TLQAP : A Topology and Link Quality Assessment Protocol For Efficient Node Allocation on Wireless Testbeds [*]

Dimitris Syrivelis
Dept of Computer and Communications Engineering
University of Thessaly
jsyr@inf.uth.gr

Angelos-Christos Anadiotis
Dept of Computer and Communications Engineering
University of Thessaly
aganadio@inf.uth.gr

Apostolos Apostolaras
Dept of Computer and Communications Engineering
University of Thessaly
apaposto@inf.uth.gr

Thanasis Korakis
Dept of Computer and Communications Engineering
University of Thessaly
korakis@poly.edu

Leandros Tassiulas
Dept of Computer and Communications Engineering
University of Thessaly
leandros@inf.uth.gr

ABSTRACT

In this paper we present Topology and Link Quality Assessment Protocol ($TLQAP$), which we have implemented as a wireless testbed management framework component, that is used to inspect link quality between wireless testbed nodes and appropriately map them to user experiment requirements. $TLQAP$ is mainly an OSI layer 2 design for fixed location, non RF-isolated wireless testbed deployments, which assesses interconnection topology and link quality by estimating packet delivery ratio (PDR) and transmission delay at each node for all requested channel, rate and transmission power combinations. Moreover, $TLQAP$ builds a measurement history log and creates a channel utilization profile, in the context of each testbed node, for all the nearby testbed-external devices that operate independently in the region and are not under the management framework control. The analysis of this information enables $TLQAP$ to choose the channels that have the highest probability of being free during an experiment. $TLQAP$ OSI layer 2 component has been implemented in the click modular router framework and the controller component has been integrated with OMF management framework for wireless testbeds. To outline $TLQAP$ benefits, we have performed experiments on our ORBIT node testbed and we compare it to an existing application level measuring tool.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

[*]The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement nř224263

Keywords

link quality measurements, wireless testbed framework

1. INTRODUCTION

As wireless computer networking is becoming mainstream for almost any type of network deployments, research at all design levels of wireless systems is very active. While for wired network solutions most researchers developed prototypes on simulation environments, the unpredictable factors that can affect the quality of wireless connectivity, especially in large-scale experiments, made respective wireless systems simulations almost impossible. To that end, there is a great need to deploy and manage wireless testbeds that can be used for the development and evaluation of wireless networking systems.

Managing and distributing a collection of wire interconnected computers to multiple users with guaranteed resource availability, like the number of dedicated processors, memory and network bandwidth, has a number of challenges which have already been addressed by managing frameworks for High Performance Computing (HPC) clusters. In the same spirit, wireless testbed deployments also feature a wired ethernet backbone that is used by the respective resource management frameworks [14]. Therefore, many of the management concepts that have been introduced for wired testbeds [3] [4], have been extended and reused by wireless testbed management frameworks. On the other hand, the latter face an additional important challenge: the distribution and management of the wireless bandwidth in terms of frequency channels, which along with the node topology and connectivity range can become a very complicated task.

To deploy an experiment, the wireless testbed user will need to allocate nodes that satisfy certain topology and link quality requirements. Moreover, the knowledge of the maximum possible throughput that each connection can achieve, allows the user to properly evaluate the observed experiment performance. In some wireless testbed deployments [5] were all the nodes are in an RF isolated room and are tightly located to avoid connectivity range problems, the allocation of frequency channel, also known as slicing [6], can be performed in a static manner. In these setups, wireless channels and bandwidth can be distributed as any other resource, like processors or memory and the user must not be able to change them.

While the described wireless testbed deployments are appropriate for a wide range of networking system experiments, there is also a need for testbed deployments that are closer to an end user setup. In such deployments there might be indoor and outdoor nodes with wireless links of varying quality, where some of them might not be able to directly communicate. Moreover, the testbed might not be RF isolated, e.g deployed on a campus site [8] [7], where several other wireless terminals, out of the testbed context, compete for channel use. Nevertheless, for a certain range of experiments, that usually belong in wireless mesh network research, the user can take advantage of the described testbed setups to evaluate networking systems (e.g. adhoc routing protocols, hidden terminal solutions). In these testbed cases, the user will need to know the actual link quality of all the wireless links, as well as the possible topologies that can be formed by the allocated nodes, in order to evaluate the experimental results. For example let's assume that a user needs to test a network coding design approach.

