# WANMon: A Resource Usage Monitoring Tool for Ad Hoc Wireless Networks

Don Ngo, Naveed Hussain, Mahbub Hassan
School of Computer Science & Engineering
The University of New South Wales
Sydney, Australia
Email: {dsn,navh,mahbub}@cse.unsw.edu.au

Jim Wu
School of Computing & Information Technology
University of Western Sydney
Sydney, Australia
Email: jimw@cit.uws.edu.au

## Abstract

*As wireless networking products and ad-hoc networks become more popular, the resource usage in routing packets for other network users will become an issue for users and developers of ad-hoc wireless products. In this paper, we propose a novel monitoring tool called* WANMon *(Wireless Ad-hoc Network Monitoring Tool). WANMon can be installed on a wireless node to monitor resource usage (such as network usage, power usage, memory usage, and CPU usage) at the node, in the context of how much of resource is used for supporting the node's own applications versus the usage for routing data of other network users in an ad-hoc wireless network. We discuss design of WANMon, and present our prototype implementation of WANMon in Linux, which provides a starting point for further research and a basis for developing a fully featured ad-hoc wireless network monitoring tool.*

## 1 Introduction

The rapid proliferation of portable computing devices (e.g. laptops, PDAs, mobile phones) has fuelled unprecedented interests in ad-hoc networking in recent years. The key requirement in ad-hoc networking is that the user devices must also act as routers to establish the network. Routing creates extra demands on the thin resources (e.g. energy, memory, CPU) in portable user devices. It is therefore important for the users to be able to accurately determine the amount of resources that are being used by the routing of other users' data. For example, a user may wish to terminate his/her participation if routing drains the battery too quickly.

In this paper, we propose a novel resource usage monitoring tool for ad-hoc wireless Ethernet networks, called *WANMon* (Wireless Ad-hoc Network Monitoring Tool). WANMon is designed with the specific goal of allowing users to determine resource usage within the context of how much of their resources are being consumed in facilitating their network applications and services versus resources being used to facilitate the network applications and services of the other ad-hoc wireless network users (i.e. routing their data).

The rest of the paper is organised as below. In Section 2, we present an overview of the monitoring tool WANMon. Section 3 describes the architecture and technical specifications of WANMon. Section 4 describes our prototype implementation of WANMon in Linux. We conclude the paper in Section 5 and discuss limitations of WANMon in Section 6.

## 2 Overview of WANMon

WANMon is a monitoring tool that allows the user to monitor the resource consumption by ad-hoc wireless network based activities. The kind of resources that WANMon will monitor includes, network usage, power usage, CPU usage and memory usage. The emphasis is on how much of these resources are being used to support the networking tasks of the user on this node versus resources being used to facilitate the networking activities of the others users (i.e. resources being used in acting as a router).

Statistics monitored by WANMon can be classified into four categories, as listed below:

- **Network**: WANMon monitors statistics on Data link, Network, ICMP, TCP/UDP layers.
- **Power**: WANMON focuses on network card (NIC). It provides power consumption figures by NIC.
- **Memory**: WANMon provides a cumulative memory count as well as memory usage over a period of time by the data link layer and network layer.
- **CPU**: WANMon monitors CPU usage and calculates the CPU time being used for routing versus the CPU time being used by user applications at a node.

Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks (LCN'03)
0742-1303/03 $ 17.00 © 2003 IEEE

### 2.1 Network Usage Statistics

WANMon provides statistics related to data that is being sent, received and routed through a node. These statistics include the number of packets and bytes that are sent and received by the wireless interface, and protocol used (e.g. TCP, UDP, and ICMP). The statistics also provide information on whether they belong to the flows of the user or belong to the flows of the other network users. Other statistics include the number of packets in errors, the number of packets dropped, the number of packets that have overruns or are not encrypted correctly.

### 2.2 Power Usage Statistics

WANMon provides usage statistics regarding power consumption by the NIC. The power consumption statistics are further categorised into the power consumption in sending and receiving data either for the purposes of the networking applications running on the node or for routing the data of the other network nodes. The information will be displayed both textually and graphically. WANMon also provides basic information regarding the total power consumption on the system through the use of APM (Advanced Power Management).

