Overhead analysis of embedded wireless testbeds

Overhead Analysis of Embedded Wireless Testbeds

Nathan Samson, Glenn Daneels, Bart Braem, Chris Blondia
Department of Mathematics and Computer Science
University of Antwerp - IBBT - PATS Research Group
Middelheimlaan 1, B-2020, Antwerp, Belgium
Email: {bart.braem,chris.blondia}@ua.ac.be

***Abstract*—As community networks are being deployed more often, interest in using them as testbeds is growing too. Frameworks like the cOntrol and Management Framework (OMF) are helpful software tools to organize, control and instrument the testbeds deployed on those networks. However, community networks have a very heterogeneous infrastructure in which less powerful, embedded devices will play an important role due to their low cost. This work will analyze the overhead generated by the OMF framework when deployed on embedded devices. Possible performance hits introduced by the overhead on both desktop and embedded systems will be presented and compared.**

I. Introduction

In the last few years more community networks are being deployed. Examples of these networks are Guifi[1] and the Athens Wireless Metropolitan Network[2]. In community networks, embedded systems are of increasing importance, providing a low performance and thus low-cost method of being part of the community. Together with the wide variety of hardware, software and network capacity, these networks provide large-scale and real world environments for researchers to test the quality and stability of new technologies and protocols.

By deploying a testbed on top of community networks, researchers can perform experiments in a realistic environment. To manage those often large-scale and complex testbeds, specific frameworks to instrument and control the environment are used. This paper considers the cOntrol and Management Framework (OMF)[3][4]. It allows to manage the testbed resources and to deploy and run experiments. During the execution of experiments, the measurements are collected by the OMF Measurement Library (OML)[5] that accompanies the OMF.

In this paper, an initial analysis of the possible performance hit introduced by the OMF framework is presented. Performance on both desktop and embedded systems will be presented and compared.

II. Hardware

Two hardware platforms were used in the experiments. Two identical devices of each hardware platform were used to study the performance impact.

One platform used a Dell Desktop system which will be referred to as the Silver nodes. The other platform is a low performance embedded system of PCEngines referred to as the Alix nodes.

An overview of the hardware in each platform is given in table I.

	Silver Nodes	Alix Nodes
CPU	Intel Pentium 4 3.0GHz	AMD Athlon Geode 1GHz
Memory	1024 MB	256 MB
Disk	Seagate 250 GB 7200 RPM SATA	Transcend 133x 4 GB Compact Flash
Wireless	Atheros AR5413/AR5414 [AR5006X(S) 802.11abg] (rev 01)	

TABLE I
OVERVIEW OF USED HARDWARE

III. SOFTWARE

A. Operating System

The experiments were performed on two different software systems. The software on the Silver nodes was based on the Ubuntu 12.04 beta distribution, the Alix nodes were running the OpenWrt Trunk. The latter was chosen because this work was performed in the scope of an ongoing project with OpenWrt on embedded devices.

An overview of the used software and version numbers can be found in table II.

	Silver Nodes	Alix Nodes
Distribution	Ubuntu 12.04 beta	OpenWrt Trunk
Linux Kernel	3.2.0-20 generic-pae #33-Ubuntu SMP	3.2.12
Optimizations	No	Yes
Ruby	1.8.7p352	1.9.2p0
Iperf	2.0.5 pthreads	2.0.5 single threaded
OMF	5.4	
OML Iperf	2.0.5 pthreads	
OTG	2.5	

TABLE II
OVERVIEW OF USED SOFTWARE

An attempt was made to make the difference between the two systems as small as possible. The used software — discussed below — has the exact same version on each platform. To be able to use a new Kernel the beta version of Ubuntu 12.04 was installed because it was the only release that contained a recent Kernel version.

The OpenWrt software on the Alix nodes was compiled with optimizations for the specific type of hardware. The comparison between the two hardware platforms still holds because embedded devices are often built with specific optimizations for their platform.

B. OMF

OMF is a framework for controlling and managing a networking testbed. It provides a set of software tools to control the testbed resources and defines a domain-specific language "OMF Experiment Description Language" (OEDL)[6]. OEDL provides a set of OMF commands and statements to describe an experiment executed on the testbed. An experimenter can describe different resources, applications and other parameters related to the experiment.

