WIRELESS COMMUNICATIONS AND MOBILE COMPUTING
Wirel. Commun. Mob. Comput. 2011; **11**:1182–1196
Published online 6 July 2010 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/wcm.908

SPECIAL ISSUE PAPER

MeshMon: a multi-tiered framework for wireless mesh network monitoring

Ramya Raghavendra*, Prashanth Acharya, Elizabeth M. Belding and Kevin C. Almeroth

Department of Computer Science, University of California, Santa Barbara, CA 93106, U.S.A.

ABSTRACT

Monitoring and troubleshooting a large wireless mesh network (WMN) presents several challenges. Diagnosis of problems related to wireless access in these networks requires a comprehensive set of metrics and network monitoring data. Collection and offloading of a large amount of data are infeasible in a bandwidth constrained mesh network. Additionally, the processing required to analyze data from the entire network restricts the scalability of the system and impacts the ability to perform real-time fault diagnosis. To this end, we propose MeshMon, a network monitoring framework that includes a multi-tiered method of data collection. MeshMon dynamically controls the granularity of data collection based on observed events in the network, thereby achieving significant bandwidth savings and enabling real-time automated management. Our evaluation of MeshMon on a real testbed shows that we can diagnose a majority (87%) of network faults with a 66% savings in bandwidth required for network monitoring. Copyright © 2010 John Wiley & Sons, Ltd.

KEYWORDS
Mesh networks, network monitoring, network management, network troubleshooting

***Correspondence**
Ramya Raghavendra, Department of Computer Science, University of California, Santa Barbara, CA 93106, U.S.A.
E-mail: ramya@cs.ucsb.edu

1. INTRODUCTION

Large scale IEEE 802.11 mesh networks promise to be a significant method of providing Internet connectivity in several cities and towns. In addition to these metro-scale deployments, wireless mesh networks (WMNs) have been proposed to provide connectivity in rural environments, especially in developing countries around the world. Such large scale mesh networks consist of hundreds to thousands of mesh routers and may be used by thousands of users. The presence of numerous wireless devices, including mesh routers and client devices, in a single administrative domain increases the complexity of the difficult task of managing these large scale mesh networks.

We believe the network administrator's ability to manage and troubleshoot these networks in real-time is a critical factor that contributes to the success of WMNs. These administrative tasks, however, present several new challenges compared to traditional wireline networks. In particular, the design of a network monitoring system is non-trivial because of the multi-hop architecture of these mesh networks and the inherent wireless-related properties of 802.11-based devices. For instance, the performance of the devices in these networks may be impacted by entities outside the network, i.e., the surrounding environment or devices that are not part of the network but share the frequency spectrum. In addition, the large number of proprietary protocols and algorithms used by different IEEE 802.11 client vendors and the interaction among these clients is not well understood. Unlike in WLANs, the backhaul links used for communication between mesh routers and the Internet Gateway consist of relatively low bandwidth multi-hop wireless links. Therefore, control traffic required for remote monitoring and administration of these mesh routers must be minimal, so as not to consume a significant portion of the available bandwidth. The unreliable nature of wireless links may result in gaps in the collected data, preventing the timely measurement analysis. Finally, unlike wired networks, the physical location of the mesh routers provides a strong spatial aspect to all data used in management and troubleshooting of mesh networks. Therefore, data from different routers that share spectrum in a geographical region may need to be analyzed in correlation with each other.

Although traditional infrastructure WLANs present similar monitoring challenges and requirements, network monitoring solutions developed for WLANs cannot be directly applied to WMNs. Most monitoring solutions for commercial WLANs only use a small fixed subset of the large set of available metrics to minimize the data collection and processing overhead. This approach may fail to capture data needed to diagnose a detected problem. Previous research has shown that the diagnosis and root cause analysis of many network faults requires a complete trace of the packets in the network [1,2]. Unfortunately, the capture and remote analysis of all data packets is infeasible in a mesh network as the bandwidth requirements are prohibitive. Further, monitoring systems that use a large set of metrics (or detailed packet traces) require resource intensive computation and thus may be unsuitable for real-time identification and remediation of problems. From our own experience in the development of a real-time network visualization tool, we found that the speed of metric collection/generation, rather than visual rendering of the data, is the computational bottleneck [3].

For the above reasons, there is a need for a methodology of monitoring and metric collection in WMNs that is bandwidth-efficient, scalable with respect to the number of devices in the network, and able to provide a comprehensive set of metrics that can be used to identify all problems in the network. Such a solution would facilitate centralized administration of a large network and also enable the use of tools, such as network visualization, to monitor the network health in real-time.

In this paper, we present MeshMon, a network monitoring framework that enables real-time identification and troubleshooting of problems in WMNs. A key observation that guides the design of MeshMon is that comprehensive metric collection is required only when there are problems in the network. A small subset of these metrics, called baseline metrics, is sufficient when the network performance is satisfactory, and can be used for coarse identification of potential problems. We propose a stateful method that intelligently adapts the metric collection process to capture the most relevant set of metrics. When the baseline metrics indicate the possible presence of a problem, the system transitions to collect a more detailed set of metrics. The goal of this methodology of metric collection is to reduce the volume of data that needs to be collected and processed without sacrificing the ability to diagnose problems in the network.

For the collection of metrics in a WMN, we incorporate our idea of dynamic and scalable hierarchical metric collection in a WLAN [4]. In our previous work, we proposed the basic idea of hierarchical metric collection and presented a simple feasibility analysis for the scheme in WLANs. In this work we develop our idea in the context of mesh networks. Mesh networks offer additional complexity because a monitoring system should address problems that affect mesh routers as well as those that affect client devices. Therefore, MeshMon incorporates metrics associated with mesh routing and connectivity into the hierarchical metric collection, in addition to metrics associated with client devices. Our design ensures that even in situations where a problem scenario is reflected in both sets of metrics (mesh related and client access related), MeshMon can successfully isolate the root cause of the problem. The bandwidth constrained environment of WMNs requires a shift from our previous centralized monitoring approach to a hybrid (partly distributed and partly centralized) approach.

Our contributions in this work are as follows:

- We present a classification of WMN metrics in a hierarchical structure to assist in automated fault diagnosis.
- We present the detailed design, implementation, and evaluation of the entire MeshMon system.
- We have implemented a prototype of MeshMon on the UCSB MeshNet testbed.[†] The prototype system is capable of identification and diagnosis of a variety of common problems that occur in WMNs. Our evaluation of MeshMon indicates that we diagnose 87% of the problems with a 66% reduction in the bandwidth required for monitoring data.

The remainder of the paper is organized as follows. Section 2 surveys related work. We present the motivation for a hierarchical monitoring solution through measurement analysis in Section 3. A detailed description of the design of MeshMon is provided in Section 4. Section 5 discusses the implementation and evaluation of our system. We conclude the paper in Section 6.

2. RELATED WORK

Wireless mesh network management has surprisingly received little research attention. The only work in the research community of which we are aware is a trace-driven approach mesh network troubleshooting approach by Qiu *et al.* [5]. This paper makes the case that troubleshooting multi-hop wireless networks is challenging, and presents a diagnostic system that employs trace-driven simulations to detect faults and perform root cause analysis. This approach is applied to diagnose performance problems caused by packet dropping, link congestion, external noise, and MAC misbehavior. While this work shares our goal of automating fault diagnosis, the problem set that considered is significantly smaller than what we consider here. We account for problems that can occur in the client tier as well as problems unique to a multi-hop network. Finally, the trace-driven simulation approach inherently restricts the real-time capabilities of their system.

Network management, health monitoring, and fault diagnosis in WLANs has been an area of active research in recent years. Meng *et. al.* [6] describe their work on building a packet capture tool for wireless LANs that can detect unauthorized users. The authors also use the concept of hierarchical monitoring to enforce accountability in wireless LANs by building multi-level, multi-resolution trace files of events and designing a flow net methodology [6]. MOJO [7] is an 802.11 troubleshooting system that outlines the importance of detailed physical layer metrics for problem diagnosis and demonstrates that many higher layer symptoms are manifestations of problems at the PHY layer. Adya *et al.* [8] propose modifications to all clients in the

[†] http://moment.cs.ucsb.edu/meshnet

network to assist in troubleshooting. APs in the network act as 'software' sensors by capturing wireless metrics and thereby avoid the cost of deploying special sensors. The authors also propose that clients associated with APs can act as a conduit for diagnostic traffic from clients not associated with APs. WiFiProfiler [9] uses a similar client conduit approach but troubleshooting can be done in a peer–peer fashion. However, given the heterogeneous client devices, instrumenting all of them may not be possible. Our solution does not assume any assistance from client devices.