In network coding, each node that acts as a gateway between other nodes that cannot communicate directly, may produce a linear combination (mix) of two or more packets that belong to different flows and avoid this way explicit transmissions by performing a single transmission of the mixed packet (which must be appropriately decoded at each destination). Obviously, the biggest challenge of network coding schemes is to properly identify coding opportunities which directly depend on the network topology and link quality. In this case the user needs to allocate nodes that satisfy certain topology requirements and needs to know explicitly the location of each node on the connectivity graph. In such cases the testbed management framework needs a system like $TLQAP$ to satisfy the node allocation requirement.

Of course, since the testbed deployment is not RF isolated and the environment is volatile, the link quality between any pair of nodes may unexpectedly vary at any point in time due to external interference. For this reason the static distribution approach, that is used in RF isolated wireless testbeds, is not efficient for these deployments.

The wireless management framework of non RF isolated testbeds should feature support that can quickly and accurately inspect the current link qualities between all the available testbed nodes and at all available channels. This should be the first step of the management framework response to a node allocation demand. Frequent inspection of the link quality will enable the framework to allocate the nodes that at least have the highest probability to satisfy the experiment requirements. Of course, during the actual experiment deployment, intrusive external interference may appear but this will happen with reduced probability compared to a static channel allocation method. Moreover, the management framework may perform inspections whenever the testbed is not in use and identify, independently for each link, time periods where external interference will be with high probability minimal. The main characteristics of an inspection subsystem that frequently determines the quality of wireless links between testbed nodes should be accuracy, speed and scalability.

In this work we propose the Topology and Link Quality Assessment Protocol ($TLQAP$) which has been designed to satisfy efficiently the aforementioned need for node allocation on non RF isolated wireless testbeds. $TLQAP$ is an OSI Layer 2 protocol, tightly integrated with the underlying MAC layer that exports a control and configuration interface which can be used appropriately by a management framework. The latter may use $TLQAP$ support to quickly and accurately determine the current quality of the wireless links between testbed nodes, on all available channels and rates. We have implemented $TLQAP$ in the click modular router [12] taking advantage of the available click extensions for the madwifi driver [1]. On the control side we have implemented an OMF based controller plugin, that interacts with $TLQAP$, observes link qualities and allocates the appropriate nodes based on the user experiment demands. In the sections that follow we describe the $TLQAP$ protocol, all system components and their organization and we compare the accuracy, the speed and the scalability of our approach with an approach that is based on existing tools.

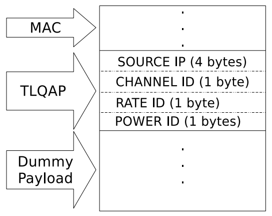


Figure 1: $TLQAP$ header.

2. TLQAP PROTOCOL

The $TLQAP$ approach is based on actual throughput measurements of a fixed number of consecutive packet transmissions that are initiated at each testbed node. A typical $TLQAP$ session is as follows. The management framework interacts with the user and creates an xml file (presented in the implementation section) that describes the required link quality between the nodes that form the network topology. Then the framework generates a list of the available nodes and channels, that are not currently assigned to other experiments, and deploys the $TLQAP$ system. After all the nodes are ready, the management framework sequentially starts for each node, channel rate and transmission power, the $TLQAP$ transmission sessions.

Each $TLQAP$ session is comprised of a user defined number of fixed size packets which are transmitted in one burst at a specific channel, rate and power combination. These packets are addressed to an arbitrary neighbor node of the current transmitter and must be transmitted without the 802.11 support for low level acknowledgements and retransmissions. Otherwise, the packet loss ratio as captured by $TLQAP$ will be far lower than the underlying, actual loss ratio. Only one $TLQAP$ session is testbed wide allowed at a time, so the next $TLQAP$ session on any node may begin after the previous one is completely finished and all the scheduled packets have been transmitted. Each node initiates $TLQAP$ transmission sessions for all the required channel, rate and power combinations.