An OMF testbed consists of several entities which work together in order to manage the testbed and run an experiment[3]. When starting a new experiment, an Experiment Description (ED) written in OEDL is passed to the Experiment Controller (EC). Based on the ED, the EC will send directives to the Aggregate Manager (AM) to initialize the referenced resources and will send commands to the Resource Controllers (RC), associated with the resources. In this paper, the RC's are the Silver nodes and the Alix nodes. By executing the commands received from the EC, the experiment is

executed. The application used in the experiment may send measurement data to a measurement library such as OML.

C. OML

In this paper, the OMF Measurement Library is the used measurement library as OMF uses it by default. However, the experimenter is not obliged to use OML and can use any measurement library in combination with his own measurement collection scheme. OML can also be used independently of OMF.

OML allows experimenters to define multiple measurement points inside their application. This measurement points can be aggregated e.g. to sum the data. That data is aggregated in a so-called sampling window. The size of this window is given by the `samples` parameter which will return measurements for every X measurements or by the `interval` parameter which will return measurements for every X seconds.

OML uses a client/server architecture. While the experiment is running, the client sends its measurement data received from the application(s) to the OML Collection Server. The client will reside on the RC and the Collection Server on an other server. The experimenter can access the results on the Collection Server.

D. Iperf

Iperf[7] is an open-source application developed by NLANR/DAST to study the maximum throughput performance of network systems. Iperf can send both TCP and UDP packets from client to server. The data stream can easily be modified by changing parameters such as `bandwidth` or datagram `length`. Network characteristics such as bandwidth, delay jitter and datagram loss are reported.

Iperf for OML[8] is a version of the Iperf application which was modified by adding OML measurement points to the source code. This way, the data reported by Iperf is sent to the OML server. This application is started by an OMF experiment.

E. OTG

OTG/OTR (ORBIT Traffic Generator / Receiver)[9] achieves the same goal as Iperf, i.e. studying the throughput of a network link. OTG supports OML out-of-the-box, but has less features than Iperf. E.g., it supports only UDP traffic while Iperf also supports TCP traffic. This application is started by an OMF experiment.

IV. Measurement Setup

When executing the experiments, the antennas (type Reverse polarity SMA) of the wireless cards in the nodes were at a small distance (~50cm). However, the nodes were also exposed to interference from other wireless networks near the nodes. The Transmit Power of the Silver and the Alix nodes is 27 dBm.

All programs were configured to use the UDP protocol. Using different packet sizes will have an impact of maximum achievable throughput[10]. All experiment configurations have a packet size of 1470, nearly the maximum size before fragmentation occurs. This is the default in some applications, like e.g. Iperf.

A. Iperf

The command (Bash shell syntax) used to run the Iperf experiments on the sending machine is

```
iperf -c IP of receiver -u -b
$((54 * 1024 * 1024)) -l 1470 -t
300
```

and on the receiving machine it is

```
iperf -s -u
```

At the end of each run the Iperf client reports how much data was sent during which time frame, and how much of it was

received by the Iperf server. The results are based on the rate that was indicated by the receiving machine.

B. OML Iperf

Iperf for OML was run in two different configurations for each machine. Both configurations use UDP.

In the *Iperf OML All* configuration all packets statistics were assembled and sent to the OML server. The transmission and reception timestamps of each packet are recorded. The Application Description part of this experiment can be found in appendix A.

In the *Iperf OML Aggregated* configuration all packet data was observed on the resource controllers, but the results were aggregated in aggregated samples over 30 seconds. For each sample the sum and average of all packet sizes is calculated and passed to the OML server. The Application Description part of this experiment can be found in appendix B.

When the first result (*Iperf OML All*) is compared with the results of the "plain" Iperf results the impact of the OML reporting can be observed. The second experiment (*Iperf OML Aggregated*) has a lower reporting rate, to study the impact of OML measurement aggregation.

C. OTG

OTG was configured, like Iperf for OML, with two different configurations similar to the ones of Iperf for OML.

The first configuration also sends all packet information like packet size and transmission/reception time to the OML server. The Application Description part of this experiment can be found in appendix C.

The second configuration aggregated packet information over an interval of 30 seconds and sent the transmission/reception time of the first and last packet and the sum of all packet sizes. The Application Description part of this experiment can be found in appendix D.