Jigsaw [1] is a comprehensive fault diagnosis system that uses a large set of dedicated wireless radio monitors to observe and record every transmission in a WLAN. The radio monitors send the captured packet trace to a central repository where the packet traces are merged to produce a single time-synchronized trace that provides a detailed view of the sequence of events. The Jigsaw system was later extended to provide automated cross-layered diagnosis of problems [2]. Although Jigsaw provides a complete view of the events in the network, it requires high overhead in terms of infrastructure. The dedicated wireless radio monitors require a backhaul network connection that consumes roughly five times the actual network traffic [2]. The high bandwidth requirements for Jigsaw make it unsuitable for a multi-hop mesh network. Additionally, the scaling properties of the trace merging process in a larger or heavily used network are not clear.

In addition to work from the research community, there are several commercial network management tools [10–12]. The proprietary nature of these tools restricts the available information to feature-sets. Based on the available documentation, we hypothesize that some tools use high level metrics accessed through SNMP MIBS [10] and other tools such as AirMagnet [12] use special radio monitors deployed throughout the network to collect packet traces.

MeshMon is a framework that determines the metrics to be collected by the health monitoring system based on the current state of the network. This methodology of metric collection ensures that the system collects the appropriate level of details for effective fault diagnosis. In effect, MeshMon may transit from a minimal set of metrics to a level that provides the same comprehensive details as Jigsaw.

3. A CASE FOR HIERARCHICAL MONITORING

Modern enterprise networks are of such complexity that even simple faults can be difficult to diagnose [2]. A problem occurring on the wireless network could arise for a variety of reasons, including RF interference, link-layer variability, dynamic addressing and authorization, incorrect VLAN setup, as well as the myriad of complexities of the wired network itself. While infrastructure WLANs present complexity, the task of troubleshooting is further daunting when the backhaul is a multi-hop wireless link. Apart from the challenges of a single hop wireless network that persist in a mesh network, additional complexities of a multi-hop network, such as routing, traffic flows, link congestion, and interference from neighboring networks and devices, must be addressed.

Network management systems are built to aid the administrator in the task of troubleshooting networks. These systems constantly monitor the network and report a set of metrics [10–12] that they are configured to monitor. However, they face the drawback that when faults occur, all information necessary to diagnose the root cause is not available. The range of interactions and the lack of sufficient information often results in manual diagnosis.

Recent research has shown that automated fault diagnosis is possible by using detailed link layer traces [1,2]. While fault detection systems that take link layer traces as their input are able to achieve better diagnosis and automation capabilities, they suffer from the drawback of imposing high overhead. These systems are reported to collect up to 100 GB of monitoring data, five times as much as the data traffic, over a period of 24 h. While a high bandwidth Ethernet connection can sustain such overhead, a mesh network that has wireless links on the backhaul typically lacks the needed bandwidth.

In this paper, we propose a light-weight fault detection system for mesh networks that does not incur the high overhead of existing automated solutions, yet can provide the detailed metrics when problem occurs. Our solution is based on the central idea that faults occur intermittently and networks are well functioned a majority of the time. When a network is functioning well, little monitoring overhead is necessary. However, when a fault occurs, more monitoring data need to be collected in order to diagnose the cause of the fault. A monitoring system that adapts its monitoring granularity can benefit significantly in terms of bandwidth savings and reduction of complexity for the administrator.

To justify an adaptive monitoring system, the question that we would like to investigate is whether faults occur intermittently. Ideally, we would like to analyze a real mesh deployment, but because of the lack of any public mesh network traces, we restrict our analysis to WLAN traces and mesh testbed experiments. We analyze traces from a large scale conference network to isolate the times when the network performance can be deemed as *faulty*. We follow this with baseline experiments on a mesh testbed deployment and perform similar analysis.

3.1. Trace Description

We use data we collected from the 67th IETF meeting held in San Diego in November 2006. The IETF network consisted of 55 Cisco and D-Link Access Points (APs), spread across the conference hotel. Thirty-eight APs were installed in the conference rooms and each device was equipped with one 802.11a and one 802.11b/g radio. The meetings were held in two separate sessions, the day and the late evening sessions, the latter of which is also called the *plenary*. We used the *vicinity sniffing* technique to collect data from the MAC layer [13,14]. We monitored the network for 4 days of the conference during both the day and plenary sessions. Details about the trace collection methodology that are relevant to the analysis presented in this paper are outlined below.

Day session: The day sessions were held between 09:00 and 17:30 h from 6–10 November 2006. The day session was divided into six to eight parallel tracks, each of which was held in one of the conference rooms. During the day sessions, we collected usage statistics in the beginning of the day and ranked the APs based on the number of users associated with each. We configured the sniffers to monitor the top 12 ranked APs for the entire day. Some of the sniffers were thus configured on the 802.11g network, while the rest were on 802.11a, depending on the usage.

Trace analysis: We analyze the traces from all the day sessions. Each parallel track had between 30 and 100 users, with an average of 15 users per AP. The medium utilization during these sessions is plotted in Figure 1(a). The *x*-axis in the figure represents the airtime or the percentage of time that the medium was utilized by transmissions. The *y*-axis represents the percentage of time for each airtime bin on the *x*-axis, computed as an average over 1 min intervals. The percentage values are computed over the span of all day sessions during the 4 days. We only include non-zero airtime values in the plot. As can be seen from the figure, the network was not highly utilized, with only 10% utilization up to 60% of the time. The network was *moderately congested* for about 25% of the time and *highly congested* for only about 1% of the time [14].

We now consider the performance and connectivity conditions of this network. To characterize the performance, we study the loss rates. Figure 1(b) shows the CDF of the loss percentage for the entire trace period. Up to 50% of the time, the loss rate was under the 5% threshold that has been used for fault detection in prior research [8]. Only about 15% of the time was over 10% loss experienced.

We next analyze the amount of 802.11 management overhead traffic that clients experienced in this network. These include the probes, association and authentication requests and responses. The overhead index is defined as the number of management packets that are transmitted for every byte of data transmitted. The overhead index provides a sense of connectivity issues that exist in the network. Figure 1(c) plots the average overhead index for the entire trace duration. The *x*-axis in the figure represents the per minute average of overhead across all the clients. The overhead indices are ordered in increasing order for visual clarity with exceedingly low values omitted. From the figure, we see that clients experienced low overhead a majority of the time; only for about an hour was the overhead index higher than 1.

The results suggest that networks experience faulty conditions for only a fraction of time and function normally in the typical case. Intuitively, networks function well for a majority of the time since large scale networks are deployed with careful planning to deliver a baseline performance. The frequency of network faults depends on a variety of factors involving deployment and usage. However, we expect similar trends in any planned wireless network. A fault detection system that takes advantage of this fact can benefit significantly by reducing the monitoring overhead. Such a system would collect a few metrics that ensure the network is functioning well, and when problems arise it would increase the granularity to collect those metrics that could identify the potential problem.

To verify that the observations from a WLAN analysis also hold in a mesh network, we conduct experiments on the mesh network at UCSB. A brief description of the testbed, the experiments, and results is presented.

3.2. Testbed Description

We start with a brief description of the mesh testbed. All experiments described in this paper were conducted on the UCSB MeshNet, an indoor wireless mesh testbed that consists of 15 wireless nodes [15]. All nodes in the testbed use 802.11a/b/g cards based on the Atheros chipset. The nodes use the IEEE 802.11b/g CSMA protocol for medium access control and SRCR [16] is used as the routing protocol. Several nodes in the testbed have multiple radios: one for the mesh tier and the other for client access tier. We configured each of the radios on orthogonal 802.11b/g channels. The nodes use Linux (kernel version 2.4) as their operating system. We use the open source MadWifi driver v0.9.2 to control the cards. RTS/CTS is disabled for all the radios.