On the other hand, each node sniffs (media is in monitor mode) and logs all the $TLQAP$ packets that it can hear. $TLQAP$ features a network packet header that is placed immediately after the ethernet header as depicted in figure 1. The header fields are the sender IP address along with globally agreed identification numbers for the channel, rate and transmission power that have been used for current packet transmission. The $TLQAP$ log is based on a counter map that holds an independent counter for each sender IP, channel, rate and power combination. Each counter is incremented when a packet with the appropriate combination arrives. After all $TLQAP$ sessions have completed, the management framework collects the counter maps from all the nodes and processes them to calculate, for each node, the packet delivery ratio (PDR) for each channel, rate and power. The PDR from node X to each node Y is calculated by dividing the number of packets received by Y by the number sent by X. As we have mentioned, during each transmission session a fixed number of packets is transmitted. The respective session transmission delay is recorded and it is immediately retrieved by the management framework which, along with the counter maps, has all the information that are needed to: i)calculate PDR for each available link direction and ii)inspect channel traffic. In the next section we give more details on how we perform all the measurements.

3. SYSTEM IMPLEMENTATION

The $TLQAP$ implementation features two main components: An OSI layer 2 component that has been integrated with the linux network stack and a controller component that has been implemented as an OMF [14] wireless management framework extension.

3.1 Network Stack Support

We have implemented the $TLQAP$ protocol in the click modular router framework [12]. Click is a linux packet processor engine that has been integrated in the linux network stack and is suitable for the development of real world, production quality, networking systems that are primarily targetted at OSI layer 2 and 3. In the $TLQAP$ implementation we have used the click extensions for the madwifi driver that controls the Atheros 5212 chipset. With these extensions, $TLQAP$ can configure the underlying driver parameters like transmission rate and power independently for each packet.

The click $TLQAP$ implementation is comprised of two so called click elements: The $TLQAP$ receiver and the $TLQAP$ transmitter. The network stack incoming and outgoing processing paths are depicted in figure 2 a) and figure 2 b) respectively.

Empty packets of a fixed size are generated by the click *RatedSource* element that along with a *Burster* element form the packet generator engine. This produces a fixed number of packets for each transmission session and pushes them all at once to the next element which is the *TLQAPtransmitter*. This element, based on its current configuration (rate, channel, power), encapsulates the packets that arrive from the generator with a properly informed $TLQAP$ header (figure 1) and enqueues them to its transmission queue.

The underlying madwifi driver is instructed to transmit addressed packets without 802.11x level acknowledgements and retransmissions but it does use the transmission slot backoff policy. This way the transmission delay of a session packet set depends entirely on the rate and the channel traffic. Transmission delay for a broadcast session is accurately recorded by the *TLQAPtransmitter* and the value is

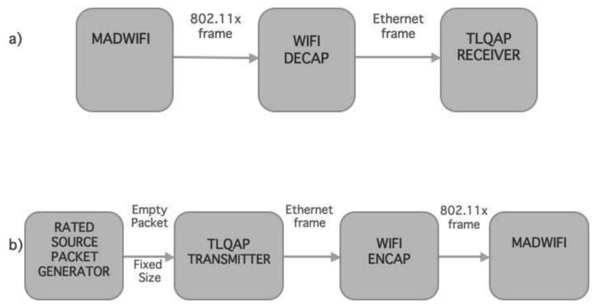


Figure 2: $TLQAP$ in the Network Stack: a) incoming path, b) outgoing path

immediately retrieved by the OMF management framework, which has to wait for the transmission session to finish. The recorded delay is used to determine possible channel traffic that originates from a testbed external source. This is a straight forward estimation because at a specific rate Y bytes/sec, a fixed number X of fixed sized packets of byte size Z, needs $W * \frac{X\ Z}{Y}$ seconds to get transmitted over a free channel. The factor W is used because the wifi header bytes are stripped before delivery to *TLQAPreceiver* without being counted, while other low level 802.11 details may introduce additional byte transfer delay. After performing numerous experiments on free channels and different rates we have seen that for 1400 byte ethernet frames W is equal to 0.7. If the channel is not free, the low level 802.11 transmission backoff delays will increase the overall packet transmission delay. The fraction of the recorded *TLQAPtransmitter* delay by the theoretical transmission delay of a session packet set gives the respective free channel ratio. Obviously the larger the packet set, the more accurate the free channel ratio estimation because a larger time slot is inspected.