### 2.3 Memory Usage Statistics

WANMon monitors buffer space being used by the networking subsystem. This means that WANMon will provide statistics regarding how much memory, over a past period of time, was used to send and receive data. These data can be categorised into memory used for networking applications of that node and memory used in routing of packets of other network nodes.

### 2.4 CPU Usage Statistics

WANMon provides CPU usage in kernel space for networking applications. As most of CPU usage time is devoted towards networking, we have a reasonable upper bound on the CPU usage by networking applications of that node. It can also provide further information (in terms of upper bounds) of the CPU usage solely by the kernel and therefore the routing that is performed [1].

## 3 WANMon Design

This section explores the design of WANMon. We describe various components that are used to create WANMon, and explain how they interact and fit with each other.

[1] Because routing utilises CPU only in the context of the kernel space.

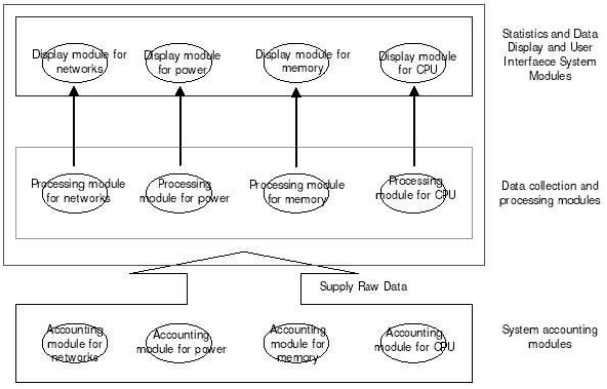


**Figure 1. Architecture of WANMon.**

### 3.1 Architecture

The design of WANMon is a three tiered architecture, as shown in Figure 1. At the bottom we have the system accounting modules. These modules are responsible for collecting raw data regarding resource usage. The middle layer represents data collection and processing modules. These modules obtain their data from the accounting modules and process it to create useful information from it. The processing involves converting into appropriate units, ratios and percentages etc. At the top layer is the statistics and data display and user interface system modules. These modules get the information from the middle layer, format it and then display it to the user through the GUI as textual and/or graphical information. The middle layer has the data collection and processing modules. The data collection modules then interact with the bottom layer, the system accounting modules, which provide the middle layer modules with the raw data for processing into information.

### 3.2 Algorithms

WANMon are designed to collect four different types of statistics, i.e. network, power, memory, and CPU usage statistics (see Figure 1). An algorithm is implemented in each component shown in Figure 1. to explain algorithms for collecting network usage statistics.

#### 3.2.1 Network Usage

The accounting module performs system accounting to keep track of the bytes being sent and received. This can be completed by using a packet sniffer (e.g. *tcpdump* [1, 2] and *ethereal* [3]). A packet sniffer provides useful information such as the MAC and IP header of the packet, the size of the packet, the type of packet (transport layer category) etc. We configure the packet sniffer to provide the following information:

- Packet size
- MAC header
- IP header
- Transport Layer Protocol type

This raw data is then passed onto the network data processing module. The data processing module uses this data to determine the following:

- Number of packets/bytes transmitted for the networking applications of this host
- Number of packets/bytes received for the networking applications of this host
- Number of packets/bytes transmitted in routing the packets of the other nodes
- Number of packets/bytes received in routing the packets of the other nodes
- Categorisation of the packets/bytes on the basis of transport layer protocol

Figure 2 shows the pseudocode of algorithm implemented in the network data processing module. Once the network based data processing module has finished processing the raw data that is made available to it by the system accounting module, it allows the statistics and data display system to have access to this processed data for displaying.