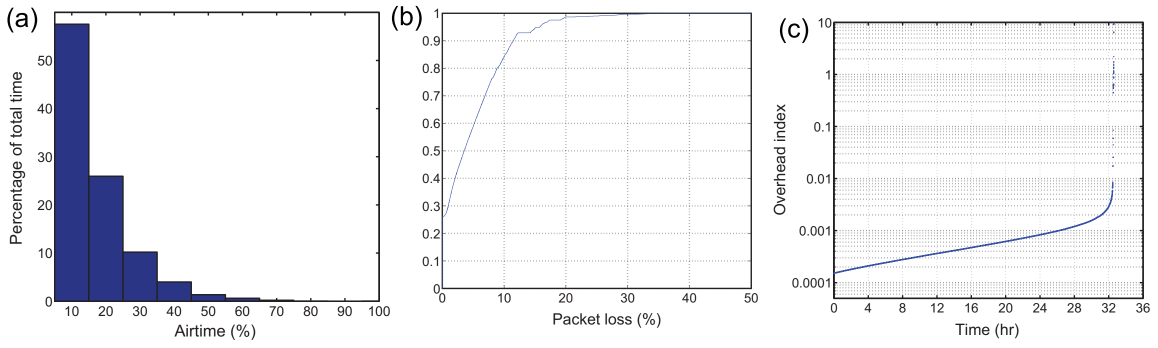

Figure 1. Analysis of WLAN traces from 67th IETF wireless network. (a) Medium utilization; (b) packet loss; (c) overhead index.

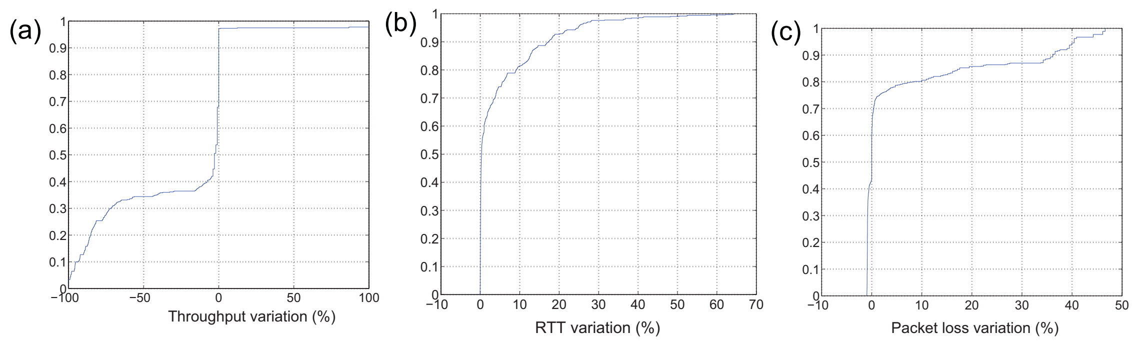

Figure 2. Study of throughput, RTT and, loss variations in UCSB MeshNet. (a) Variation of throughput from the highest 90th percentile; (b) variation of RTT values from the lowest 90th percentile; (c) variation of packet loss percentage from the lowest 90th percentile.

The nodes are placed in different locations on three floors of the building. The testbed coexists with an 802.11g wireless LAN that provides Internet connectivity throughout the building.

3.3. Baseline Experiment

Using constant rate flows, we perform experiments on the mesh network to study the variations in throughput, loss, and delay values.

The experiment consisted of eight client laptops, placed at different locations and connected to the mesh network on the client access tier. The laptops were used to run eight UDP sessions of constant 1 Mbps bitrate with the gateway. The corresponding receiver for each session was the gateway node. Since the clients were connected to the gateway through a variable number of hops, the average throughput achieved varied from 1 Mbps to 300 kbps. The flows were measured over a period of 4 h and the achieved throughput, RTT, and packet losses were measured over every 1 min interval during this period.

Figure 2 shows the results from the experiment. The x-axis in each graph represents the variation, which we define to be the difference between the observed value and the 90th percentile value of that metric. We observe that in the mesh network, similar to the WLAN, the problems were restricted to a small percentage of time. For instance, in Figure 2(a), we see that about 50% of the time the nodes achieved the 90th percentile throughput, and up to 70% of the time achieved throughput within 50% variation. Similarly, from Figure 2(b) and (c) we see that up to 60% of the time, the nodes had the no loss variation and up to 70% of the time they no delay variation.

These instances clearly indicate that much can be gained from having an intelligent monitoring system that adapts the monitoring granularity based on the network conditions. To this end, we design MeshMon, a light-weight mesh monitoring system that runs on the mesh nodes and provides real-time fault detection capabilities. We design such a system using a hierarchical structure of metrics and demonstrate its usefulness in monitoring the MeshNet.

4. DESIGN OF MeshMon

In this section we first present the network architecture in which MeshMon operates. We then present a brief overview of the design philosophy, followed by a detailed description of the design. Some aspects of the baseline design were first presented in our previous work [4]. For the sake of completeness, we outline those details again in this paper, and describe the evolution of our design for mesh networks.

4.1. Network Attributes

Our solution is designed for a multi-hop IEEE 802.11-based mesh network. We assume that each mesh router is equipped with two radios—one used for the backhaul connectivity to the Internet and the second radio as an AP that services client 802.11 devices. This network architecture of separate client access layer and network backhaul layer is commonly used by several commercial mesh router manufacturers.[‡] Some networks from the research community (e.g., MIT Roofnet[§] [16]) also adopt this logical separation of network backhaul and client access layers. Traffic to the client devices is routed, using the backhaul layer, from the gateway to the mesh node to which the client is associated. The client access radio at this node is then used to transmit the traffic to the client device. We believe that minor modifications to the current design of MeshMon, specifically to the decision tree (presented in Section 4.6), will also enable its use in single radio mesh networks wherein both backhaul and client access layers use the same radio. The challenge in such single radio systems is the isolation of the source of a problem due to the complex interactions between the two network layers. We do not address such networks in this paper.

All nodes in the mesh network run MeshMon and communicate with a central controller. The central controller

[‡] Strix Systems - http://www.strixsystems.com, BelAir Networks-http://www.belair.com, FireTide - http://www.firetide.com

[§] For MIT Roofnet, the client access layer uses Ethernet instead of the second 802.11 radio.

performs the following functions: it collects data from the mesh nodes and stores the data in a database; it issues commands to mesh nodes to control the data collection; and it provides data to the network administrator. The data collected by the controller may also be accessed by a network health monitoring tool such as SCUBA [3], and can be used for automatic rule-based remediation of problems.

We assume the client devices to be autonomous and largely outside the control of the network administrator. As part of our future work, we intend to incorporate statistics collected from cooperative client devices, i.e., devices that can communicate with the MeshMon controller. Currently, we restrict the focus of the metric collection system to the wireless access part of the network and do not consider metrics from high layers (e.g. events from DHCP, DNS queries).

4.2. Design Rationale

The basic idea in the design of MeshMon is to use a few baseline metrics that capture the general health of the network. When problems are detected, the system intelligently increases metric collection to capture only those metrics that are needed to diagnose the root cause of the problem. The principle behind the design of such a system is that in the general case networks are in a stable state, during which time it is sufficient to have a light-weight monitoring system. On the other hand, when a problem arises, collection of detailed metrics in the area where the problem is detected can facilitate fine-tuned problem diagnosis.

In the design of MeshMon, we use the concept of tiers of metrics, wherein each tier collects a level of detail more than the previous level. The system goal is to diagnose the network problem at the lowest possible tier, i.e., with the minimum level of detail necessary. When diagnosis cannot be made with certainty at a particular tier, the next tier is triggered to collect more metrics. The biggest challenge in designing a multi-tiered metric collection system is to identify the metrics that are necessary and sufficient for making decisions at each tier for the particular problem set that the system should handle. The classification of metrics into tiers is presented in Section 4.6.

One design consideration is whether to make the monitoring system centralized or distributed or use a hybrid control mode. The intelligence to transition the metric collection among the different tiers can be either at the central controller or at the mesh nodes. The latter option provides a distributed approach that scales better as the size of the network increases. The metric collection process is more responsive to local events. Further, a distributed approach can reduce monitoring and related control traffic. On the other hand, the central controller has a global view of the network and may be able to correlate symptoms of nearby mesh nodes. This ability to detect non-localized problems is critical in a mesh network. For example, an event such as high traffic load at a mesh node close to the gateway would impact the performance of other nodes routing through it. Therefore, we choose to use a hybrid model of control wherein the mesh nodes attempt to diagnose the problem locally when possible, and in other cases communicate with the central controller to coordinate metric collection among the various mesh nodes.

Even in situations where the mesh node can diagnose the problem locally, we send the metrics associated with the diagnosis to the central conroller. This data is stored in the central database and used for audits and post-mortem analysis. The transmission of this data is considered optional; however, the system currently does not support automated remediation schemes. We therefore believe that the data associated with the problem diagnosis is essential for the network administrators in order to fix the problem.

Next we describe the system architecture of MeshMon, followed by the classification of metrics into tiers and the rules/triggers that govern the transition of the metric collection among the different tiers.

