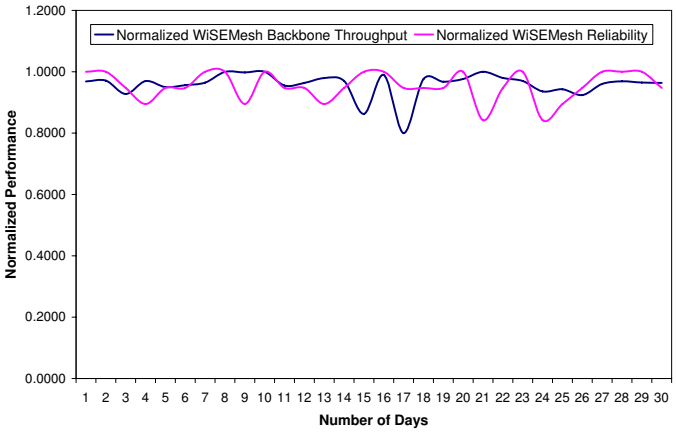

Fig. 9. WiSEMesh Performance.

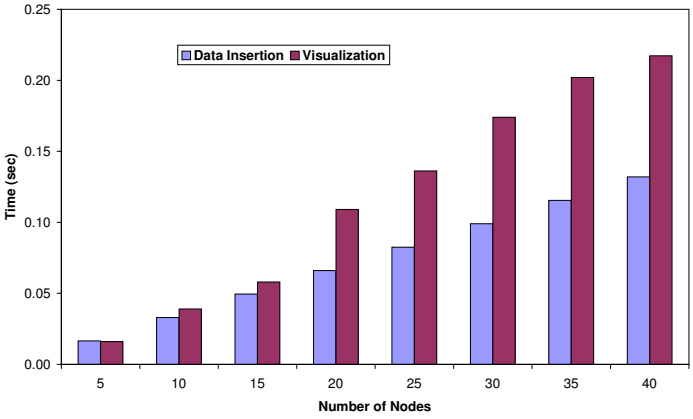

Fig. 10. WiVi Performance.

A realtime visualization requires minimum query time. For a 40 node network, data translation and visualization on to the browser take 200 milliseconds. For optimizing visualization process, the transaction data is actually updated into master tables too. As a result, a large number of nodes can be visualized within the human eyes response time.

Fig. 11 shows a snap-shot picture of WiVi visualization of the WiSEMesh undergraduate dormitory testbed. On the left hand side column, the first item is the list of testbeds maintained in WiVi followed by color code arrangement of the range of connected users and link throughput. Another item in the left column is nodes of the network. A user can select a node and a date for detailed node information. The

Fig. 11. WiVi Visualization.

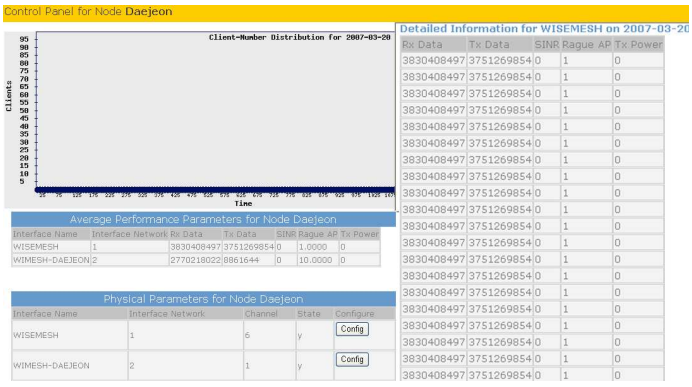

Fig. 12. WiVi Data Arrangement.

last item in the column is a selection panel of networks stored in WiVi for configuration purpose.

WiVi systematically presents data of a given date for each node. WiVi can show different numbers of graphs that users request. An example graph (Fig. 12) is the distribution of number of clients connected to the nodes in a day. Each node is further divided into interfaces and are shown with their average network parameter values such as average transmitted and received bytes, average number of rogue AP, etc. All these parameters are also available in detail for viewing purpose. A user can configure each interface or the node as well.

VI. Conclusion

WMN testbeds are popular among research institutes and many commercial companies are investing in WMN product line. Both the research and the commercial worlds are deploying WMN testbeds to learn and enhance performance of the network. In this paper, we introduced our WMN testbed, WiSEMesh, that uses PCs available in households and business offices. WiSEMesh nodes can have multiple heterogeneous radios; thus making it a high performance and highly scalable network. The testbed serves dual purpose by providing free wireless internet connectivity to a community of over 1000 students living in the dormitories and a testing platform for new and improved communication network layers. The software operating in the testbed nodes is a package of open source OS, drivers and tools. Moreover, WiSEMesh can be accessed from institutions and companies located anywhere in the world.

Once the testbed with necessary properties for community service and/or research was installed, we focused on designing a network management system (NMS) that can be implemented for any wireless network. We developed WiVi, a NMS that has dynamic and scalable data collection and storage facility, software distribution platform, fault management and configuration modules, and visualization tool. The paper discussed a complete WMN solution that has a flexible testbed with a dynamic NMS.

Further research is being carried out in the field of fault management that involves self-diagnosis, self-repair and self-healing.

References

[1] I. F. Akyildiz and X. Wang, *A Survey on Wireless Mesh Networks*, IEEE Radio Communications, September 2005.
[2] Network Working Group, RFC 3411-RFC 3418.
[3] K. Ramachandran, K. C. Almeroth and E. M. Belding-Royer, *A Novel Framework for the Management of Large-scale Wireless Network Testbeds*, Proceeding of Wireless Networks Measurements, April 2005.
[4] K. Ramachandran, E. M. Belding-Royer and K. C. Almeroth, *DAMON: A Distributed Architecture for Monitoring Multi-hop Mobile Networks*, proceedings of the IEEE Sensor and Ad hoc Communications and Networks (SECON), October 2004.
[5] MIT Roofnet, *http://pdos.csail.mit.edu/roofnet/doku.php*
[6] Berline Roofnet, *http://sarwiki.informatik.hu-berlin.de/BerlinRoofNet*
[7] Microsoft mesh networking, *http://research.microsoft.com/mesh/*
[8] UCSB MeshNet, *http://moment.cs.ucsb.edu/meshnet/*
[9] Purdue University Wireless Mesh Network Testbed, MAP, *http://engineering.purdue.edu/MESH*
[10] Wireless Mesh Networks Research at Georgia Tech, *http://www.ece.gatech.edu/research/labs/bwn/mesh/*
[11] Wireless Mesh Network at Carleton University, *http://kunz-pc.sce.carleton.ca/MESH/index.htm*
[12] Hyacinth: An IEEE 802.11-based Multi-channel Wireless Mesh Network, *http://www.ecsl.cs.sunysb.edu/multichannel/*
[13] John Hopkins Smesh, *http://www.smesh.org/*
[14] Firetide HotView Pro Mesh Management System, *http://www.firetide.com/*
[15] BelAir BelView Network Management System, *http://www.belairnetworks.com/*
[16] Mesh Dynamics, *http://www.meshdynamics.com/WNetworkMgr.html*
[17] Ralink Technology, *http://www.ralinktech.com*