Note that a bug was found in the receiving program of OTG, namely OTR. The original OTR version reported more incoming packets than outgoing but with a smaller size (1024 bytes). This was caused by a hard coded maximum read buffer size of 1024 bytes. This constant was changed to 1470 bytes, so packets could be read with one `recvfrom` call.

In the following section, the results are compared and analyzed.

V. Results

Every type of experiment was repeated 100 times and the duration of each experiment was 300 seconds. After all experiments were executed the results of the different runs of each experiment were averaged. In figure 1 the average throughput of the different systems is shown, the error bars in the graph indicate the standard deviation. No result is shown for the OML Iperf Alix All experiment, as the reported performance was very low (see section V-B). In what follows, the experiments will be discussed separately. In the tables, the average, minimum, maximum and standard deviation σ of the throughput will be considered.

A. Iperf

	Silver Nodes	Alix Nodes
Avg.	27.02	26.52
Min	23.40	23.10
Max	28.30	27.80
σ	1.02	1.09

TABLE III
Throughput for the OTG Experiments (in Mbit/s)

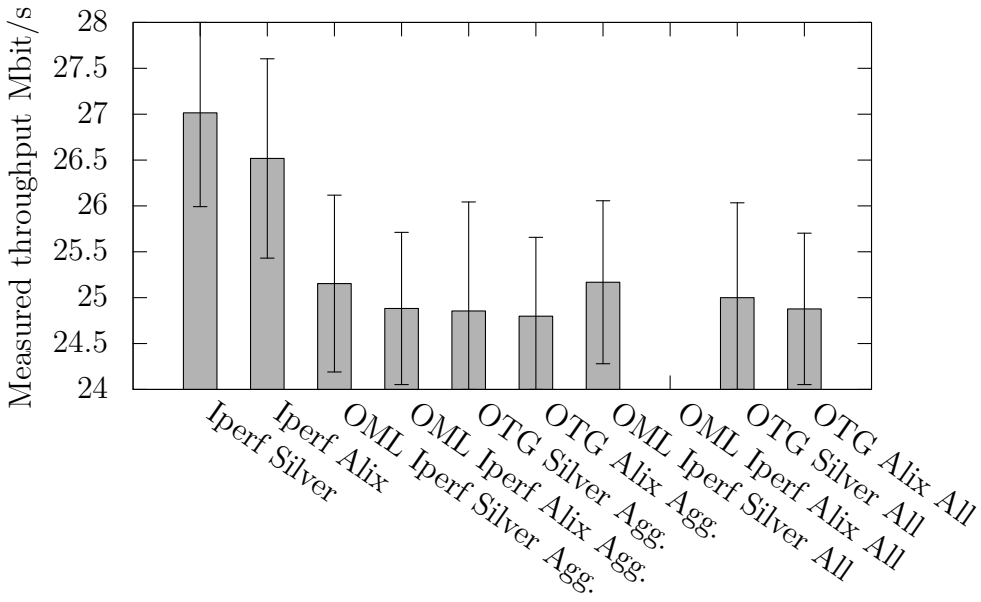

Fig. 1. Detailed overview of observed throughput

The maximum throughput achieved by Iperf at the Silver nodes is 28.30 Mbit/s which is an appropriate speed according to Bredel and Fidler[10].

The average throughput is nearly 2% higher for the Silver nodes, but since the standard deviation is also quite high for both systems this result can be expected.

B. OML Iperf

	Silver All	Alix All	Silver Agg.	Alix Agg.
Avg.	25.17	3.97	25.15	24.88
Min	22.98	2.68	22.32	22.46
Max	26.34	8.31	26.50	26.27
σ	0.89	0.53	0.96	0.83

TABLE IV
THROUGHPUT FOR THE OML IPERF EXPERIMENTS (MBIT/S)

In table IV the results of the OML Iperf experiment configurations are displayed. *Silver All* and *Alix All* refers to the *OML Iperf All* experiment on respectively the Silver and Alix nodes. *Silver Agg.* and *Alix Agg.* refer to the *OML Iperf Aggregated* experiment on respectively the Silver and Alix nodes.