Notably, since $TLQAP$ sessions are transmitted in a single burst and immediately saturate the MAC layer at all available transmission rates, the channel utilization can be again estimated during each transmission session and an average can be returned by $TLQAP$ system for each channel. This approach is employed to inspect channel utilization for a larger time slot which improves the traffic observation. For this reason, $TLQAP$ sessions, which are started from the highest provided rate, do not stop when PDR is found equal to one at all recipient nodes. Note that, in this case, PDR for lower rates will be definitely equal to 1 for all nodes and the broadcasts from that point onwards are performed only to inspect channel utilization. Of course, the fact that a channel is found free during the $TLQAP$ measurements does not provide any form of guarantee that it will continue to be available during the actual deployment of the user experiment. In the next subsection we describe framework support that increases the probability that a channel will continue to be free during the actual testbed use.

In the incoming path (figure 2 a), the madwifi driver is configured to operate in monitor mode and delivers for processing all the frames it receives. After the wifi header decapsulation the packet is delivered to the *TLQAPreceiver* element, which checks if it contains a *TLQAP* header. If it does, the element reads all the *TLQAP* field values and forms a hash identifier that is used to retrieve and increment the appropriate counter from a hash table (counter map). The received packet is then discarded. The counter map contents can be retrieved at any time from the management framework.

Both *TLQAPtransmitter* and *TLQAPreceiver* elements can be configured via the click communication interface. The latter can be accessed on each node via telnet (over the wired ethernet) or locally via loopback interface. This communication channel is also used for data retrievals. The described support allows easy integration of *TLQAP* with any type of management framework.

3.2 Management Framework Support

We have integrated *TLQAP* with the OMF framework for wireless testbeds. *OMF* is a Control, Management and Measurement Framework that provides the users with a set of tools to describe, execute and collect the results of an experiment in a straight forward manner. There are three main components that comprise OMF: i) the gridservices, ii) the nodehandler and iii) the nodeagent. Below, we give a short description for each one of these components.

Gridservices is a set of web services that are used by *OMF* to fetch information and perform actions remotely on the nodes. These services can be used for the node system image loading, the experiment execution and the results collection.

Nodehandler resides on the central server that interacts with the user for the experiment submission. Moreover, it provides the necessary applications for node system image loading, experiment execution, image saving and node status check. Nodehandler communicates with both the gridservices and the nodeagent to get the required information and perform actions. Regarding the experiment deployment, nodehandler contains a set of prototypes that can be used for experiment definition. Based on a message passing system, the nodehandler uses either multicast or unicast communication to contact nodeagent(s) in order to initiate and control experiment deployment. Finally, nodehandler watches the experiment execution and notifies the user for any problems that may arise.

A nodeagent instance is deployed on each testbed node. Contrary to the nodehandler which is triggered upon load, execution, save or status call, the nodeagent is constantly active. It is waiting for information to arrive from the nodehandler, which contain instructions for the experiment deployment. Since nodeagent runs as a background process, it reports its state to the nodehandler, which in turn notifies the user.

Apart from these existing components, OMF is being currently extended with a new component that performs scheduling. In this first scheduler version, the user may request the topology and the resources needed for his experiments. Obviously, the scheduler needs to determine the current testbed topology, link quality and channel utilization to properly decide which node set matches the experiment requirements.

More specifically, we developed an OMF based *TLQAP* controller component to accomplish the following tasks: i) interact with our scheduler and get a list with nodes and spectrum availability as well as topology and link quality requirements, ii) configure *TLQAPtransmitter* and start transmission sessions sequentially on each node, iii) collect the transmission delay for each broadcast session after it is completed, iv) collect the counter maps from all nodes, v) process results, allocate the nodes and reply to the scheduler.

Initially the user should provide the framework with an abstract description of the nodes and the link characteristics that are needed between them. Based on those data, the scheduler creates an XML description of the users' request. Before starting the XML dialog, the scheduler also collects information about the available nodes and the spectrum. Then it initiates communication with the *TLQAP* controller, that runs on the OMF server and provides the available nodes using the following XML syntax that is illustrated via an example:

```
<TestbedAvailability>
  <Domain name="nitos">
    <AvailableSpectrum>
        <Channel>1</Channel>
        <Channel>2</Channel>
    </AvailableSpectrum>
    <AvailableNodes>
        <Node>node001</Node>
        <Node>node003</Node>
    </AvailableNodes>
   </Domain>
  <Domain name="sb2">
    <AvailableSpectrum>
        <Channel>3</Channel>
        <Channel>4</Channel>
    </AvailableSpectrum>
    <AvailableNodes>
        <Node>node001</Node>
        <Node>node002</Node>
    </AvailableNodes>
   </Domain>
</TestbedAvailability>
```