### 3.2.2 Power Usage

To determine the power consumption by NIC card, we use the linear equations proposed in [5]. In [5], the power consumption of a NIC in sending/receiving packets is modelled using a set of linear equations of the form:

$$aX + b \tag{1}$$

in which $X$ represents the packet size. $a$ is an coefficient, and $b$ is a constant. Coefficient $a$ and constant $b$ in Eq. (1) are determined empirically based on various packet transmission/reception scenarios identified in [5]. The equations gives power consumption by the NIC in sending/receiving a packet in an ad-hoc wireless Ethernet network. Figure 3 shows the pseudocode of algorithm implemented in the power data processing module as shown in Figure 1.

```
On packet arrival:
IF packet is an IP packet THEN
    IF packet's dest MAC address is mine THEN
        IF packets dest IP address is mine THEN
            packet is meant for me;
        ELSE
            packet is for routing;
        END IF
    ELSE
        packet was received in promiscuous mode;
    END IF
END IF

On packet departure:
IF packet is an IP packet THEN
    IF packet's src MAC address is mine THEN
        IF packets src IP address is mine THEN
            packet is meant for me;
        ELSE
            packet is for routing;
        END IF
    END IF
END IF
```

**Figure 2. Pseudocode for network data processing module.**

```
On packet arrival:
IF packet is an IP packet THEN
    IF packet's dest MAC address is mine THEN
        IF packets dest IP address is mine THEN
            packet is meant for me;
        ELSE
            packet is for routing;
        END IF
    ELSE
        Packet was received in promiscuous mode;
    END IF
END IF

On packet departure:
IF packet is an IP packet THEN
    IF packet's src MAC address is mine THEN
        IF packets src IP address is mine THEN
            packet is meant for me;
        ELSE
            packet is for routing;
        END IF
    END IF
END IF
M̂
```

**Figure 3. Pseudocode for power data processing module.**

```
On packet arrival:
IF packet is an IP packet THEN
    IF packet's src IP address is my IP address THEN
       add the memory used to store the packet to
       my memory transmit usage count;
    ELSE
       add the memory used to store the packet to
       other memory transmit usage count;
    END IF
END IF

On packet departure:
IF packet is an IP packet THEN
    IF dest IP address is my IP OR
    packet's dest IP address is broadcast address THEN
        add the memory used to store the packet to
        my memory receive usage count;
    ELSE
        add the memory used to store the packet to
        other memory receive usage count;
    END IF
END IF
```

**Figure 4. Pseudocode for memory data processing module.**

#### 3.2.3 Memory Usage

As the NIC device driver keeps track of memory usage related to packet transmission/reception, we just need to modify the device driver to perform accounting. Figure 4 shows the pseudocode of algorithm that collects memory usage information. Once the data has been collected by the driver, the driver will forward the collected data to the data processing module. In this case, the data processing module does not have much to do. It simply prepares the raw data (for displaying) and then forward it to the display module.

#### 3.2.4 CPU Usage

The system accounting module of CPU usage (refer to Figure 1) is implemented as a filesystem which keeps track of the total number of CPU cycles used by the kernel as a whole, as well as that used by each of the userland processes. We assume [2] that all the networking related processing, whether to route the packet or to pass it up to the host, will register as part of the CPU time spent in the kernel mode. This CPU usage time is distributed among the userland processes and the kernel process. The time that each process spends in kernel mode is determined. This information is then correlated with the total kernel mode time to calculate the upper bound for CPU usage by the networking component for the benefit of the user's networking processes and for routing of the data of other ad-hoc network users. Figure 5 shows the pseudocode of the algorithm implemented in the accounting module.

[2] This is a safe assumption given the monolithic kernel architecture of the Linux kernel.

```
listNetworkingProcesses = get list of all networking processes
FOR EACH process IN listNetworkingProcesses DO
     totalNetworkProcessesKernelTime =
     totalNetworkProcessesKernelTime + processKernelTime
LOOP

totalKernelTime = Total kernel mode time
totalProcessesKernelTime = Total time by processes
   in kernel mode
kernelOnlyTime = totalKernelTime   totalProcessesKernelTime
```

**Figure 5. Pseudocode for CPU data processing module.**

### 3.3 GUI Interface

The design of the statistics and data display system modules revolves around the interaction between the data processing modules and the display modules. First the statistics and data display system obtains the relevant information they require. It then present this information to the user through GUI interfaces.