The first clear result is the *OML Iperf All* experiment, which was reporting very low throughput on the Alix nodes. This experiment observed only a throughput around 4Mbit/s most of the time, much lower than other experiments. Further investigation made clear that the CPU time spent in the OMF program on the RC was much higher than in the same experiment on the Silver nodes. In a similar experiment where all packets were reported to OML, namely the OTG experiment (discussed later on), the CPU usage of the OMF program was negligible. This is the only experiment where we have seen a dramatic decrease in performance on the embedded devices. The same configuration on the Silver nodes did not show any anomalies and reached an averaged rate of 25.17 MBit/s.

The Alix nodes show an average throughput of 24.88 MBit/s for the *OML Iperf OML Aggregated* configuration. The difference is minimal with the Silver nodes which show an average of 25.15 MBit/s.

As can be seen from these results, the impact of aggregating the results does not have any impact on the Silver nodes. The Alix nodes experienced a strong performance increase by aggregating the data.

All throughput results are smaller than the baseline performance obtained in the Iperf experiments, up to 7% for the Silver nodes.

C. OTG

	Silver All	Alix All	Silver Agg.	Alix Agg.
Avg	25.00	24.88	24.94	24.80
Min	22.16	22.52	22.14	22.55
Max	26.06	26.33	26.05	26.25
σ	1.03	0.83	1.07	0.86

TABLE V
THROUGHPUT FOR THE OTG EXPERIMENTS (MBIT/S)

In table V the results of all OTG experiment configurations are displayed. *Silver All* and *Alix All* refers to the *OTG All* experiment on respectively the Silver and Alix nodes. *Silver Agg.* and *Alix Agg.* refer to the *OTG Aggregated* experiment on respectively the Silver and Alix nodes.

In both configurations the throughput difference between the Silver and Alix nodes stays under 1%.

Aggregating the results, and thus minimizing reporting rate, does not have a significant impact on either the Silver and Alix nodes. The performance throughput even decreases slightly on both platforms.

D. Analysis

In all studied configurations the reported throughput results were lower than the one measured in the normal Iperf experiment. No significant difference can be seen between the OTG experiments and the OML Iperf experiments, except for the All experiment on the Alix nodes. This can be expected since the two applications in these configurations should be performing the same task.

VI. Conclusion & Future Work

Different traffic generators were used to perform overhead analysis on different systems. It is clear that OML introduced some overhead on the throughput of the systems on both powerful machines and on embedded devices. In only one experiment significant differences were found between the Silver nodes and the Alix nodes.

Using the OML framework on the other hand will have an impact of about 6%, whether run on embedded system or on powerful machines. In one particular case the impact was so dramatic that the throughput was less than 25% of the throughput in the other experiments.

Future work will involve a more complete analysis, where metrics such as delay will be studied. Also, the performance hit introduced by OML and OMF will be studied to reduce it for the specific characteristics of embedded platforms. In this scope, the authors hope to resolve the low throughput in the particular OML Iperf experiment on the embedded system.

Appendix

A. OML Iperf All Experiment Description

```
1 defGroup('Sender',property.theSender)
2   do |node|
3   node.addApplication("iperf-5.4") {|
        app|
4     app.setProperty('port', 3000)
```

```
    app.setProperty('client', property.receiverip)
    app.setProperty('reportstyle', 'O')
    app.setProperty('interval', '30')
    app.setProperty('bandwidth', property.bitrate)
    app.setProperty('udp', true)
    app.measure('packets', :samples=>1)
    app.setProperty('time', property.runtime)
  }
#...
end

defGroup('Receiver',property.theReceiver)
  do |node|
  node.addApplication("iperf-5.4") {|app|
    app.setProperty('port', 3000)
    app.setProperty('server', true)
    app.setProperty('reportstyle', 'O')
    app.setProperty('interval', '30')
    app.setProperty('udp', true)
    app.measure('packets', :samples=>1)
  }
# ...
end
```