4.3. MeshMon Architecture

Figure 3 illustrates the architecture of the MeshMon system. At each mesh node, the MeshMon system is comprised of three main components—the monitoring engine, the analysis engine, and the communication engine. The monitoring engine takes as input a list of metrics to collect. The list of metrics to collect depends on the current metric collection tier for the node. The collected metrics are output to the analysis engine and also to the communication engine.

The analysis engine is responsible for generating the list of metrics to collect at the node. For this purpose, the data collected by the monitoring engine are processed using a repository of rules. The rules are specifications of network conditions or events that trigger a transition in the metric collection tiers. Each rule represents the possible presence of a problem in the network and is usually described in terms of conditional statements that compare current metric values against pre-determined threshold values. Since each rule is inherently associated with a tier of metric collection, the set of rules form a structure similar to a decision tree. Section 4.6 provides a detailed discussion of the rules and the decision tree. A second output of the analysis engine is the diagnosis of faults in the network. Based on the problem hypothesis presented by the rules and the corresponding metrics, the analysis engine can perform root cause analysis in the network and suggest potential remedial actions.

The third component of the system is the communication engine and is responsible for communication between the node and other nodes in the network, including the gateway node. For reasons outlined in Section 4.2, the metrics collected by the monitoring engine are sent to the central controller via the gateway. In a mesh network, the problem diagnosis may require metrics from neighboring, upstream or downstream mesh nodes, in addition to metrics collected locally. The communication and actions required to trigger this non-local metric collection are handled by the communication engine.

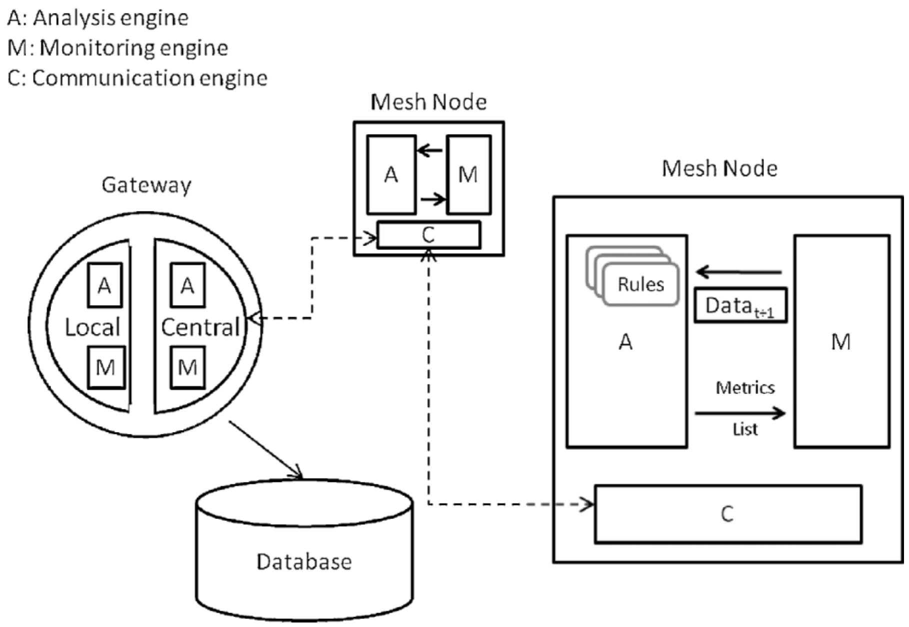

Figure 3. Architecture of MeshMon showing mesh nodes and gateway. MeshMon runs on each mesh router.

4.4. Baseline Metrics

In order to characterize the performance of wireless networks, a plethora of metrics have been studied: throughput, medium utilization, control overhead, loss rates, retransmissions, data rate, and received signal strength are all good candidates. When a change occurs in the network condition, it is often reflected in one or more of these metrics. Conversely, when the value of a metric changes, several other metrics are needed to find the problem that caused the change.

To select a baseline set of metrics, we consider the typical performance goals of a mesh network [17]. Broadly, there are two goals that a WMN tries to achieve: (1) provide connectivity to clients within the network's coverage area and (2) ensure high quality routes to the gateway. We note that ubiquitous coverage is a goal during the deployment phase of a network; we are concerned with detecting performance issues during post-deployment operation. Therefore, the ultimate objective is to ensure clients are able to connect to the network and obtain good performance from the mesh network.

These objectives lead us to three baseline metrics: maximum client overhead index ($O_{\max}$), load-aware WCETT (L-WCETT), and minimum client throughput ($T_{\min}$). The overhead index is defined as the ratio of control and management traffic (in bytes) to data traffic (in bytes) [18]. When a client has connectivity problems, $O_{\max}$ will be high. The second metric, L-WCETT, provides a measure of the mesh performance. As we describe later in Section 4.5, L-WCETT at a mesh router is closely related to the mesh path throughput achieved between the mesh router and its gateway. A high value of L-WCETT indicates reduced path throughput for a mesh router. The third baseline metric, $T_{\min}$, tracks the performance of connected clients. When a client obtains low throughput, $T_{\min}$ will be low.[‖] We piggyback these baseline metrics on a heartbeat message sent by each mesh node to the central controller. The baseline metrics can be tailored based on the goals of the network. If one of the objectives of the network is to support Voice-over-IP applications, then low packet delay is another goal of the network. In such networks, packet delay would be the fourth baseline metric. In this paper, we consider networks that are unaware of specific traffic type.

We believe that these metrics are sufficient at the high level to detect a network problem. Three of the most important problems related to wireless network access are connectivity problems, performance problems, and authentication problems [8]. Connectivity and authentication problems result in high $O_{\max}$, whereas performance problems result in a high L-WCETT, a low $T_{\min}$ or both. In a mesh network, a problem that is manifest on the client access tier could be either because of a faulty condition on the client access tier or the mesh backbone.[¶] While performance problems manifest in a variety of other metrics, such as data rates, loss rates, and signal strength, these metrics can be used to provide a deeper understanding of the cause of low throughput and thus are not first tier metrics.

[‖] In order to distinguish clients with little or no offered load, we only consider active clients (defined by a minimum activity threshold in bytes transferred) for computation of $T_{\min}$.

[¶] This challenge remains in case of a single radio mesh as well, wherein both the tiers are on a common frequency, but the root cause of the problems is disparate.

4.5. Metric Collection

A large number of metrics such as throughput, goodput, packet loss, delay, signal strength, ETX, and ETT are available to indicate the performance of the network. These metrics serve as indicators to potential problems in different layers of the network stack. In this section, we present the metrics that are monitored in MeshMon, along with an outline of the implementation description.

Load-aware WCETT (L-WCETT): The end-to-end throughput from a mesh node to the gateway is the path throughput for the node, and serves of the indication of the mesh performance. Path throughput cannot be measured directly since a node does not have information about the packet reception at the other nodes. Hence we need a metric that a mesh node can access, and that has a good correlation with the path throughput.

One likely metric for this purpose is round trip time (RTT). Prior work has shown that a correlation exists between RTT and TCP throughput [19]. In order to verify that the relation holds in a WMN as well, we conducted TCP tests on the UCSB Meshnet. The experiment consists of saturation throughput measurements between 10 mesh nodes to the gateway. We ran 5 min TCP sessions upstream and downstream between each of the 10 mesh nodes and the gateway using the *iperf* utility and measured the end-to-end throughput and RTT. Figure 4 shows the average throughput and RTT over 20 s intervals. We observe the relationship between RTT and throughput closely follows the model described by Padhye *et al.* [19] and conclude that RTT can indeed serve as a good indicator of throughput.

The measurement of RTT, however, requires transmission of probe packets and proactive measurement of RTTs. Probes can be avoided through the use of a routing metric such as WCETT [20] as a measure of path throughput. Since the mesh backbone is on a single channel, we use the definition of WCETT without the channel diversity, as given by:

$$\mathrm{WCETT} = \sum_{i=1}^{n} \mathrm{ETT}_i$$

Here, n is the number of links in the path. WCETT was originally designed to be load independent, since the routing algorithm should be resilient to route flapping. However, for the purpose of detecting network faults, we would like to be able to account for the throughput degradation due to network load. We do this by adding to WCETT the queuing and contention delays along the path. We call this metric load-aware WCETT (L-WCETT). Figure 4 shows the variation of throughput with L-WCETT for the same experiments. We observe that L-WCETT has a high degree of correlation with RTT. The degree of correlation is high (linear correlation coefficient of 0.96) in the region where throughput is less than 4 Mbps, which is a region of interest in the diagnosis of performance problems. Therefore, we choose to use L-WCETT, a metric that can be calculated locally in collaboration with the routing module, to model the backhaul layer throughput. Note that L-WCETT is used only by MeshMon; the routing algorithm continues to use the WCETT metric.