The statistics and data display system periodically updates the resource usage statistics and data provided to the user. To this end, a timer is maintained to determine the time at which the data needs to be provided to the statistics and data display system module for updating. The data processing module keeps track of the time it last provided the statistics and data display module with the resource usage data, and when the timer expires it will push the data to the statistics and data display module and request the statistics and data display module to redraw.

The statistics and data display module produces output in both textual and graphical formats. The textual information provides simple resource usage information (e.g. counter figures, ratios percentages etc.). The graphical information provides information in forms of line graphs, bar graphs and pie graphs.

## 4 WANMon Implementation

We have implemented WANMon on RedHat Linux platform. The network card (NIC) used in our implementation is Lucent Wavelan/IEEE Card [4] (running in ad-hoc mode). This section describes implementation of the proposed monitoring tool WANMon.

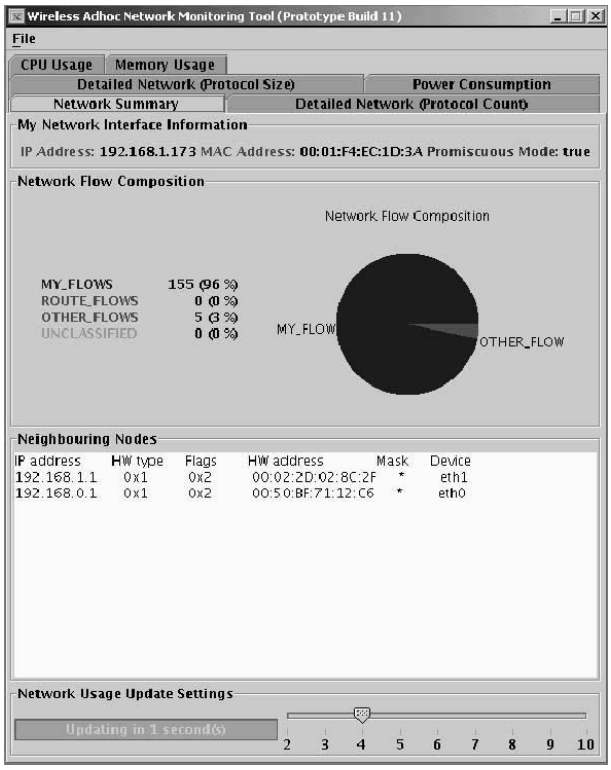


**Figure 6. WANMon Network Summary Panel.**

## 4.1 Network Usage

The implementation contains two important system account modules namely *Network_PacketAnalyser* and *Network_Neighbours*. These modules pass important network information to *Network_PacketParser*, which is the most important data processing module. It supports other modules in WANMon such as network statistics, power consumption and memory usage of the wireless network device.

The core of the network monitoring module *Network_PacketAnalyser* uses a popular network packet monitoring program *tcpdump*. The output of *tcpdump* is fed to *Network_PacketParser* network data processing module for further processing.

### 4.1.1 Network Summary

Figure 6 shows the screenshot of *Network Summary Panel* which displays a summary of all network related information of the node monitored. As shown in Figure 6, Network Summary Panel displays three types of information, i.e. network interface, network flows, and network neighbours information. Each of the these statistics is displayed in its subpanel.

The *My Network Interface information* subpanel shows identifying information of the monitored wireless node. The information displayed includes current IP address, MAC address of the wireless device and whether this device is in promiscuous mode. The *Network Flow Composition* subpanel displays the types of network flows and packet count for each type of network flow. The percentage of each flow's packet count is shown in a pie graph. To increase clarity to the user, the pie graphs sector colours correspond to the colour of each flow label. For example, in Figure 6, traffic flows initiated by the wireless node make up 96% of total flows traversing the wireless node. This implies that only a small portion (4%) of network resources at the wireless node is used to support traffic from other users (i.e. routing packets from other users). The *Neighbouring Nodes* subpanel shows IP addresses and hardware address (MAC addresses) of any nearby neighbours have been found at the wireless node.