The *TLQAP* controller now knows which nodes and channels are available, but it also needs to know which are the experiment link requirements. For this reason the scheduler, which has already interacted with the user, issues the following link quality request:

```
<NetTopoGraphReq>
  <link id="1" type="bidirectional">
    <MaxChannelUtil>30</MaxChannelUtil>
    <direction>
        <MinRate>10</MinRate>
        <MinPDR>60</MinPDR>
        <TransPower>60</TransPower>
    </direction>
    <direction>
        <MinRate>10</MinRate>
        <MinPDR>60</MinPDR>
        <TransPower>60</TransPower>
    </direction>
  </link>
   ...............
</NetTopoGraphReq>
```

This request describes independently for each link and direction the minimum requirements. Note that the maximum channel utilization refers to the maximum allowed percentage of channel that may occupied by testbed external devices. The *TLQAP* system should determine and reply with all the links that meet these requirements. The reply XML syntax is as follows:

```
<NetTopoGraphRes>
 <link id="1">
  <node src="Node001">
    <connection dest="Node005">
      <Channel>48</Channel>
      <Rate>10</Rate>
      <PDR>60</PDR>
      </connection>
      <connection dest="Node007">
         <Channel>4</Channel>
         <Rate>12</Rate>
         <PDR>70</PDR>
      </connection>
       . . . . . . . . . . . . . . . . .
   </node>
   <node src="Node005">
      <connection dest="Node001">
         <Channel>48</Channel>
         <Rate>10</Rate>
         <PDR>60</PDR>
      </connection>
      <connection dest="Node007">
         <Channel>8</Channel>
         <Rate>15</Rate>
         <PDR>60</PDR>
      </connection>
       . . . . . . . . . . . . . . . . . . . . . . . .
   </node>
    . . . . . . . . . . . . . . . . . . . . . . . . .
  </link>
</NetTopoGraphRes>
```

$TLQAP$ controller is now ready to perform the $TLQAP$ based measurements. Firstly, it inquiries the testbed scheduler to get the list of available nodes and channels on which $TLQAP$ measurements may be performed. Then the system uses OMF support to load $TLQAP$ images to the nodes and start $TLQAP$ click instances. The transmission sessions are sequentially started on each node for all available channels, rate and power combinations. The controller stores the reported transmission time after a session finishes. Finally, when the last session is finished, the controller collects all the counter maps, processes results and replies to the scheduler.

Admittedly, determining the connectivity between a group of nodes that belong to a non RF isolated testbed, immediately before using them, does not necessarily mean that the observed quality of all the links will remain stable when the actual experiment is deployed. At least, $TLQAP$ increases the probability that the chosen nodes will satisfy the experiment needs. To further strengthen this probability, we have developed a mysql based history log for the quality of each testbed link on all available channels. Each time $TLQAP$ is deployed and results are collected and processed, the database is inquired for the quality of the chosen links on each channel around this time of day and for the past week. If the link quality and/or channel does not appear to have stable availability, the system tries to find a substitute node if it is possible. After this process is finished, the database is populated with the current link quality measurements that have been recorded by $TLQAP$ which will be used for future reference. In order to increase the density of the history log and improve its validity, we schedule $TLQAP$ sessions for all the idle testbed nodes and channels, every fifteen minutes, for the sole purpose of expanding the history log.

4. EVALUATION

To evaluate $TLQAP$ we have used our OMF [14] testbed that is deployed on the main building of our department which is located in the center of Volos city in Greece. The testbed currently features a total of 9 nodes which are placed both outdoors and indoors. Around the testbed area and out of the testbed context, are independently operating 56 access points which are located in the nearby buildings. Notably, most of the neighbors use 802.11g channels during business hours. Below we describe our testbed nodes and organization in detail and we then present and compare $TLQAP$ system with the existing bandwidth measurement tools approach, that could have been used instead on this testbed.