Airtime: We use the term Airtime (A) to denote the fraction of time that a node senses the medium as unavailable for transmission. This includes the time during which the channel is busy, and the additional wait time the 802.11 MAC spends for the DIFS, SIFS, and backoff periods. High airtime is an indication of high transmit or receive times at a node due to high traffic or high contention. The airtime metric is thus a good indicator of medium congestion.

Loss: Loss (L) is used to indicate the MAC layer losses. To compute loss, we modify the driver to report the number of packets that are retransmitted at the MAC layer. The loss rate of a link is used to diagnose problems such as congestion and poor signal strength.

Signal strength: Signal strength (S) is used to refer to the RSSI measurements reported by the card.

Transmit rate: Transmit rate (R) is the data rate or the modulation rate at which a packet is transmitted.

Overhead index: Overhead index (O) is the number of management frames that a mesh node transmits. Management frames include association and reassociation responses, probe responses, disassociation, authentication, and deauthentication frames. Management overhead (M) is used to represent the overhead computed for each management frame type.

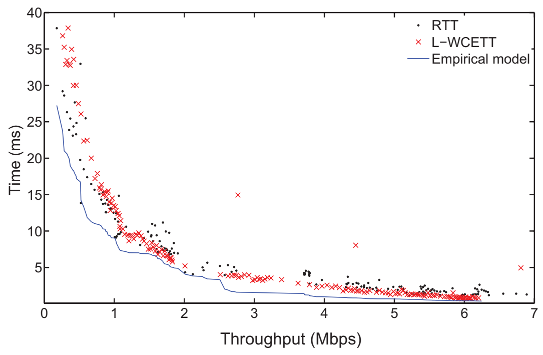

Figure 4. Measure of throughput as a function of round trip time (RTT) and load-aware WCETT (L-WCETT). Throughput and RTT are measured by saturating the link using *iperf* measurements. L-WCETT is computed as the sum of the ETT measurements and the queuing delays along the path.

4.6. Decision Tree

The set of metrics to collect at each tier, together with the rules that trigger the transition between tiers, can be visualized as a decision tree. A rule is essentially a threshold check to look for the manifestation of a network problem, indicated through one or more metrics.

Based on the network model described in Section 4.1, we note that the operation of the backhaul layer and the client access layer of the nodes are largely independent of each other. In other words, although the symptoms of a problem in the backhaul layer may be reflected in metrics associated with the client layer (and sometimes vice-versa), the diagnosis of the problem does not require detailed client layer metrics. Therefore we use two logical decision trees—one for the client access layer and one for the network backhaul layer. Both these trees are active simultaneously and the tiers of metric collection may be different in each depending on the problem suspected.

Algorithm 1 Client layer decision algorithm.

```
Parameters: $T_t$,$S_t$,$L_t$,$A_t$
*Current Set* = {}
—— *For each monitoring cycle do:*
**Tier 1:** *Current Set* = {$T_{min}$, $O_{max}$}
**if** ($T_{min} < T_t$) **then** go to Tier 2
**Tier 2:**
    **if** $T_{min} < T_t$ **then** *Current Set* += {$A$, $L$}
    **else if** $O > O_t$ for each client $i$ *Current Set* += {$O$, $O_i$};
    **if** $A < A_t$ **and** $L < L_t$ **and** $O < O_t$ **then** go to Tier 1
    **else** go to Tier 3
**Tier 3:**
    **if** $A > A_t$ **then** for each client $i$ *Current Set* += {$A_i$}
    **if** $L > L_t$ **then** for each client $i$ *Current Set* += {$L_i$}
    textbfif $O_i > O_t$ **then** for each client $i$ *Current Set* += {$M_i$}
    **if** $\sum_{i=0}^{n} A_i < A$ **then EXTERNAL INTERFERENCE**
    **if** for all clients $i$ $A < A_t$ **and** $L < L_t$ **and** $O_i < O_t$
    **then** go to Tier 2

**else** go to Tier 4
**Tier 4:**
    **if** $\sum_{i=0}^{n} A_i > A_t$ **then** for all clients $i$ *Current Set* += {$R_i$, $S_i$}
    **if** $M_i > M_t$ **then**
        **if** Association responses high
        **then ASSOCIATION PROBLEM**
        **else if** Auth/Deauth messages high **then AUTH PROBLEM**
    **if** for all clients $i$ $S_i > S_t$ **and** $R_i < R_t$ **then CONGESTION**
    **else if** for all clients $i$ $S_i < S_t$ **and** $R_i < R_t$ **then POOR LINK**
    **else** Packet traces
—— *On diagnosis, go to Tier 1*
```

Note that the tiers represent a logical organization of the various metrics for the purpose of diagnosing a fault. In our implementation of MeshMon, the state of metric collection is represented by *Current Set*. This set is obtained as union of all the metrics in all the active paths. An active path is the path in the decision tree based on the current hypothesis of a problem. If more than one problem is suspected, there may be more than one path active simultaneously. This enables MeshMon to detect multiple concurrent problems. Another scenario wherein multiple paths may be active at once is as follows. A mesh router may be collecting metrics in the mesh layer decision tree, under the direction of the central controller, to diagnose a non-local problem. At the same time, a problem in the client access layer at the mesh router itself would trigger metric collection in the client access decision tree.

Figure 5 presents the visual representation of the hierarchy tree. Each of the metrics is described in Section 4.5. In the normal state of network operation, MeshMon operates in Tier 1 and the Current Set is comprised of the baseline metrics. The rules for transition between tiers and the state of the Current Set in each of these tiers are outlined in Algorithm 1 for the client access layer and Algorithm 2 for the mesh access layer.

When a baseline metric for the mesh layer crosses its threshold, only the mesh layer decision tree is activated. However, when a problem is detected in the client layer, both the client access tree and the mesh access tree are activated, since the fault could lie in either tier. If the problem lies in the client layer, all information required for diagnosing is present locally and hence fault diagnosis occurs on the mesh node. However, when the fault lies in the mesh layer, a mesh node attempts to locally diagnose a problem. If unsuccessful, it contacts the gateway, which in turn will turn on the diagnosis on the other mesh nodes on the node's upstream path.

As an example, consider the scenario where a mesh node's throughput has dropped below the threshold because a mesh router further up the route to the gateway is congested. Tier 2 of metric collection is triggered to collect the node's local airtime and ETX metrics. Both these metrics would not indicate the problem that lies upstream. At this point, the mesh router triggers the gateway to initiate centralized diagnosis, and the gateway triggers the collection on each mesh router along the node's upstream path. Congestion will be detected by the gateway since

$$\sum_{i=0}^{n} \mathrm{LETT}_i < \mathrm{LETT}_t$$

where LETT_i is the per-link load aware ETT (L-ETT) value along the path and L-ETT_t is the threshold computed. This is shown in Algorithm 2.

4.7. Choosing Parameters

We now discuss the selection of the two parameters that influence performance of the monitoring system: threshold values and periodicity of monitoring.

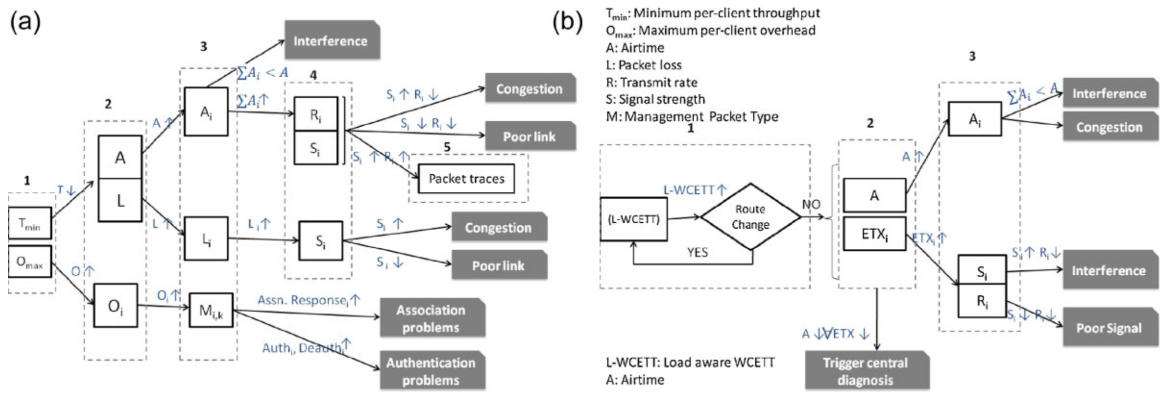

Figure 5. Multi-tiered metric collection decision tree implemented in the analysis engine. The numbers at the top indicate the tier of metric collection. White boxes represent the metrics collected at each tier. Arrows indicate the triggers used to transition between tiers. Black boxes indicate the fault diagnosis. (a) Client layer decision tree. (b) Mesh layer decision tree.