The update bar (in the bottom of Figure 6) informs the user of when an update is to occur. When an update is due, statistical data in that panel will be updated such as tables, graphs and labels. An option of a slider has been included to allow the user to adjust the monitoring interval, as the user may feel the updates may be excessive or insufficient.

### 4.1.2 Detailed Network Information

Detailed network information about protocol can be obtained by clicking at the corresponding button on the menu bar. For example, clicking at the *Detailed Network(Protocol Count)* will bring up a new statistic panel as shown in Figure 7.

As shown in Figure 7, the Detailed Network Panel is divided into three subpanels. Each subpanel displays breakdown information on network protocols, for 'My Flows', 'Routing Flows', and 'Other Flows' respectively. For example, for the node's own flows (My Flows), 68% are UDP flows, 15% are TCP flows. ICMP and ARP traffic flows account for 2% each.

## 4.2 Power Usage

Power usage statistics are obtained by clicking at the *Power Consumption* button on the menu bar. Figure 8 shows the screenshot of *Power Consumption Panel* which displays power consumption information, including power consumption of the system as a whole, power consumption of the network device only and power consumption data update settings.

As shown in Figure 8, the *System Power Consumption* subpanel shows the overall power consumption, battery status, battery left, and estimated battery life (in minutes) at the wireless node. The *Network Interface Power Consumption* subpanel shows power consumption for the network