4.1 Testbed Description

Our testbed is comprised of ORBIT-like nodes, as depicted in Figure 3(b). More specifically, in Figure 3(a) we can see a diagram of an ORBIT node design. Each node consists of a 1GHz VIA C3 processor, 512MB of RAM, 40GB of hard disk, two ethernet ports and two miniPCI slots which are used to host two 5212 Atheros WiFi cards.

All the nodes are connected through wired Ethernet with the testbed's server - *console*. On console we have all the required testbed services running. These services are both network services, such as Dynamic Host Configuration Protocol (DHCP) server which gives IP address to the nodes, Domain Name System (DNS) server which gives names to the nodes, Network File System (NFS) server and OMF services. We also maintain a web server where we keep the web interface of our scheduler. On this server, we also keep some scripts mandatory for remotely booking the nodes and a MySQL server for keeping records of the testbed status at each slot. More information about our testbed (e.g. node connectivity graph) can be found on our web site [2].

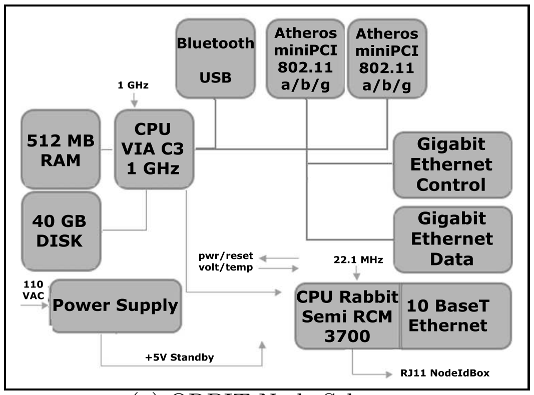
4.2 Using an existing measurement tool

The most popular approach to assess the quality of a wireless link between testbed nodes is using a bandwidth measurement application. Unfortunately, in this approach, the default network stack allows packet broadcasts to be transmitted only at the basic rate, so the $TLQAP$ style transmit-receive sessions cannot be used with these tools. Therefore, for each node that is within range, the classic single addressed transmit-receive flows must be used. In our case we used the *iperf* tool as follows. For every fixed rate, channel and power combination we deploy *iperf* transmitter-receiver pairs for UDP unicasts on the respective nodes and for each link direction. *Iperf* provides results every second and usually needs to receive packets at least for three seconds to start reporting the actual throughput.

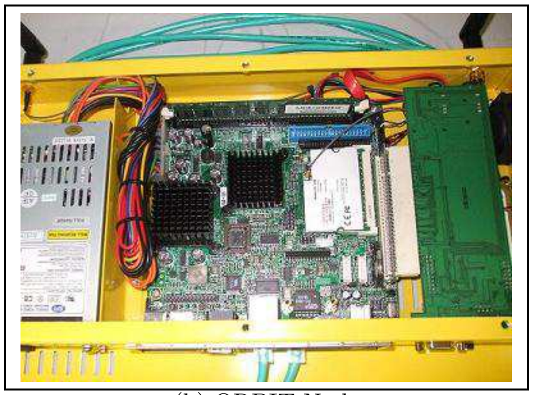
Depending on the current fixed rate, the iperf packet generator is configured to saturate the MAC layer. We have seen in practice, after performing numerous experiments on free channels, that for *iperf* default configuration the observed application level throughput of a high quality single link direction between two testbed nodes, is 55% of the (fixed) transmission rate. Therefore, if the *iperf* reported throughput is less than 55% of the used transmission rate we assume that the channel is not totally free. The free channel ratio is calculated by dividing the observed throughput by the maximum throughput that can be observed in practice (55% of the transmission rate). Moreover, *iperf* receiver reports the actual packet delivery ratio and during the measurements we have disabled 802.11 support for low level acknowledgements and retransmissions.