Algorithm 2 Mesh tier decision algorithm.

```
Parameters: $lwcett_t$, $lETT_t$, $S_t$, $R_t$, $A_t$
*Current Set* = {}
—— *For each monitoring cycle do:*
**Tier 1:**
*Current Set* += {*lwcett*}
    **if** $lwcett < lwcett_t$ **then**
        Has there been a route change?
        **if** yes **then** go to Tier 1
        **else** go to Tier 2
**Tier 2:**
    *Current Set* = {$A$, $ETX$}
    **if** $A < A_t$ **and** $ETX < ETX_t$ **then** go to Tier 1
    **else if** $P < P_t$ **and** $A < A_t$ **and** $ETX < ETX_t$
    **then TRIGGER CENTRAL DIAGNOSIS**
    **else** go to Tier 3;
**Tier 3:**
    **if** $A > A_t$ **then** *Current Set* += {$A_i$}
        **if** $\sum_{i=0}^{n} A_i < A$ **then INTERFERENCE**
        **else CONGESTION**
    **if** $lETT_i > lETT$ **then CONGESTION**
    **if** $\sum_{i=0}^{n} lETT_i > lETT_t$ **then***Current Set* +=
{$S_i$, $R_i$}
    **if** for all links $i$ $S_i > S_t$ **and** $R_i < R_t$ **then INTER-
FERENCE**
    **else if** for all links $i$ $S_i < S_t$ **and** $R_i < R_t$ **then
POOR LINK**
—— *On diagnosis, go to Tier 1*
```

Thresholds: An important aspect of the decision tree is the choice of thresholds used in the triggers. Thresholds can be of two types: static thresholds and dynamic thresholds. A static threshold is one that does not change with the network usage or condition, and is used for a metric that has a well-defined threshold value. An example is throughput; the network has a minimum throughput goal that should be met at all times. A dynamic threshold is one that is computed online by the system and is used for a metric that does not have a well defined threshold. Examples include signal strength and transmit rate. For these metrics, we track the history and dynamically compute a threshold. For every monitoring cycle, the new value of the metric is compared against this dynamic threshold to check whether the value has degraded by a statistically significant quantity, indicating a potential problem.

The static metrics in our design are throughput and overhead index. We derive some thresholds from the network goals, e.g., minimum throughput for a connected client is obtained from the network deployment goal. For the testbed evaluations, we fixed this threshold to be the minimum average per-client throughput achieved during controlled throughput measurements in the testbed. Prior research [18], Reference [21] has guided us to arrive at a threshold for the overhead metric. Clients operating in highly congested networks are shown to have high overhead of 1–1000. Since we want to identify connectivity problems, we fix the threshold for overhead index to 1.

For the remaining metrics, we compute the thresholds dynamically to distinguish between normal and anomalous network conditions. The metric values are summarized using the exponentially decaying moving average (EDMA) statistics. EDMA has been used in the past to dynamically study temporal sequences [22,23]. This method allows us to store only four floating point values and derive from these values the number of data points collected, their average value, and their variance (the square of the standard deviation). The technique to compute the EDMA is outlined in the Appendix.

A desirable property of MeshMon is to have minimal false negatives in problem identification, i.e., we do not want to miss detection of a fault. On the other hand, too many false positives (i.e., transitions to collect detailed metrics when there are no problems in reality) will generate a large number of alerts and reduce the usefulness of the system. While the EDMA technique facilitates tracking a sudden change in the metric by computing thresholds online, it is not completely resilient to transient spikes. However, the hybrid architecture of MeshMon is such that the only result of a false positive will be to collect metrics on the next tier. Since the decision making is local, no bandwidth overhead is incurred and the only result of a false positive is to send an alert . For the mesh layer, however, false positives should be avoided since they can potentially turn on centralized diagnosis. We can easily overcome this by introducing a dampening factor to the EDMA model. We introduce a small window of k points and compare the behavior in this window to the EDMA model for the preceding points.

Periodicity: In contrast to a system such a Jigsaw [1] that captures every packet transmission, MeshMon works mainly on a statistical view of the network. Thus MeshMon may be unable to capture transient conditions in the network. Instead it focuses on more persistent problems. The response time of MeshMon depends on the time granularity of metric collection. A smaller period of collection makes the system more responsive to temporary conditions in the network but increases the bandwidth requirements and the workload of the central controller. A large window saves bandwidth but may cause the system to miss some network faults. In our initial design we chose to collect metrics every 5 s. We believe this value provides a balance between transient fault detection and system responsiveness.

5. EVALUATION

In this section, we evaluate the fault diagnosis capability of MeshMon. The system prototype implemented on the mesh testbed is described below. We evaluate the system under two scenarios, each consisting of a 4 h experiment run, as described in Section 5.2. The system is evaluated by injecting faults into the network and comparing the number of faults detected with the number injected. We then discuss the implications of incomplete monitoring data as a result of using the wireless link for monitoring.

5.1. Implementation

A prototype of the MeshMon system has been implemented on the UCSB Meshnet, described in Section 3.2. The system consists of three components: the monitoring engine, analysis engine, and the communication engine. Each of these modules runs on every mesh router. The monitoring engine has been implemented as simple extensions to the madwifi driver. The analysis engine is a user level program that parses the incoming data from the monitoring engine, runs through a rules file to check whether any threshold has been crossed, and triggers the appropriate level of metric collection. Communication to the gateway occurs via the mesh backhaul link. The communication engine is responsible for sending problem notifications and metrics to the gateway. Similarly, the gateway can issue commands to a mesh node to collect specific metrics. The metrics are stored in a relational database at the central controller.

Two basic principles that have guided our implementation are simplicity and flexibility. Many of the metrics that we have used are already computed by the driver. The analytical engine has been implemented to be flexible so as to plug in any fault detection algorithm.

5.2. Experiment Setup

We use the mesh testbed described in Section 3.2 in all our evaluations. Eight client laptops are deployed at different locations in the building. Each laptop is equipped with an Atheros 802.11b/g card and is associated with the closest mesh router. Evaluations are conducted with two types of traffic: (a) constant rate flows as described in Section 3.3, which we call the CBR traffic and (b) traces from a large WLAN, which we call the Replay traffic.

In the first scenario, each laptop sends CBR traffic at a constant rate of 1 Mbps to the gateway. The setup is the same as the one described in Section 3.3.

In the second scenario, we use the WLAN traces described in Section 3.1 to extract link layer data traffic patterns and use this information to replay the traffic on the mesh testbed. Use of real traces allows us to evaluate the system for more realistic network usage with varying periods of low usage and high usage with congestion. From the entire 4 day trace, a 4 h plenary period is chosen. In this part of the trace, eight 802.11b/g AP were simultaneously being used by about 600 users. We extract per-client activity from these traces and choose traces from the most active 32 clients to be replayed. For each client, we create a traffic profile consisting of packet size and inter-packet interval that the client replays to the mesh gateway. Similarly, the gateway replays traffic to the clients, which in the original trace were packets from the access point to the client. The nodes are synchronized to retain the packet sequence in the original conversation. The client traces from the 32 selected clients are replayed using eight laptops, so each laptop runs four conversations with the gateway. We verified that there is indeed a variation in traffic load from each laptop.

The two traffic scenarios are summarized in Table I.

5.3. Fault Diagnosis

We now evaluate the fault detection capabilities of MeshMon. Our general evaluation methodology is as follows. We inject a set of faults into the system. The nodes run

Table I. Description of the two evaluation scenarios.

Traffic	Length (h)	# of clients	Description
CBR	4	8	CBR traffic of 1 Mbps with gateway
Replay	4	8	Most active 32 clients from IETF traces

MeshMon and attempt to diagnose the faults through increased metric collection and send alerts to the central controller when the fault is detected. We quantify the diagnosis accuracy by comparing the inferred fault and its source with the original fault we injected. We inject faults in both the client access layer and the mesh layer.

The following classes of faults were injected into the system.