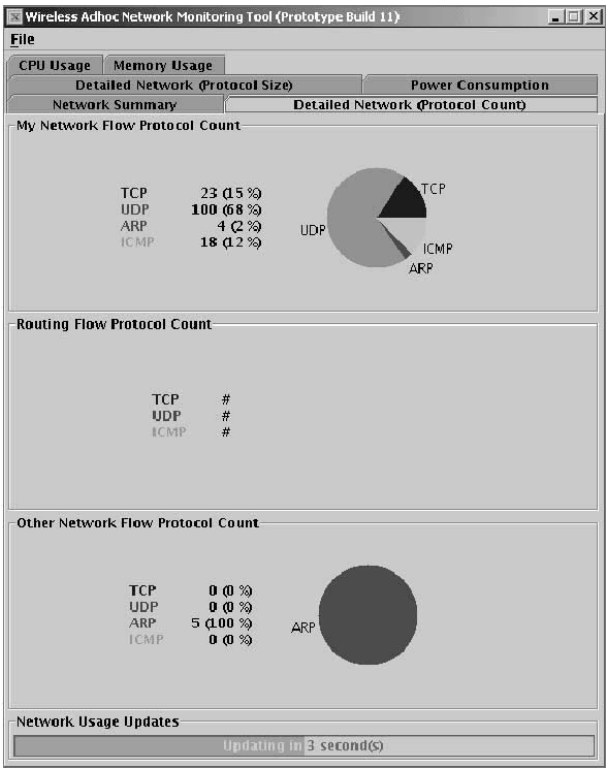


**Figure 7. WANMon Detailed Network Panel.**

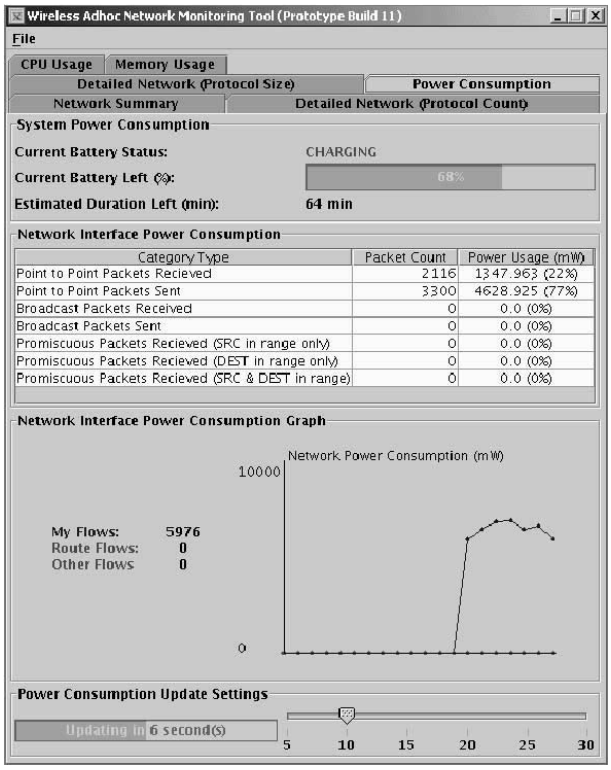


**Figure 8. WANMon Power Consumption Panel.**

card (NIC card). It presents various packet types and their resource usage in a tabular format. The total power consumption in the NIC card is shown graphically in the *System Power Consumption Graph* subpanel. As we can see in the figure, most battery power (99%) is consumed by the node's own applications.

### 4.3 Memory Usage

Memory usage statistics are obtained by clicking at the *Memory Usage* button on the menu bar. Figure 9 shows the screenshot of *Memory Usage Panel* which displays memory usage information at the wireless node. The Memory Usage Panel is divided into two parts, one subpanel displays memory usage for transmitting data and the other subpanel displays memory usage for receiving data.

As shown in Figure 9, the *Memory Usage for Transmitting* subpanel shows memory used for transmitting packets generated by the wireless node, and memory used for transmitting other packets (e.g. routing). The figure shows that the memory used to support applications at the wireless node is about 900 KB, and memory used to support other traffic is about 230 bytes. Similarly, the *Memory Usage for Receiving* subpanel shows memory used for receiving packets destined to the node, as well as memory used for receiving packets for other nodes. As shown in the figure,