4.3 Experiments and Results

We performed a series of experiments in order to: i)verify that $TLQAP$ properly determines PDR and channel utilization, ii) evaluate the contribution of the history log and iii) assess the scalability of $TLQAP$ approach. Moreover, in some cases we compare $TLQAP$ with the *iperf* approach. Note that our testbed nodes are not tightly located, so some



(a) ORBIT Node Schema



(b) ORBIT Node

Figure 3: ORBIT-Like Nodes

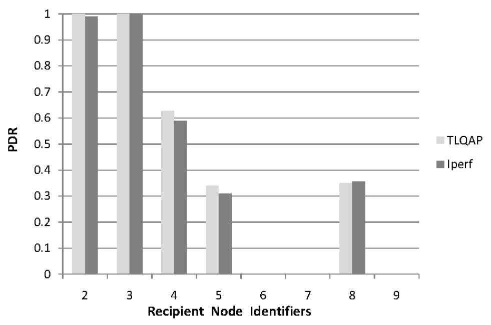


Figure 4: *TLQAP* and *Iperf* PDR measurements at nodes 2-9 on a free channel and fixed rate. The transmitter is node 1

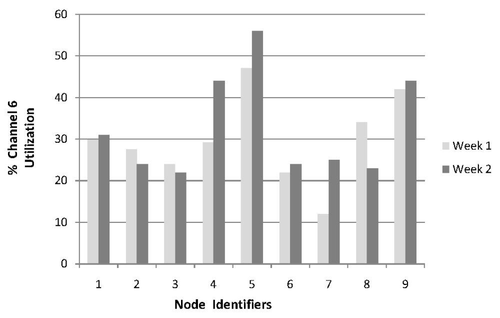


Figure 5: Average channel 6 utilization by testbed external devices for 2 consecutive weeks. (measured at each node between 11:00a.m and 11.20a.m)

nodes cannot directly communicate and channel interference may not be the same for all the nodes as well.

We first examined the reported PDR using both $TLQAP$ and *iperf* approaches on channel 48. We chose this channel because it is not being used by any neighbour and it is guaranteed to be interference free from any factor that is not under the testbed administration control. In figure 4 we present the packet delivery ratio (PDR) when node 1 transmits to the neighbors 500 1400 byte packets using $TLQAP$ at $54Mbits/sec$ on channel 48. We have repeated the experiment using *iperf* reports for the same rate and channel and we depict the results on the same figure. As expected, there are no serious PDR deviations between the two approaches and they both determine it accurately.

The next figure depicts the channel 6 utilization by testbed external devices which is determined independently at each node. We have employed the history log to improve the probability that a channel will be available, because wireless network traffic usually follows a steady pattern during working hours on business days. Of course, history log can only provide rough estimations. In figure 5 we present the average channel 6 utilization that has been observed for two consecutive weeks, only on business days, in the time frame between 11:00a.m and 11:20a.m. On average, 30% of channel 6 is occupied each second for testbed external transmissions. For these measurements we transmitted 1000 packets during each $TLQAP$ session.

The basic advantages of $TLQAP$ approach over the application level measuring tools regarding the measuring delay are i) the use of a single session per node, transmission rate and channel and ii) the fact that the layer 2 implementation can accurately determine when actual packet transmission takes place and allows a fixed number of consecutive packet transmissions to provide accurate delay measurements.

On the contrary, application level bandwidth measurement tools need to generate traffic that saturates all the lower layer buffering mechanisms and to observe the reception side for a fixed amount of time to make sure that they are capturing the actual throughput. *Iperf* needs 3 seconds for each link direction and, for example, for all 9 nodes of our experimental setup 3024 seconds are required to exhaustively check all links on 14 channels and 9 different rates using fixed transmission power.

In figure 6 , we present the $TLQAP$ delay for the same example and for different sizes of session packet numbers. As it is depicted, if we use 1000 packets during each broadcast session $TLQAP$ approach is 4 times faster than *iperf* and produces the same results. Note that during these experiments, we did not employ any additional support for the *iperf* approach to examine if there is any connectivity at all (e.g. via ping) between two nodes, which would avoid performing measurements between nodes that are not directly linked. As we have explained, $TLQAP$ performs all broadcasts at all available rates anyway, even for the sole purpose

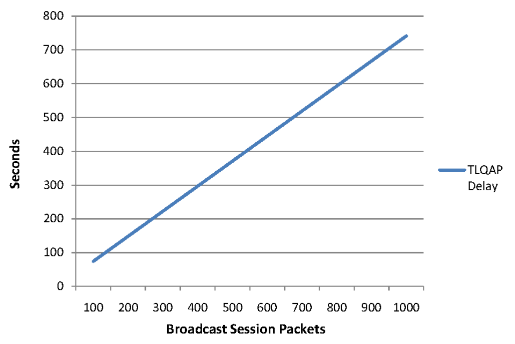


Figure 6: *TLQAP* measured delay for inspecting link of 9 nodes on 14 channels and 9 different rates

of determining channel utilization.