B. OML Iperf Aggregated Experiment Description

```
defGroup('Sender',property.theSender)
  do |node|
  node.addApplication("iperf-5.4") do |app|
    app.setProperty('port', 3000)
    app.setProperty('client', property.receiverip)
    app.setProperty('reportstyle', 'O')
    app.setProperty('interval', '1')
    app.setProperty('bandwidth', property.bitrate)
    app.setProperty('udp', true)
    app.measure('packets', :interval => 30) do |mp|
      mp.filter('packet_size', 'sum')
      mp.filter('packet_size', 'avg')
      mp.filter('packet_time_s', 'first')
      mp.filter('packet_time_s', 'last')
      mp.filter('packet_time_us', 'first')
      mp.filter('packet_time_us', 'last')
    end
    app.setProperty('time', property.runtime)
  end
# ...
end

defGroup('Receiver',property.theReceiver)
  do |node|
  node.addApplication("iperf-5.4") do |app|
    app.setProperty('port', 3000)
    app.setProperty('server', true)
    app.setProperty('reportstyle', 'O')
    app.setProperty('interval', '1')
    app.setProperty('udp', true)
    app.measure('packets', :interval => 30) do |mp|
      mp.filter('packet_size', 'sum')
      mp.filter('packet_size', 'avg')
      mp.filter('packet_time_s','first')
      mp.filter('packet_time_s', 'last')
      mp.filter('packet_time_us', 'first')
      mp.filter('packet_time_us', 'last')
    end
  end
# ...
end
```

C. OTG All Experiment Description

```
defGroup('Sender',"omf.pats.alix1")
  do |node|
  node.addApplication("test:app:otg2") do |app|
    app.setProperty('udp:local_host', property.senderip)
    app.setProperty('udp:dst_host', property.receiverip)
    app.setProperty('udp:dst_port', 3000)
    app.setProperty('cbr:size', 1470)
    app.setProperty('cbr:rate', 54*1024*1024)
    app.setProperty('generator', 'cbr')
    app.measure('udp_out', :samples => 1)
  end
# ...
end
```

```
defGroup('Receiver',"omf.pats.alix2")
  do |node|
  node.addApplication("test:app:otr2")
       do |app|
    app.setProperty('udp:local_host',
        property.receiverip)
    app.setProperty('udp:local_port',
        3000)
    app.measure('udp_in', :samples =>
        1)
    end
# ...
end
```

D. OTG Aggregated Experiment Description

```
defGroup('Sender',"omf.pats.alix1")
  do |node|
  node.addApplication("test:app:otg2")
       do |app|
    app.setProperty('udp:local_host',
        property.senderip)
    app.setProperty('udp:dst_host',
        property.receiverip)
    app.setProperty('udp:dst_port',
        3000)
    app.setProperty('cbr:size', 1470)
    app.setProperty('cbr:rate',
        54*1024*1024)
    app.setProperty('generator', 'cbr'
        )
    app.measure('udp_out', :interval
        => 30) do |mp|
    mp.filter('pkt_length', 'sum')
    mp.filter('pkt_length', 'avg')
    mp.filter('ts', 'first')
    mp.filter('ts', 'last')
    end
  end
# ...
end
```

```
defGroup('Receiver',"omf.pats.alix2")
  do |node|
  node.addApplication("test:app:otr2")
       do |app|
    app.setProperty('udp:local_host',
        property.receiverip)
    app.setProperty('udp:local_port',
        3000)
    app.measure('udp_in', :interval =>
         30) do |mp|
    mp.filter('pkt_length', 'sum')
    mp.filter('pkt_length', 'avg')
    mp.filter('ts', 'first')
    mp.filter('ts', 'last')
    end
  end
# ...
end
```

References

[1] Guifi.net. http://guifi.net/.

[2] Athens wireless metropolitan network. http://www.awmn.net/.

[3] T. Rakotoarivelo, G. Jourjon, M. Ott, and I. Seskar. Omf: a control and management framework for networking testbeds. *Operating systems review*, 43(4):54, 2009.

[4] NICTA. Control and management framework. http://omf.mytestbed.net/.

[5] NICTA. Omf measurement library. http://oml.mytestbed.net/projects/oml/wiki.

[6] Oedl - the omf experiment description language. http://oml.mytestbed.net/projects/omf/wiki/OEDL-5-2.

[7] NLANR/DAST. Iperf. http://sourceforge.net/projects/iperf/.

[8] OMF. Iperf with oml support. http://mytestbed.net/projects/iperf/wiki.

[9] OMF. Otg2. http://mytestbed.net/projects/oml/wiki/Otg2.

[10] M. Bredel and M. Fidler. A measurement study of bandwidth estimation in ieee 802.11 g wireless lans using the dcf. *NETWORKING 2008 Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, pages 314–325, 2008.