- *Congestion (CONG):* A link is said to be congested when the load on the link exceeds its capacity, resulting in queue overflows and packet drops. We inject congestion on a link by starting several high bandwidth conversations to the node's upstream neighbor using *iperf*. In our setup, we introduce congestion by starting seven UDP flows, each flow with an application data rate of 25 Mbps. This causes the 1 Mbps CBR link to drop down to 20–100 Kbps till the duration of the other flows.
- *Poor link (LINK):* A poor link is characterized by low signal strength. We first place the receiver at a distance where it can just sustain a 1 Mbps throughput. In order to inject this fault, we use a metal case to cover the receiver's radio. This reduces the RSSI by 1–2 dB and creates a bad link, and the client is unable to sustain a throughput of 1 Mbps.
- *Interference from neighboring networks (INT):* A mesh network often shares the spectrum with neighboring Wi-Fi or other mesh networks. The network may be affected by interference from these neighboring networks, resulting in throughput degradation. In order to create interference of this kind, we setup seven laptops in *ad hoc* mode, in the vicinity of the mesh node, and start several high bandwidth conversations between them so that the mesh link is unable to sustain the 1 Mbps traffic.
- *Non-802.11 interference (EXT):* A mesh typically shares the spectrum with several non-802.11 devices such as microwaves, bluetooth devices, cordless phones, etc., that operate on the same band. We model this scenario by operating a microwave oven in the vicinity of the receiver.
- *Connectivity problems (CONN):* A client can have connectivity issues due to inability to associate with an access point or authentication failure. Association could fail because of a lossy link, high levels of congestion [18,21], ACLs or whitelists. Authentication problems could arise due to incorrect keys or missing or expired certificates. We inject association problems for a client by excluding its MAC address from the whitelist and thereby preventing association.
- *Non-local problems in mesh layer (CTRL):* Specific to the mesh layer are a class of problems that require the intervention of the central controller. Consider the case where a link in a mesh router's upstream path is congested. The router will suffer from high L-WCETT and hence low end-to-end throughput. However, the router's local metrics fail to diagnose the problem. We inject this fault by starting several high bandwidth conversations with a mesh router in the upstream path.

Note that the CONN faults class is applicable to the client access layer only, and the CTRL faults class to the mesh layer only. The other four fault classes are applicable to both layers. Therefore there are five fault classes for each layer.

We inject three instances of every fault described above for both traffic scenarios and we do this on both the client and the mesh layers. Hence each fault class is injected in 12 instances. Each mesh router runs MeshMon and collects metrics based on the hierarchical decision tree. When a fault is detected, the router sends to the central controller an alert corresponding to the fault and the metrics that support this diagnosis. For example, when a client is excluded from the whitelist and this fault is detected, an alert for association problems, along with the associated metrics of overhead index, and the number of probe and association requests for the client is sent to the central controller.

We evaluate the system in terms of the following metrics:

- *Faults injected:* Since we inject three instances of each type of fault class described above, on both the mesh and client access layers, a total of 30 faults is injected in each traffic scenario during the 4 h experiment. These faults represent the ground-truth for the experiment scenarios.
- *Faults detected:* A fault is said to be detected when the monitoring system correctly identifies the type and location of the fault. Ideally, the system should be able to detect all the injected faults.
- *Overhead reduction:* We compare the bandwidth savings of MeshMon with a non-hierarchical approach that uses the entire set of metrics in every monitoring cycle. In other words, overhead reduction is the percentage change in the size of the monitoring data. A high value of overhead reduction is desirable as it indicates the low bandwidth utilization of MeshMon.
- *False positive:* A false positive represents the scenario wherein MeshMon indicates the presence of a fault when none was injected in reality. We would like to have a low number of false positives to minimize false alarms and the associated overhead of metric collection.

Fault diagnosis accuracy and overhead reduction: The complete set of results from the experiments is presented in Table II. Of the total 60 faults injected in the two scenarios, 52 were successfully detected by MeshMon. The

Table II. Fault diagnosis performance of MeshMon.

Trace	CBR	Replay
Faults injected	30	30
Faults detected	27	25
False positives	0	2
Overhead reduction	68%	64%

average reduction in overhead for the two scenarios was 66%. In other words, MeshMon was able to detect a high percentage (87%) of faults using only one-third of the monitoring bandwidth as compared to the simple approach of using all the available metrics. For our simple testbed setup with 15 nodes and a maximum of one client per mesh node, the simple monitoring approach collected about 400 MB of monitoring data for a 4 h period, while MeshMon required about 134 MB. This is an encouraging result that indicates that MeshMon can scale better and can support larger mesh networks.

In order to understand the performance of the hierarchical metric collection system, we track the highest monitoring tier at each node during each 5 s monitoring interval. Figure 6 plots the histogram of the number of intervals for the highest tier in both the mesh backhaul layer and the client access layer. We normalize the histogram with respect to the total number of monitoring intervals in the experiment. We observe that for a majority of the time, the system operates at Tier 1. For the client layer, we observe that the fraction of intervals for the highest monitoring tier decreases with the increase in the tier number. This is in accordance with our design hypothesis of hierarchical metric collection. For the mesh layer, we observe that the system operates for a higher percentage of time in Tier 3 than Tier 2. This is because we observed fewer non-local problems in the mesh layer. As per the decision tree in Figure 5(b), the metric collection remains at Tier 2 only when the central controller is triggered for non-local fault diagnosis; otherwise it transitions to Tier 3 to diagnose local faults.

False positives: The results in Table II indicate a high number of false positives and hence we further investigate this behavior. Figure 7 plots the time series of mesh layer events during one of the experiment trials. The data points in Figure 7(a) represent the injected faults, and data points in Figure 7(b) represent the alerts recorded. We observe that for some injected faults, the central controller receives alerts from multiple mesh routers. MeshMon currently does not have the capability of correlating alerts posted by multiple mesh routers. Such a capability would enable MeshMon to distinguish a fault that simultaneously impacts the performance of multiple mesh routers and reduce the misleading false positive rate.

Fault classes: We now study the fault diagnosis capabilities of MeshMon for each class of faults that were injected. This study helps us obtain further insight into the behavior of the system and may be used for further tuning of the thresholds or the decision-trees. Figure 8 plots the various classes of faults injected and the number of faults detected.

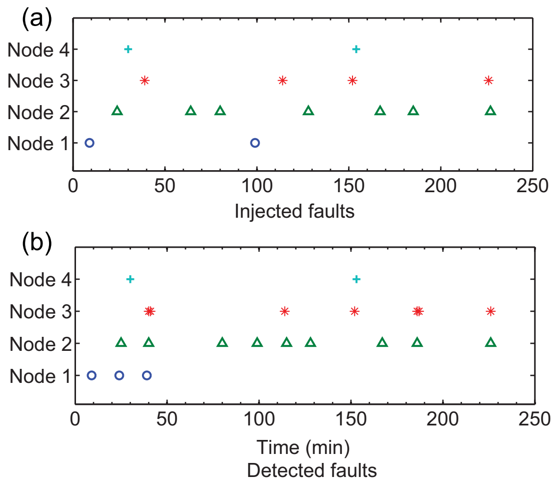

Figure 7. Time series of events (faults injected and faults detected) at the mesh layer in a representative experiment trial.

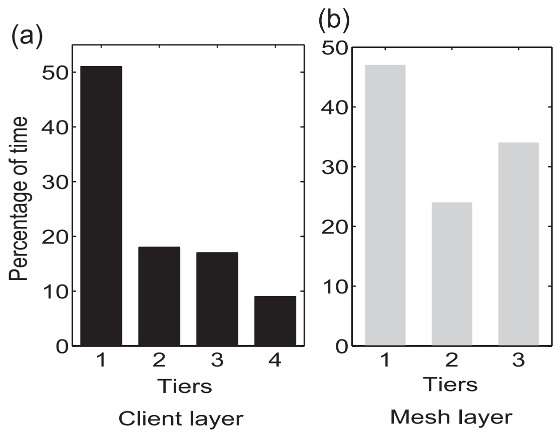

Figure 6. Percentage of time spent in each tier at the client and mesh layers.

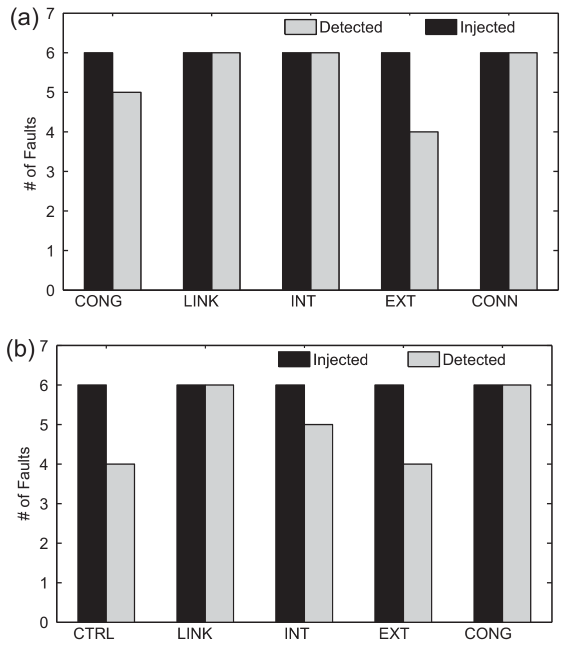

Figure 8. Number of faults injected and detected at client and mesh layers. These results include both the CBR and replay traces. (a) Fault detection at the client layer; (b) fault detection at the mesh layer.