5. RELATED WORK

Adhoc routing protocols for wireless mesh networks, employ link quality metrics that can be updated quickly without being particularly intrusive and determine the best routing path to a destination. Most popular of these metrics like ETX [9], mETX [13], ETT [10] are based on the PDR and/or the average packet transmission delay, which are the only measurements that must be performed at each node for all its immediate links. Exactly as it happens with *TLQAP*, these routing protocols calculate PDR and transmission delay by periodically transmitting the so called packet probes at all available rates, without using 802.11 level acks and retransmissions. Increasing the probing frequency, results in more accurate estimations but also increases the bandwidth overhead. For this reason, in routing protocols, packet probes are emitted from each node every few seconds. Since *TLQAP* has been designed for offline measurements it can use relatively large number of consecutive probe packet broadcasts that increase the accuracy of the estimations and, most importantly, better determine channel traffic. Moreover, designers of link quality metrics can take advantage of *TLQAP* feedback and use it as a reference to evaluate the performance of their design during an experiment.

In [11] authors determine channel traffic by sending back-to-back just two probes and then measure their dispersion. They have observed in their experiments strong correlation between probe packet dispersion and traffic in the air. While we can use this approach in *TLQAP* to determine channel traffic by implementing the proposed probe priority queue scheme, we decided to estimate the average dispersion of large sets of consecutive packet probes. We believe that this approach enhances the offline measurements because it captures activity for a wider time frame. This is also why we have used history log as well.

6. CONCLUSION

Distributing the network bandwidth between experiments on wireless testbeds can be a very complicated task, especially when the testbed nodes are not operating in an RF isolated environment. The respective management frameworks should employ support to frequently inspect the link quality between the nodes. On testbeds where the nodes are in fixed locations, a history log of quality measurements can significantly increase the probability that a link will retain the measured quality during the actual experiment. What is more important, the testbed user has a good reference of the achievable bandwidth between the reserved nodes and can better evaluate the observed performance of the deployed experiment. Management framework extensions that measure performance should feature a low level subsystem that can bypass the network stack buffering and retransmission mechanisms, to make more accurate measurements and perform faster. *TLQAP* system design addresses the aforementioned considerations.

7. REFERENCES

[1] http://madwifi.org/.

[2] http://nitlab.inf.uth.gr/.

[3] http://www.emulab.net/.

[4] http://www.planet-lab.org/.

[5] http://www.winlab.rutgers.edu/.

[6] A. Anadiotis, A. Apostolaras, T. Korakis, and L. Tassiulas. A new scheme for slicing over experimental wireless testbeds. Technical report, University of Thessaly, 2009.

[7] J. Bicket, D. Aguayo, S. Biswas, and R. Morris. Architecture and evaluation of an unplanned 802.11b mesh network. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, 2005.

[8] I. Broustis, J. Eriksson, S. V. Krishnamurthy, and M. Faloutsos. A blueprint for a manageable and affordable wireless testbed: Design, pitfalls and lessons learned. In *IEEE International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities.*

[9] D. S. J. D. Couto, R. Morris, D. Aguayo, and J. Bicket. A high-throughput path metric for multi-hop wireless routing. *Wireless Networking*, 11(4), 2005.

[10] R. Draves, J. Padhye, and B. Zill. Routing in multi-radio, multi-hop wireless mesh networks. In *MobiCom '04: Proceedings of the 10th annual international conference on Mobile computing and networking.* ACM, 2004.

[11] M. A. Ergin and M. Gruteser. Using packet probes for available bandwidth estimation: a wireless testbed experience. In *WiNTECH '06: Proceedings of the 1st international workshop on Wireless network testbeds, experimental evaluation & characterization.* ACM.

[12] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 2000.

[13] C. Koksal and H. Balakrishnan. Quality-aware routing metrics for time-varying wireless mesh networks. *IEEE Selected Areas In Communications*, 2006.

[14] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. In *IEEE Wireless Communications and Networking Conference (WCNC 2005).*