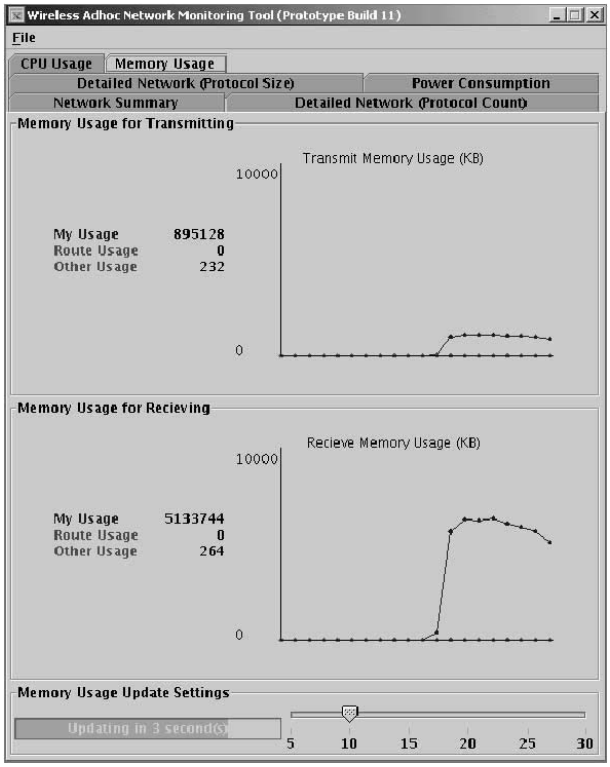


**Figure 9. WANMon Memory Usage Panel.**

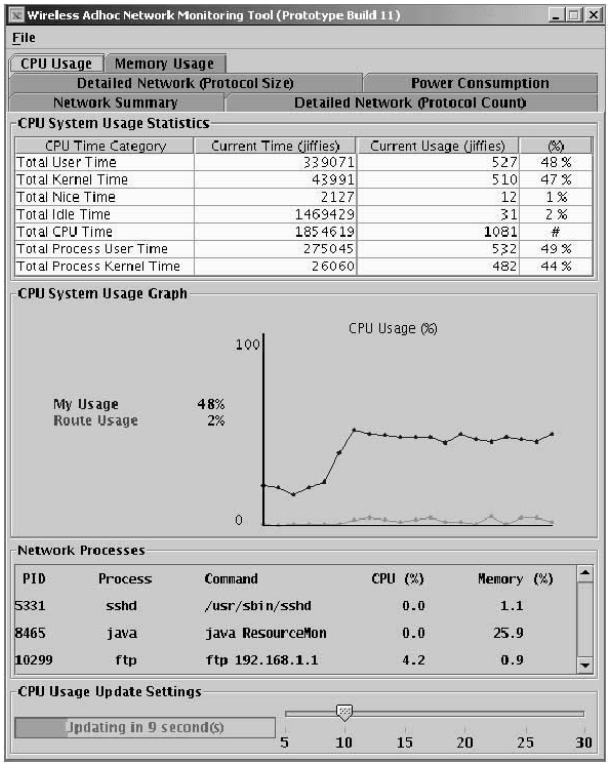


**Figure 10. WANMon CPU Usage Panel.**

about 5 MB of memory are used for receiving packets for the wireless node, and only about 260 bytes are used for receiving packets for other nodes. This means that most of the memory at the wireless node is used to support the node's own applications.

### 4.4 CPU Usage

CPU usage statistics are obtained by clicking at the *CPU Usage* button on the menu bar. Figure 10 shows the screenshot of *CPU Usage Panel* which displays CPU usage information at the wireless node. The CPU Usage Panel is divided into three parts: *CPU System Usage Statistics*, *CPU System Usage graph*, and *Network processes*.

As shown in Figure 10, the *CPU System Usage Statistics* subpanel provides a summary of CPU usage in various categories. The *CPU System Usage graph* subpanel shows percentage of CPU usage time by applications of the wireless node, and percentage of CPU usage time by routing. Consistent with other usages at the wireless node, most CPU resource (48%)is consumed by the node's own applications, with only 2% used for routing.

The *Network Processes* subpanel lists all network processes currently running on the wireless node. It provides information such as process ID, process name, command, CPU usage and memory usage for each process.

## 5 Conclusion

We have presented the design and implementation of a novel resource monitoring tool (we call it *WANMon*) for wireless Ad Hoc networks. WANMon allows wireless network users to monitor resource usage (in terms of network usage, power consumption, memory usage, and CPU usage) of their wireless nodes. It provides a breakdown of resource usage in the context of how much of resource is used for supporting a wireless node's own applications versus the usage for routing data of other network users in an ad-hoc wireless network. Using Lucent Wavelan/IEEE wireless network card (or Orinoco card), we have implemented WANMon on Linux platform. We have described detailed functionality and various features of WANMon in the paper. We believe that our prototype monitoring tool can provide Ad Hoc wireless network researchers and ad-hoc wireless network product developers with the means to develop a tool that will help them monitor the resource usage of their research projects or products in an ad-hoc wireless network environment.

## 6 Limitations and Future Work

To our best knowledge, WANMon is the first monitoring tool of its kind. It provides a means for Ad Hoc network users to closely monitor resource usage of their wireless devices. However, as a prototype monitoring tool, WANMon has its limitations. Firstly, WANMon cannot provide real-time information. This is because WANMon collects information over a period of time and then displays the resource usage over that past period of time. Secondly, there is a limitation on memory that WANMon can use to store resource usage counters. This may cause resource usage counters of WANMon to overflow. While memory overflow is considered as an extreme circumstance, it is however a necessary limitation of WANMon. Lastly, our prototype WANMon provides only rough CPU usage figures. The CPU usage statistic provided by WANMon is just an upper bound for the CPU usage involved in networking activities.

As a future work, we intend to improve the accuracy of CPU usage statistics. This requires major changes to the Linux kernel. The changes would involve modifications to the scheduler, such that whenever it is scheduling anything that is networking related it must account for the CPU usage. Another possible extension to WANMon is to provide resource usage for programs/processes that perform dynamic routing. We assume static routing in our prototype implementation of WANMon.

## References


[1] Main web site for tcpdump, http://ee.lbl.gov/

[2] Web site providing info about tcpdump patch site, http://www.tcpdump.org/

[3] Main web site for ethereal. http://www.ethereal.com

[4] Web page about Lucent Wavelan/IEEE or Orinoco Card, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Orinoco.html

[5] Laura Feeney and Martin Nilsson,"Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment", in Proceedings of the Conference on Computer Communications (IEEE Infocom), Anchorage, Alaska, April 2001.