We plot this information for the mesh layer faults as well as the client access layer faults.

We observe that problems such as poor link, connectivity issues, congestion at the client access layer, local congestion at the mesh layer, and interference from neighboring 802.11 networks can be detected with a high degree of accuracy. Diagnosis of faults such as external non-802.11 interference and non-local problems at the mesh layer is less accurate and requires further investigation. Our intuition on the inaccuracy for the non-local problems is that non-local problems injected on one node is affecting the node further up in the upstream path; Figure 7 supports this hypothesis. The detection algorithm required both the type of fault and source to be identified correctly, resulting in higher false negatives.

6. CONCLUSION

Management and troubleshooting of a wireless network is a challenging task because of the variety of problems that may arise, and the various metrics that characterize these problems. The multi-hop architecture of the mesh networks adds to the complexity of this task. In this paper, we described MeshMon, a network monitoring system specifically designed to address the problem of real-time management of a WMN. MeshMon employs a dynamic and adaptive metric collection system. We presented the design of hierarchical metric collection in a mesh network and the classification of metrics in tiers.

Our implementation and evaluation of MeshMon demonstrated the advantages of multi-tiered metric collection, in terms of significant reduction in the bandwidth requirement. While we have designed our solution for mesh networks, we note that a simpler version of MeshMon (specifically, the decision tree with the client access layer only) can be used for infrastructure WLANs as well. Currently, MeshMon addresses the problems of metric collection and real-time fault diagnosis. As part of our future work we would like to extend the system to include automated remedial actions, i.e., based on the fault diagnosis, MeshMon should apply a pre-configured solution to rectify the problem.

ACKNOWLEDGMENT

This work is supported in part by NSF Wireless Networks award CNS-0722075.

REFERENCES

1. Cheng Y-C, Bellardo J, Benko P, Snoeren AC, Voelker GM, Savage S. Jigsaw: solving the puzzle of enterprise 802.11 analysis. In *Proceedings of SIGCOMM*, Pisa, Italy, September 2006.
2. Cheng Y-C, Afanasyev M, Verkaik P, *et al*. Automating cross-layer diagnosis of enterprise wireless networks. In *Proceedings of SIGCOMM*, Kyoto, Japan, August 2007.
3. Jardosh A, Suwannatat P, Hollerer T, Belding E, Almeroth K. SCUBA: focus and context for real-time mesh network health diagnosis. In *Proceedings of PAM*, Cleveland, OH, April 2008.
4. Raghavendra R, Acharya PAK, Belding EM, Almeroth KC. Antler: a multi-tiered approach to automated wireless network management. In *Proceedings of the 1st IEEE Workshop on Automated Network Management*, Phoenix, AZ, April 2008.
5. Qiu L, Bahl P, Rao A, Zhou L. Troubleshooting wireless mesh networks. *SIGCOMM Computer Communication Review* 2006; **36**(5): 17–28, 2006.
6. Xiao Y. Flow-net methodology for accountability in wireless networks. *IEEE Network* 2009; **23**(5): 30–37.
7. Sheth A, Doerr C, Grunwald D, Han R, Sicker D. MOJO: a distributed physical layer anomaly detection system for 802.11 WLANs. In *Proceedings of MobiSys*, Uppsala, Sweden, June 2006.
8. Adya A, Bahl P, Chandra R, Qiu L. Architecture and techniques for diagnosing faults in IEEE 802.11 infrastructure networks. In *Proceedings of MobiCom*, Philadelphia, PA, September 2004.
9. Chandra R, Padmanabhan V, Zhang M. WiFiProfiler: cooperative diagnosis in wireless LANs. In *Proceedings of MobiSys*, Uppsala, Sweden, June 2006.
10. Netdisco—Network Discovery and Management, February 2008. Available online at: http://www.netdisco.org/
11. AirWave Management Platform, February 2008. Available online at: http://www.airwave.com/
12. AirMagnet, February 2008. Available online at: http://www. airmagnet.com/
13. Yeo J, Youssef M, Agrawala A. A framework for wireless LAN monitoring and its applications. In *Proceedings of WiSe*, Philadelphia, PA, October 2004.
14. Jardosh AP, Ramachandran KN, Almeroth KC, Belding-Royer EM. Understanding congestion in IEEE 802.11b wireless networks. In *Proceedings of IMC*, Berkeley, CA, October 2005.
15. UCSB MeshNet. Available online at: http://moment. cs.ucsb. edu/meshnet
16. Bicket J, Aguayo D, Biswas S, Morris R. Architecture and evaluation of an unplanned 802.11b mesh network. In *Proceedings of MobiCom*, Cologne, Germany, August 2005.
17. Robinson J, Knightly E. A performance study of deployment factors in wireless mesh networks. In *Proceedings of INFOCOM*, Anchorage, AK, May 2007.
18. Jardosh AP, Mittal K, Ramachandran KN, Belding EM, Almeroth KC. IQU: practical queue-based user associa-

tion management for WLANs. In *Proceedings of MobiCom*, Los Angeles, CA, September 2006.

19. Padhye J, Firoiu V, Towsley D, Kurose J. Modeling TCP throughput: a simple model and its empirical validation . In *Proceedings of SIGCOMM*, Vancouver, Canada, September 1998.
20. Draves R, Padhye J, Zill B. Routing in multi-radio, multi-hop wireless mesh networks. In *Proceedings of MobiCom*, Philadelphia, PA, September 2004.
21. Raghavendra R, Belding EM, Papagiannaki K, Almeroth KC. Understanding handoffs in large IEEE 802.11 wireless networks. In *Proceedings of IMC*, San Diego, CA, October 2007.
22. von Eicken T, Karpinski S. Systems and methods for selecting efficient connection paths between computing devices, March 2008. Patent Pending 20080069104.
23. Howard SL, Schlegel C. Differential turbo-coded modulation with APP channel estimation. *IEEE Transactions On Communications* 2006; **54**(8): 1397–1406.

AUTHORS' BIOGRAPHIES

Ramya Raghavendra is a Ph.D. candidate with Professor Elizabeth Belding in the Department of Computer Science, University of California Santa Barbara. Her research interests lie in network computing and wireless networking, specifically in studying the performance of network deployments and building protocols to make them more reliable and robust. Prior to this, she received her B.Eng. degree from Visvesvaraya Technological University, India.

Prashanth A.K. Acharya received his Ph.D. in 2009 from the Department of Computer Science at the University of California, Santa Barbara. He received his B.Eng. degree at the National Institute of Technology, Karnataka, India in 2002. His research interests include mobile and wireless networks, wireless multi-hop and mesh networks, Quality of Service, and multimedia networks. He is currently with Amazon Web Services.

Elizabeth M. Belding is a Professor in the Department of Computer Science at the University of California, Santa Barbara. Elizabeth's research focuses on mobile networking, specifically mesh networks, multimedia, monitoring, and solutions for networking in under-developed regions. She is the founder of the Mobility Management and Networking (MOMENT) Laboratory (http://moment.cs.ucsb.edu) at UCSB. Elizabeth is the author of over 80 papers related to mobile networking and has served on over 50 program committees for networking conferences. Elizabeth served as the TPC Co-Chair of ACM MobiCom 2005 and IEEE SECON 2005, and the TPC Co-Chair of ACM MobiHoc 2007. She also served on the editorial board for the *IEEE Transactions on Mobile Computing*. Elizabeth is the recipient of an NSF CAREER award, and a 2002 Technology Review 100 award, awarded to the world's top young investigators.

Kevin C. Almeroth is currently a Professor in the Department of Computer Science at the University of California in Santa Barbara where his main research interests include computer networks and protocols, wireless networking, multicast communication, large-scale multimedia systems, and mobile applications. He has published extensively with more than 150 journal and conference papers. He is also heavily engaged in stewardship activities for a variety of research outlets including journal editorial boards, conference steering committees, new workshops, and the IETF. He is a Member of the ACM and a Senior Member of the IEEE.